**ВСЕРОССИЙСКАЯ ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»**

**«ШАГ В БУДУЩЕЕ, МОСКВА»**

регистрационный номер

**Информатика и системы управления**

# Разработка программно-аппаратного комплекса для управления проходной

**Автор:** Грачев Константин Алексеевич,

ученик 11 класса, ГБОУ школы №1279 «Эврика» г. Москва

**Научный руководитель:** Балдин Александр Викторович,

д.т.н. профессор кафедры ИУ-5

**2023 год**

# АННОТАЦИЯ

Сейчас стало обычной практикой, что доступ на территорию или на предприятие (организацию) осуществляется через автоматизированную систему, которая состоит из технических средств и программного обеспечения. Однако, существующие на данный момент системы имеют ряд недостатков. В частности, очень ограниченный объем памяти для хранения данных и низкая эффективность при обновлении больших списков доступа на турникетах.

Целью работы является создание совершенно новой системы, не имеющая приведенных выше недостатков. Вместо нескольких отдельных контроллеров в турникетах предлагается использовать один более мощный, созданный на основе одноплатного микрокомпьютера Orange PI или аналогов для группы турникетов. Это в разы уменьшит время, необходимое для обновления списков доступа, а также позволит хранить данные на внешних носителях с практически неограниченным объемом памяти.

Для предлагаемой системы разработано программное обеспечение на языке Python с использованием библиотек Opi.GPIO, MFRC522 и СУБД sqlite3, а также встроенных средств языка программирования. В сравнении с аналогами основным преимуществом является открытый исходный код, дающий возможность настраивать систему в соответствии с особенностями различных мест эксплуатации.

В результате работы будет создан программно-аппаратный комплекс для управления проходной с одним контроллером на всю проходную и практически неограниченной памятью для хранения данных.

# Оглавление

# ВВЕДЕНИЕ

Сейчас стало обычной практикой, что доступ на территорию или на предприятие (организацию) осуществляется через автоматизированную систему, которая состоит из технических средств и программного обеспечения, управляющего техническими средствами.

Практически каждый человек сталкивался с подобными системами, например, при проходе в метро. Такие системы используются в школах Москвы. Проход учащихся в здания учебных заведений осуществляется с помощью карт «Москвенок» и позволяет родителям и сотрудникам школ отслеживать посещаемость учеников в автоматическом режиме. Помимо этого многие люди пользуются общественным транспортом, где для учета и оплаты прохода используется карта «Тройка».

Для прохода на территорию, или в здание, используют системы контроля и управления доступом — СКУД. В настоящее время существует большое количество вариантов СКУД, однако все они имеют схожую конструкцию. Они состоят из:

1. Сервера
2. Компьютера проходной
3. Преграждающего устройства (например, турникета)
4. Идентификатора (пропуска, QR-кода)
5. Считывателя
6. Контроллера

Сервер хранит всю информацию; компьютер в проходной управляет турникетами; турникеты с контроллерами и считывателями открывают доступ.

СКУД выполняют ряд различных задач, таких как:

- ограничение доступа на заданную территорию
- идентификация лица, имеющего доступ на заданную территорию
- учёт рабочего/учебного времени

Однако такая система имеет ряд недостатков, одним из которых является ограниченность возможностей контроллера, в частности его памяти. К примеру, в МГТУ им. Н. Э. Баумана используется оборудование, изготовленное фирмой «Parsec», обслуживающее около 27000 обучающихся, 4500 преподавателей и порядка 5000 сотрудников. Каждый из них имеет пропуск, хранящий 16-символьный идентификатор, в котором 1 символ занимает 1 байт памяти, также каждый из них может привязать пропуск к банковской карте, пластиковому пропуску с чипом или сгенерировать временно действующий QR-код. Если считать что объём памяти, выделяемый для привязка карты и создания QR-кода равен объёму памяти для идентификатора, получаем, что контроллеру потребуется около (27000 + 4500 + 5000) * 16 * 3 = 1 752 000 байт памяти, помимо этого в Университете могут проходить различные мероприятия, гостям которого тоже потребуется пропуск или QR-код для входа. С таким объемом информации контроллер может просто не справиться. Еще одна проблема, с которой столкнулись в ходе эксплуатации - это обновление списков доступа на турникетах. В каждом турникете стоит свой контроллер. Изменение одной записи на каждом из них занимает около трех секунд, что приводит к проблемам при внесении большого количества изменений (например, при зачислении новых студентов или проведении массовых мероприятий). Время необходимое на обновление всех устройств может занимать несколько дней, что крайне затрудняет оперативное внесение изменений.

Для решения проблем, которые есть в СКУД я хочу сделать контроллер с большой памятью и возможностью управлять несколькими турникетами.

## СУЩЕСТВУЮЩИЕ АНАЛОГИ

Существует множество разных производителей СКУД. В качестве аналогов были выбраны контроллеры, способные работать с турникетами (триподами) и позволяющие хранить от 100 000 карт доступа. Характеристики устройств и их стоимость взята с официальных сайтов компаний, выпускающих устройства. Для разработанного контроллера характеристики получены на основании результатов тестирования и исходя из аппаратных ограничений одноплатного компьютера (количество портов ввода-вывода, объема памяти). Стоимость указана приблизительно, исходя из затрат на приобретения необходимых деталей и расходных материалов на маркетплейсе Ozon.

| | RUBEZH STRAZH STR20-IP-Ent | RusGuard ACS-202-CE | Parsec NC-100K-IP | Разрабатываемое устройство |
|---|---|---|---|---|
| Количество карт доступа | 100 000 | 1 000 000 | 100 000 | 500 000+ |
| Количество событий в журнале | 400 000 | 60 000 | 53 000 | 5 000 000+ |
| Интеграция с внешним сервером | + | + | + | + |
| Интеграция с существующим оборудованием | Требует дополнительное устройство (STR-1AP на каждый турникет) | + | + | + |

| | | | | |
|---|---|---|---|---|
| Доступность исходных кодов | - | - | - | + |
| Количество подключаемых турникетов | 10 | 2 | 1 | 6 |
| Стоимость | 82500 руб. | 76424 руб. | 45200 руб. | 10000 руб. |

Контроллеры RusGuard ACS-202-CE и Parsec NC-100K-IP основаны на микроконтроллерах.

Достоинства:

- Высокая скорость первоначальной загрузки (несколько секунд)
- Низкое энергопотребление

Недостатки:

- Ограниченный объем памяти, без возможности увеличения
- Низкая скорость обновления карт доступа

От разработки с применением микроконтроллера было решено отказаться ввиду сложности реализации и отладки, необходимости компиляции прошивки с последующей записью в устройство. А также отсутствием необходимых знаний в данной области.

Контроллеры RUBEZH STRAZH STR20-IP-Ent основан на одноплатный компьютере Raspberry PI CM3 (CM3+).

Достоинства:

- Большой объема памяти (1 Гбайт ОЗУ, 8 Гбайт флеш памяти)
- Большое количество портов ввода-вывода (48 портов)
- Возможность автономной работы
- Быстрое обновление карт доступа

Недостатки:

- Низкая скорость первоначальной загрузки (более минуты), вызванная необходимостью запуска ОС Linux.
- Ограниченный объем памяти, без возможности увеличения
- Для использования турникетов не поддерживающих OSPD (Open Supervised Device Protocol) интерфейс требуется докупать модуль доступа STR-1AP на каждый турникет

## ВЫБОР ТЕХНИЧЕСКИХ СРЕДСТВ ДЛЯ РАЗРАБОТКИ ПРОЕКТА

Для реализации проекта было принято решение использовать одноплатный микрокомпьютер. Такой вариант имеет следующие преимущества:

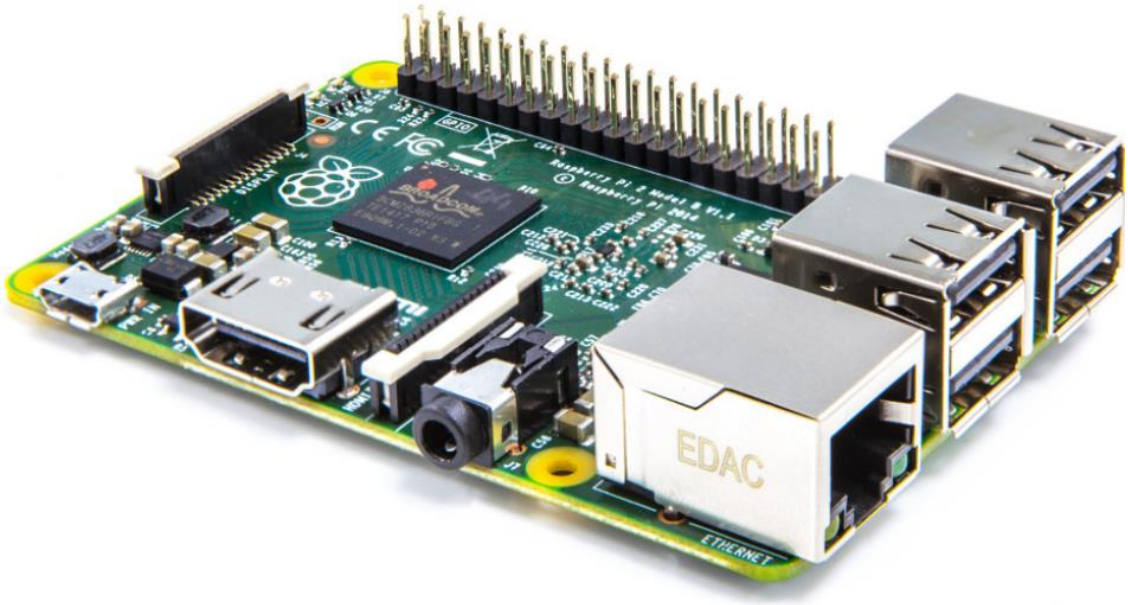
- Высокая производительность (по сравнению с решениями на микроконтроллерах)
- Большой объем памяти
- Возможность работы с внешними устройствами с помощью портов ввода/вывода
- Невысокая стоимость
- Доступность
- Полноценная операционная система (Linux)

Были рассмотрены следующие модели:

- Orange Pi Zero

Orange Pi PC Plus

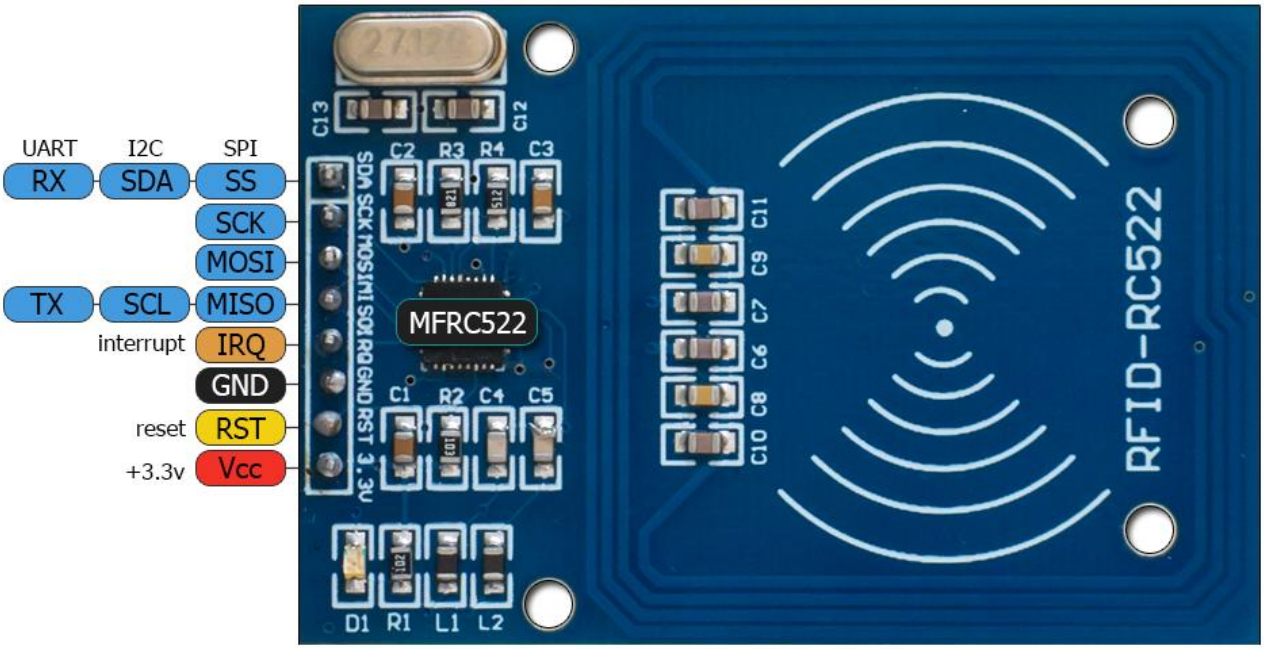

- Raspberry Pi 2 Model B



| | Orange Pi PC Plus | Orange Pi Zero | Raspberry Pi 2 Model B |
|---|---|---|---|
| CPU | H3 Quad-core Cortex-A7 1.2 ГГц | H2 Quad-core Cortex-A7 1.2 ГГц | A 900MHz quad-core ARM Cortex-A7 CPU |
| RAM | 1GB DDR3 | 512MB DDR3 | 1GB DDR3 |
| Хранение данных | MicroSD (Max. 32 ГБ) 8GB EMMC флеш | MicroSD (Max. 32ГБ) | MicroSD (Max. 32ГБ) |
| Сеть | 10/100 Ethernet RJ45 | 10/100 Ethernet RJ45 | 10/100 Ethernet RJ45 |
| Питание | DC блок питания USB OTG вход | USB OTG вход | USB OTG вход |
| USB 2.0 Порты | 3x USB 2.0, 1x USB 2.0 OTG | 1x USB 2.0, 1x USB 2.0 OTG | 4x USB 2.0, 1x USB 2.0 OTG |
| Подключение переферии | Разъем на 40 выводов | Разъем на 26 выводов | Разъем на 40 выводов |
| Индикация | Зеленый и красный светодиоды | Зеленый и красный светодиоды | Зеленый и красный светодиоды |

В результате сравнительного анализа выбор был сделан в пользу Orange pi pc plus, поскольку она обладает более высокопроизводительным процессором и встроенной памятью EMMC объемом 8 Гб для установки операционной системы и функционирования СКУД.

В качестве считывателя RFID-меток был использован модуль RC522.




Для подключения одного считывателя используется 4 вывода. Каждый дополнительный потребует еще 1 вывод.

Таким образом максимальное количество турникетов, которые можно подключить с учетом выбранного оборудования можно посчитать по формуле:

$N = (P – 3) / (S + T)$, где

N – количество турникетов

P – количество доступных выводов

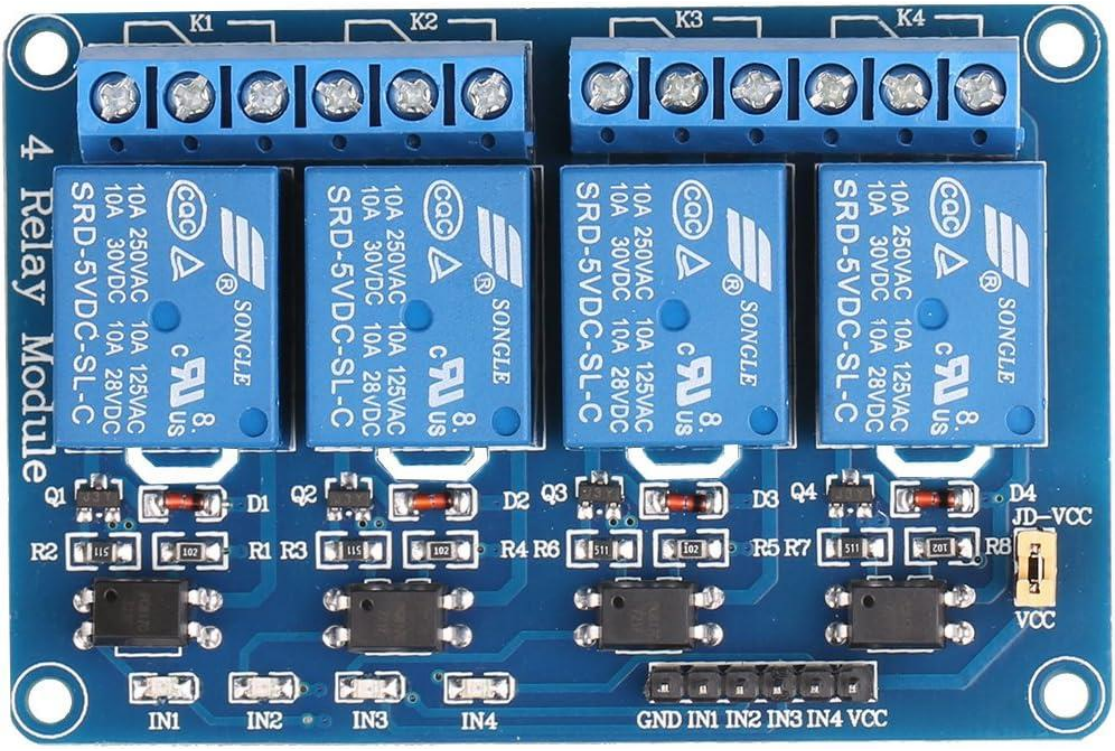
S – количество считывателей, подключенных к одному турникету

T – количество выводов, необходимых для управления триподом

$N = (28 – 3) / (2 + 2) = 6.25$

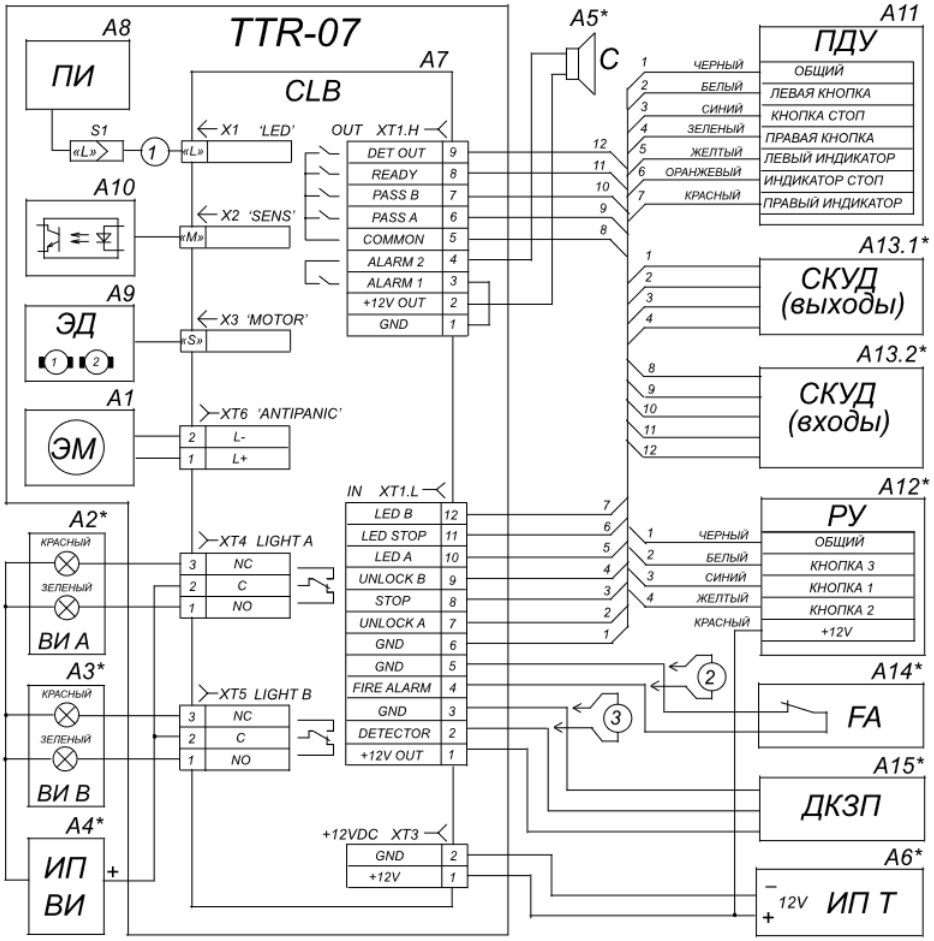
Получается можно подключить 6 двунаправленных турникета или 12 однонаправленных.

Управление турникетом осуществляется посредством четырехканального модуля реле.




Разработанный мной контроллер будет работать совместно с турникетом-триподом TTR-07.1 с автоматической «Антипаникой» фирмы PERCo, которые установлены в МГТУ им. Н.Э. Баумана на проходных.

Схема внешних подключений к плате CLB.2 турникета выглядит следующим образом:




Общая схема подключения всех компонентов выглядит следующим образом:

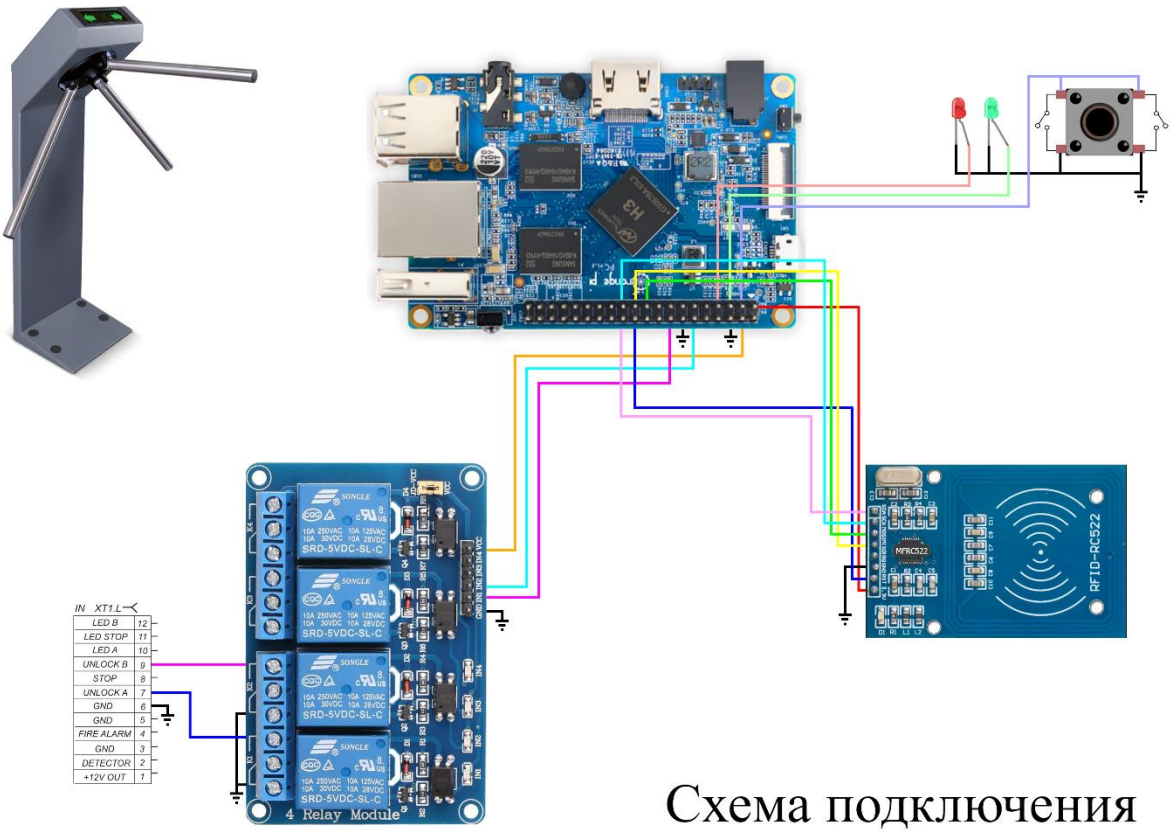
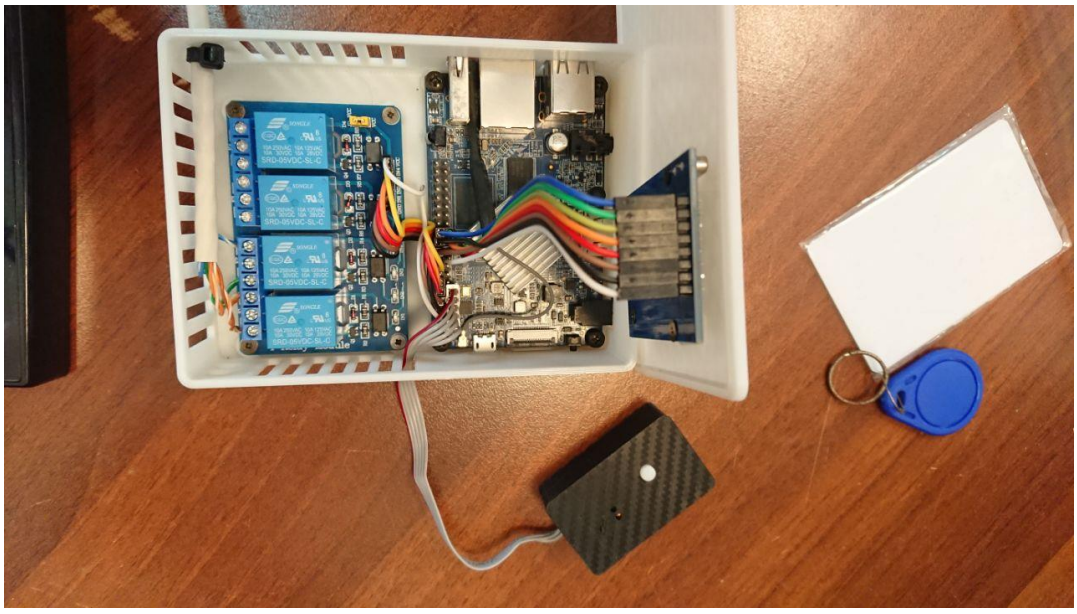



Рис. Схема подключения

Мной был разработан и напечатан на 3D-принтере корпус контроллера. Итогом разработки стало следующее устройство:



# ВЫБОР ПРОГРАММНЫХ СРЕДСТВ ДЛЯ РАЗРАБОТКИ ПРОЕКТА

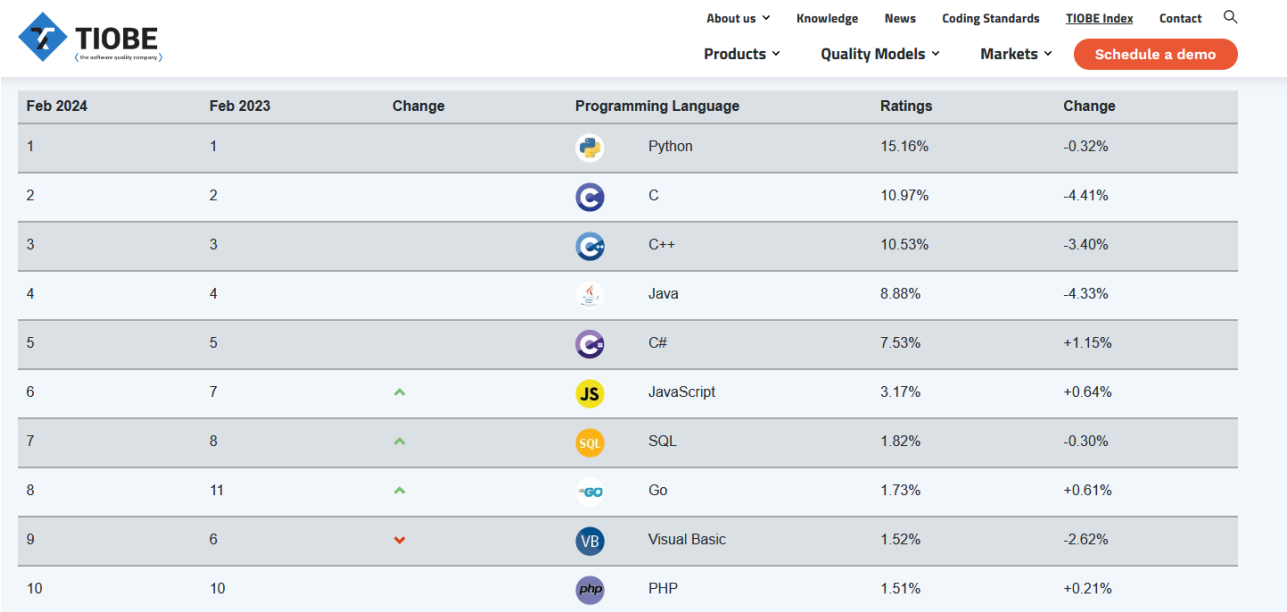
На основании анализа различных языков программирования, на которых возможно создавать программное обеспечение для микрокомпьютеров, был выбран язык Python.

Python — интерпретируемый язык программирования высокого уровня. В проекте помимо базовых средств языка для создания проекта были использованы библиотеки OPi.GPIO и sqlite3, а также микрофреймворк Flask.

Основными достоинствами данного языка являются:

- Популярность

Python остается одним из самых популярных языков программирования уже многие годы, согласно статистике TIOBE [1].



TIOBE

About us | Knowledge | News | Coding Standards | TIOBE Index | Contact

Products | Quality Models | Markets | Schedule a demo

| Feb 2024 | Feb 2023 | Change | Programming Language | Ratings | Change |
|---|---|---|---|---|---|
| 1 | 1 | | Python | 15.16% | -0.32% |
| 2 | 2 | | C | 10.97% | -4.41% |
| 3 | 3 | | C++ | 10.53% | -3.40% |
| 4 | 4 | | Java | 8.88% | -4.33% |
| 5 | 5 | | C# | 7.53% | +1.15% |
| 6 | 7 | ↑ | JavaScript | 3.17% | +0.64% |
| 7 | 8 | ↑ | SQL | 1.82% | -0.30% |
| 8 | 11 | ↑ | Go | 1.73% | +0.61% |
| 9 | 6 | ↓ | Visual Basic | 1.52% | -2.62% |
| 10 | 10 | | PHP | 1.51% | +0.21% |

- Кроссплатформенность

Благодаря интерпретируемости языка он поддерживается практически на всех платформах, что позволяет использовать программы почти на любом

устройстве не зависимо от операционной системы без необходимости внесения каких-либо изменений.

- Простота визуального восприятия

Python обладает простым и понятным синтаксисом, а блоки кода выделяются отступами. Это значительно упрощает восприятие кода.

Для работы с данными была использована библиотека sqlite3.

SQLite — встраиваемая СУБД на языке C.  Основными её достоинствами являются:

- Высокая скорость
- Надежность
- Кроссплатформенность

Для реализации проекта была создана база данных, структура которой имеет следующий вид:

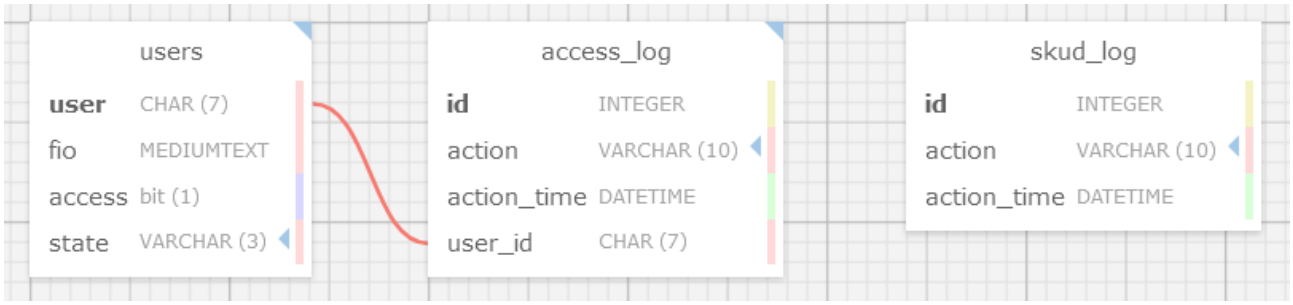



Рис. Структура БД

Для того, чтобы оценить какое количество пользователей и событий может вместить мой контроллер был написан проверочный скрипт для заполнения БД тестовыми данными. После выполнения производилась оценка изменения объема занятого дискового пространства.

Результаты получились следующие:

- 500 000 пользователей — 41 Мбайт

- 1 000 000 событий — 26 Мбайт
- 5 000 000 событий – 135 МБайт

Программное обеспечение контроллера состоит из трех основных частей: модуль управления проходной, модуль обновления и модуля администрирования. Рассмотрим их более подробно.

## Модуль управления проходной

Данный модуль выполняет следующие функции:

- первичная инициализация базы данных
- считывание карт
- добавление новых пользователей
- ведение лога прохода
- ведение системного лога
- управление турникетом

Основным файлом этого модуля является skud.py.

После его запуска происходит создание и первичная инициализация базы данных (если это необходимо). В системный лог добавляется запись об успешном запуске системы. Контроллер переходит в режим ожидания.

Когда к турникету прикладывают карту, происходит выход из режима ожидания. Контроллер считывает её идентификатор, после чего делает запрос в базу данных. Если пользователь найден, проверятся имеет ли он право на вход. В соответствии с результатом программа отправляет команду реле, которое управляет разрешением на проход в турникете. В лог прохода заносится информация о пользователе и времени события. При отсутствии информации о приложенной карте в базе данных доступ не предоставляется и в лог прохода вносится запись о попытке прохода неизвестного посетителя.

При необходимости при помощи пульта управления контроллер можно перевести в режим записи карт. Для этого необходимо нажать кнопку. Когда на

пульте загорится красный светодиод, необходимо приложить карту к считывателю. Контроллер осуществит запрос идентификатора карты, который будет добавлен в базу данных пользователей в качестве гостевой учетной записи. На пульте загорится зеленый светодиод, подтверждающий успешность операции. Также событие записывается в системный лог.

Для выхода из режима записи необходимо повторно нажать кнопку на пульте управления. Красный светодиод погаснет.

Для обеспечения корректной работы турникета было создано программное обеспечение с помощью библиотеки OPi.GPIO. Программа считывает id карты, проверяет наличие пользователя в базе данных, после чего оправляет команду реле.

Блок-схема управления проходной выглядит следующим образом:

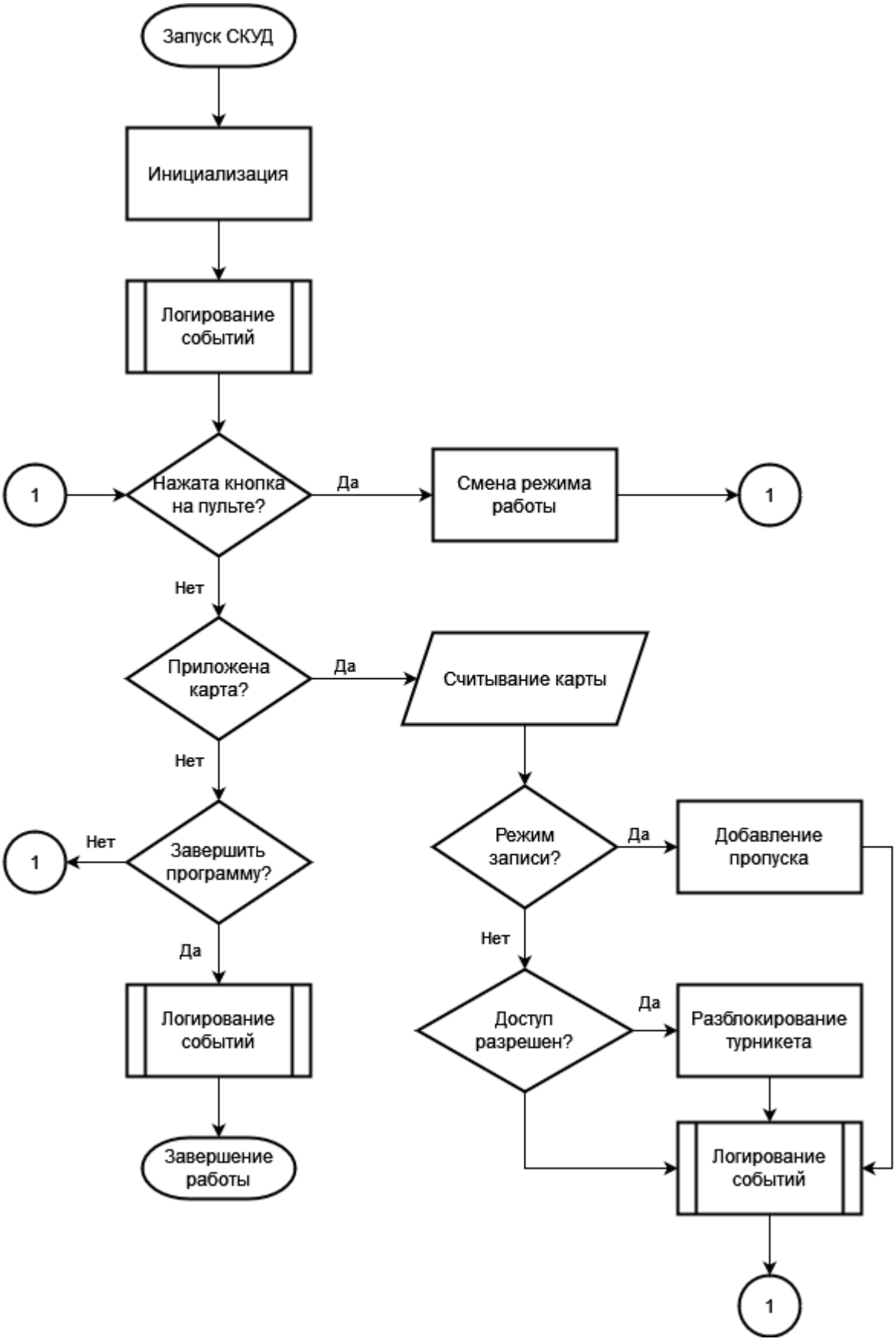



Рис. Блок-схема модуля управления проходной

# Модуль обновления

Для автоматизированного добавления, изменения и удаления информации о пользователях и параметров доступа предназначен update.py. Сервер отправляет на контроллер файлы update_users.json и update_users.sha1, которые представляют собой файл с описанием необходимых изменений и файл с контрольной суммой соответственно. Перед обновлением базы данных проверяется целостность файлов при помощи подсчета контрольной суммы и в случае, если ошибок не было выявлено, вносятся изменения.

Пример файла update_users.json:

```
{
   "add": [
      ["f3f9fa1aeaffff", "Иванов Иван Иваныч", true, "out"]
   ],
   "update": [
      ["Петров Петр Петрович", true, "out", "5c4e7217ffffff"]
   ],
   "delete": [
      ["4c4e7217ffffff"]
   ]
}
```

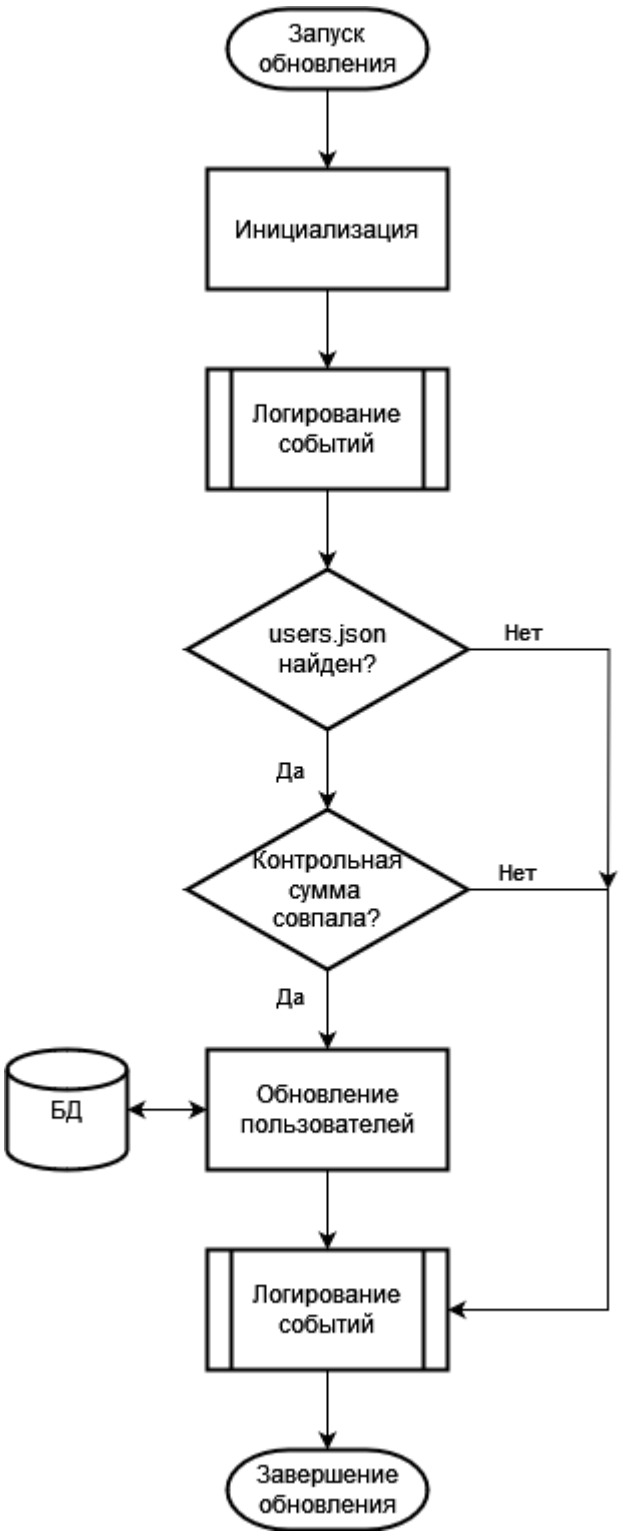
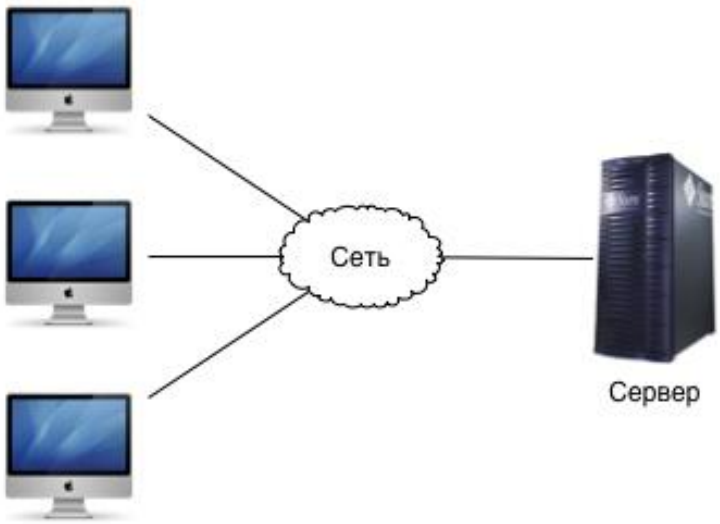



Рис. Блок-схема модуля обновления

## Модуль администрирования

Для взаимодействия с базой данных было создано web-приложение с использованием библиотеки Flask. Flask — фреймворк для создания web-приложений. Среди прочих он выделяется небольшим объемом. Его

использование позволило сэкономить память контроллера. В роли сервера выступает сам микрокомпьютер. Пользователь может вносить изменения в базу данных пользователей, а также просматривать лог СКУД, лог прохода и список пользователей.

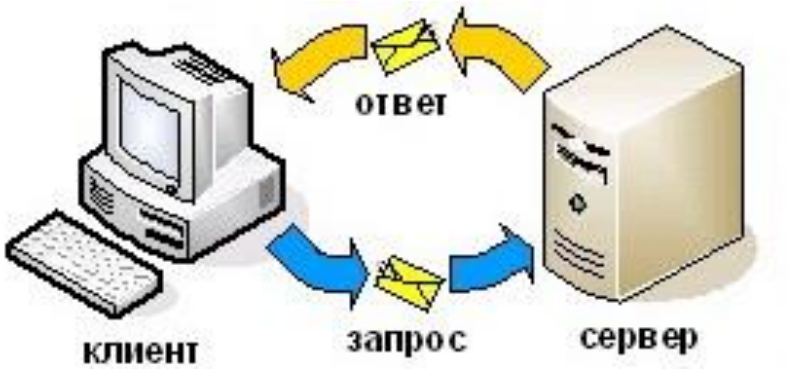



В основе написания web-приложения на Python лежит объектный подход, основанный на понятиях объекта, в котором сочетаются как свойства, сгруппированные данные, так и методы их обработки (подпрограммы).

Web-приложение состоит из клиентской и серверной частей, тем самым реализуя технологию «клиент-сервер».

Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него.

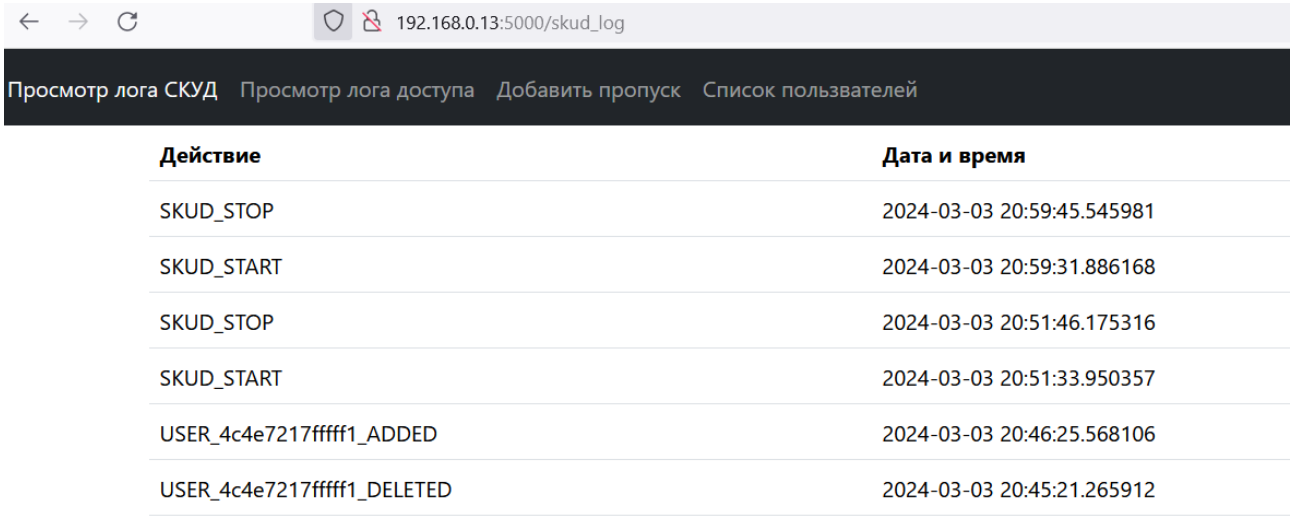
Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует Web-страницу и отправляет её клиенту по сети с использованием протокола HTTP.




**Просмотр лога СКУД.**

Для просмотра журнала СКУД необходимо выбрать пункт «Просмотр лога СКУД» в меню. На экране отобразится таблица с идентификатором события и его датой. Возможные варианты действий:

- SKUD_START – контроллер запущен
- SKUD_STOP – контроллер остановлен
- USER_<id>_ADDED – пользователь добавлен
- USER_<id>_DELETED – пользователь удален
- USER_<id>_UPDATED – информация о пользователе обновлена
- STARTING_UPDATE – запущено обновление БД пользователей
- UPDATED – обновление успешно завершено
- UPDATE_FAILED_CHECKSUM_ERROR – проверка контрольной суммы файла update_users.json выявила ошибку
- UPDATE_FAILED_FILE_NOT_FOUND – файл обновления не найден




| Действие | Дата и время |
|---|---|
| SKUD_STOP | 2024-03-03 20:59:45.545981 |
| SKUD_START | 2024-03-03 20:59:31.886168 |
| SKUD_STOP | 2024-03-03 20:51:46.175316 |
| SKUD_START | 2024-03-03 20:51:33.950357 |
| USER_4c4e7217fffff1_ADDED | 2024-03-03 20:46:25.568106 |
| USER_4c4e7217fffff1_DELETED | 2024-03-03 20:45:21.265912 |

Рис. Лог СКУД

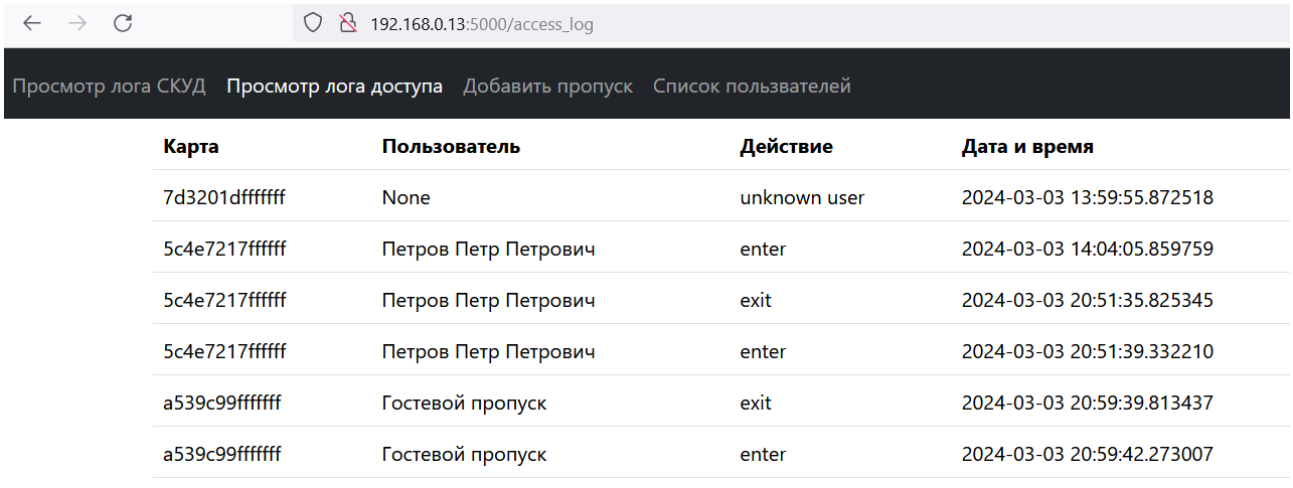
**Просмотр лога доступа.**

Для просмотра журнала доступа необходимо выбрать пункт «Просмотр лога доступа» в меню. На экране отобразится таблица со следующими элементами:

- Карта — идентификатор пропуска
- Пользователь — ФИО пользователя (при наличии)
- Действие:

1. enter – вход
2. exit – выход
3. unknown user – карта пользователя не найдена в БД

- Дата и время




| Карта | Пользователь | Действие | Дата и время |
|---|---|---|---|
| 7d3201dfffffff | None | unknown user | 2024-03-03 13:59:55.872518 |
| 5c4e7217ffffff | Петров Петр Петрович | enter | 2024-03-03 14:04:05.859759 |
| 5c4e7217ffffff | Петров Петр Петрович | exit | 2024-03-03 20:51:35.825345 |
| 5c4e7217ffffff | Петров Петр Петрович | enter | 2024-03-03 20:51:39.332210 |
| a539c99fffffff | Гостевой пропуск | exit | 2024-03-03 20:59:39.813437 |
| a539c99fffffff | Гостевой пропуск | enter | 2024-03-03 20:59:42.273007 |

Рис. Лог доступа

## Добавление пользователя

Для добавления пользователя необходимо выбрать пункт «Добавить пропуск» в меню. На экране отобразится форма со следующими элементами:

- Фамилия Имя Отчество
- id карты

Все поля являются обязательными для заполнения.

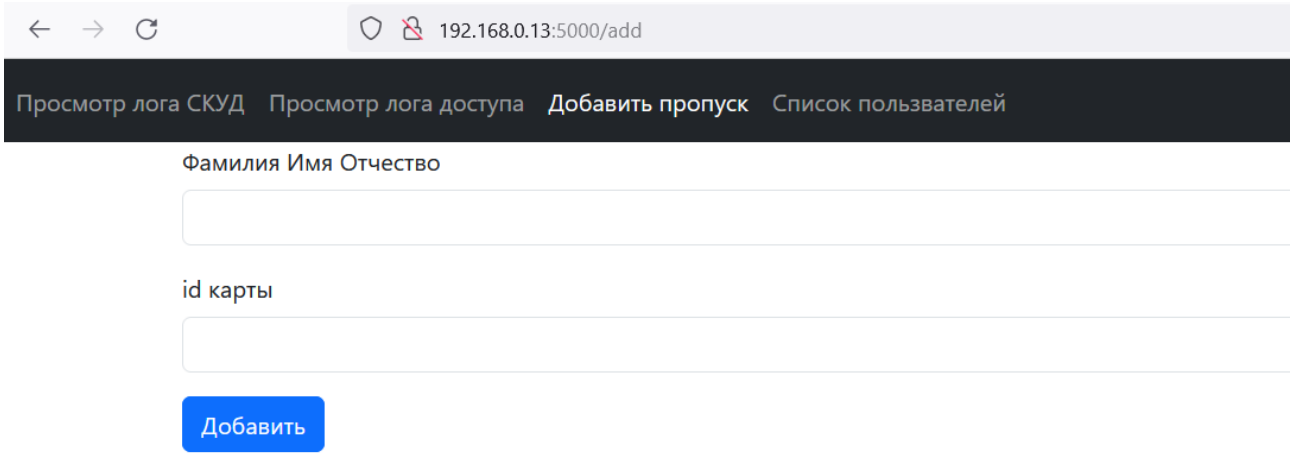



Рис. Добавление пропуска

## Просмотр списка пользователей.

Для просмотра списка пользователей необходимо выбрать пункт «Список пользователей» в меню. На экране отобразится таблица со следующими элементами:

- Карта — идентификатор пропуска
- Пользователь — ФИО пользователя
- Доступ — флаг разрешения доступа
- Состояние — местонахождение посетителя (in – посетитель вошел через проходную, out – посетитель вышел через проходную)
- Кнопка «Удалить»
- Кнопка «Изменить»

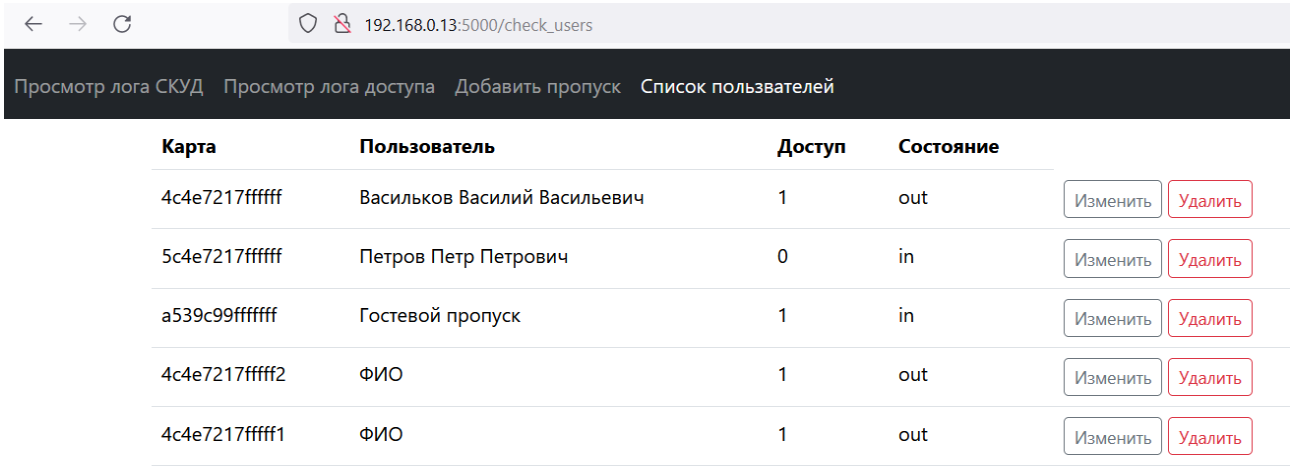



Рис. Список пользователей

**Удаление пользователя.**

При нажатии на кнопку удаления приложение запрашивает подтверждение при помощи диалогового окна (см. рис.).

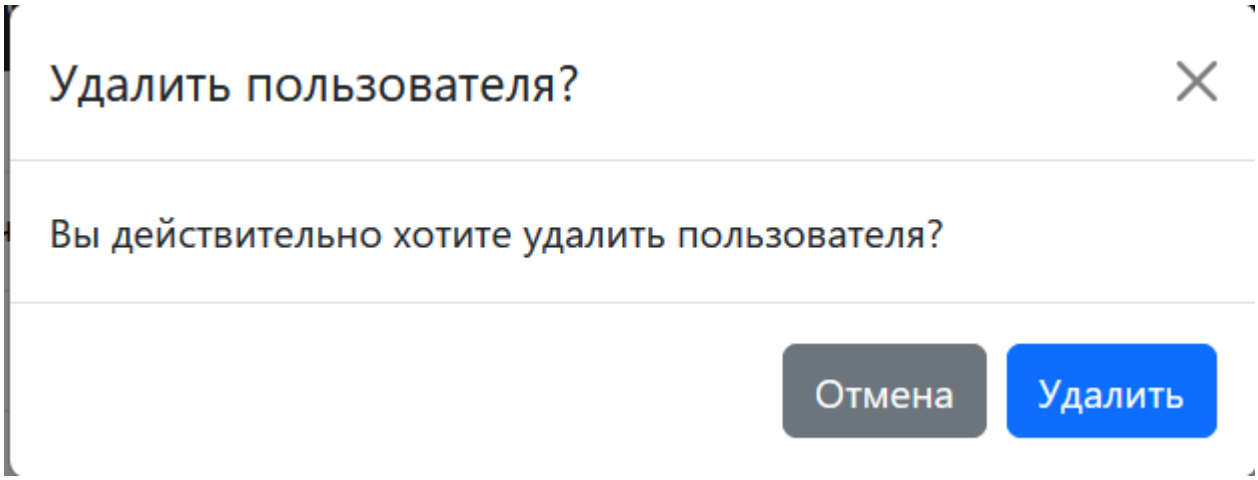



Рис. Удаление пользователя

**Редактирование информации о пользователе.**

Для изменения информации о пользователе в базе данных необходимо нажать на кнопку «Изменить». Откроется окно с формой редактирования данных пользователя (см. рис.). После внесения изменений необходимо нажать на кнопку «Сохранить». Программа внесет изменения в БД и откроет список пользователей.

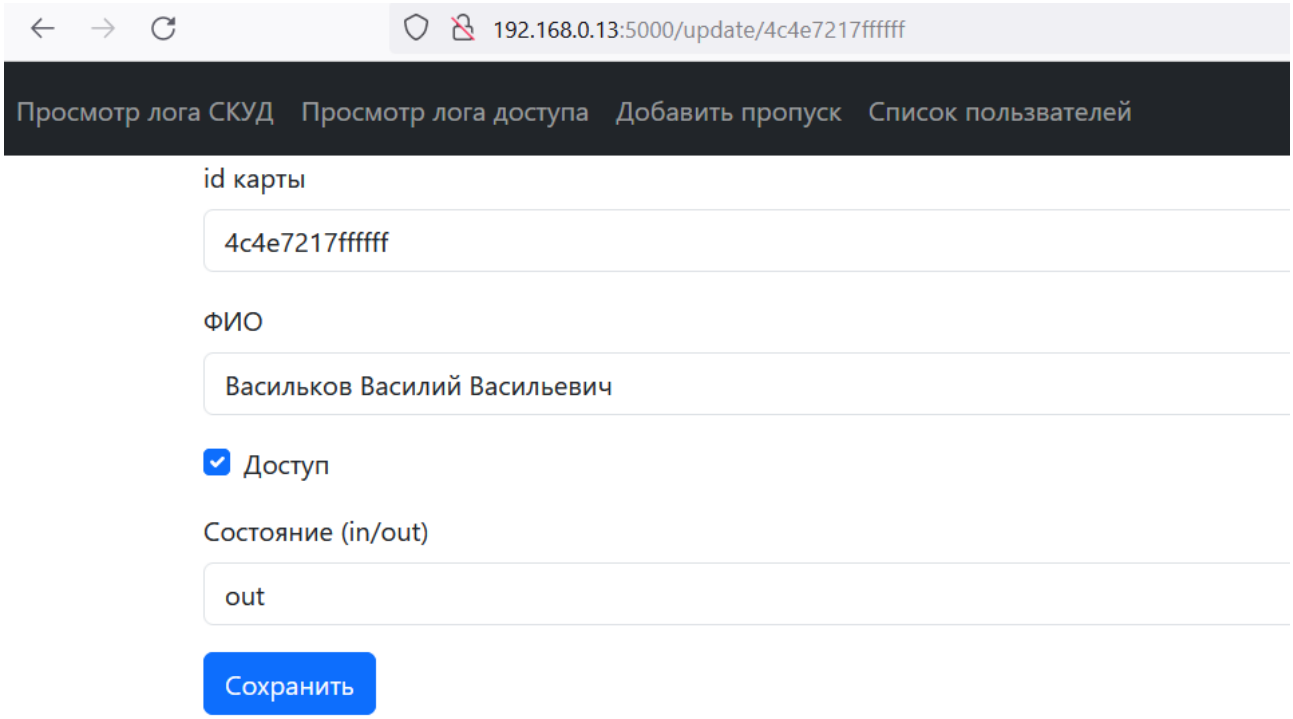



Рис. Редактирование пользователя

# ВЫВОДЫ

В результате выполненной работы был спроектирован программно-технический комплекс управления проходной. Разработаны контроллер на основе одноплатного компьютера, к которому подключаются считыватели и турникеты, и программа управления и администрирования комплекса на языке Python.

Были проведены испытания с подключением к исполнительному устройству, которые показали его работоспособность при подключении к реальному оборудованию.

Разработанный web-интерфейс позволяет использовать контроллер в автономном режиме без необходимости подключения к внешнему серверу СКУД, что является плюсом для предприятий с небольшим числом работников, а также при начальной установке системы, когда отсутствует доступ к сети.

Модульная конструкция устройства и выбор языка программирования Python позволят в дальнейшем модифицировать как программную, так и аппаратную части комплекса с минимальными трудозатратами.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. https://www.tiobe.com/tiobe-index/
2. Гринберг, М. Разработка веб-приложений с использованием Flask на языке Python / М. Гринберг. — Москва: ДМК Пресс, 2014. — 272 с. — ISBN 978-5-97060-138-9.
3. Лутц, Марк. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с.: ил. — Парад, тит. англ. ISBN 978-5-907144-52-1 (рус., том 1).
4. https://descubriendolaorangepi.wordpress.com/2017/04/09/gpio-en-python-conectando-un-lector-rfid-nfc-por-spi/

# ПРИЛОЖЕНИЕ. ЛИСТИНГИ ПРОГРАММ

## skud/skud/skud.py

```
#!/usr/bin/python3
# -*- coding: utf8 -*-

from datetime import datetime

import MFRC522
import signal
from init_db import init_db, con, cur
import orangepi.pcplus
from OPi import GPIO
from time import sleep

continue_reading = True
BTN = 3
GRN = 5
RED = 7
IN4 = 8
IN3 = 10
IN2 = 12
IN1 = 16
reading_mode = False
relay_in = [IN1, IN2, IN3, IN4]
CHANGE_STATE = {'in': 'out', 'out': 'in'}
ACTION = {'in': 'exit', 'out': 'enter'}


def button_event(channel):
    GPIO.output(RED, 0)
    GPIO.output(GRN, 0)
    global reading_mode
    reading_mode = not reading_mode


def gpio_init():
    GPIO.setmode(orangepi.pcplus.BOARD)
    GPIO.setup(BTN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    GPIO.setup(RED, GPIO.OUT, initial=GPIO.LOW)
    GPIO.setup(GRN, GPIO.OUT, initial=GPIO.LOW)
    GPIO.add_event_detect(BTN, GPIO.FALLING, button_event, bouncetime=200)
    GPIO.setup(relay_in, GPIO.OUT, initial=GPIO.HIGH)
    print("GPIO init success!")


def end_read(signal, frame):
    skud_log('SKUD_STOP', datetime.now())
    global continue_reading
    print("Ctrl+C captured, ending read.")
```

```
    continue_reading = False
    GPIO.cleanup()


def skud_log(*values):
    cur.execute("INSERT INTO skud_log (action, action_time) VALUES (?, ?);", val-
ues)
    con.commit()


def access_log(*values):
    cur.execute("INSERT INTO access_log (action, action_time, user_id) VALUES (?,
?, ?);", values)
    con.commit()


if __name__ == '__main__':

    init_db()
    gpio_init()

    skud_log('SKUD_START', datetime.now())

    signal.signal(signal.SIGINT, end_read)

    MIFAREReader = MFRC522.MFRC522()

    print("Welcome to the MFRC522 data read example")
    print("Press Ctrl-C to stop.")

    while continue_reading:
        if reading_mode:
            GPIO.output(RED, GPIO.HIGH)
            GPIO.output(GRN, GPIO.LOW)

        (status, TagType) = MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

        if status == MIFAREReader.MI_OK:
            print("Card detected")

        (status,uid) = MIFAREReader.MFRC522_Anticoll()


        if status == MIFAREReader.MI_OK:

            print("Card read UID: %s,%s,%s,%s" % (uid[0], uid[1], uid[2], uid[3]))
            str_uid = ''
            if uid[0] != 0x88:
                for i in map(hex, uid[:4]):
                    str_uid += i[2:]
                str_uid = str_uid + 'f' * (14 - len(str_uid))
```

```
            else:
                for i in map(hex, uid[1:4]):
                    str_uid += i[2:]
                (status,uid) = MIFAREReader.MFRC522_Anticoll2()
                for i in map(hex, uid):
                    str_uid += i[2:]

            print(str_uid)
            search_result = cur.execute("SELECT * FROM users WHERE user = ?;",
(str_uid, )).fetchone()



            if reading_mode:
                GPIO.output(RED, GPIO.HIGH)
                GPIO.output(GRN, GPIO.LOW)
                if search_result:
                    continue
                print(str_uid)

                cur.execute("INSERT INTO users VALUES (?, ?, ?, ?);", (str_uid,
'Гостевой пропуск', True, 'out'))
                cur.execute("INSERT INTO skud_log (action, action_time) VALUES
('GUEST_CARD_ADDED', ?);", (datetime.now(),))
                con.commit()
                print("uid добавлен")

                GPIO.output(GRN, GPIO.HIGH)
                GPIO.output(RED, GPIO.LOW)
                sleep(1)

            else:
                if search_result:
                    user, fio, access, state = search_result
                    if access:
                        cur.execute("""UPDATE users SET state = ? WHERE user =
?;""", (CHANGE_STATE[state], user))
                        cur.execute("""INSERT INTO access_log (action, action_time,
user_id) VALUES (?, ?, ?);""",
                                    (ACTION[state], datetime.now(), user))
                        print(search_result)
                        GPIO.output(IN4, GPIO.LOW)
                        print('ВХОДИТЕ')
                        GPIO.output(GRN, GPIO.HIGH)
                        sleep(1)
                        GPIO.output(GRN, GPIO.LOW)
                else:
                    cur.execute("""INSERT INTO access_log (action, action_time,
user_id) VALUES (?, ?, ?);""",
                                ('unknown user', datetime.now(), str_uid))
                    print('ДОСТУП ЗАПРЕЩЕН!')
```

```
                    GPIO.output(RED, GPIO.HIGH)
                    sleep(1)
                    GPIO.output(RED, GPIO.LOW)
                con.commit()
                GPIO.output(IN4, GPIO.HIGH)
```

## skud/web_server/web_server.py

```
import os
from datetime import datetime

import sqlite3
from flask_bootstrap import Bootstrap5
from flask import Flask, render_template, redirect

from forms.AddForm import AddForm
from forms.UpdateForm import UpdateForm
from forms.DeleteForm import DeleteForm


app = Flask(__name__)
bootstrap = Bootstrap5(app)
app.config['SECRET_KEY'] = 'secret_key'  # TODO: secret key
con = sqlite3.connect('../users_db.sqlite', check_same_thread=False)
cur = con.cursor()


def write_skud_log(event):
    cur.execute("INSERT INTO skud_log (action, action_time) VALUES (?, ?);",
(event, datetime.now()))
    con.commit()


@app.route('/')
def main():
    return render_template('main_page.html')


@app.route('/skud_log')
def skud_log():
    search_result = cur.execute("SELECT action, action_time FROM skud_log ORDER BY
id DESC;").fetchall()
    return render_template('skud_log.html', data=search_result)


@app.route('/access_log')
def access_log():
    search_result = cur.execute("SELECT user_id, fio, action, action_time FROM ac-
cess_log LEFT JOIN users ON (user = user_id) ORDER BY id DESC;").fetchall()
```

```
    return render_template('access_log.html', data=search_result)


@app.route('/check_users')
def check_users():
    form = AddForm()
    search_result = cur.execute("SELECT * FROM users;").fetchall()
    return render_template('user_log.html', data=search_result)


@app.route('/update/<string:id>', methods=['GET', 'POST'])
def update_user(id):
    user = cur.execute("SELECT * FROM users WHERE user = ?;", (id,)).fetchone()
    form = UpdateForm()
    if form.validate_on_submit():
        user_id = form.id.data
        if form.fio.data:
            cur.execute("UPDATE users SET fio = ? WHERE user = ?;", (form.fio.data,
user_id))
        cur.execute("UPDATE users SET access = ? WHERE user = ?;", (form.ac-
cess.data, user_id))
        if form.state.data:
            cur.execute("UPDATE users SET state = ? WHERE user = ?;",
(form.state.data, user_id))
        write_skud_log('USER_' + user_id + '_UPDATED')
        con.commit()
        return redirect('/check_users')
    form.id.data = user[0]
    form.fio.data = user[1]
    form.access.data = user[2]
    form.state.data = user[3]
    return render_template('update.html', form=form)


@app.route('/delete', methods=['GET', 'POST'])
def delete():
    form = DeleteForm()
    if form.validate_on_submit():
        cur.execute("DELETE FROM users WHERE user = ?;", (user_id,))
        con.commit()
        write_skud_log()
        return redirect('/check_users')
    return render_template('delete.html', form=form)


@app.route('/del/<string:id>')
def delete_user(id):
    cur.execute("DELETE FROM users WHERE user = ?;", (id,))
    con.commit()
    write_skud_log('USER_' + id + '_DELETED')
    return 'DELETED'
```

```
@app.route('/add', methods=['GET', 'POST'])
def add():
    form = AddForm()
    if form.validate_on_submit():
        form_id = form.id.data
        str_id = form_id + 'f' * (14 - len(form_id))
        fio = form.fio.data
        cur.execute("INSERT INTO users VALUES (?, ?, 1, 'out');", (str_id, fio))
        con.commit()
        write_skud_log('USER_' + form_id + '_ADDED')
        return redirect('/check_users')
    return render_template('add.html', form=form)


def run():
    port = int(os.environ.get("PORT", 5000))
    app.run(host='0.0.0.0', port=port)


def test():
    port = int(os.environ.get("PORT", 5000))
    app.run(host='127.0.0.1', port=port)


if __name__ == '__main__':
    run()
```

## skud/update/update.py

```
from datetime import datetime
import json
import os
import hashlib

import sqlite3


con = sqlite3.connect("users_db.sqlite")
cur = con.cursor()


def update():
    cur.execute("INSERT INTO skud_log (action, action_time) VALUES ('STARTING_UP-
DATE', ?);", (datetime.now(), ))
    try:
        with open('server/update_users.json', encoding='utf-8') as f:
            data = json.load(f)
        with open('server/update_users.sha1') as f:
            checksum = f.readline()
```

```
        with open('server/update_users.json', 'rb') as f:
            my_checksum = hashlib.sha1(f.read()).hexdigest()
    except FileNotFoundError:
        cur.execute("INSERT INTO skud_log (action, action_time) VALUES ('UP-
DATE_FAILED_FILE_NOT_FOUND', ?);",
                    (datetime.now(),))
    if checksum == my_checksum:
        cur.executemany("DELETE FROM users WHERE user = ?;", data['delete'])
        cur.executemany("INSERT INTO users VALUES (?, ?, ?, ?);", data["add"])
        cur.executemany("UPDATE users SET fio = ?, access = ?, state = ? WHERE user
= ?;", data["update"])
        cur.execute("INSERT INTO skud_log (action, action_time) VALUES ('UPDATED',
?);", (datetime.now(),))
    else:
        cur.execute("INSERT INTO skud_log (action, action_time) VALUES ('UP-
DATE_FAILED_CHECKSUM_ERROR', ?);",
                    (datetime.now(), ))
        os.remove('server/update_users.json')
        os.remove('server/update_users.sha1')
    con.commit()


if __name__ == '__main__':
    update()
```