Министерство образования Республики Беларусь

Учреждение образования

ГРОДНЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ ЯНКИ КУПАЛЫ

Физико-технический факультет

Кафедра информационных систем и технологий

К защите допустить*:*

Заведующий кафедрой ИСиТ

\_\_\_\_\_\_\_\_\_\_\_\_\_Ю.Р.Бейтюк

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

на тему

**СИСТЕМА АУТЕНТИФИКАЦИИ И АВТОРИЗАЦИИ ДОСТУПА К ИНФОРМАЦИОННЫМ РЕСУРСАМ НА ОСНОВЕ РАСПОЗНАВАНИЯ ЛИЦ**

ГРГУ ДП 1-38 02 01 08 006 ПЗ

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.М. Дробыш

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Белко

Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гродно 2019

# Содержание

[Введение 7](#_Toc7266056)

[1 Краткое описание технологий проекта 9](#_Toc7266057)

[1.1 Язык программирования Python 9](#_Toc7266058)

[1.2 Веб-фреймворк Flask 10](#_Toc7266059)

[1.3 Docker 11](#_Toc7266060)

[1.4 Dlib 13](#_Toc7266061)

[1.5 Язык программирования JavaScript 13](#_Toc7266062)

[1.6 Язык программирования TypeScript 14](#_Toc7266063)

[1.7 Node.js 14](#_Toc7266064)

[1.8 Express.js 15](#_Toc7266065)

[1.9 Babel 15](#_Toc7266066)

[1.10 MongoDB 16](#_Toc7266067)

[1.11 Webpack 16](#_Toc7266068)

[1.12 React 16](#_Toc7266069)

[2 Практическая реализация 18](#_Toc7266070)

[2.1 Общая структура 18](#_Toc7266071)

[2.2 Сервис распознавания лиц 19](#_Toc7266072)

[2.2.1 Конфигурация 19](#_Toc7266073)

[2.2.2 Функционал 20](#_Toc7266074)

[2.2.3 Распознавание лица 22](#_Toc7266075)

[2.2.4 Результаты тестирования приложения 25](#_Toc7266076)

[2.3 Сервер приложения 26](#_Toc7266077)

[2.3.1 Конфигурация 26](#_Toc7266078)

[2.3.2 Функционал 26](#_Toc7266079)

[2.4 База данных 29](#_Toc7266080)

[2.5 Веб-клиент 30](#_Toc7266081)

[2.5.1 Конфигурация 30](#_Toc7266082)

[2.5.2 Функционал 30](#_Toc7266083)

[2.5.3 Результаты тестирования приложения 34](#_Toc7266084)

[2.6 Оценка точности распознавания лица 35](#_Toc7266085)

[3 Техника безопасности при работе за персональным компьютером 39](#_Toc7266086)

[3.1 Требования по электрической безопасности 39](#_Toc7266087)

[3.2 Особенности электропитания системного блока 40](#_Toc7266088)

[3.3 Система гигиенических требований 41](#_Toc7266089)

[3.4 Требования к рабочему месту 42](#_Toc7266090)

[3.5 Требования к организации пространства 44](#_Toc7266091)

[4 Энергосбережение 45](#_Toc7266092)

[4.1 Основные способы экономии энергии 45](#_Toc7266093)

[4.2 Энергосберегающие режимы работы компьютера 46](#_Toc7266094)

[4.3 Энергоэффективность в серверных помещениях 47](#_Toc7266095)

[4.4 Стандарты энергосберегающих технологий 47](#_Toc7266096)

[4.4.1 Стандарт усовершенствованной системы управления питанием (Advanced Power Management — APM) 48](#_Toc7266097)

[4.4.2 Стандарт усовершенствованной системы управления питанием (Advanced Power Management—APM) 50](#_Toc7266098)

[4.4.3 Стандарт DPMS (Display Power Management Signaling - система сигналов управления питанием монитора) 50](#_Toc7266099)

[4.5 Энергосберегающие технологии в мобильных ПК 51](#_Toc7266100)

[4.5.1 Технология Enhanced Intel SpeedStep 51](#_Toc7266101)

[4.5.2 Режимы Deep Sleep и Deeper Sleep 54](#_Toc7266102)

[4.5.3 Технология Intel Mobile Voltage Positioning (IMVP) 55](#_Toc7266103)

[4.5.4 Чипсеты для мобильных процессоров 56](#_Toc7266104)

[4.5.5 Технология Centrino 57](#_Toc7266105)

[5 Экономическое обоснование 59](#_Toc7266106)

[5.1 Оптимальные характеристики оборудования 59](#_Toc7266107)

[5.2 Расчет себестоимости программного обеспечения 60](#_Toc7266108)

[Заключение 68](#_Toc7266109)

[Список использованных источников 69](#_Toc7266110)

[Приложение 72](#_Toc7266111)

[Исходный код сервиса по распознаванию лиц 72](#_Toc7266112)

[Исходный код сервера приложения 76](#_Toc7266113)

[Исходный код веб-клиента 82](#_Toc7266114)

# 

# Реферат

Дипломный проект «Система аутентификации и авторизации доступа к информационным ресурсам на основе распознавания лиц» содержит 99 страниц, 11 изображений, 10 таблиц, 17 источников литературы.

ЦЕЛЬ РАБОТЫ: Создание максимально кроссплатформенного и с адаптивными, заменяемыми программными модулями комплекса ПО распознавания лица на изображении, поступающем с веб-камеры компьютера и авторизации на основе результата распознавания, а также возможностью расширения функционала с минимальными изменениями основных компонентов проекта.

МЕТОДЫ ИССЛЕДОВАНИЯ: Для реализации проекта использовались операционная система Ubuntu 16.04, текстовый редактор Visual Studio Code, языки программирования Python, TypeScript и JavaScript.

ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ:

* Docker-изображение с веб-сервисом, распознающим лицо на входном изображении
* Веб-сервер, сохраняющий результаты распознавания в базу данных
* Браузерный клиент

КЛЮЧЕВЫЕ СЛОВА: Docker, Python, TypeScript, JavaScript, Node.js, MongoDB, ReactJS, Dlib, Machine learning, Facial recognition.

ОБЛАСТЬ ПРИМЕНЕНИЯ: Проект может быть использован на любом предприятии, где есть необходимость ограничить доступ к информационным ресурсам.

# Essay

The diploma project «The system for information resources access authentication and authorization based on facial recognition» contains 99 pages, 11 figures, 10 tables, 17 literature sources.

PURPOSE OF WORK: Creation of the most cross-platform and with adaptive, replaceable software modules software complex for facial recognition on an image, provided by the computer web camera and authorization based on the recognition result, also with the possibility of extending the functionality with minimal changes to the main components of the project.

RESEARCH METHODS: Operating system Ubuntu 16.04, text editor Visual Studio Code, Python, TypeScript and JavaScript programming languages were used to implement the project.

RESULTS:

* Docker-image with a web-service which recognizes a face in an input image
* Web-server which recognizes recognition results into the database
* Browser client

KEYWORDS: Docker, Python, TypeScript, JavaScript, Node.js, MongoDB, ReactJS, Dlib, Machine learning, Facial recognition.

SCOPE: The project can be used in any enterprise where there is a need to restrict an access to information resources.

# Введение

Сегодня в мире наблюдается растущий интерес к проблеме машинного распознавания лиц. Системы распознавания лиц внедряются в современные смартфоны, берутся на вооружение властями некоторых государств для поимки преступников. Так, к примеру, применив технологию распознавания лиц на фестивале пива в китайском Циндао, полиция арестовала 25 преступников в розыске. Среди задержанных был преступник, который скрывался от правосудия в течение десяти лет. Также алгоритмы распознавания лиц используются в популярной сети Facebook - добавив совместное фото с друзьями, можно видеть, как социальная сеть самостоятельно предлагает пользователям отметить на снимке конкретных людей.

Идентификация человека по его фотографии имеет следующий ряд преимущества по сравнению с другими методами распознавания:

1. нет необходимости в специальной технике
2. не нужен физический контакт с какими-либо устройствами, достаточно лишь пройти перед камерой

Помимо достоинств компьютерное распознавание лиц имеет и свои недостатки

1. сам по себе один лишь способ не гарантирует стопроцентной идентификации, там где такая точность необходима стоит использовать несколько методов распознавания в их комбинации
2. на точность распознавания могут повлиять различные травмы лица, шрамы
3. при различном повороте головы и освещении лицо выглядит по разному

В настоящее время используются десятки методов распознавания лиц и вот некоторые из них:

* использование нейронных сетей
* метод гибкого сравнения на графах (elastic graph matching)
* скрытые Марковские модели (СММ, HMM)
* метод главных компонентов (principal component analysis, PCA)

Целью данной работы является разработка кроссплатформенного ПО для распознавания лиц с возможностью использования для авторизации пользователя в системе.

# Краткое описание технологий проекта

В проекте использовался комплекс программного обеспечения для создания составного программного продукта. Было необходимо ПО как для разработки сервиса распознавания лиц, веб-сервера и клиентской части. Сервис распознавания лиц и веб-сервер были соединены в один программный продукт. Технологии, связанные с созданием сервиса для распознавания лиц: Python, Flask, Docker, dlib. Технологии, использованные для создания веб-сервера: TypeScript, Node.js, MongoDB, Express.js. Технологии, использованные для создания веб-клиента: JavaScript, Babel, Webpack, React.

## Язык программирования Python

Python (в русском языке распространено название питон) — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное.

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ. Он распространяется под свободной лицензией Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные.

Python — активно развивающийся язык программирования, новые версии (с добавлением/изменением языковых свойств) выходят примерно раз в два с половиной года. Вследствие этого и некоторых других причин на Python отсутствуют стандарт ANSI, ISO или другие официальные стандарты, их роль выполняет CPython.

[Разработка языка Python была начата в конце 1980-х годов сотрудником голландского института CWI Гвидо ван Россумом. Для распределенной ОС Amoeba требовался расширяемый скриптовый язык, и Гвидо начал писать Python на досуге, позаимствовав некоторые наработки для языка ABC (Гвидо участвовал в разработке этого языка, ориентированного на обучение программированию). В феврале 1991 года Гвидо опубликовал исходный текст в группе новостей alt.sources. С самого начала Python проектировался как объектно-ориентированный язык.](https://ru.wikipedia.org/wiki/Python#cite_note-18)

[Наличие дружелюбного, отзывчивого сообщества пользователей считается наряду с дизайнерской интуицией Гвидо одним из факторов успеха Python. Развитие языка происходит согласно четко регламентированному процессу создания, обсуждения, отбора и реализации документов PEP (англ. Python Enhancement Proposal) — предложений по развитию Python](https://ru.wikipedia.org/wiki/Python#cite_note-18) [1].

## Веб-фреймворк Flask

Flask — фреймворк для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2. Относится к категории так называемых микрофреймворков — минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности.

Поддерживается установка посредством пакетного менеджера PyPI, версия 1.0 совместима с Python 2.7, Python 3.3 и выше.

Создатель и основной автор — австрийский программист Армин Ронахер, начал работу над проектом в 2010 году.

Пример веб-приложения, которое показывает «Hello World!»:

from flask import Flask  
app = Flask(\_\_name\_\_)  
  
@app.route("/")  
def hello():  
 return "Hello World!"  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app.run()

Сообществом поддерживается серия пакетов-расширений для Flask, их название обычно начинается с flask-, например flask-login, flask-sqlalchemy, flask-wtf [2].

## Docker

Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в среде виртуализации на уровне операционной системы. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесён на любую Linux-систему с поддержкой cgroups в ядре, а также предоставляет среду по управлению контейнерами.

Разрабатывается и поддерживается одноимённой компанией-стартапом, распространяется в двух редакциях — общественной (Community Edition) по лицензии Apache 2.0 и для организаций (Enterprise Edition) по проприетарной лицензии. Написан на языке Go.

Программное обеспечение функционирует в среде Linux с ядром, поддерживающим cgroups и изоляцию пространств имён (namespaces); существуют сборки только для платформ x86-64 и ARM. Начиная с версии 1.6 возможно использование в ОС Windows.

В состав программных средств входит демон — сервер контейнеров (запускается командой docker -d), клиентские средства, позволяющие из интерфейса командной строки управлять образами и контейнерами, а также API, позволяющий в стиле REST управлять контейнерами программно.

Демон обеспечивает полную изоляцию запускаемых на узле контейнеров на уровне файловой системы (у каждого контейнера собственная корневая файловая система), на уровне процессов (процессы имеют доступ только к собственной файловой системе контейнера, а ресурсы разделены средствами libcontainer), на уровне сети (каждый контейнер имеет доступ только к привязанному к нему сетевому пространству имён и соответствующим виртуальным сетевым интерфейсам).

Набор клиентских средств позволяет запускать процессы в новых контейнерах (docker run), останавливать и запускать контейнеры (docker stop и docker start), приостанавливать и возобновлять процессы в контейнерах (docker pause и docker unpause). Серия команд позволяет осуществлять мониторинг запущенных процессов (docker ps по аналогии с ps в Unix-системах, docker top по аналогии с top и другие). Новые образы возможно создавать из специального сценарного файла (docker build, файл сценария носит название dockerfile), возможно записать все изменения, сделанные в контейнере в новый образ (docker commit). Все команды могут работать как с docker-демоном локальной системы, так и с любым сервером Docker, доступным по сети. Кроме того, в интерфейсе командной строки встроены возможности по взаимодействию с публичным репозиторием Docker Hub, в котором размещены предварительно собранные образы контейнеров, например, команда docker search позволяет осуществить поиск образов среди размещенных в нем, образы можно скачивать в локальную систему (docker pull), возможно также отправить локально собранные образы в Docker Hub (docker push).

[Также Docker имеет пакетный менеджер Docker Compose, позволяющий описывать и запускать многоконтейнерные приложения. Конфигурационные файлы Compose описываются на языке YAML [3].](https://ru.wikipedia.org/wiki/Docker#cite_note-20)

## [Dlib](https://ru.wikipedia.org/wiki/Docker#cite_note-20)

Dlib — кроссплатформенная библиотека общего назначения с открытым исходным кодом, написанная на языке C++. На ее устройство очень сильно повлияли идеи контрактной архитектуры и компонентно-ориентированной разработки программного обеспечения. Таким образом это, в первую очередь, набор независимых программных компонентов.

Разработка Dlib началась в 2002 и с тех пор в Dlib было включено множество инструментов, таких как работа с сетью, потоками, пользовательским графическим интерфейсом, структурами данных, линейной алгеброй, машинным обучением, обработкой изображений, разбором текста и XML и многими другими задачами. В последнее время фокус разработки сконцентрирован на создании широкого набора инструментов для машинного обучения [4].

## Язык программирования JavaScript

JavaScript (аббр. JS) — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией языка ECMAScript (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

JavaScript является объектно-ориентированным языком, но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам — функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкания — что придаёт языку дополнительную гибкость [5].

## Язык программирования TypeScript

TypeScript — язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript.

TypeScript является обратно совместимым с JavaScript и компилируется в последний. Фактически, после компиляции программу на TypeScript можно выполнять в любом современном браузере или использовать совместно с серверной платформой Node.js.

TypeScript отличается от JavaScript возможностью явного статического назначения типов, поддержкой использования полноценных классов (как в традиционных объектно-ориентированных языках), а также поддержкой подключения модулей, что призвано повысить скорость разработки, облегчить читаемость, рефакторинг и повторное использование кода, помочь осуществлять поиск ошибок на этапе разработки и компиляции, и, возможно, ускорить выполнение программ [6].

## Node.js

Node или Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом [7].

## Express.js

Express.js, или просто Express, фреймворк web-приложений для Node.js, реализованный как свободное и открытое программное обеспечение под лицензией MIT. Он спроектирован для создания веб-приложений и API[2]. Express можно использовать как backend для программного стека MEAN, вместе с базой данных MongoDB и фреймворком AngularJS для frontend'а [8].

## Babel

Babel — это инструмент, помогающий в написании кода на последней версии языка JavaScript. Если целевые окружения не поддерживают исполнение необходимой версии языка нативно, использование Babel дает возможность скомпилировать написанный код в поддерживаемый.

Пример такой компиляции ниже:

Вход

// ES2015 стрелочная функция  
[1, 2, 3].map((n) => n + 1);

Выход

[1, 2, 3].map(function(n) {  
 return n + 1;  
});

Babel является программным обеспечением с открытым исходным кодом и поддерживается в основном сообществом волонтеров [9].

## MongoDB

MongoDB — документоориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных. Написана на языке C++ [10].

## Webpack

Webpack — это опенсорсный сборщик модулей для языка JavaScript. Webpack принимает на вход список модулей с зависимостями и генерирует на выходе набор статических файлов, готовых к встраиванию в веб-страницу. Сборщик может быть использован через коммандную строку, либо сконфигурирован через специальный файл, называемый webpack.config.js [11].

## React

React — это JavaScript библиотека с открытым исходным кодом для построения пользовательских интерфейсов.

React может использоваться при разработке одностраничных веб-приложений и мобильных приложений. React нацелен в первую очередь на обеспечение высокой скорости разработки, простоты и расширяемости.

React был разработан Джорданом Уолком (Jordan Walke), программистом компании Facebook. Впервые React был использован в 2011 году в новостной ленте Facebook и впоследствии на сайте Instagram в 2012. Его исходный код был открыт в мае 2013 года.

Ниже представлен простейший пример использования React (вместе с синтаксисом JSX) в HTML странице:

<**div** id="myReactApp"></**div**>  
  
<**script** type="text/babel">  
 **class** Greeter **extends** React.Component {   
 render() {   
 **return** <h1>{**this**.props.greeting}</h1>  
 }   
 }   
  
 ReactDOM.render(<Greeter greeting="Hello World!" />, document.getElementById('myReactApp'));  
</**script**>

Класс Greeter — это React-компонент, принимающий свойство greeting. Метод ReactDOM.render создает экземпляр компонента Greeter, устанавливает свойству greeting значение ‘Hello World’ и вставляет отрисованный компонент в документ как дочерний элемент DOM-элемента #myReactApp.

Отображаемый в браузере результат [12]:

<**div** id="myReactApp">  
 <**h1**>Hello World!</**h1**>  
</**div**>

# Практическая реализация

## Общая структура

Проект назван FR (face recognition) и разделен на две функционально независимые части - fr-service (сервис, распознающий лица), fr-mediator (сервер приложения с БД, выполняющий авторизацию и хранение пользовательский данных) и fr-client (собственно клиент, которому необходимо авторизоваться).

Принцип работы этой реализации представлен ниже:

1. На ПК, который будет выступать сервером сервиса распознавания , устанавливается необходимое ПО: Docker. Затем в папке с исходным кодом сервиса выполняются команды docker build -t fr . и docker run -v $(pwd):/root/app/ -p 5001:5001 fr. Сервис теперь готов к использованию и слушает входящие соединения на порту 5001.
2. На ПК, который будет выступать сервером приложения устанавливается необходимое ПО: Node.js и MongoDB. Далее в папке с исходным кодом выполняются команды npm run build и npm start. Веб сервер теперь готов к использованию и слушает входящие соединения на порту 8001.
3. К HTTP-серверу выполняют подключение клиенты. В рамках работы реализована возможность подключения клиентов из WEB-браузера. Тем не менее при разработке сервера реализован API для возможности последующего дополнения проекта другими типами клиентов.

Далее будут более подробно рассмотрены уже созданные части проекта.

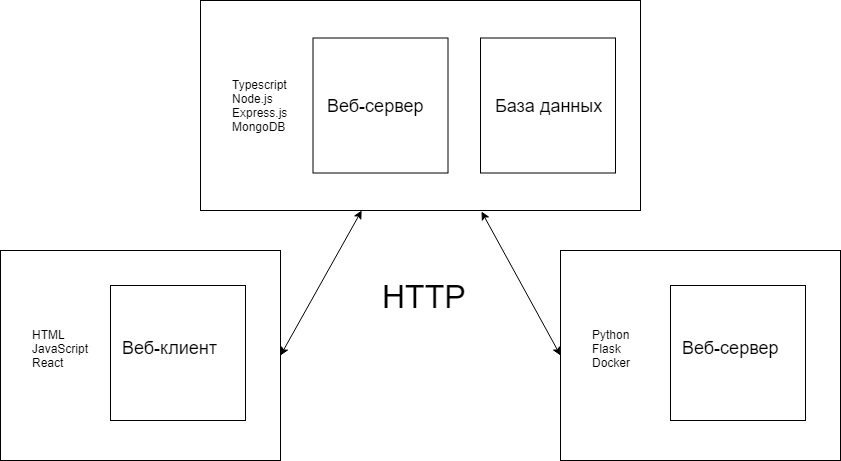


Рисунок Блок схема взаимодействия между различными частями системы

## Сервис распознавания лиц

Понятие сервиса распознавания лиц в контексте проекта включает Docker изображение, с запущенным внутри приложением на языке Python и фреймворке Flask. Docker слушает входящие соединения на порту 5001 и транслирует их запущенному внутри приложению.

Данная сервис является универсальным и может быть переиспользован в любом другом программном комплексе. Все, что для этого необходимо, это развернуть докер-изображение и изспользовать описанное ниже API.

### Конфигурация

Машина (ПК, ноутбук), выступающая в роли сервера сервиса по распознаванию лиц, должна иметь совместимую ОС и достаточно ресурсов для установки последней версии Docker CE. Сервис запускается при помощи следующих комманд:

docker build -t fr/service .

docker run -v $(pwd):/root/app/ -p 5001:5001 fr/service

### Функционал

1. Прослушивание входящих соединений
2. “Кодировка” лица в 128-размерный вектор для последующего использования при входящем POST запросе на путь /encode, содержащем в теле изображения для кодирования.
3. “Узнавание лица” при входящем POST запросе на путь /recognize, содержащем в теле следующие данные:
   * encoding – числовой массив, содержащий код распознаваемого лица
   * storage – массив, элементами которого являются объекты, содержащие информацию об уже известных системе лицах, формат элементов следующий:

{

name: String;

encoding: Array<Number>;

}

* + tolerance – максимальное евклидово расстояние между двумя кодами, при котором они еще считаются соответствующими одному и тому же лицу.

Исходный код вынесен в приложение.

Ниже приведены блок-схема работы веб-сервиса и код методов-обработчика входящих запросов.

@app.route('/encode, methods=['POST'])

def encode():

# Check if a valid image file was uploaded

if 'file' not in request.files:

return no\_file\_uploaded\_response

file = request.files['file']

if file.filename == '':

return no\_file\_uploaded\_response

if file and allowed\_file(file.filename):

# The image file seems valid! Detect faces and return the result.

return detect\_faces\_in\_image(file)

# If no valid image file was uploaded, go to index:

return no\_file\_uploaded\_response

@app.route('/recognize', methods=['POST'])

def recognize():

data = json.loads(request.get\_json())

storage = data['storage']

if not storage:

return ('', 204)

tolerance = data['tolerance']

encoding = data['encoding']

for item in storage:

if (compare\_faces(item['encoding'], encoding, tolerance)):

return item['name']

return ('', 204)

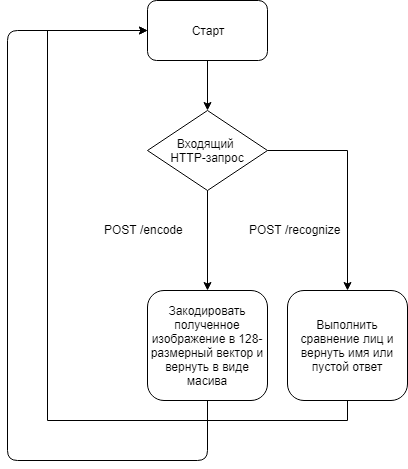


Рисунок Блок-схема работы сервиса распознавания лиц

### Распознавание лица

За распознавание лиц в проекте отвечают библиотека dlib, предоставляющая уже обученную нейронную сеть.

Тем не менее далее этот процесс рассмотрен более подробно. В предыдущей главе сказано, что отправка POST запроса, содержащего изображение в теле, на путь /encode, вызывает метод detect\_faces\_in\_image.

def detect\_faces\_in\_image(file\_stream):

no\_faces\_recognized\_response = 'No faces detected'

# Load the uploaded image file

img = np.array(Image.open(file\_stream).convert('RGB'))

try:

# Get face encodings for any faces in the uploaded image

result = recognize\_face(img)

return result

except IndexError:

return no\_faces\_recognized\_response

Выражение np.array(Image.open(file\_stream).convert('RGB')) конвертирует изображение в массив пикселей, каждый из которых в свою очередь представляет собой массив из трех чисел в промежутке 0-255, каждое из этих чисел - это яркость отдельно взятого пикселя в R, G или B канале.

def recognize\_face(face\_image):

'''Face recognizer function'''

num\_jitters = 1

face\_recognition\_model = face\_recognition\_model\_location()

face\_encoder = dlib.face\_recognition\_model\_v1(face\_recognition\_model)

face\_rect = detect\_face(face\_image)

if not face\_rect:

raise IndexError

landmarks = detect\_landmarks(face\_image, face\_rect)

unknown\_face = np.array(face\_encoder.compute\_face\_descriptor(

face\_image, landmarks, num\_jitters))

return json.dumps(unknown\_face.tolist())

Приведенная выше функция recognize\_face на вход изображения, представляющее собой массив пикселей и возвращает закодированное лицо, которое представляет собой 128-размерный вектор.

Рассмотрим процесс получения такого кода. Это происходит в несколько этапов. Первым шагом выполняется поиск лиц на входном изображении.

face\_recognition\_model = face\_recognition\_model\_location()

face\_encoder = dlib.face\_recognition\_model\_v1(face\_recognition\_model)

face\_detector = dlib.get\_frontal\_face\_detector()

face\_rects = face\_detector(face\_image)

Распознавание выполняется с использованием метода направленных градиентов[13]. В переменной face\_rects теперь хранится массив прямоугольников, каждый из которых обозначает местоположение найденного лица.

Следующим шагом является выявление на каждом из лиц характерных точек, характеризующих положение глаз, бровей, губ, носа и подбородка.

model = face\_recognition\_models.pose\_predictor\_model\_location()

predictor = dlib.shape\_predictor(model)

landmarks = predictor(image, face\_location)

Здесь shape\_predictor – алгоритм, основанный на деревьях решений[14], pose\_predictor\_model\_location – результаты тренировки этого алгоритма на большом объеме данных, выполненные и предоставляемые командой dlib [15]. Landmarks – массив из 68 точек, каждаю из которых имеет координаты x и y.

Полученный массив характерных точек далее кодируется в 128-размерный вектор.

face\_recognition\_model =

face\_recognition\_models.face\_recognition\_model\_location()

face\_encoder = dlib.face\_recognition\_model\_v1(face\_recognition\_model)

face\_encoder.compute\_face\_descriptor(face\_image, landmarks, num\_jitters)



Рисунок Результат обнаружения лица на фотографии



Рисунок Результат обнаружения характерных точек на фотографии

Здесь face\_recognition\_model\_v1 – остаточная нейронная сеть [16] содержащая 29 сверточных слоев, face\_recognition\_model\_location – натренированная на трех миллионах лиц модель [17], параметр num\_jitters – число. Если num\_jitters больше одного, то исходное изображение искажается (num\_jitters - 1) раз, для каждого искажения высчитывается код, после полученный массив из num\_jitters векторов усредняется. Это уточняет полученный результат, но замедляет работу алгоритма. В работе этот параметр равен единице.

Задачу же верификации, то есть принятия решения принадлежат ли два изображения одному и тому же человеку, выполняет следующий код:

def compare\_faces(encoding1, encoding2, tolerance):

if len(encoding1) is not len(encoding2):

raise Exception('Encodings length are not compatible')

distance = 0

length = len(encoding1)

for i in range(length):

distance = distance + (encoding1[i] - encoding2[i]) \*\* 2

return distance < tolerance

Функция compare\_faces попросту рассчитывает евклидово растояние между двумя кодами, и если расстояние оказывается меньше какого-то определенного числа, то два изображения лица считаются принадлежащими одному и тому же человеку.

### Результаты тестирования приложения

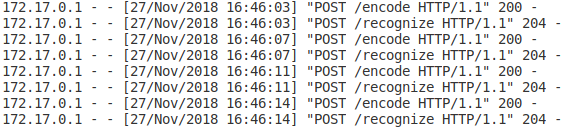


Рисунок 5 Скриншот запущенного сервера

## Сервер приложения

### Конфигурация

Исходный код с логикой, написанной на TypeScript с использованием фреймворка Express.js, компилируется при помощи команды npm run build, которая, в свою очередь, запускает компиляцию с помощью tsc. Затем команда npm start запускает приложение.

Дополнительно должен быть создан файл .env, содержащий конфигурационные данные, характерные для данной конкретной установки, и содержащий следующую информацию:

DB\_USER=username

DB\_PASS=password

DB\_URL=databasehost:55915/databasename

RECOGNIZER\_URL=http://localhost:5001

### Функционал

1. Прослушивание входящих соединений
2. При входящем POST запросе на /recognize выполняется запрос к сервису распознавания лиц и полученный идентификатор сравнивается с хранимыми в базе
3. При входящем POST запросе на /register полученный идентификатор сохраняется в базу данных
4. При входящем GET запросе на /serialports возвращается список последовательных портов на компьютере

Ниже приведены некоторые ключевые вырезки из исходного кода.

Обработчики входящих запросов:

/\* formData \*/

router.post('/recognize', upload.any(), function (req, res) {

/\*\*

\* Proxies request to a recognizer

\*/

if (!req.files) {

res.send({ message: ApiResponse.NoFileUploaded });

}

const file = req.files[0];

if (!file) {

res.send({ message: ApiResponse.NoFileUploaded });

}

if (!isFileAllowed(file.originalname)) {

res.send({ message: ApiResponse.NoFileUploaded });

}

encode(file, async (error, proxyRes, body) => {

if (error) {

console.log(error);

console.error(ApiResponse.NoFacesDetected);

res.send({ message: ApiResponse.NoFacesDetected });

return;

}

if (body === ApiResponse.NoFacesDetected) {

console.error(ApiResponse.NoFacesDetected);

res.send({ message: ApiResponse.NoFacesDetected });

return;

}

try {

const encoding: Encoding = JSON.parse(body);

const name = await recognize(encoding);

if (name) {

res.send({

name,

message: ApiResponse.FaceRecognized,

});

} else {

console.error(ApiResponse.FaceNotRecognized);

res.send({

encoding,

message: ApiResponse.FaceNotRecognized,

});

}

} catch (e) {

console.log(body);

console.error(e);

res.status(500).send(e, body);

}

});

});

/\* { name, encoding } \*/

router.post('/register', function (req, res) {

const face: Face = req.body;

if (face.name == null || !Array.isArray(face.encoding)) {

res.status(400).send({ message: ApiResponse.InvalidFaceReceived });

return;

}

registerFace(face);

res.send({ message: ApiResponse.FaceRegistered });

});

router.get('/serialports', async function (req, res) {

const ports = await listSerialPorts();

res.send(ports);

})

Работа с базой данных:

import \* as Mongoose from 'mongoose';

import { Face } from './core';

const {

DB\_USER,

DB\_PASS,

DB\_URL,

} = process.env;

Mongoose.connect(

`mongodb://${DB\_USER}:${DB\_PASS}@${DB\_URL}`,

{ useNewUrlParser: true },

(err) => {

if (err) {

console.error('Error while connecting to the DB');

console.error(err);

return;

}

console.log('Connected to the DB');

});

Mongoose.set('debug', (collectionName, method, query, doc) => {

console.log(`${collectionName}.${method}`, JSON.stringify(query), doc);

});

const FaceSchema = {

name: String,

encoding: [Number],

}

export const FaceModel: Mongoose.Model<Face> = Mongoose.model('Face', FaceSchema);

Сохранение лица в БД:

export function registerFace(face: Face): Promise<void> {

return FaceModel.create(face);

}

Ниже приведена блок-схема работы сервера:



Рисунок Блок-схема работы серверной части приложения

## База данных

В качестве системы управления базы данных используется документоориентированная СУБД MongoDB. Такой выбор продиктован отсутствием необходимости в реляционности, то есть отношениях между различными наборами данных. Вся хранимая инормация – это список JSON документов вида { name: string, encoding: Array<number> }.

База данных не развертывается локально, а используется облачный сервис https://mlab.com/

Структурно БД состоит из одной коллекции faces и описана следующим образом:

const FaceSchema = {

name: String,

encoding: [Number],

}

export const FaceModel: Mongoose.Model<Face> = Mongoose.model('Face', FaceSchema);

## Веб-клиент

### Конфигурация

Исходный код с логикой, написанной на Javascript с использованием фреймворка React и синтаксиса разметки JSX, компилируется при помощи команды npm run build, которая, в свою очередь, запускает компиляцию с помощью Babel и сборку модулей в один файл с помощью Webpack. Затем папки public и dist могут отдаваться клиенту любым выбраным веб-сервером, например nginx или Apache. Для запуска дев-версии приложения, необходимо выполнить команду npm start.

### Функционал

1. Хранение данных в кэше приложения (local storage)
2. Отправка изображения с веб-камеры на веб-север для распознавания
3. Отправка изображения с веб-камеры на веб-сервер для регистрации
4. Авторизация при распознавании
5. Отправка запроса на получение списка последовательных портов и его отображение.

Ниже приведены некоторые ключевые вырезки из исходного кода.

Общение с серверной частью:

import axios from 'axios';

export const recognize = (data /\* formData \*/) => axios({

data,

method: 'post',

url: '/recognize',

config: { headers: { 'Content-Type': 'multipart/form-data' } },

});

export const register = (data /\* { name, encoding } \*/) => axios({

data,

method: 'post',

url: '/register',

});

export const getSerialPortsList = () => axios('/serialports');

Реализация конпонента-инжектора состояния авторизации:

import React from 'react';

import PropTypes from 'prop-types';

const AuthenticationContext = React.createContext();

export class AuthenticationProvider extends React.Component {

constructor(props) {

super(props);

const localStorageName = localStorage.getItem('name');

const localStorageNameIsDefined = localStorageName && localStorageName !== 'undefined';

const name = localStorageNameIsDefined ? localStorageName : undefined;

this.state = {

name,

};

}

setName = (name) => {

this.setState({ name });

localStorage.setItem('name', name);

}

render() {

const providerValue = {

name: this.state.name,

setName: this.setName,

};

return (

<AuthenticationContext.Provider value={providerValue}>

{this.props.children}

</AuthenticationContext.Provider>

);

}

}

export const withAuthentication = (Component) => {

function WithAuthenticationHoc(props) {

return (

<AuthenticationContext.Consumer>

{(context) => <Component {...props} authentication={context} />}

</AuthenticationContext.Consumer>

);

}

WithAuthenticationHoc.displayName = `WithAuthentication(${Component.displayName || Component.name})`

return WithAuthenticationHoc;

};

export const withAuthenticationPropTypes = {

authentication: PropTypes.shape({

name: PropTypes.string,

setName: PropTypes.func.isRequired,

}).isRequired,

};

Реализация компонента, перенаправляющего на страницу авторизации в случае, если пользователь не авторизован:

import React from 'react';

import { Redirect, Route } from 'react-router-dom';

import { withAuthentication } from './AuthenticationProvider';

function PrivateRoute(

{

component: Component,

authentication,

...rest

}) {

return (

<Route

{...rest}

render={(props) =>

authentication.name ? (

<Component {...props} />

) : (

<Redirect

to={{

pathname: "/authenticate",

state: { from: props.location }

}}

/>

)

}

/>

);

}

export default withAuthentication(PrivateRoute);

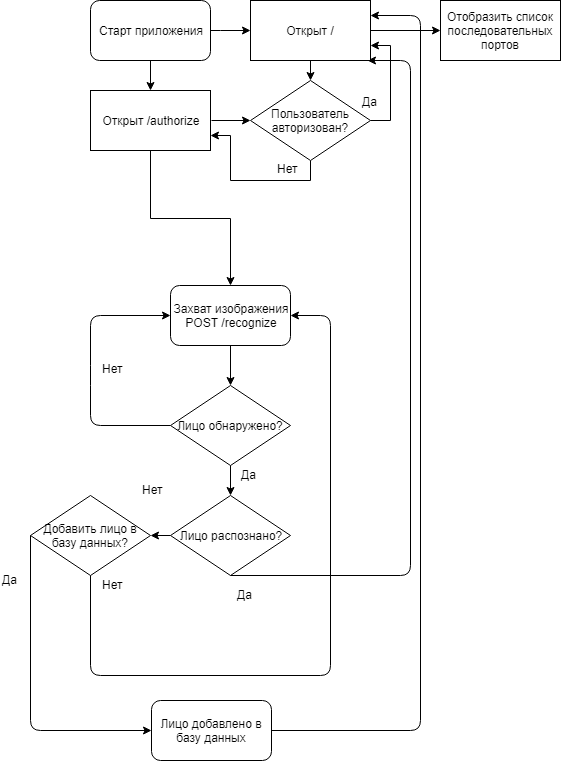


Рисунок 7 Блок схема работы клиентской части приложения

Исходный код целиком можно посмотреть в приложении.

### Результаты тестирования приложения

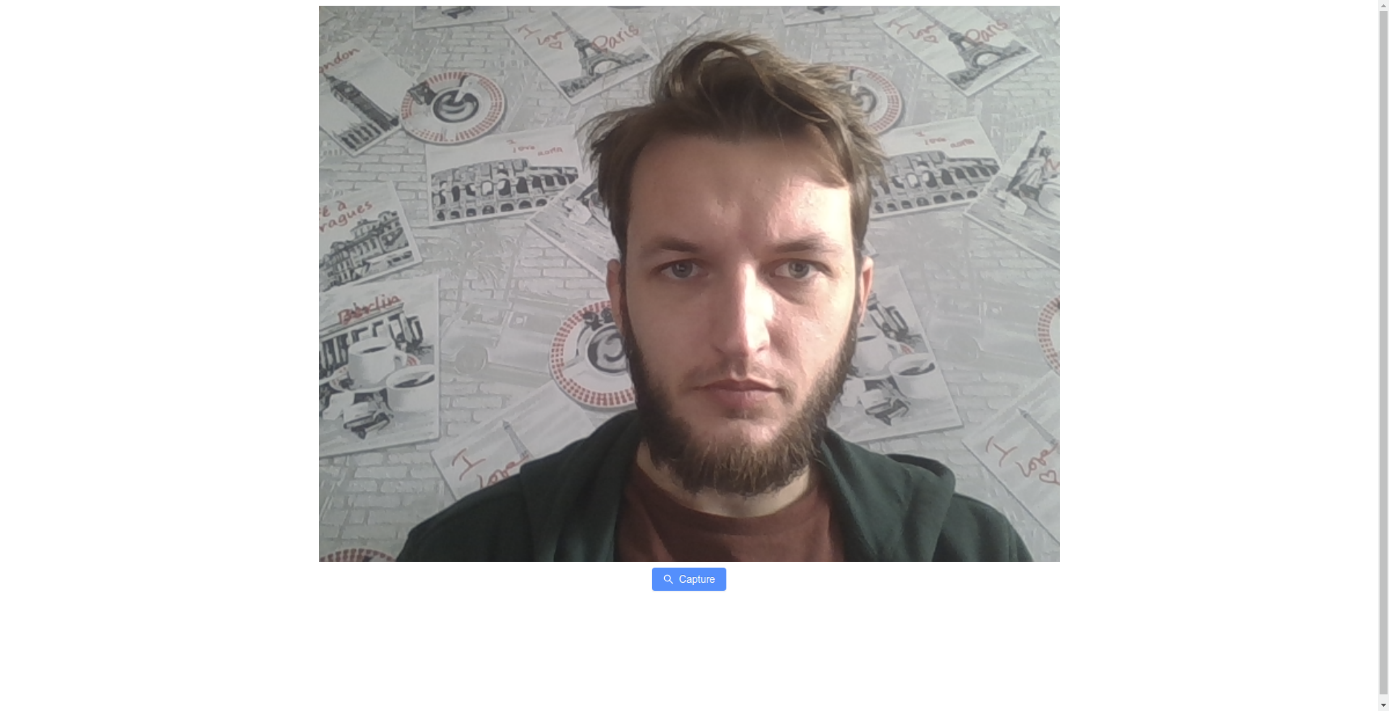


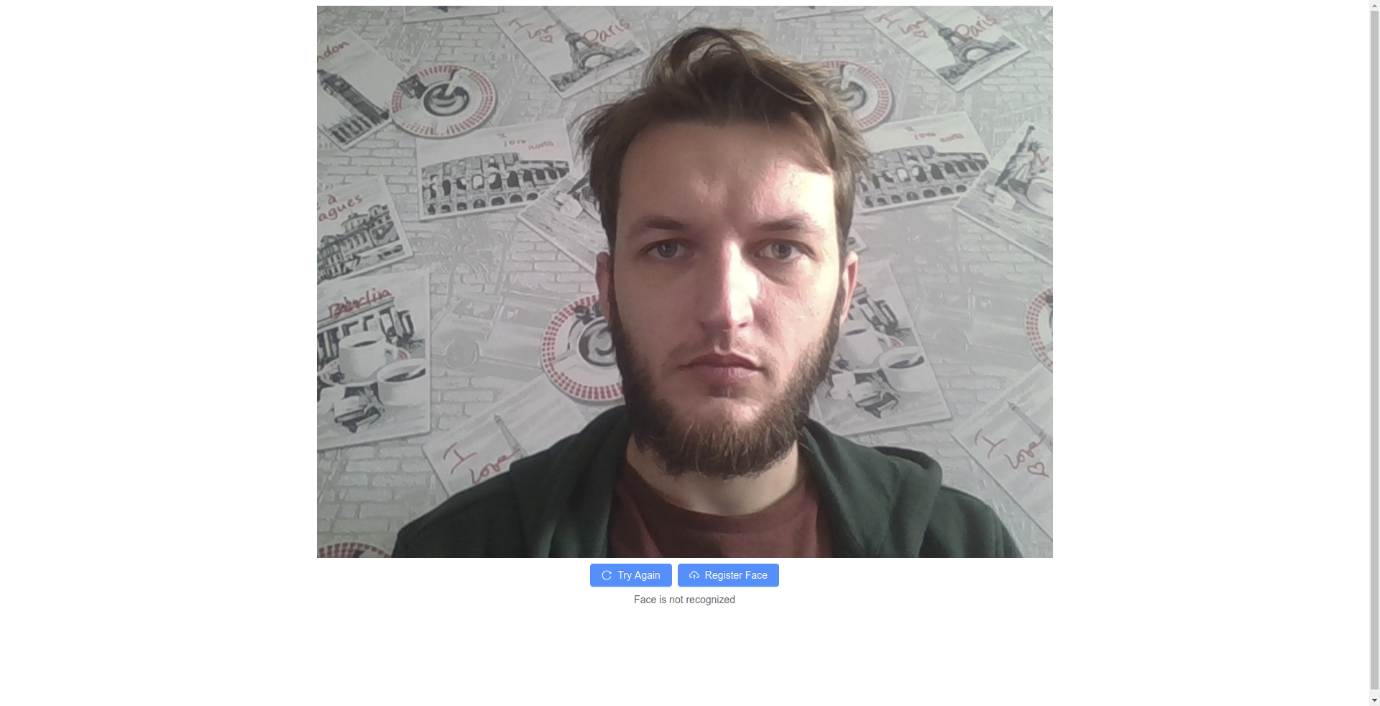
Рисунок Старт работы приложения

Рисунок Результат нажатия кнопки "Capture" - лицо не распознано

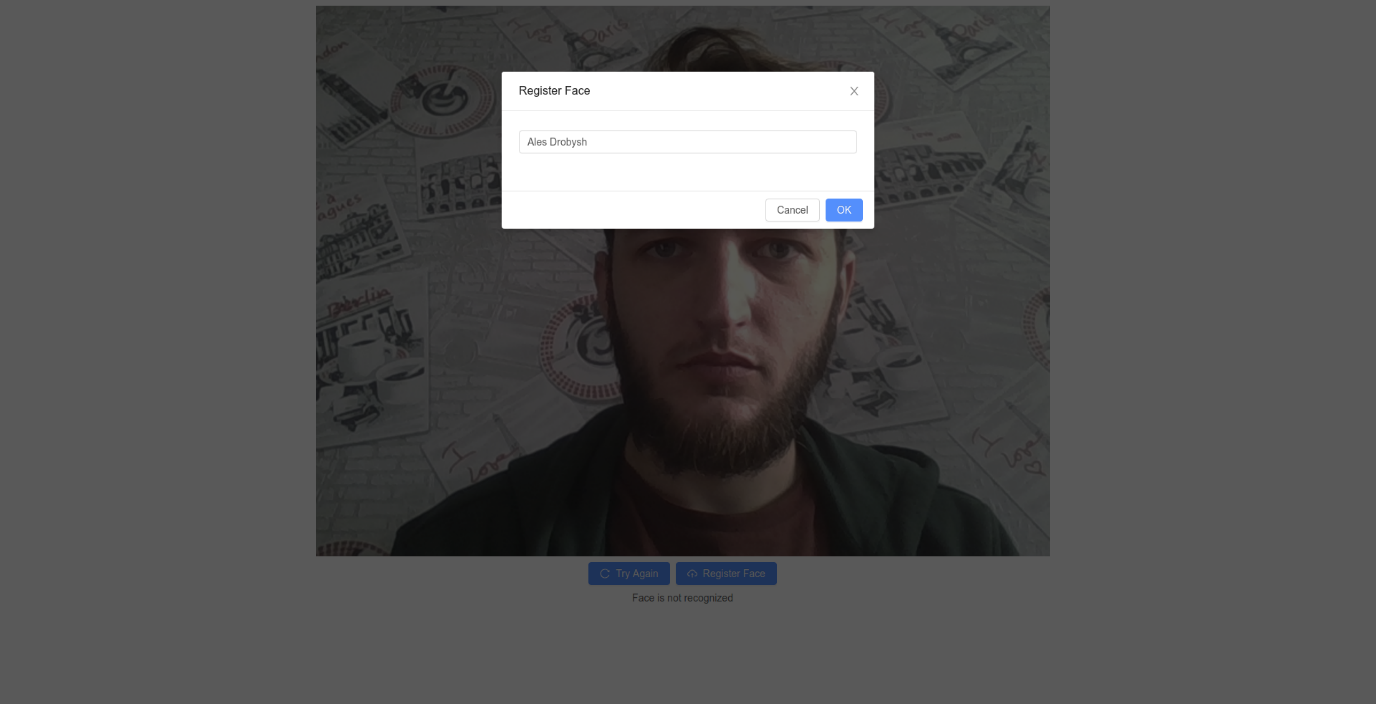


Рисунок Результат нажатия кнопки "Register face"



Рисунок Интерфейс приложения после авторизации

## Оценка точности распознавания лица

Поскольку используемая нейронная сеть была натренированна на наборе лиц Labeled Faces in Wild, то для тестирования был выбран другой датасет (Glasgow Unfamiliar Face Database), чтобы избежать феномена переобучения (когда алгоритм работает хорошо на данных, испозовавшихся при обучении, но результаты на новых данных получаются значительно хуже).

Были подготовлены две базы данных, одна была наполнена 181 дескрипторами лиц, другая только одним. Таким образом имитировались два режима работы – многопользовательская и однопользовательская системы. Эти две базы тестировались на наборе из 199 дескрипторов лиц (из которых в первой базе были сохранены только 150, во второй базе ни один дескриптор из этих 199 не присутствовал).

Учитывая, что возможны два варианта ответа системы и два правильных ответа, то всего возможно 4 исхода:

|  |  |  |
| --- | --- | --- |
|  | Система отвечает «да» | Система отвечает «нет» |
| Да | Положительный исход | Ложноотрицательный исход |
| Нет | Ложноположительный исход | Отрицательный исход |

Таблица Возможные исходы распознавания лица

Из которых удоблетворительными являются только два. С некоторым допущением удовлетворительным можно назвать и ложноотрицательный исход, поскольку в данной ситуации пользователю необходимо лишь еще раз сделать фото, однако это число все равно не должно быть большим во избежание необходимости делать много фото подряд и получать раз за разом отказ во входе в систему.

За принятие решения отвечает следующая функция:

def compare\_faces(encoding1, encoding2, tolerance):

if len(encoding1) is not len(encoding2):

raise Exception('Encodings length are not compatible')

distance = 0

length = len(encoding1)

for i in range(length):

distance = distance + (encoding1[i] - encoding2[i]) \*\* 2

return distance < tolerance

По сути функция рассчитывает евклидово расстояние между двумя векторами и, если оно меньше какого-то значения tolerance, товозвращает «Да», иначе – «Нет».

Далее представлены результаты тестирования при различных значениях tolerance для базы данных с 188 сохраненными дескрипторами.

|  |  |  |
| --- | --- | --- |
| 0.6 188 | Система отвечает «да» | Система отвечает «нет» |
| Да | 2 | 0 |
| Нет | 197 | 0 |

Таблица Результаты для набора данных размером 188 и tolerance равного 0.6

|  |  |  |
| --- | --- | --- |
| 0.1 188 | Система отвечает «да» | Система отвечает «нет» |
| Да | 142 | 8 |
| Нет | 2 | 47 |

Таблица Результаты для набора данных размером 188 и tolerance равного 0.1

|  |  |  |
| --- | --- | --- |
| 0.05 188 | Система отвечает «да» | Система отвечает «нет» |
| Да | 58 | 92 |
| Нет | 1 | 48 |

Таблица Результаты для набора данных размером 188 и tolerance равного 0.05

|  |  |  |
| --- | --- | --- |
| 0.03 188 | Система отвечает «да» | Система отвечает «нет» |
| Да | 1 | 149 |
| Нет | 0 | 49 |

Таблица Результаты для набора данных размером 188 и tolerance равного 0.03

|  |  |  |
| --- | --- | --- |
| 0.01 188 | Система отвечает «да» | Система отвечает «нет» |
| Да | 0 | 150 |
| Нет | 0 | 49 |

Таблица Результаты для набора данных размером 188 и tolerance равного 0.01

Система перестает выдавать ложноположительные ответы при tolerance равном 0.03, однако процент ложноотрицательных ответов 149/199 = 74.8% делает систему практически непригодной для использования.

Ниже предоставлены результаты тестирования на базе данных с одним лицом.

|  |  |  |
| --- | --- | --- |
| 0.6 1 | Система отвечает «да» | Система отвечает «нет» |
| Да | 0 | 0 |
| Нет | 136 | 63 |

Таблица Результаты для набора данных размером 1 и tolerance равного 0.6

|  |  |  |
| --- | --- | --- |
| 0.1 1 | Система отвечает «да» | Система отвечает «нет» |
| Да | 0 | 0 |
| Нет | 0 | 199 |

Таблица Результаты для набора данных размером 1 и tolerance равного 0.1

Как видно при tolerance равном 0.1 система ошибается в 2/199 = 1% случаев на относительно большом наборе данных и не ошибается на наборе данных, состоящем из одного лица.

По результатам тестирования в работе приложения используется tolerance равный 0.1.

# Техника безопасности при работе за персональным компьютером

Персональный компьютер — электроприбор. От прочих электроприборов он отличается тем, что для него предусмотрена возможность длительной эксплуатации без отключения от электрической сети. Кроме обычного режима работы компьютер может находиться в режиме работы с пониженным электропотреблением или в дежурном режиме ожидания запроса. В связи с возможностью продолжительной работы компьютера без отключения от электросети следует уделить особое внимание качеству организации электропитания.

## Требования по электрической безопасности

Недопустимо использование некачественных и изношенных компонентов в системе электроснабжения, а также их суррогатных заменителей: розеток, удлинителей, переходников, тройников. Недопустимо самостоятельно модифицировать розетки для подключения вилок, соответствующих иным стандартам. Электрические контакты розеток не должны испытывать механических нагрузок, связанных с подключением массивных компонентов (адаптеров, тройников и т. п.).

Все питающие кабели и провода должны располагаться с задней стороны компьютера и периферийных устройств. Их размещение в рабочей зоне пользователя недопустимо.

Запрещается производить какие-либо операции, связанные с подключением, отключением или перемещением компонентов компьютерной системы без предварительного отключения питания.

Компьютер не следует устанавливать вблизи электронагревательных приборов и систем отопления.

Недопустимо размещать на системном блоке, мониторе и периферийных устройствах посторонние предметы: книги, листы бумаги, салфетки, чехлы для защиты от пыли. Это приводит к постоянному или временному перекрытию вентиляционных отверстий.

Запрещается внедрять посторонние предметы в эксплуатационные или вентиляционные отверстия компонентов компьютерной системы.

Особенности электропитания монитора. Монитор имеет элементы, способные сохранять высокое напряжение в течение длительного времени после отключения от сети. Вскрытие монитора пользователем недопустимо ни при каких условиях. Это не только опасно для жизни, но и технически бесполезно, так как внутри монитора нет никаких органов, регулировкой или настройкой которых пользователь мог бы улучшить его работу. Вскрытие и обслуживание мониторов может производиться только в специальных мастерских.

## Особенности электропитания системного блока

Все компоненты системного блока получают электроэнергию от блока питания. Блок питания ПК — это автономный узел, находящийся в верхней части системного блока. Правила техники безопасности не запрещают вскрывать системный блок, например при установке дополнительных внутренних устройств или их модернизации, но это не относится к блоку питания. Блок питания компьютера — источник повышенной пожаро-опасности, поэтому вскрытию и ремонту он подлежит только в специализированных мастерских.

Блок питания имеет встроенный вентилятор и вентиляционные отверстия. В связи с этим в нем неминуемо накапливается пыль, которая может вызвать короткое замыкание. Рекомендуется периодически (один - два раза в год) с помощью пылесоса удалять пыль из блока питания через вентиляционные отверстия без вскрытия системного блока. Особенно важно производить эту операцию перед каждой транспортировкой или наклоном системного блока.

## Система гигиенических требований

Длительная работа с компьютером может приводить к расстройствам состояния здоровья. Кратковременная работа с компьютером, установленным с грубыми нарушениям гигиенических норм и правил, приводит к повышенному утомлению. Вредное воздействие компьютерной системы на организм человека является комплексным. Параметры монитора оказывают влияние на органы зрения. Оборудование рабочего места влияет на органы опорно-двигательной системы. Характер расположения оборудования в компьютерном классе и режим его использования влияет как на общее психофизиологическое состояние организма, так и им органы зрения.

В прошлом монитор рассматривали в основном как источник вредных излучений, воздействующих прежде всего на глаза. Сегодня такой подход считается недостаточным. Кроме вредных электромагнитных излучений (которые на современных мониторах понижены до сравнительно безопасного уровня) должны учитываться параметры качества изображения, а они определяются не только монитором, но и видеоадаптером, то есть всей видеосистемы в целом.

Монитор компьютера должен удовлетворять следующим между народным стандартам безопасности:

* по уровню электромагнитных излучений — ТСО 95
* по параметрам качества изображения (яркость, контрастность, мерцание, антибликовые свойства и др. ) — ТСО 99

Узнать о соответствии конкретной модели данным стандартам можно в сопроводительной документации. Для работы с мониторами, удовлетворяющими данным стандартам, специальные защитные экраны не требуется.

На рабочем месте монитор должен устанавливаться таким образом, чтобы исключить возможность отражения от его экрана в сторону пользователя источников общего освещения помещения.

Расстояние от экрана монитора до глаз пользователя должно составлять от 50 до 70 см. Оптимально размещение монитора на расстоянии 1,5 D от глаз пользователя, где D — размер экрана монитора, измеренный по диагонали. Завыщенное расстояния от глаз до монитора приводит к дополнительному напряжению органов зрения, сказывается на затруднении перехода от работы с монитором к работе с книгой и проявляется в преждевременном развитии дальнозоркости.

Важным параметром является частота кадров, которая зависит от свойств монитора, видеоадаптера и программных настроек видеосистемы. Для работы с текстами минимально допустима частота кадров 72 Гц. Для работы с графикой рекомендуется частота кадров от 85 Гц и выше.

## Требования к рабочему месту

В требования к рабочему месту входят требования к рабочему столу, посадочному месту (стулу, креслу), Подставкам для рук и ног. Несмотря на кажущуюся простоту, обеспечить правильное размещение элементов компьютерной системы и правильную посадку пользователя чрезвычайно трудно. Полное решение проблемы требует дополнительных затрат, сопоставимых по величине со стоимостью отдельных узлов компьютерной системы, поэтому и биту и на производстве этими требованиями часто пренебрегают.

Монитор должен быть установлен прямо перед пользователем и не требовать поворота головы или корпуса тела.

Рабочий стол и посадочное место должны иметь такую высоту, чтобы уровень глаз пользователя находился чуть выше центра монитора. На экран монитора следует смотреть сверху вниз, а не наоборот. Даже кратковременная работа с монитором, установленным слишком высоко, приводит к утомлению шейных отделов позвоночника.

Если при правильной установке монитора относительно уровня глаз выясняется, что ноги пользователя не могут свободно покоиться на полу, следует установить подставку для ног, желательно наклонную. Если ноги не имеют надежной опоры, это непременно ведет к нарушению осанки и утомлению позвоночника. Удобно, когда компьютерная мебель (стол и рабочее кресло) имеют средства для регулировки по высоте. В этом случае проще добиться оптимального положения.

Клавиатура должна быть расположена на такой высоте, чтобы пальцы рук располагались на ней свободно, без напряжения, а угол между плечом и предплечьем составлял 100° — 110°. При использовании обычных школьно-письменных столов добиться одновременно правильного положения и монитора, и клавиатуры практически невозможно. Для работы рекомендуется использовать специальные компьютерные столы, имеющие выдвижные полочки для клавиатуры.

При длительной работе с клавиатурой возможно утомление сухожилий кистевого сустава. Известно тяжелое профессиональное заболевание — кистевой туннельный синдром, связанное с неправильным положением рук на клавиатуре. Во избежание чрезмерных нагрузок на кисть желательно предоставить рабочее кресло с подлокотниками, уровень высоты которых, замеренный от пола, совпадает с уровнем высоты расположения клавиатуры.

При работе с мышью рука не должна находиться на весу. Локоть руки или хотя бы запястье должны иметь твердую опору. Если предусмотреть необходимое расположение рабочего стола и кресла затруднительно, рекомендуется применить коврик для мыши, имеющий специальный опорный валик. Нередки случаи, когда в поисках опоры для руки (обычно правой) располагают монитор сбоку от пользователя (соответственно, слева), чтобы он работал вполоборота, опирая локоть или запястье правой руки о стол. Этот прием недопустим. Монитор должен обязательно находиться прямо перед пользователем.

## Требования к организации пространства

Экран монитора — не единственный источник вредных электромагнитных излучений. Разработчики мониторов достаточно давно и успешно занимаются их преодолением. Меньше внимания уделяется вредным побочным излучениям, возникающим со стороны боковых и задней стенок оборудования. В современных компьютерных системах эти зоны наиболее опасны.

Монитор компьютера следует располагать так, чтобы задней стенкой он был обращен не к людям, а к стене помещения. В компьютерных помещениях, имеющих несколько компьютеров, рабочие места должны располагаться по периферии помещения, оставляя свободным центр. При этом дополнительно необходимо проверить каждое из рабочих мест на отсутствие прямого отражения внешних источников освещения. Как правило, добиться этого для всех рабочих мест одновременно достаточно трудно. Возможное решение состоит в использовании штор на окнах и продуманном размещении искусственных источников общего и местного освещения.

Сильными источниками электромагнитных излучений являются устройства бесперебойного питания. Располагать их следует как можно дальше от посадочных мест пользователей.

# Энергосбережение

Энергосбережение — реализация правовых, организационных, научных, производственных, технических и экономических мер, направленных на эффективное (рациональное) использование (и экономное расходование) топливно-энергетических ресурсов и на вовлечение в хозяйственный оборот возобновляемых источников энергии. При разработке ПО самые большие траты приходятся на электроэнергию, потребляемую компьютерами.

## Основные способы экономии энергии

Ключевыми мероприятиями оптимизации потребления электроэнергии на освещение являются:

* максимальное использование дневного света (повышение прозрачности и увеличение площади окон, дополнительные окна);
* повышение отражающей способности (белые стены и потолок);
* оптимальное размещение световых источников (местное освещение, направленное освещение);
* использование осветительных приборов только по необходимости;
* повышение светоотдачи существующих источников (замена люстр, плафонов, удаление грязи с плафонов, применение более эффективных отражателей);
* замена ламп накаливания на энергосберегающие (люминесцентные, компактные люминесцентные, светодиодные);
* применение устройств управления освещением (датчики движения и акустические датчики, датчики освещенности, таймеры, системы дистанционного управления);
* внедрение автоматизированной системы диспетчерского управления наружным освещением (АСДУ НО);
* установка интеллектуальных распределённых систем управления освещением (минимизирующих затраты на электроэнергию для данного объекта).

## Энергосберегающие режимы работы компьютера

Спящий режим — это режим пониженного потребления электроэнергии, который позволяет быстро возобновить работу в режиме обычного потребления энергии (обычно в течение нескольких секунд) по требованию пользователя.

Перевод компьютера в спящий режим напоминает нажатие кнопки "Пауза" на проигрывателе DVD — компьютер немедленно останавливает все операции и готов к возврату в рабочий режим при необходимости.

Режим гибернации — это режим пониженного потребления электроэнергии, разработанный в первую очередь для ноутбуков. При переходе в спящий режим все открытые документы и параметры сохраняются в памяти и компьютер переходит в режим пониженного потребления электроэнергии, а при переходе в режим гибернации все открытые документы и программы сохраняются на жестком диске и затем компьютер выключается. Из всех энергосберегающих режимов, используемых в ОС Windows, для поддержания режима гибернации требуется наименьшее количество электроэнергии. Если в течение длительного промежутка времени не планируется использовать ноутбук и нет возможности подзарядить батарею, рекомендуется перевести ноутбук в режим гибернации.

Гибридный спящий режим — это режим, который разработан преимущественно для настольных компьютеров. Гибридный спящий режим сочетает в себе спящий режим и режим гибернации, поскольку все открытые документы и программы сохраняются в памяти и на жестком диске и компьютер переводится в режим пониженного потребления электроэнергии. При неожиданном сбое питания операционная система Windows может легко восстановить данные с диска. Если гибридный спящий режим включен, переход в спящий режим автоматически переводит компьютер в гибридный спящий режим. На настольных компьютерах гибридный спящий режим обычно включен по умолчанию.

Таким образом, наиболее эффективное энергосбережение достигается при применении спящего режима в сочетании с режимом гибернации, либо при применении гибридного спящего режима.

## Энергоэффективность в серверных помещениях

Обслуживание касается как серверного оборудования, так и системы кондиционирования. Нужно поддерживать правильные настройки. Например, не нужно задавать на кондиционере слишком низкую температуру, которая не нужна для работы серверов. Нужно следить за расположением коммутационных кабелей в серверных шкафах. Обычно со временем клубок кабелей копится, потоки воздуха начинают испытывать излишнее сопротивление. Пустые юниты в стойках стоит закрывать панелями-заглушками, чтобы холодный воздух не перемешивался с горячим. Нужно вовремя чистить и мыть оборудование, т.к. грязный наружный блок кондиционера или грязные воздушные фильтры во внутренних блоках существенно снижают теплообмен. Если не следить за всем выше перечисленным, существенно увеличиваются энергозатраты (компрессор и вентиляторы вынуждены работать более производительно).

## Стандарты энергосберегающих технологий

В начале 90-х годов компания EPA (Environmental Protection Agency — Агентство по защите окружающей среды) начало проводить кампанию по сертификации энергосберегающих персональных компьютеров и периферийного оборудования. Компьютер или монитор во время продолжительного простоя должен снизить энергопотребление до 30 Вт и более. Система, удовлетворяющая этим требованиям, может получить сертификат Energy Star.

### Стандарт усовершенствованной системы управления питанием (Advanced Power Management — APM)

Разработан фирмой Intel совместно с Microsoft и определяет ряд интерфейсов между аппаратными средствами управления питанием и операционной системой компьютера. Полностью реализованный стандарт APM позволяет автоматически переключать компьютер между пятью состояниями в зависимости от текущего состояния системы. Каждое последующее состояние в приведенном ниже списке характеризуется уменьшением потребления энергии.

* Full On

Система полностью включена.

* APM Enabled

Система работает, некоторые устройства являются объектами управления для системы управления питанием. Неиспользуемые устройства могут быть выключены, может быть также остановлена или замедлена (т.е. снижена тактовая частота) работа тактового генератора центрального процессора.

* APM Standby (резервный режим)

Система не работает, большинство устройств находятся в состоянии потребления малой мощности. Работа тактового генератора центрального процессора может быть замедлена или остановлена, но необходимые параметры функционирования хранятся в памяти. Пользователь или операционная система могут запустить компьютер из этого состояния почти мгновенно.

* APM Suspend (режим приостановки).

Система не работает, большинство устройств пассивны. Тактовый генератор центрального процессора остановлен, а параметры функционирования хранятся на диске и при необходимости могут быть считаны в память для восстановления работы системы. Чтобы запустить систему из этого состояния, требуется некоторое время.

* Off (система отключена).

Система не работает. Источник питания выключен.

Для реализации режимов APM требуются аппаратные средства и программное обеспечение. Источниками питания ATX можно управлять с помощью сигнала Power\_On и факультативного разъема питания с шестью контактами. (Необходимые для этого команды выдаются программой.) Изготовители также встраивают подобные устройства управления в другие элементы системы, например в системные платы, мониторы и дисководы. Операционные системы (такие как Windows), которые поддерживают APM, при наступлении соответствующих событий запускают программы управления питанием, наблюдая за действиями пользователя и прикладных программ. Однако операционная система непосредственно не посылает сигналы управления питанием аппаратным средствам. Система может иметь множество различных аппаратных устройств и программных функций, используемых при выполнении функций APM. Чтобы разрешить проблему сопряжения этих средств в операционной системе и аппаратных средствах предусмотрен специальный абстрактный уровень, который облегчает связь между различными элементами архитектуры

При запуске операционной системы загружается программа — драйвер APM, который связывается с различными прикладными программами и программными функциями. Именно они запускают действия управления питанием, причем все аппаратные средства, совместимые с APM, связываются с системной BIOS. Драйвер APM и BIOS связаны напрямую; именно эту связь использует операционная система для управления режимами аппаратных средств.

Таким образом, чтобы функционировали средства APM, необходим стандарт, поддерживаемый схемами, встроенными в конкретные аппаратные устройства системы, системная BIOS и операционная система с драйвером APM. Если хотя бы один из этих компонентов отсутствует, APM работать не будет.

### Стандарт усовершенствованной системы управления питанием (Advanced Power Management—APM)

Усовершенствованная конфигурация и интерфейс питания (Advanced Configuration and Power Interface— ACPI) впервые реализованы в современных BIOS и операционных системах Windows 98 и более поздних. Если BIOS компьютера поддерживает систему ACPI, то все управление питанием передается операционной системе. Это упрощает конфигурирование параметров, все они находятся в одном месте — в операционной системе.

### Стандарт DPMS (Display Power Management Signaling - система сигналов управления питанием монитора)

Стандарт ассоциации VESA. Определяет состав сигналов, передаваемых компьютером в монитор, при вхождении системы от состояния простоя в режимы пониженного потребления энергии. В этих системных процедурах контроль берет на себя драйвер, посылающий соответствующие сигналы через графическую карту. При нажатии клавиши на клавиатуре или движении "мыши" монитор переходит в нормальный режим работы.

## Энергосберегающие технологии в мобильных ПК

Для решения проблемы энергопотребления в корпорации Intel были созданы специальные версии мобильных процессоров, например Intel Pentium III-M, Intel Pentium 4-M и процессор Intel Pentium M для мобильных ПК с поддержкой технологии Intel Centrino. Они отличаются от своих собратьев для стационарных ПК (исключение составляет процессор Intel Pentium M, не имеющий аналога для стационарного ПК) средствами управления энергопотреблением, позволяющими увеличить продолжительность автономной работы ноутбука на мобильном процессоре. К таким средствам относится:

* технология Enhanced Intel SpeedStep;
* режимы ожидания Deep Sleep и Deeper Sleep;
* технология Intel Mobile Voltage Positioning (IMVP).

Использование мобильных версий процессоров позволяет отчасти решить проблему производительности в совокупности с увеличением времени работы ноутбука от батареи.

### Технология Enhanced Intel SpeedStep

Улучшенная технология SpeedStep (Enhanced Intel SpeedStep) дает пользователям возможность увеличить время автономной работы от батареи за счет динамического изменения напряжения ядра процессора и его тактовой частоты. Изменение условий работы процессора зависит от его загрузки (степени утилизации), от температурного режима, а также от установленных пользователем предпочтений через задания схемы энергопотребления (Power Schemes) в настройках операционной системы.

В отличие от предыдущей версии технологии Intel SpeedStep, предусматривающей возможность работы мобильного процессора лишь на двух тактовых частотах, улучшенная технология Enhanced Intel SpeedStep определяет использование нескольких возможных напряжений питания и частот (в совокупности — рабочих точек), что позволяет достичь лучшего соотношения «напряжение/частота» и более эффективного режима функционирования, когда производительность согласуется с рабочей нагрузкой.

Крайние рабочие точки процессора задаются аппаратно, а промежуточные точки устанавливаются программно. Управление переходами между различными рабочими точками выполняется только самим процессором и блоком регулятора напряжения (VRM).

Для установки требуемого напряжения процессор Intel Pentium M посылает служебные VID-последовательности непосредственно в VRM-модуль. При этом не используются никакие другие компоненты системы при осуществлении перехода между рабочими состояниями процессора.

Переход между различными рабочими точками процессора, характеризующимися напряжением и частотой, происходит таким образом, чтобы обеспечивать работоспособность процессора в процессе самого перехода (который не может осуществляться мгновенно). Для того чтобы осуществить переход на более высокую тактовую частоту, сначала до требуемого уровня меняется напряжение процессора. Процесс изменения напряжения длится порядка 100 мкс, то есть является достаточно длительным. Чтобы сохранить работоспособность процессора при изменении напряжения, частота процессора при этом не меняется. Когда же напряжение изменится и достигнет требуемого уровня, происходит скачкообразное увеличение частоты процессора, которое длится порядка 10 мкс. Если требуется осуществить переход к меньшей частоте, сначала происходит практически мгновенное изменение частоты (в течение 10 мкс), а после этого постепенно уменьшается напряжение самого процессора — уже при неизменной частоте.

Всего в технологии Enhanced Intel SpeedStep рассматриваются четыре схемы энергопотребления:

* Maximum Performance Mode;
* Automatic Mode;
* Battery-Optimized Performance Mode;
* Maximum Battery Mode.

Схема Maximum Performance Mode — это режим по умолчанию работы ноутбука в случае питания от сети (внешнего источника питания). В этом режиме процессор работает на максимальной тактовой частоте, что обеспечивает максимальную производительность.

Схема Automatic Mode является схемой по умолчанию при автономной работе ноутбука от аккумуляторной батареи. В данном режиме средствами операционной системы определяется степень загруженности процессора и в зависимости от полученного значения динамически устанавливаются требуемые значения тактовой частоты и напряжения ядра процессора. Тем самым режим Automatic Mode обеспечивает баланс между производительностью ноутбука и временем автономной работы от батареи. Отметим также, что режим Automatic Mode автоматически устанавливается и при выборе схемы Battery-Optimized Performance Mode, если температура процессора превышает допустимый уровень, заданный в настройках BIOS.

Battery-Optimized Performance Mode — это режим работы ноутбука, устанавливаемый программным способом средствами операционной системы (Windows XP/Me/2000) через настройки схемы энергопотребления (Power Schemes). В данном режиме работы тактовая частота и напряжение процессора при выполнении им многих нересурсоемких задач понижаются до минимального значения, что позволяет существенно снизить энергопотребление (и соответственно увеличить время автономной работы от батареи) по сравнению с режимом работы процессора на номинальной тактовой частоте.

Maximum Battery Mode. Данный режим, так же как и режим Battery-Optimized Performance Mode, устанавливается программным способом. При его выборе тактовая частота и напряжение процессора понижаются до минимального значения, что позволяет значительно снизить энергопотребление. Следует отметить, что в данном режиме процессор работает на пониженной тактовой частоте при любой степени загрузки. В результате за счет снижения производительности достигается максимально возможное время автономной работы от аккумуляторной батареи. Данный режим предназначен для тех случаев, когда для пользователей наиболее критично именно время автономной работы от батареи, даже в ущерб производительности ноутбука.

Автоматическое переключение между различными схемами энергопотребления (например, при отключении внешнего питания) происходит незаметно для пользователя, так как для этого требуется менее 0,001 с. Естественно, сам процесс переключения не нарушает режима работы всех запущенных приложений. Кроме того, переключение между различными режимами работы возможно и вручную. При использовании операционной системы Windows XP установка требуемого режима работы осуществляется через настройки схемы питания (Power Schemes) в диалоговом окне Power Options.

### Режимы Deep Sleep и Deeper Sleep

Другими средствами энергосбережения, реализованными в мобильных процессорах, являются технологии Deep Sleep (глубокий сон) и ее усовершенствованная версия Deeper Sleep (еще более глубокий сон). Технология Deep Sleep известна также как режим C3 ACPI, а технология Deeper Sleep — как режим C4 ACPI.

Эти технологии позволяют процессору динамически переключаться в режим минимально возможного энергопотребления. Так, для процессоров семейства Intel Pentium M в режиме Deeper Sleep напряжение питания ядра составляет от 0,705 до 0,785 В.

Переход в состояние Deeper Sleep происходит каждый раз, когда регулятор напряжения понижает напряжение ядра процессора по сигналу, получаемому от хаба ввода-вывода (I/O hub).

Несмотря на то что режим Deeper Sleep позволяет снизить общее энергопотребление компьютера, он никак не отражается на его производительности. Дело в том, что динамическое переключение в режим «спячки» происходит только в том случае, если система неактивна. К примеру, когда пользователь набирает текст, то в промежутках между нажатием клавиш (для компьютера это очень большие периоды времени) ноутбук неактивен и может динамически переключаться в режим Deeper Sleep. Вообще, переключение в режим Deeper Sleep происходит каждый раз, когда система неактивна менее 1 мс. Обратный переход из режима Deeper Sleep происходит практически мгновенно, как только система начинает проявлять активность.

В принципе режим Deeper Sleep полностью идентичен режиму Deep Sleep — за тем лишь исключением, что в режиме Deeper Sleep напряжение питания процессора снижается на 30% больше, чем в режиме Deep Sleep.

### Технология Intel Mobile Voltage Positioning (IMVP)

Intel Mobile Voltage Positioning (IMVP) — это технология интеллектуального регулирования напряжения (smart voltage regulation), позволяющая снижать напряжение ядра процессора при одновременном повышении питающего тока, что дает возможность поддерживать требуемый уровень производительности при одновременном снижении энергопотребления, а также обеспечивать условия, необходимые для режимов Deep Sleep и Deeper Sleep.

Технология IMVP также оказывает влияние на тепловыделение процессора (Thermal Design Power, TDP), которое должно поддерживаться в заданных пределах. Снижение TDP позволяет производителям ноутбуков использовать более мощные процессоры в тонких и компактных ноутбуках.

Не так давно корпорация Intel представила улучшенную версию технологии IMVP, которая называется IMVP-IV. В технологии IMVP-IV применяются инновационные методы, позволяющие еще больше снизить требования по энергопотреблению и TDP процессоров. Новую технологию поддерживают мобильные процессоры Intel Pentium 4-M и Intel Pentium M.

### Чипсеты для мобильных процессоров

Для реализации всех технологий энергосбережения (Enhanced Intel SpeedStep, Deeper Sleep, IMVP), заложенных в мобильных версиях процессоров, необходима соответствующая поддержка со стороны чипсета, то есть мобильные процессоры можно использовать только в совокупности с соответствующими мобильными чипсетами. Так, для процессора Intel Pentium III-M — это мобильный чипсет Intel 830, для процессора Intel Pentium 4-M — мобильная версия чипсета Intel 845, а для нового процессора Intel Pentium M — семейство мобильных чипсетов Intel 855.

Все мобильные чипсеты должны удовлетворять определенным требованиям, к которым относятся:

* поддержка интерфейса ACPI (Advanced Configuration and Power Interface) версии 2.0;
* поддержка стандарта AMP (Advanced Power Management) версии 1.2, в котором определяются режимы нормального и пониженного энергопотребления процессора;
* возможность динамического управления частотой чипсета для снижения энергопотребления в периоды неактивности чипсета;
* поддержка режимов низкого энергопотребления чипсета;
* управление питанием AGP-порта.

### Технология Centrino

Говоря о технологиях энергосбережения для мобильных ПК, нельзя не упомянуть о новом поколении ноутбуков, поддерживающих технологию Intel Centrino. Эта технология представляет собой сочетание трех основополагающих компонентов: процессора Intel Pentium M, ранее известного под кодовым названием Banias, чипсета Intel 855 с прежним кодовым названием Odem (а также чипсета 855GМ, чье кодовое название было Montara-GM) и интегрированного беспроводного решения Intel PRO/Wireless network connection. При этом в новой платформе впервые реализован комплексный подход, позволяющий объединить производительности, жизнеспособность батарей, формфактор и возможность установления связи.

Основное преимущество новой платформы заключается в том, что ноутбуки с поддержкой технологии Intel Centrino сочетают в себе высокую производительность наряду с рекордно долгим временем автономной работы от батареи. В основе платформы лежит принципиально новый процессор Intel Pentium M.

Существует три его варианта: Intel Pentium M, Intel Pentium M Low Voltage (LW) и Intel Pentium M Ultra Low Voltage (ULV), различающиеся напряжением питания и возможными тактовым частотами.

Отличительной особенностью новой микроархитектуры процессора Intel Pentium M является сочетание высокой производительности при низком энергопотреблении и соответственно малом тепловыделении.

Кроме уже рассмотренных выше технологий энергосбережения, в процессоре Intel Pentium M реализована технология оптимизации энергопотребления процессорной шины, которая призвана снизить энергопотребление процессора. Как правило, процессоры оставляют свою системную шину в рабочем состоянии даже тогда, когда она не используется; при этом значительную долю энергии потребляют усилители считывания. Эти усилители применяются на шине данных (64 вывода), стробах данных (8 выводов) и для сигналов инверсии данных (4 вывода). Для оптимизации энергопотребления процессор Intel Pentium M включает усилители считывания только непосредственно при приеме данных и отключает их при отсутствии транзакций данных, что приводит к существенной экономии энергопотребления.

В наборе микросхем Intel 855 также используются следующие средства оптимизации энергопотребления:

* сниженное до 1,2 В (вместо 1,5 В) напряжение Vcc ядра процессора;
* управление питанием памяти;
* управление питанием блока ввода-вывода Intel 855 DDR I/O;
* управление блоком FSB I/O;
* поддержка технологии DDR Read Throttling с помощью внешнего датчика температуры;
* управление вводом-выводом DDR I/O;
* сокращение питания в C3.

Управление питанием памяти DDR подразумевает отключение питания при любой возможности и использование оптимизированного метода управления страницами, при котором количество одновременно открытых страниц (что само по себе ведет к увеличению энергопотребления) сводится к минимуму.

# Экономическое обоснование

В разделе представлена информация, из которой выводится целесообразность (или нецелесообразность) создания ПО. Содержит анализ затрат и результатов проекта. Позволяет определить, стоит ли вкладывать деньги в предлагаемый проект.

## Оптимальные характеристики оборудования

Как инструмент разработки программ использовался ноутбук ASUS K54HR со следующими ключевыми техническими характеристиками:

* Процессор Intel(R) Pentium(R) CPU B960 @ 2.20GHz
* Встроенный дисплей с разрешением 1366x768
* ОЗУ DDR3 4 Гбайт

Отметим пункты для значительного увеличения скорости разработки программ:

* Наличие дополнительного монитора
* Мониторы с высоким разрешением экрана
* Высокопроизводительный многоядерный процессор
* Большой размер ОЗУ (8Гбайт - 16Гбайт)
* Наличие SSD в качестве основного жесткого диска

Указанные пункты помогут уменьшить время разработки программ за счет увеличения скорости работы сборщика проектов, возможности одновременного доступа к нескольким источникам информации без необходимости переключения внимания и скорости проведения тестов разработанного ПО за счет возможности параллельной работы сразу с несколькими эмуляторами вместо последовательной в случае работы с устаревшим оборудованием.

Минимальные системные требования:

* Процессор 2ГГц
* Разрешение экрана 1024х768
* ОЗУ 2Гбайта
* Наличие монитора, клавиатуры и мыши

## Расчет себестоимости программного обеспечения

При расчете себестоимости необходимо учитывать целый ряд технико-экономических показателей, к которым относятся время разработки, размер и объем программного кода.

Совокупные затраты на создание скрипта состоят из нескольких компонентов. При расчете все эти компоненты необходимо приводить к одной единице измерения с целью обеспечения их однородности и удобства сопоставления. В качестве такой единицы удобнее всего рассматривать денежные выражения либо человеко-часы.

При разработке программных комплексов наибольшее значение в составе затрат имеют следующие составляющие:

1. затраты на подготовку и применение технологии и программных средств автоматизации разработки программ;
2. затраты на проектирование, программирование, отладку и испытания программ согласно требований пользователя или заказчика;
3. затраты на ЭВМ, используемые при разработке данного программного продукта;
4. затраты на изготовление опытного образца программного продукта как продукции производственно–технического назначения;
5. затраты на подготовку и повышение квалификации специалистов–разработчиков.

Затраты типа 1 и 4 можно исключить, учитывая условия и характер разработки данного программного продукта. Затраты типа 5 тоже не учитываются, так как их наиболее трудно учитывать и формализовать в конкретной работе. Имеются и дополнительные составляющие стоимости, которые не учитываются исходя из условий работы. К этим составляющим относятся: заработная плата, мебель, коммунальные услуги, охрана, противопожарные мероприятия, налоги и т.д.

Для определения составляющих затрат труда воспользуемся расчетом условного числа выполняемых строк программного кода в программе:

, (1)

где *с* = 1,2— коэффициент увеличения затрат, характеризует увеличение затрат труда вследствие недостаточно полного описания задачи, уточнений и некоторой доработки. Этот коэффициент может принимать значения от 1,2 до 5;

*p* = 0.08 — коэффициент коррекции программы в ходе ее разработки, принимает значения от 0.05 до 0.1;

*q* = 1604— число строк программного кода.

В итоге получаем, что условное число операторов в программе принимает значение:



Важной составляющей при расчете себестоимости программного продукта является коэффициент квалификации *k*. Коэффициент квалификации берется из таблицы коэффициентов, приведенной ниже.

|  |  |
| --- | --- |
| Опыт работы | Коэффициент |
| До 2–х лет | 0,8 |
| 2 — 3 года | 1,0 |
| 3 — 5 лет | 1,1–1,2 |
| 5 — 7 лет | 1,3–1,4 |
| Больше 7 лет | 1,5–1,6 |

Таблица 9 Коэффициенты квалификации

Согласно данным таблицы, необходимый коэффициент квалификации принимаем равным *k =* 1,2*.*

Для определения трудоемкости разработки приложения необходимо рассчитать следующие виды затрат:

* затраты труда на подготовку описания задачи;
* затраты труда на разработку блок–схемы приложения;
* затраты труда на программирование по блок–схеме;
* затраты труда на отладку кода;
* затраты труда на подготовку документации/описания.

Затраты труда на подготовку описания задачи рассчитываются по следующей формуле:

, (2)

где  — минимально возможная трудоемкость выполнения работы;

 — наиболее вероятная трудоемкость;

 — максимально возможная трудоемкость.

По формуле 2 получаем, что затраты составляют

 (чел/час).

Затраты труда на разработку блок–схемы алгоритма:

, (3)

где *B* — коэффициент увеличения затрат в зависимости от качества постановки задачи [1,2...5];

*k* — коэффициент квалификации берется из диапазона [75…85], (примем его равным 80):

 (чел/час).

Затраты труда на программирование по блок–схеме рассчитывается по формуле (коэффициент квалификации берется из диапазона [60…75], примем его равным 70):

 (4)

(чел/час).

Затраты труда на отладку кода (коэффициент квалификации берется из диапазона [40…50], примем его равным 40):

 (5)

(чел/час).

Затраты труда на подготовку документации:

(чел/час), (6)

где *t*пр — затраты труда на подготовку

*t*оф — время на оформление документов

,

(чел/час),



В итоге получаем:

(чел/час).

Тогда итоговая трудоемкость разработки данного программного обеспечения равна:

(чел/час),

(чел/час).

Это время приблизительно равняется одному месяцу при длине рабочего дня в 8 часов, что соответствует реальному времени разработки данного программного продукта.

При расчете затрат на разработку программного обеспечения необходимо разделить на составляющие: заработная плата (основная и дополнительная), отчисления на социальные нужды, эксплуатационные затраты (электроэнергия, техническое обслуживание и текущий ремонт), накладные расходы, материалы и комплектующие.

Основная заработная плата рассчитывается по формуле:

(руб.), (7)

где  – суммарные затраты труда, 148(чел/час);

*t*ср – среднее число рабочих дней в месяце: (365–107)/12 = 21,5 дней;

Тарифная ставка (ТС) представляет собой минимальный размер оплаты труда (МРОТ), увеличенный в зависимости от тарифного коэффициента *k*т, соответствующего данному разряду работ. Для 13–ого разряда работ, который соответствует работе программиста, тарифный коэффициент равен 3,04.

ТС = МРОТ.*k*т =200 ⋅ 3,04 = 608 (руб/мес),

(руб.),

(мес.),

где *t*з - время, затраченное на создание программного обеспечения (в месяцах).

Надбавка за стаж составляет 5% от основной заработной платы

(руб.)

Премия 30 % от основной заработной платы за успешно выполненное задание в срок

(руб.).

Отчисление на социальное страхование составляют 35% от всей заработной платы:

(руб.)

Эксплуатационные затраты возникают при эксплуатации ЭВМ:

Стоимость электроэнергии:

, (8)

где *М* — мощность ЭВМ (0,06 кВт);

*к*з — коэффициент загрузки (0,7);

*С*кВт.ч. — стоимость 1 кВт час электроэнергии (0,165 руб.);

*F*эф — эффективный фонд, рассчитывается по формуле:

, (9)

где  — номинальное число рабочих дней в году (255);

 — планируемый процент времени на ремонт ЭВМ (2%);

тогда

(час.),

тогда стоимость электроэнергии составит:

(руб.).

Это стоимость электроэнергии за год, а за период разработки программного обеспечения стоимость электроэнергии составит:

, (10)

где  — суммарные затраты труда, 1273 (чел/час);

 — номинальное число рабочих дней в году (255);

 — стоимость электроэнергии за год.

Тогда стоимость электроэнергии за период разработки программного обеспечения составит:

(руб.).

Предположительная стоимость компьютера с необходимыми параметрами будет составлять 1000 (руб.).

Техническое обслуживание и ремонт составляют 2.5% от стоимости компьютера, тогда получаем:

(руб.).

Это стоимость и обслуживание технического ремонта за год, а за период разработки стоимость технического обслуживания и текущего ремонта составит:

(руб.),

где  — суммарные затраты труда, 1273 (чел.час);

 — номинальное число рабочих дней в году (255).

Накладные расходы составляют 10% от основной заработной платы:

(руб.).

Материалы и комплектующие составляют 1.5% от стоимости оборудования:

(руб.)

Таким образом, суммарные расходы на разработку программного обеспечения составили:

 (руб.) (11)

 (руб.)

|  |  |
| --- | --- |
| Составляющие себестоимости | Сумма, BYN |
| Зарплата | 523 |
| Социальное страхование | 247 |
| Эксплуатационные расходы | 1 |
| Накладные расходы | 52,3 |
| Материалы и комплектующие | 15 |
| Техническое обслуживание | 1,8 |
| **Итого** | 840,1 |

Таблица 10 Смета затрат а разработку

Согласно данным таблицы 10 себестоимость разработанного программного продукта составляет 840,1(руб.).

# Заключение

Было разработано клиент-серверное приложение для распознавания лиц по биометрическим параметрам на основе нейронных сетей и авторизации по результатам распознавания лица. Взаимодействие между отдельными частями программного комплекса организовано по протоколу HTTP. Сервис, распознающий лица, запускается в изолированном контейнере, что упрощает последующее его развертывание в облаке и переиспользование в других приложениях. Клиентская часть не требует особой подготовки клиентской машины, необходим лишь современный веб-браузер.

# Список использованных источников

1. Python — Википедия [Электронный ресурс] / Википедия — свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Python. Дата доступа: 21.05.2018.
2. Flask (веб-фреймворк) — Википедия [Электронный ресурс] / Википедия — свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Flask\_(%D0%B2%D0%B5%D0%B1-%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA). Дата доступа: 21.05.2018.
3. Docker — Википедия [Электронный ресурс] / Википедия — свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Docker. Дата доступа: 21.05.2018.
4. Dlib – Wikipedia [Electronic resource] / Wikipedia, the free encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Dlib. – Date of access: 21.05.2018.
5. JavaScript — Википедия [Электронный ресурс] / Википедия — свободная энциклопедя. – Режим доступа: https://ru.wikipedia.org/wiki/JavaScript. Дата доступа: 21.05.2018.
6. TypeScript — Википедия [Электронный ресурс] / Википедия — свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/TypeScript. Дата доступа: 25.11.2018.
7. Node.js — Википедия [Электронный ресурс] / Википедия — свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Node.js. Дата доступа: 25.11.2018.
8. Node.js — Википедия [Электронный ресурс] / Википедия — свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Express.js. Дата доступа: 25.11.2018.
9. GitHub - babel/babel: Babel is a compiler for writing next generation JavaScript. [Electronic resource] / The world’s leading software development platform · GitHub – Mode of access: https://github.com/babel/babel. Date of access: 21.05.2018.
10. MongoDB — Википедия [Электронный ресурс] / Википедия — свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Express.js. Дата доступа: 25.11.2018
11. Webpack – Wikipedia [Electronic resource] / Wikipedia, the free encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Webpack. Date of access: 21.05.2018.
12. React (JavaScript library) – Wikipedia [Electronic resource] / Wikipedia, the free encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/React\_(JavaScript\_library). Date of access: 21.05.2018.
13. dlib.net/face\_detector.py.html [Electronic resource] / dlib C++ Library. Mode of access: http://dlib.net/face\_detector.py.html. Date of access: 21.05.2018.
14. Decision tree learning – Wikipedia [Electronic resource] / Wikipedia, the free encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Decision\_tree\_learning. Date of access: 21.05.2018.
15. dlib.net/face\_landmark\_detection.py.html [Electronic resource] / dlib C++ Library. Mode of access: http://dlib.net/face\_landmark\_detection.py.html. Date of access: 21.05.2018.
16. Residual neural network – Wikipedia [Electronic resource] / Wikipedia, the free encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Residual\_neural\_network. Date of access: 21.05.2018.
17. dlib C++ Library: High Quality Face Recognition with Deep Metric [Electronic resource] / dlib C++ Library. Mode of access: http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html. Date of access: 21.05.2018.

# 

# Приложение

## Исходный код сервиса по распознаванию лиц

#### init.sh

#!/bin/bash

docker build -t fr .

#### start.sh

#!/bin/bash

docker run -v $(pwd):/root/app/ -p 5001:5001 fr

#### requirements.txt

opencv-python==3.3.0.10

opencv-contrib-python==3.4.0.12

boost==0.1

dlib==19.8.2

face-recognition==1.0.0

flask==0.12.2

#### Dockerfile

FROM python:3.4-slim

RUN apt-get -y update

RUN apt-get install -y --fix-missing \

build-essential \

cmake \

gfortran \

git \

wget \

curl \

graphicsmagick \

libgraphicsmagick1-dev \

libatlas-dev \

libavcodec-dev \

libavformat-dev \

libboost-all-dev \

libgtk2.0-dev \

libjpeg-dev \

liblapack-dev \

libswscale-dev \

pkg-config \

python3-dev \

python3-numpy \

software-properties-common \

zip \

&& apt-get clean && rm -rf /tmp/\* /var/tmp/\*

RUN cd ~ && \

mkdir -p dlib && \

git clone -b 'v19.7' --single-branch https://github.com/davisking/dlib.git dlib/ && \

cd dlib/ && \

python3 setup.py install --yes USE\_AVX\_INSTRUCTIONS

COPY . /root/app

RUN cd /root/app && \

pip3 install -r requirements.txt

EXPOSE 5001

CMD cd /root/app/ && python3 main.py

#### .dockerignore

init.sh

start.sh

#### main.py

from flask import Flask, request

import json

from PIL import Image

import numpy as np

import dlib

from pkg\_resources import resource\_filename

# You can change this to any folder on your system

ALLOWED\_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}

def main():

app = Flask(\_\_name\_\_)

no\_file\_uploaded\_response = 'No file uploaded'

@app.route('/encode', methods=['POST'])

def encode():

# Check if a valid image file was uploaded

# TODO: replace no\_file\_uploaded\_response with HTTP error

if 'file' not in request.files:

return no\_file\_uploaded\_response

file = request.files['file']

if file.filename == '':

return no\_file\_uploaded\_response

if file and allowed\_file(file.filename):

# The image file seems valid! Detect faces and return the result.

return detect\_faces\_in\_image(file)

# If no valid image file was uploaded, go to index:

return no\_file\_uploaded\_response

@app.route('/recognize', methods=['POST'])

def recognize():

data = json.loads(request.get\_json())

storage = data['storage']

if not storage:

return None

tolerance = data['tolerance']

encoding = data['encoding']

for item in storage:

if (compare\_faces(item['encoding'], encoding, tolerance)):

return item['name']

return 'json.dumps(ex)'

app.run(host='0.0.0.0', port=5001, debug=True)

def detect\_faces\_in\_image(file\_stream):

no\_faces\_recognized\_response = 'No faces detected'

# Load the uploaded image file

img = np.array(Image.open(file\_stream).convert('RGB'))

try:

# Get face encodings for any faces in the uploaded image

result = recognize\_face(img)

return result

except IndexError:

return no\_faces\_recognized\_response

def recognize\_face(face\_image):

'''Face recognizer function'''

# unknown\_face = fr.face\_encodings(face\_image)[0]

num\_jitters = 1

face\_recognition\_model = face\_recognition\_model\_location()

face\_encoder = dlib.face\_recognition\_model\_v1(face\_recognition\_model)

face\_rect = detect\_face(face\_image)

if not face\_rect:

raise IndexError # TODO Implement a custom exception type

landmarks = detect\_landmarks(face\_image, face\_rect)

unknown\_face = np.array(face\_encoder.compute\_face\_descriptor(

face\_image, landmarks, num\_jitters))

return json.dumps(unknown\_face.tolist())

def detect\_landmarks(image, face\_location):

"""

Detects face landmarks im a passed image and returns a dlib rectangle object

:param image: numpy array that represent an image

:param face\_location: dlib rectangle that represents the face location

:return: dlib full\_object\_detection object that represents face landmarks

"""

model = pose\_predictor\_model\_location()

predictor = dlib.shape\_predictor(model)

return predictor(image, face\_location)

def detect\_face(image):

"""

Detects face im a passed image and returns a dlib rectangle object

:param image: an image a face on it

:return: a dlib 'rect' object

"""

face\_detector = dlib.get\_frontal\_face\_detector()

face\_rects = face\_detector(image)

if len(face\_rects) is 0:

print("no face found")

return np.empty((0))

return face\_rects[0]

def compare\_faces(encoding1, encoding2, tolerance):

if len(encoding1) is not len(encoding2):

raise Exception('Encodings length are not compatible')

distance = 0

length = len(encoding1)

for i in range(length):

distance = distance + (encoding1[i] - encoding2[i]) \*\* 2

return distance < tolerance

def face\_recognition\_model\_location():

return resource\_filename(\_\_name\_\_, "models/dlib\_face\_recognition\_resnet\_model\_v1.dat")

def pose\_predictor\_model\_location():

return resource\_filename(\_\_name\_\_, "models/shape\_predictor\_68\_face\_landmarks.dat")

def allowed\_file(filename):

return '.' in filename and \

filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS

if \_\_name\_\_ == "\_\_main\_\_":

main()

## Исходный код сервера приложения

#### package.json

{

"name": "fr-mediator",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"build": "tsc",

"start": "nodemon lib/index.js",

"test": "echo \"Error: no test specified\" && exit 1",

"watch": "npm run build -- -w"

},

"author": "",

"license": "ISC",

"dependencies": {

"body-parser": "^1.18.3",

"dotenv": "^6.1.0",

"express": "^4.16.3",

"mongodb": "^3.1.9",

"mongoose": "^5.3.13",

"morgan": "^1.9.1",

"multer": "^1.4.1",

"nodemon": "^1.18.4",

"request": "^2.88.0",

"request-promise-native": "^1.0.5",

"serialport": "^7.0.2"

},

"devDependencies": {

"@types/node": "^10.12.5",

"typescript": "^3.1.6"

}

}

#### tsconfig.json

{

"compilerOptions": {

"target": "es6",

"module": "commonjs",

"outDir": "lib",

"strictNullChecks": true

},

"files": [

"./node\_modules/@types/node/index.d.ts"

],

"include": [

"src/\*\*/\*.ts"

],

"exclude": [

"node\_modules"

]

}

#### src/api.ts

import { Face, Encoding, Name, FacesStorage } from './core';

import { FaceModel } from './db';

const request = require('request');

const requestPromise = require('request-promise-native');

const { RECOGNIZER\_URL } = process.env;

export function registerFace(face: Face): Promise<void> {

return FaceModel.create(face);

}

export function encode(file, callback) {

console.log(`${RECOGNIZER\_URL}/encode`);

const options = {

url: `${RECOGNIZER\_URL}/encode`,

method: 'POST',

headers: {

'Content-Type': 'multipart/form-data',

accept: 'application/json',

},

};

/\* body: encoding \*/

const r = request.post(options, callback);

const form = r.form();

form.append('file', file.buffer,

{ contentType: 'image/png', filename: file.originalname });

}

export function compare(encoding: Encoding, storage: FacesStorage, tolerance: number): Promise<any> {

const options = {

url: `${RECOGNIZER\_URL}/recognize`,

method: 'POST',

headers: {

accept: 'application/json',

},

body: JSON.stringify({

encoding,

storage,

tolerance,

}),

json: true,

};

return requestPromise.post(options);

}

export async function recognize(encoding: Encoding, tolerance: number = 0.1): Promise<Name | null> {

try {

const storage: FacesStorage = await FaceModel.find({}).exec();

const storageLength = storage.length;

if (storageLength === 0) {

return null;

}

return compare(encoding, storage, tolerance);

} catch (e) {

console.error('ERROR', e);

throw (e);

}

}

#### src/core.ts

export type Name = string;

export type Encoding = number[];

export type Face = {

name: Name;

encoding: Encoding;

};

export type FacesStorage = Face[];

export enum ApiResponse {

NoFileUploaded = 'No file uploaded',

NoFacesDetected = 'No faces detected',

FaceNotRecognized = 'Face is not recognized',

FaceRecognized = 'Face is recognized',

InvalidFaceReceived = 'Invalid face received',

FaceRegistered = 'Face is registered',

}

#### src/db.ts

import \* as Mongoose from 'mongoose';

import { Face } from './core';

const {

DB\_USER,

DB\_PASS,

DB\_URL,

} = process.env;

Mongoose.connect(

`mongodb://${DB\_USER}:${DB\_PASS}@${DB\_URL}`,

{ useNewUrlParser: true },

(err) => {

if (err) {

console.error('Error while connecting to the DB');

console.error(err);

return;

}

console.log('Connected to the DB');

});

Mongoose.set('debug', (collectionName, method, query, doc) => {

console.log(`${collectionName}.${method}`, JSON.stringify(query), doc);

});

const FaceSchema = {

name: String,

encoding: [Number],

}

export const FaceModel: Mongoose.Model<Face> = Mongoose.model('Face', FaceSchema);

#### src/index.ts

const express = require('express');

const morgan = require('morgan');

const bodyParser = require('body-parser');

require('dotenv').config();

import router from './router';

const PORT = 8081;

const HOST = '0.0.0.0';

const app = express();

app.use(morgan('combined'));

app.use(bodyParser.json());

app.use(router);

app.listen(PORT, HOST);

console.log(`Running on http://${HOST}:${PORT}`);

#### src/router.ts

const express = require('express');

const multer = require('multer');

import { ApiResponse, Encoding, Face } from './core';

import { recognize, registerFace, encode } from './api';

import { listSerialPorts } from './serialport';

const router = express.Router();

const upload = multer();

/\* formData \*/

router.post('/recognize', upload.any(), function (req, res) {

/\*\*

\* Proxies request to a recognizer

\*/

if (!req.files) {

res.send({ message: ApiResponse.NoFileUploaded });

}

const file = req.files[0];

if (!file) {

res.send({ message: ApiResponse.NoFileUploaded });

}

if (!isFileAllowed(file.originalname)) {

res.send({ message: ApiResponse.NoFileUploaded });

}

encode(file, async (error, proxyRes, body) => {

if (error) {

console.log(error);

console.error(ApiResponse.NoFacesDetected);

res.send({ message: ApiResponse.NoFacesDetected });

return;

}

if (body === ApiResponse.NoFacesDetected) {

console.error(ApiResponse.NoFacesDetected);

res.send({ message: ApiResponse.NoFacesDetected });

return;

}

try {

const encoding: Encoding = JSON.parse(body);

const name = await recognize(encoding);

if (name) {

res.send({

name,

message: ApiResponse.FaceRecognized,

});

} else {

console.error(ApiResponse.FaceNotRecognized);

res.send({

encoding,

message: ApiResponse.FaceNotRecognized,

});

}

} catch (e) {

console.log(body);

console.error(e);

res.status(500).send(e, body);

}

});

});

/\* { name, encoding } \*/

router.post('/register', function (req, res) {

const face: Face = req.body;

if (face.name == null || !Array.isArray(face.encoding)) {

res.status(400).send({ message: ApiResponse.InvalidFaceReceived });

return;

}

registerFace(face);

res.send({ message: ApiResponse.FaceRegistered });

});

router.get('/serialports', async function (req, res) {

const ports = await listSerialPorts();

res.send(ports);

})

function isFileAllowed(fileName) {

const allowedExtensions = ['png', 'jpg', 'jpeg', 'gif'];

return (

fileName.includes('.')

&&

allowedExtensions.indexOf(fileName.split('.')[1].toLowerCase()) !== -1

);

}

export default router;

#### src/serialport.ts

const SerialPort = require('serialport');

export const listSerialPorts = () => new Promise((resolve, reject) => {

SerialPort.list(function (err, ports) {

if (err) {

reject(err);

}

resolve(ports);

});

});

## Исходный код веб-клиента

#### package.json

{

"name": "fr-client",

"version": "0.1.0",

"private": true,

"proxy": "http://localhost:8081/",

"dependencies": {

"antd": "^3.10.8",

"axios": "^0.17.1",

"classnames": "^2.2.6",

"prop-types": "^15.6.2",

"react": "^16.6.3",

"react-dom": "^16.6.3",

"react-json-view": "^1.19.1",

"react-jss": "^8.6.1",

"react-router-dom": "^4.3.1",

"react-scripts": "1.1.0"

},

"scripts": {

"start": "react-scripts start",

"build": "react-scripts build",

"test": "react-scripts test --env=jsdom",

"eject": "react-scripts eject"

}

}

#### src/index.js

import React from 'react';

import ReactDOM from 'react-dom';

import { App } from './components';

import registerServiceWorker from './registerServiceWorker';

import 'antd/dist/antd.css';

ReactDOM.render(<App />, document.getElementById('root'));

registerServiceWorker();

#### src/components/index.js

export { default as App } from './App';

#### src/components/App.js

import React, { Component } from 'react';

import injectSheet from 'react-jss';

import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';

import Authorize from '../modules/Authentication/components/Authorize';

import Home from './Home';

import { AuthenticationProvider } from '../modules/Authentication'

import { ProtectedRoute } from '../modules/Authentication';

function App() {

return (

<AuthenticationProvider>

<Router>

<Switch>

<ProtectedRoute path="/" exact component={Home} />

<Route path="/authenticate/" component={Authorize} />

</Switch>

</Router>

</AuthenticationProvider>

);

};

export default App;

#### src/components/Home.js

import { Layout, Menu } from 'antd';

import React from 'react';

import injectSheet from 'react-jss';

import {

withAuthentication,

withAuthenticationPropTypes,

} from '../modules/Authentication';

import logo50 from '../images/logo50.png';

import { compose } from '../modules/util';

import { constants } from '../config';

import { SerialPortsList } from './SerialPortsList';

const { Header, Content, Footer } = Layout;

const styles = {

img: {

height: '50px',

width: '50px',

margin: '6px',

},

header: {

display: 'flex',

justifyContent: 'space-between',

},

menu: {

lineHeight: '64px',

},

content: {

padding: '50px',

backgroundColor: '#fff',

}

};

function Home(props) {

const { classes } = props;

const onMenuClick = ({ key }) => {

if (key === 'logout') {

props.authentication.setName(undefined);

}

}

return (

<Layout>

<Header className={classes.header}>

<img src={logo50} className={classes.img} alt="logo" />

<Menu

theme="dark"

mode="horizontal"

className={classes.menu}

onClick={onMenuClick}

>

<Menu.Item key="logout">{constants.LOG\_OUT}</Menu.Item>

</Menu>

</Header>

<Content className={classes.content}>

<SerialPortsList />

</Content>

<Footer />

</Layout>

);

}

Home.proptypes = {

...withAuthenticationPropTypes,

}

export default compose(

withAuthentication,

injectSheet(styles),

)(Home);

#### src/components/SerialPortsList.js

import React from 'react';

import ReactJson from 'react-json-view';

import { api, notification } from '../services';

import { constants } from '../config';

export class SerialPortsList extends React.Component {

state = {

list: [],

};

async componentDidMount() {

try {

const listResponse = await api.getSerialPortsList();

this.setState({ list: listResponse.data })

} catch (e) {

console.error(e);

notification.error(constants.FAILED\_FETCHING\_SERIAL\_PORTS);

}

}

render() {

const { list } = this.state;

return (

<React.Fragment>

<h1>{constants.SERIAL\_PORTS}</h1>

<ReactJson src={list} />

</React.Fragment>

)

}

}

#### src/config/index.js

import \* as constants from './constants';

export { constants };

#### src/config/constants.js

export const FACE\_NOT\_RECOGNIZED\_RESPONSE = 'Face is not recognized';

export const FACE\_RECOGNIZED\_RESPONSE = 'Face is recognized';

export const nameIsRegistered = (name) => `${name} is registered`;

export const CAPTURE = 'Capture';

export const TRY\_AGAIN = 'Try Again';

export const REGISTER\_FACE = 'Register Face';

export const PLEASE\_INPUT\_YOUR\_NAME = 'Please input your name!';

export const NAME = 'Name!';

export const FAILED\_FETCHING\_SERIAL\_PORTS = 'Failed fetching serial ports';

export const SERIAL\_PORTS = 'Serial Ports';

export const LOG\_OUT = 'Log out';

#### src/services/index.js

import \* as api from './api';

import \* as notification from './notification';

export { api, storage, notification };

#### src/services/api.js

import axios from 'axios';

export const recognize = (data /\* formData \*/) => axios({

data,

method: 'post',

url: '/recognize',

config: { headers: { 'Content-Type': 'multipart/form-data' } },

});

export const register = (data /\* { name, encoding } \*/) => axios({

data,

method: 'post',

url: '/register',

});

export const getSerialPortsList = () => axios('/serialports');

#### src/services/notification.js

import { message } from 'antd';

export const error = (text) => message.error(text);

export const success = (text) => message.success(text);

#### modules/util/compose.js

// https://github.com/reduxjs/redux/blob/master/src/compose.js

/\*\*

\* Composes single-argument functions from right to left. The rightmost

\* function can take multiple arguments as it provides the signature for

\* the resulting composite function.

\*

\* @param {...Function} funcs The functions to compose.

\* @returns {Function} A function obtained by composing the argument functions

\* from right to left. For example, compose(f, g, h) is identical to doing

\* (...args) => f(g(h(...args))).

\*/

export default function compose(...funcs) {

if (funcs.length === 0) {

return arg => arg

}

if (funcs.length === 1) {

return funcs[0]

}

return funcs.reduce((a, b) => (...args) => a(b(...args)))

}

#### module/util/index.js

export { default as compose } from './compose';

#### module/Authentication/index.js

export {

AuthenticationProvider,

withAuthentication,

withAuthenticationPropTypes,

} from './AuthenticationProvider';

export { default as ProtectedRoute } from './ProtectedRoute';

#### module/Authentication/AuthenticationProvider.js

import React from 'react';

import PropTypes from 'prop-types';

const AuthenticationContext = React.createContext();

export class AuthenticationProvider extends React.Component {

constructor(props) {

super(props);

const localStorageName = localStorage.getItem('name');

const localStorageNameIsDefined = localStorageName && localStorageName !== 'undefined';

const name = localStorageNameIsDefined ? localStorageName : undefined;

this.state = {

name,

};

}

setName = (name) => {

this.setState({ name });

localStorage.setItem('name', name);

}

render() {

const providerValue = {

name: this.state.name,

setName: this.setName,

};

return (

<AuthenticationContext.Provider value={providerValue}>

{this.props.children}

</AuthenticationContext.Provider>

);

}

}

export const withAuthentication = (Component) => {

function WithAuthenticationHoc(props) {

return (

<AuthenticationContext.Consumer>

{(context) => <Component {...props} authentication={context} />}

</AuthenticationContext.Consumer>

);

}

WithAuthenticationHoc.displayName = `WithAuthentication(${Component.displayName || Component.name})`

return WithAuthenticationHoc;

};

export const withAuthenticationPropTypes = {

authentication: PropTypes.shape({

name: PropTypes.string,

setName: PropTypes.func.isRequired,

}).isRequired,

};

#### module/Authentication/ProtectedRoute.js

import React from 'react';

import { Redirect, Route } from 'react-router-dom';

import { withAuthentication } from './AuthenticationProvider';

function PrivateRoute(

{

component: Component,

authentication,

...rest

}) {

return (

<Route

{...rest}

render={(props) =>

authentication.name ? (

<Component {...props} />

) : (

<Redirect

to={{

pathname: "/authenticate",

state: { from: props.location }

}}

/>

)

}

/>

);

}

export default withAuthentication(PrivateRoute);

#### module/Authentication/components/Authorize.js

import React, { Component } from 'react';

import injectSheet from 'react-jss';

import { Redirect } from 'react-router-dom';

import PropTypes from 'prop-types';

import Video from './Video';

import RegisterFaceForm from './RegisterFaceForm';

import {

withAuthentication,

withAuthenticationPropTypes,

} from '..';

import { compose } from '../../util';

const styles = {

result: {

width: '640px',

margin: '8px auto',

textAlign: 'center',

},

}

class Authorize extends Component {

static propTypes = {

...withAuthenticationPropTypes,

classes: PropTypes.objectOf(PropTypes.string).isRequired,

};

state = {

result: '',

unrecognizedFaceEncoding: [],

modalVisible: false,

confirmLoading: false,

};

openRegisterModal = (unrecognizedFaceEncoding) => this.setState({

unrecognizedFaceEncoding,

modalVisible: true,

});

closeRegisterModal = () => this.setState({

unrecognizedFaceEncoding: [],

modalVisible: false,

});

onVideoResult = (result) => this.setState({ result });

render() {

const { classes } = this.props;

const {

modalVisible,

confirmLoading,

} = this.state;

if (this.props.authentication.name) {

return (

<Redirect to="/" />

);

}

return (

<React.Fragment>

<Video

onRegisterClick={this.openRegisterModal}

onResult={this.onVideoResult}

/>

<div className={classes.result}>

{this.state.result}

</div>

<RegisterFaceForm

modalVisible={modalVisible}

confirmLoading={confirmLoading}

closeModal={this.closeRegisterModal}

unrecognizedFaceEncoding={this.state.unrecognizedFaceEncoding}

/>

</React.Fragment>

);

}

}

export default compose(

injectSheet(styles),

withAuthentication,

)(Authorize);

#### module/Authentication/components/RegisterFaceForm.js

import React, { Component } from 'react';

import PropTypes from 'prop-types';

import { Form, Input, Modal } from 'antd';

import { constants } from '../../../config';

import { api, notification } from '../../../services';

import { compose } from '../../util';

import { withAuthentication, withAuthenticationPropTypes } from '../AuthenticationProvider';

const FormItem = Form.Item;

class RegisterFaceForm extends Component {

static propTypes = {

modalVisible: PropTypes.bool.isRequired,

confirmLoading: PropTypes.bool.isRequired,

closeModal: PropTypes.func.isRequired,

unrecognizedFaceEncoding: PropTypes.arrayOf(PropTypes.number).isRequired,

...withAuthenticationPropTypes,

};

setFormRef = (element) => {

this.form = element;

};

handleRegisterOk = (event, values) => {

this.props.form.validateFields(async (err, values) => {

if (err) {

return;

}

this.setState({

confirmLoading: true,

});

const { name } = values;

const data = {

name,

encoding: this.props.unrecognizedFaceEncoding,

};

try {

await api.register(data);

notification.success(constants.nameIsRegistered(name));

this.props.authentication.setName(name);

} catch (e) {

console.error(e.response);

notification.error(e.message);

this.props.closeModal();

}

});

}

render() {

const { getFieldDecorator } = this.props.form;

const { modalVisible, confirmLoading, closeModal } = this.props;

return (

<Modal title={constants.REGISTER\_FACE}

visible={modalVisible}

onOk={this.handleRegisterOk}

confirmLoading={confirmLoading}

onCancel={closeModal}

>

<Form>

<FormItem>

{getFieldDecorator('name', {

rules: [{ required: true, message: constants.PLEASE\_INPUT\_YOUR\_NAME }],

})(

<Input placeholder={constants.NAME} />

)}

</FormItem>

</Form>

</Modal>

);

}

}

export default compose(

Form.create(),

withAuthentication,

)(RegisterFaceForm);

#### module/Authentication/components/Video.js

import React, { Component } from 'react';

import injectSheet from 'react-jss';

import classNames from 'classnames';

import PropTypes from 'prop-types';

import VideoControls from './VideoControls';

import { constants } from '../../../config';

import { api, notification } from '../../../services';

import {

withAuthentication,

withAuthenticationPropTypes,

} from '../AuthenticationProvider';

import { compose } from '../../util';

const styles = {

image: {

width: '640px',

height: '480px',

margin: '8px auto',

},

video: {

height: '100%',

width: '100%',

},

canvas: {

height: '100%',

width: '100%',

},

hidden: {

display: 'none',

},

};

class Video extends Component {

static propTypes = {

...withAuthenticationPropTypes,

classes: PropTypes.objectOf(PropTypes.string).isRequired,

onRegisterClick: PropTypes.func.isRequired,

onResult: PropTypes.func.isRequired,

};

state = {

result: '',

unrecognizedFaceEncoding: [],

videoHidden: false,

canvasHidden: true,

captureHidden: false,

againHidden: true,

registerHidden: true,

spinHidden: true,

confirmLoading: false,

};

async componentDidMount() {

try {

this.video.srcObject = await navigator.mediaDevices.getUserMedia({ video: true });

} catch (e) {

console.error(e);

}

}

setVideoRef = (element) => {

this.video = element;

};

setCanvasRef = (element) => {

this.canvas = element;

};

buildVideoClassNames = () => classNames({

[this.props.classes.video]: true,

[this.props.classes.hidden]: this.state.videoHidden,

});

buildCanvasClassNames = () => classNames({

[this.props.classes.canvas]: true,

[this.props.classes.hidden]: this.state.canvasHidden,

});

processRecognitionResponse = (response) => {

const faceNotRecognized = response.data.message === constants.FACE\_NOT\_RECOGNIZED\_RESPONSE;

const faceRecognized = response.data.message === constants.FACE\_RECOGNIZED\_RESPONSE;

const registerHidden = !faceNotRecognized;

const result = faceRecognized ? undefined : response.data.message;

const unrecognizedFaceEncoding = faceNotRecognized ? response.data.encoding : [];

if (faceRecognized) {

this.props.authentication.setName(response.data.name);

return;

}

return { registerHidden, result, unrecognizedFaceEncoding };

}

handleCaptureClick = () => {

this.setState({

videoHidden: true,

canvasHidden: false,

captureHidden: true,

spinHidden: false,

});

// Draw the video frame to the canvas.

const context = this.canvas.getContext('2d');

context.drawImage(this.video, 0, 0, this.canvas.width, this.canvas.height);

this.canvas.toBlob(async (blob) => {

const formData = new FormData();

formData.append('file', blob, 'selfie.jpg');

try {

const response = await api.recognize(formData)

const prosessedResponse = this.processRecognitionResponse(response);

if (prosessedResponse) {

this.props.onResult(prosessedResponse.result);

this.setState({

...prosessedResponse,

spinHidden: true,

againHidden: false,

});

}

} catch (e) {

console.error(e);

notification.error(e.message);

this.setState({

spinHidden: true,

againHidden: false,

});

}

});

};

handleTryAgainClick = () => {

this.setState({

result: '',

videoHidden: false,

canvasHidden: true,

captureHidden: false,

againHidden: true,

registerHidden: true,

unrecognizedFaceEncoding: [],

});

};

handleRegisterClick = () => {

this.props.onRegisterClick(this.state.unrecognizedFaceEncoding);

};

render() {

const { classes } = this.props;

const {

captureHidden,

againHidden,

registerHidden,

spinHidden

} = this.state;

return (

<React.Fragment>

<div className={classes.image}>

<video

className={this.buildVideoClassNames()}

ref={this.setVideoRef}

autoPlay

/>

<canvas

className={this.buildCanvasClassNames()}

ref={this.setCanvasRef}

/>

</div>

<VideoControls

handleCaptureClick={this.handleCaptureClick}

handleTryAgainClick={this.handleTryAgainClick}

handleRegisterClick={this.handleRegisterClick}

captureHidden={captureHidden}

againHidden={againHidden}

registerHidden={registerHidden}

spinHidden={spinHidden}

/>

</React.Fragment>

);

}

};

export default compose(

injectSheet(styles),

withAuthentication,

)(Video);

#### module/Authentication/components/VideoControls.js

import React from 'react';

import { Button, Spin } from 'antd';

import classNames from 'classnames';

import injectSheet from 'react-jss';

import { constants } from '../../../config';

const styles = {

actionButton: {

margin: '0 4px',

},

hidden: {

display: 'none',

},

actions: {

width: '640px',

margin: '8px auto',

textAlign: 'center',

},

}

const VideoControls = (

{

classes,

handleCaptureClick,

handleTryAgainClick,

handleRegisterClick,

captureHidden,

againHidden,

registerHidden,

spinHidden,

}

) => {

const buildCaptureButtonClassNames = () => classNames({

[classes.actionButton]: true,

[classes.hidden]: captureHidden,

});

const buildTryAgainButtonClassNames = () => classNames({

[classes.actionButton]: true,

[classes.hidden]: againHidden,

});

const buildRegisterButtonClassNames = () => classNames({

[classes.actionButton]: true,

[classes.hidden]: registerHidden,

});

const buildSpinClassNames = () => classNames({

[classes.hidden]: spinHidden,

});

return (

<div className={classes.actions}>

<Button

type="primary"

icon="search"

className={buildCaptureButtonClassNames()}

onClick={handleCaptureClick}

>

{constants.CAPTURE}

</Button>

<Button

type="primary"

icon="reload"

className={buildTryAgainButtonClassNames()}

onClick={handleTryAgainClick}

>

{constants.TRY\_AGAIN}

</Button>

<Button

type="primary"

icon="cloud-upload-o"

className={buildRegisterButtonClassNames()}

onClick={handleRegisterClick}

>

{constants.REGISTER\_FACE}

</Button>

<Spin className={buildSpinClassNames()} />

</div>

);

};

export default injectSheet(styles)(VideoControls);

#### public/index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="theme-color" content="#000000">

*<!--*

*manifest.json provides metadata used when your web app is added to the*

*homescreen on Android. See https://developers.google.com/web/fundamentals/engage-and-retain/web-app-manifest/*

*-->*

<link rel="manifest" href="%PUBLIC\_URL%/manifest.json">

<link rel="shortcut icon" href="%PUBLIC\_URL%/favicon.ico">

*<!--*

*Notice the use of %PUBLIC\_URL% in the tags above.*

*It will be replaced with the URL of the `public` folder during the build.*

*Only files inside the `public` folder can be referenced from the HTML.*

*Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will*

*work correctly both with client-side routing and a non-root public URL.*

*Learn how to configure a non-root public URL by running `npm run build`.*

*-->*

<title>React App</title>

</head>

<body>

<noscript>

You need to enable JavaScript to run this app.

</noscript>

<div id="root"></div>

</body>