

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 «database» – содержит описание подключения к БД;
- папка logs – содержит файл «script.log», содержащий логи работы программы.

Далее будут подробнее рассмотрено содержание папок.

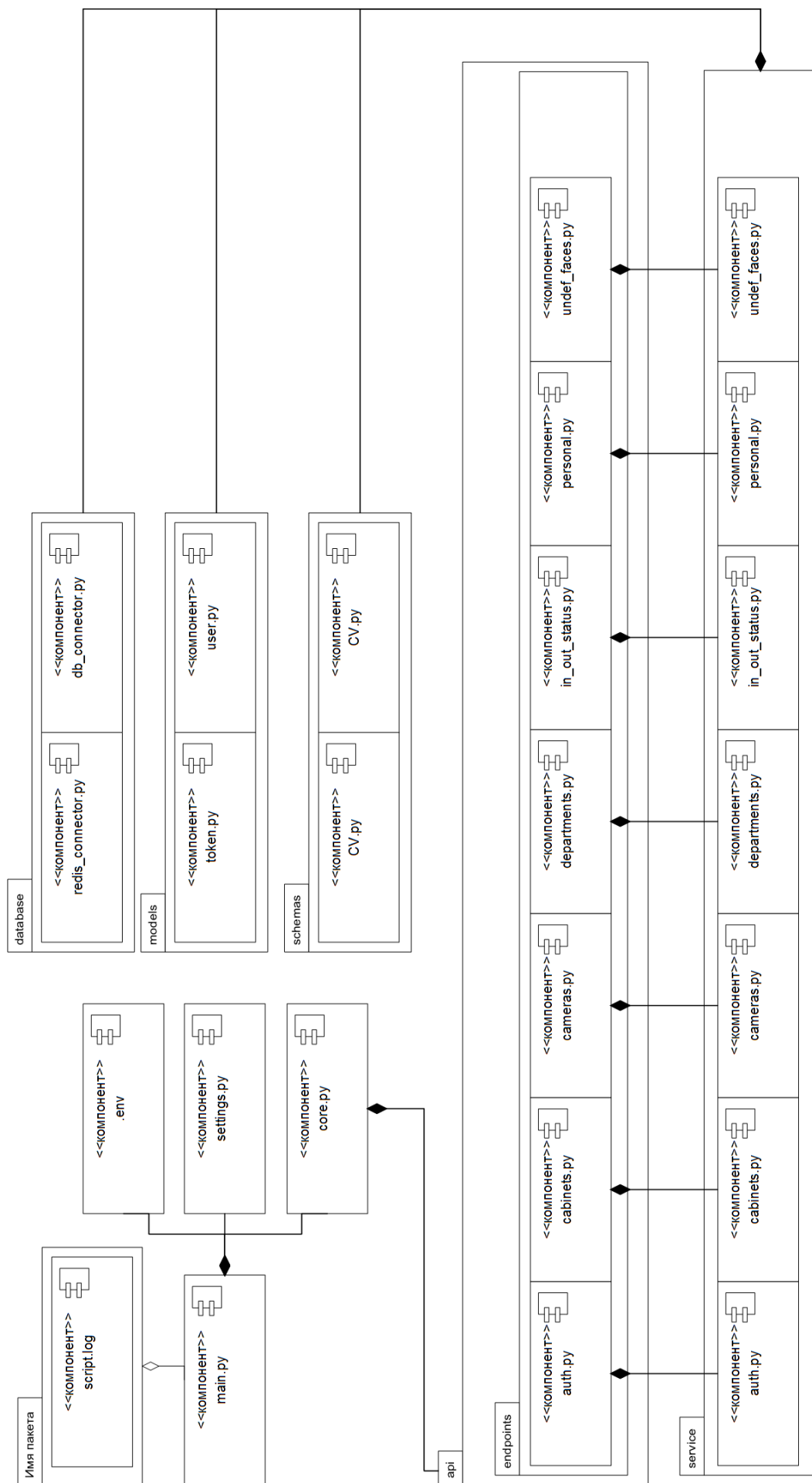


Рисунок 9 – диаграмма компонентов backend

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 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 «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;

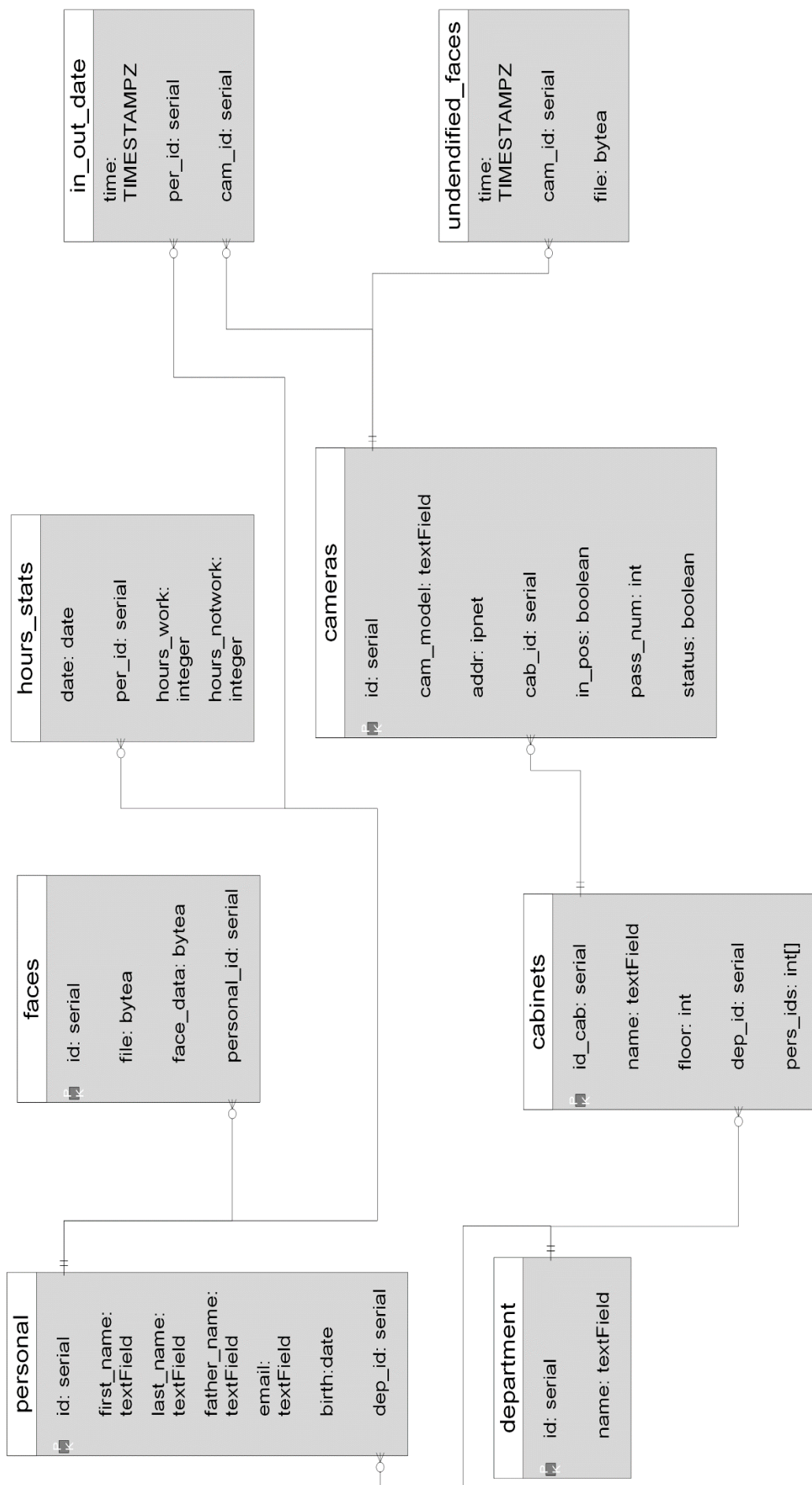


Рисунок 10 – Схема базы данных

- 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 данный 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.

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 лиц. Если на запрос пришло 0 векторов, то программа логирует сообщение об ошибке, затем останавливается на 5 секунд и повторяет запрос и так, пока

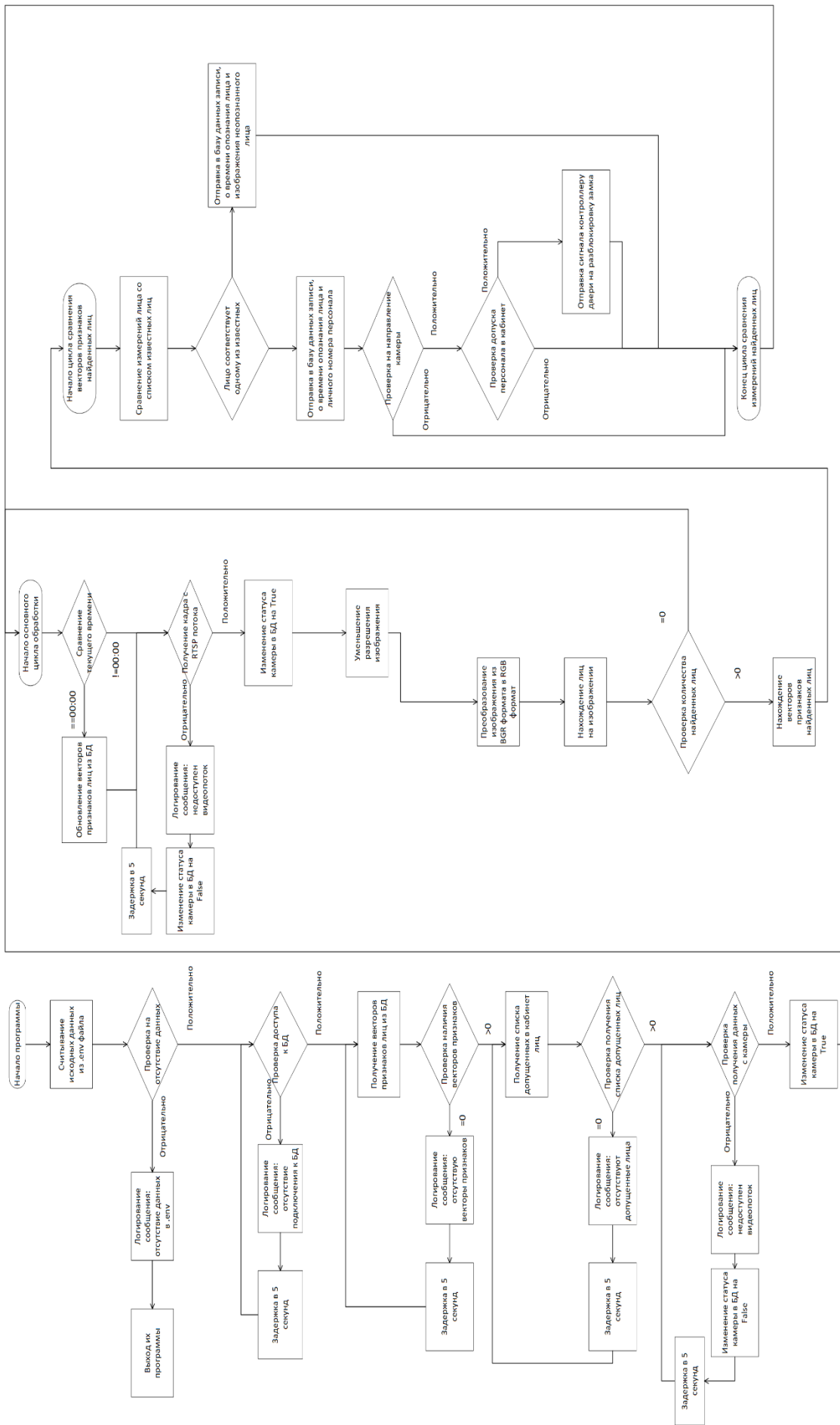


Рисунок 11 – Алгоритм программы обработки

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 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 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 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 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

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.466152.006 ПД.

Страница содержит навигационную панель (см. рисунок 12), предназначенная для переключения между разделами.

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

Раздел «Personal info» – предназначен для отображения списка персонала (см. рисунок 13), 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 идентификационному номеру в базе данных (см. рисунок 14) и также фото работников (см. рисунок 15).

Раздел «In-Out status» – предназначен для отображения регистрации посещений помещений в виде таблицы (см. рисунок 16) со следующими столбцами:

– столбец «timedate» – дата и время посещения;

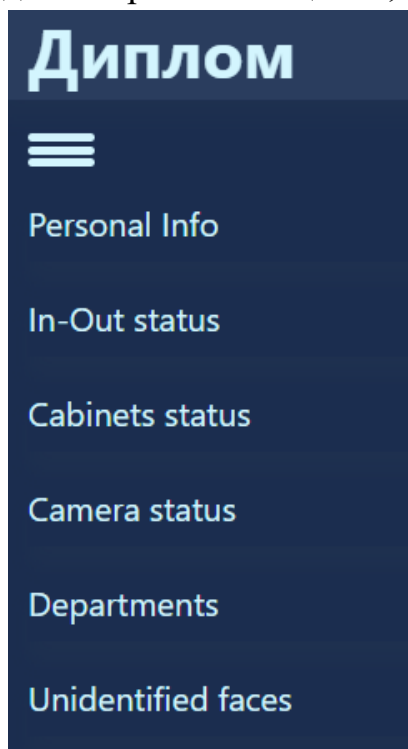


Рисунок 12 – Навигационная панель



Рисунок 13 – Список персонала



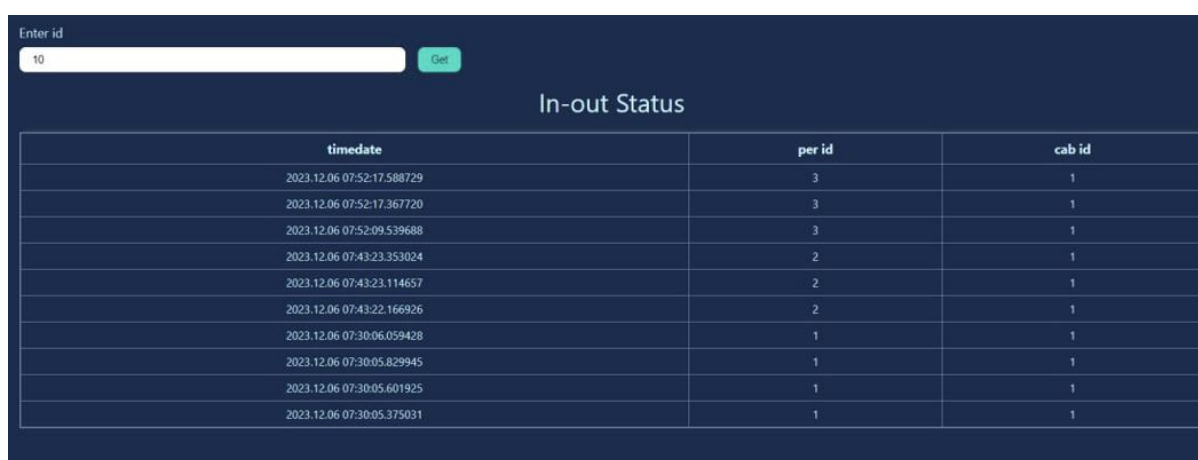
Рисунок 14 – Окно поиска



Enter id

Рисунок 15 – Окно поиска фотографий

- столбец «per id» – идентификационный номер опознанной персоны;
- столбец «cab id» – идентификационный номер кабинета, где зафиксировалось событие.



Enter id

In-out Status

timdate	per id	cab id
2023.12.06 07:52:17.588729	3	1
2023.12.06 07:52:17.367720	3	1
2023.12.06 07:52:09.539688	3	1
2023.12.06 07:43:23.353024	2	1
2023.12.06 07:43:23.114657	2	1
2023.12.06 07:43:22.166926	2	1
2023.12.06 07:30:06.059428	1	1
2023.12.06 07:30:05.829945	1	1
2023.12.06 07:30:05.601925	1	1
2023.12.06 07:30:05.375031	1	1

Рисунок 16 – Список посещения помещений

Раздел «Cameras status» предназначен для отображения списка камер, задействованных в системе, в виде таблицы (см. рисунок 17) 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.

Cameras Status						
id cam	cam model	ip addr	cab id	in pos	pass num	status
1	www	192.168.0.1/32	2	●	3	●
4	www	192.168.0.1/32	2	●	4	●
3	www	192.168.0.1/32	2	●	4	●
2	www	192.168.0.1/32	2	●	3	●
6	www	192.168.0.1/32	3	●	1	●
5	www	192.168.0.1/32	3	●	1	●

Рисунок 17 – Список камер

Так же этот раздел содержит окно поиска помещений по идентификационному номеру в базе данных, окно полностью повторяет окно поиска персонала (см. рисунок 14).

Раздел «Cabinets status» предназначен для отображения списка помещений, представленного в виде таблицы (см. рисунок 18) 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 идентификационному номеру в базе данных, окно полностью повторяет окно поиска персонала (см. рисунок 14).

Enter id

Show all

Get

Cabinets Status

id cab	name	floor	dep id
2	101	1	1
3	502	5	2

Рисунок 18 – Список помещений

Раздел «Departments» – предназначен для отображения списка отделов в виде таблицы (см. рисунок 19) со следующими столбцами:

- столбец «id» – идентификационный номер отдела;
- столбец «name» – название отдела.

Departments	
id cab	name
1	depA
2	depB

Рисунок 19 – Список отделов

Раздел «Unidentified faces» – предназначен для отображения списка неопознанных лиц в виде таблицы (см. рисунок 20) со следующими столбцами:

- столбец «timedate» – дата и время фиксации лица;
- столбец «cam id» – идентификационный номер камеры;
- столбец «file» – фото неопознанного лица.



Unidentified faces		
timedate	cam id	file
2023-12-06 18:03:16.746048	1	1
2023-12-06 18:03:16.493775	1	1
2023-12-06 18:03:16.252257	1	1
2023-12-06 18:03:16.011363	1	1
2023-12-06 18:03:15.769918	1	1
2023-12-06 18:03:15.528269	1	1
2023-12-06 18:03:15.283506	1	1
2023-12-06 18:03:15.036956	1	1
2023-12-06 18:03:14.125854	1	1
2023-12-06 18:03:13.850869	1	1

Рисунок 20 – Список неопознанных лиц

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.