СОДЕРЖАНИЕ

Введение....3
Основная часть....4
1 Постановка задачи автоматизации....4
2 Проектирование информационной системы....6
3 Разработка пользовательского интерфейса информационной системы....8
4 Анализ разработанной информационной системы, выявление достоинств и недостатков 14
5 Разработка методики внедрения и сопровождения информационной системы....15
Заключение....18
Список использованных источников....19
Приложение А. Диаграмма основного бизнес-процесса....20
Приложение Б. Диаграмма связей сущностей....21
Приложение В. Словарь данных....22
Приложение Г. Руководство пользователя....23
Приложение Д. Руководство пользователя....31
Приложение Ж. Листинги кода программного продукта....38
Приложение К. Программный продукт....51

					ДП.011.09.02.07.000.ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		
Разраб.		Королёв			СОДЕРЖАНИЕ	Лит. / Лист 1 / Листов 1
Провер.		Пивоваров				
Н. Контр.		Новикова				ВПМТ 4ИС
Утверд.		Галимова				

ВВЕДЕНИЕ

Актуальность темы: поиск фотографий в медиатеке на которых запечатлен определённый студент может занять большое время, особенно если общее количество фотографий исчисляется тысячами. Проект стремится решить эту проблему, автоматизируя ручной процес.

Цель и задачи: разработать систему для поиска фотографий, на которых запечатлено лицо пользователя. Система должна иметь возможность сохранить результаты поиска удобным для пользователя способом.

Объект исследования: создание биометрического слепка по загруженной фотографии пользоваеля, поиск подобной биометрии по сформированной базе данных, формирование архива, сохранение результата на флеш-накопитель пользователя.

Предмет исследования: процесс распознавания и сравнения лиц, сохранение результата.

Структура работы:

1 Постановка задачи автоматизации;

2 Проектирование информационной системы;

2 Разработка пользовательского интерфейса информационной системы;

4 Анализ разработанной информационной системы, выявление достоинств и недостатков;

5 Разработка методики внедрения и сопровождения информационной системы.

ОСНОВНАЯ ЧАСТЬ

1 Постановка задачи автоматизации

1.1 Задачи системы:

- Обработка медиатеки техникума, поиск и идентификация лиц, найденных на фотографиях;
- Получение фото пользователя, распознавание полученного снимка;
- Формирование списка распознанных фотографий, известных системе, на которых запечатлен пользователь;
- Сохранение результата работы системы на флеш-накопитель.

1.2 Входные данные:

- Фотографии в медиатеке техникума;
- Фотография лица пользователя.

1.3 Выходные данные:

Файл архива в формате zip, содержащий фотографии, на которых запечатлен пользователь.

1.4 Возможные изменения информационных потребностей пользователей:

- Сбор не только фотографий, но и видео;
- Сохранение результата не только на внешний накопитель, но и отправка на электронную почту.

1.5 Алгоритм решения задачи:

1.5.1 Разработать структуру базы данных для сопоставления одного лица множеству и быстрой выборке файлов фотографий;

1.5.2 Разработать программу, работающую в фоновом режиме, которая обрабатывает фотографии, выполняет анонимную идентификацию всех найденных лиц и сохраняет информацию в базу данных;

1.5.3 Разработать графический интерфейс, предназначенный для взаимодействия пользователей с системой, позволяющий сфотографировать лицо пользователя, отправить его на обработку в систему и получить результат;

1.5.4 Разработать механизм, позволяющий пользователям сохранять результат работы системы.

С системой взаимодействует два типа пользователей — студенты, которые хотят получить результат работы и администратор, который может взаимодействовать с механизмом сканирования фотографий системы.

1.6 Похожие проекты и их сравнение

В ходе изучения предметной области были проанализированы похожие решения, выявлены достоинства и недостатки (таблица 1.1).

Таблица 1.1 — Описание похожих проектов

Проект	Достоинства	Недостатки
Сбор изображений студентов и сотрудников МИЭМ [1]	Интеграция с корпоративным мессенджером.	Не самостоятельное приложение, а адаптер mongodb.
Google Photos [2], Яндекс Диск [3]	Точное распознавание лиц, облачная платформа.	Ограничение по размеру для некоммерческой версии, нет автоматической компоновки результата.

2 Проектирование информационной системы

При проектировании системы были выделены два типа пользователей: студенты и администраторы, а так же определены функции, которые каждый из типов может использовать. Описание функций приведено в use-case диаграмме (рис. 2.1).

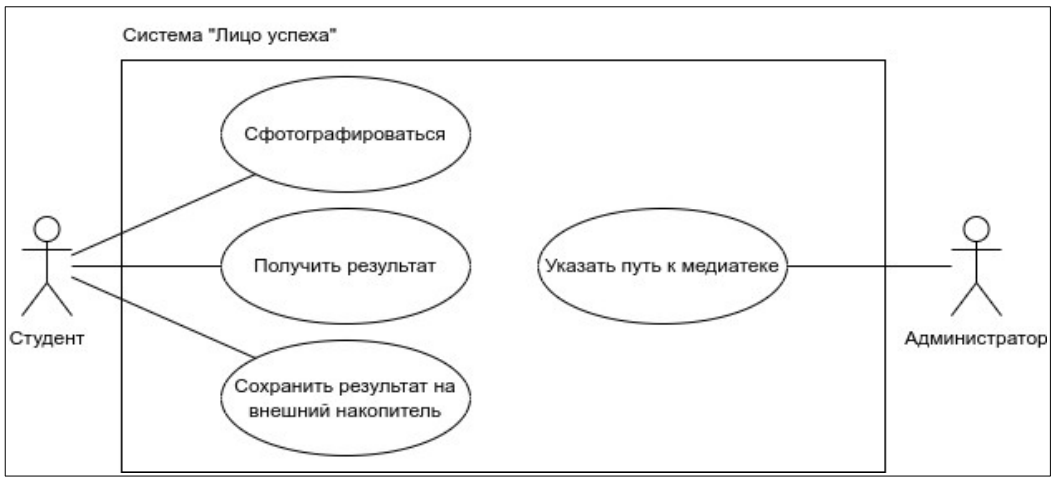

Рисунок 2.1 — use-case диаграмма системы

При проектировании также была разработана диаграмма, отображающая основной бизнес процесс проекта. Диаграмма выполнена в нотации IDEF0 и приведена в приложении А.

Для эффективной обработки лиц потребовалось разработать базу данных, хранящую в себе уже распознанные лица студентов и пути к изображениям. Это решение позволяет не обрабатывать заново тысячи фотографий медиатеки, а вместо этого сравнить полученное лицо с гораздо меньшим количеством данных. Разработана диаграмма связей сущностей, приведена в приложении Б.

Для пояснения структуры данных был разработан словарь данных (приложение В).

Проанализировав альтернативные решения, были выявлены следующие общие характеристики:

- Использование искусственного интеллекта для распознавания лиц;
- Исполнение в браузере.

Таблица 2.1 — Параметры среды разработки

Параметр	Значение
Распознавание лиц	Язык программирования python, библиотека face_recognition [4], так как python имеет обширные средства работы с искусственным интеллектом и обработкой изображений.
Пользовательский интерфейс	Язык программирования javascript, фреймворк vue.js [5], так как платформа браузера, на которой исполняется код, предоставляет много возможностей в плане разработки интерфейса.

3 Разработка пользовательского интерфейса информационной системы

Пользовательский интерфейс разработан с помощью веб-технологий, таких как:

- HTML;
- CSS;
- JavaScript.

Типичный сценарий взаимодействия пользователя с системой:

1 Подойти к устройству, на котором развёрнут интерфейс;

Пользователю будут представлены краткие инструкции по работе с системой и кнопка «Начать» (рис. 3.1). Кнопка «Нажать» проверит, подключено ли внешнее хранилище, и, в случае если хранилище не подключено, выдаст ошибку с соответствующим сообщением (рис. 3). Проверка выполняется с помощью отправки HTTP запроса на API, работающее на том же устройстве, что и общий интерфейс. API собирает все устройства, подключенные к системе, имеющие тип «disk», а так же имеющие атрибут «removable» с помощью библиотеки pyudev и функции list_devices [6]. Затем, выполняется поиск всех разделов на этом устройстве, вычисляется точка монтирования. Если какой-либо шаг в этой последовательности был провален, система выдаёт ошибку подключения хранилища (рис. 3.2). Пример кода, отвечающего за взаимодействие пользователя с системой приведены в листинга Ж.1 - Ж.3.

Рисунок 3.1 — Начальная страница системы

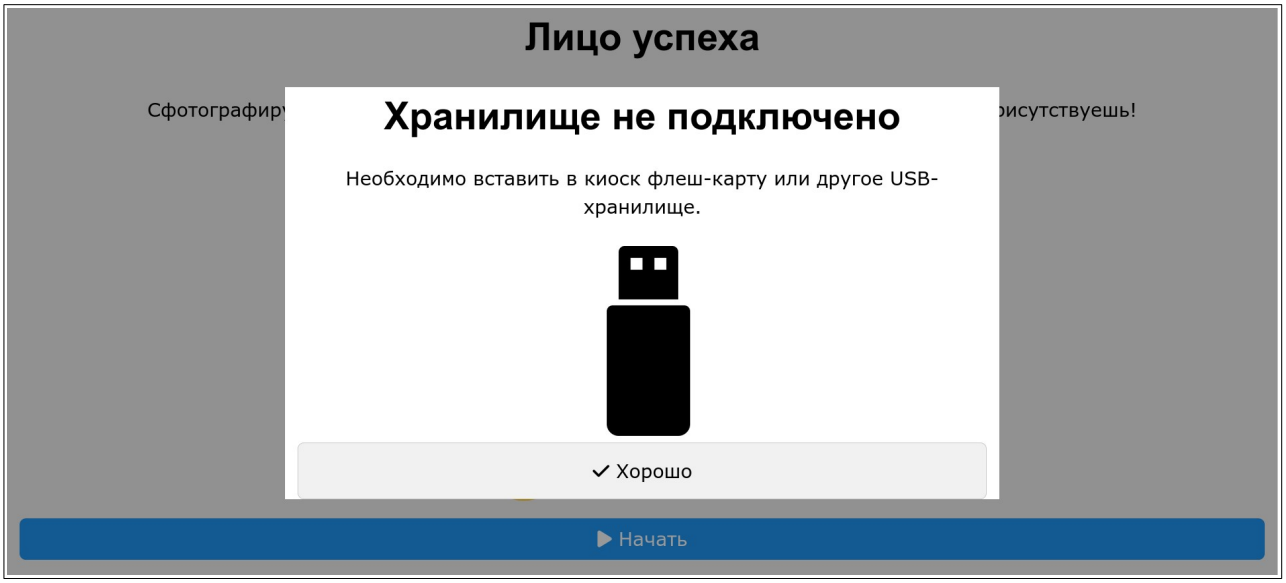

Рисунок 3.2 — Ошибка, сообщающая о том, что необходимо подключить внешнее хранилище

2 Подключить внешнее хранилище;

3 Сфотографироваться;

Система кодирует данные фотографии в base64, передаёт по сети в подсистему распознавания, где данные декодируются и загружаются в память. На странице отображён видеопоток для предпросмотра фотографии (рис. 3.3). Пример кода, управляющий работой с фотографиями приведён в листингах Ж.4-Ж.6.

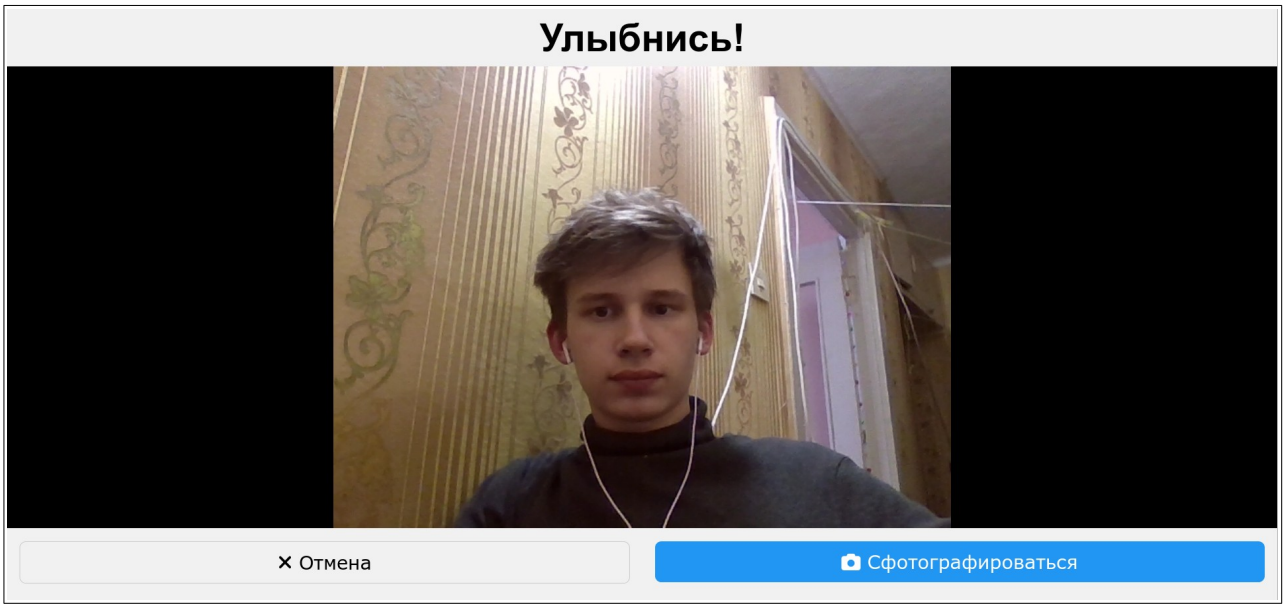

Рисунок 3.3 — Страница снятия фотографии

4 Ждать процесса распознавания;

Распознавание занимает меньше двух секунд при тестовой выборке из 1000 фотографий. Но, в случае если ожидание окажется слишком долгим, интерфейс системы будет отображать экран загрузки с одним случайно выбранным интересным фактом о техникуме (рис. 3.4).

Процесс распознавания заключается в расшифровке изображения, закодированного на предыдущем шаге. Изображение сохраняется во временный каталог системы, затем читается библиотекой face_recognition. После этого, библиотека ищет лица на фотографии. Если найдено больше или меньше одного лица, возвращается ошибка, в интерфейсе открывается экран с описанием ошибки (рис. 3.5). Экран позволяет перефотографироваться или вернуться на главный экран.

Система затем десериализует и загружает данные всех известных кодировок лиц из базы данных в память. Кодировка лиц представляет собой 128-размерный массив [7]. Массив сериализуется в последовательность байтов для сохранения в базе данных, и десериализуется обратно в массив при запросе пользователя.

Система проверяет кодировки лиц из базы данных с полученной, формирует выборку лиц, сортирует, выполняя поиск самого похожего лица. Далее, все фотографии, на которых это лицо было запечатлено, возвращаются в интерфейс для отображения и/или сохранения.

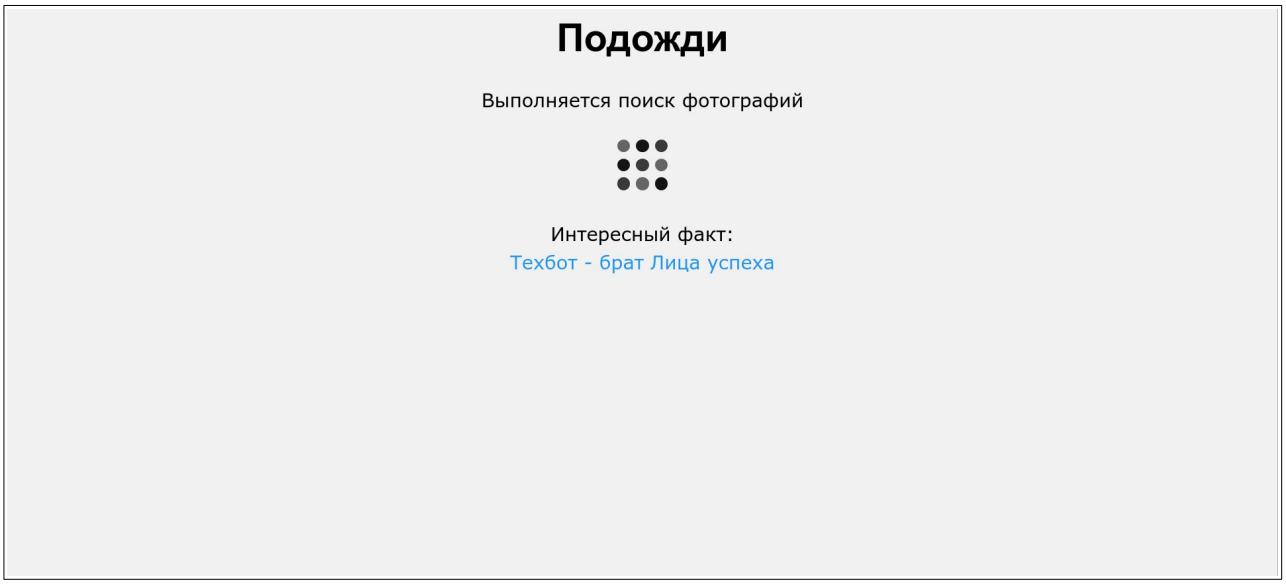

Рисунок 3.4 — Экран ожидания результатов

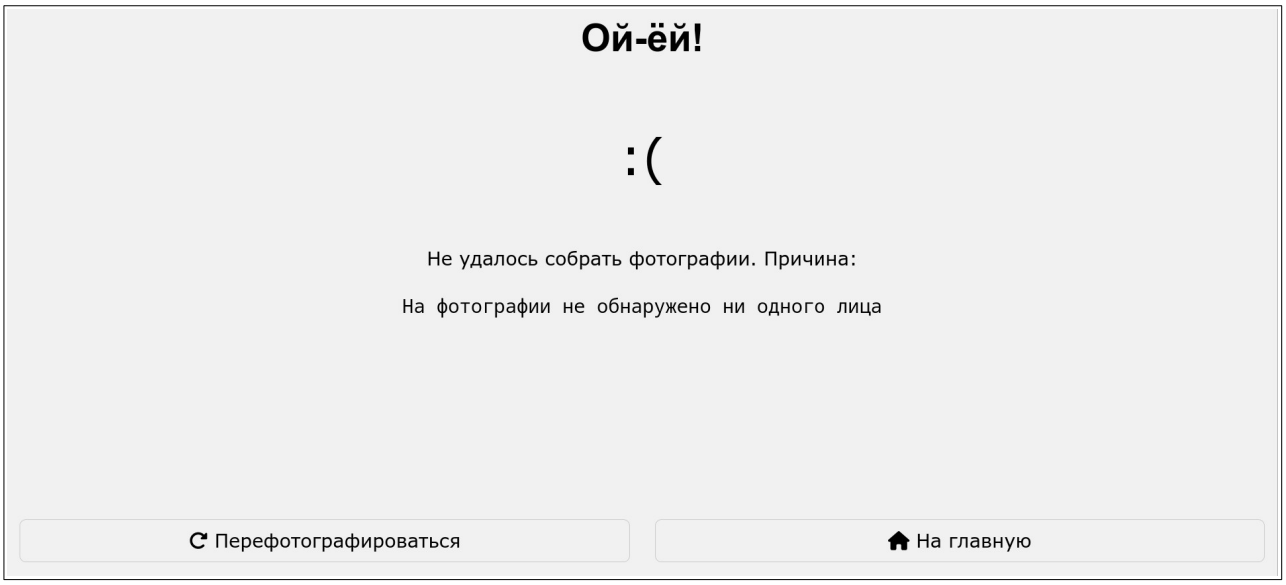

Рисунок 3.5 — Экран с описанием ошибки

5 Просмотреть результат сборки фотографий;

На этом экране система отображает количество фотографий, которое было найдено, а так же выборку из результата в размере не более 6 фотографий (рис. 3.6). На экране расположены две кнопки: «На главную» и «Сохранить результат». Код страницы просмотра результата приведён в листинге Ж.7.

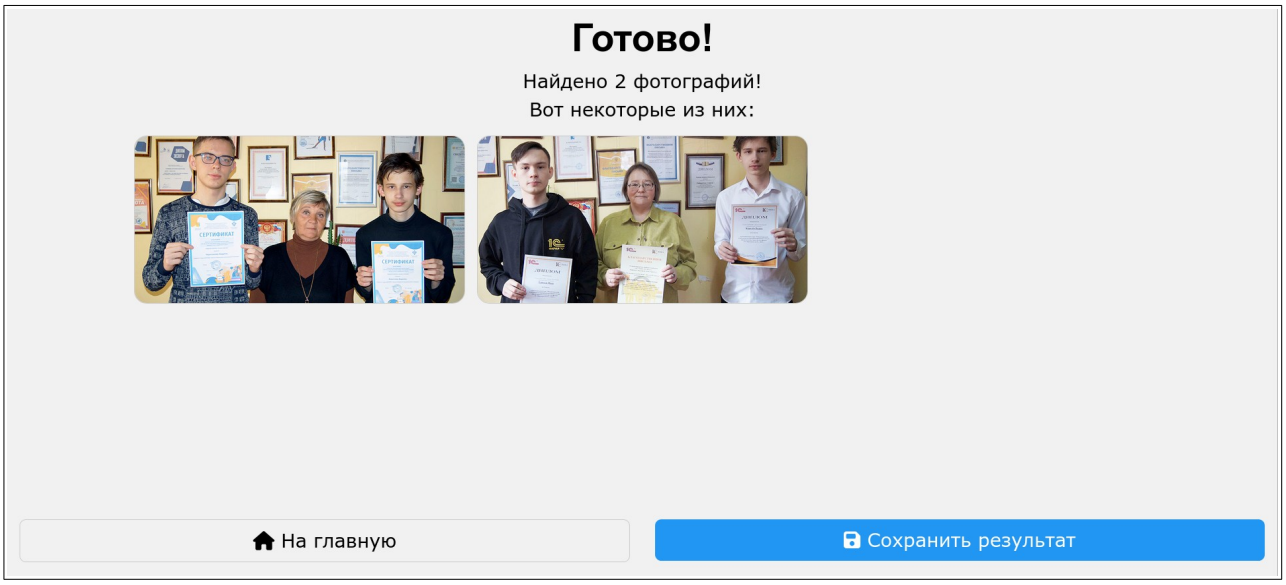

Рисунок 3.6 — Экран просмотра результата

6 Сохранить результат работы системы.

Перед сохранением пользователю выдаётся предупреждение с информацией о сохранении файла (рис. 3.7). Имя файла генерируется динамически.

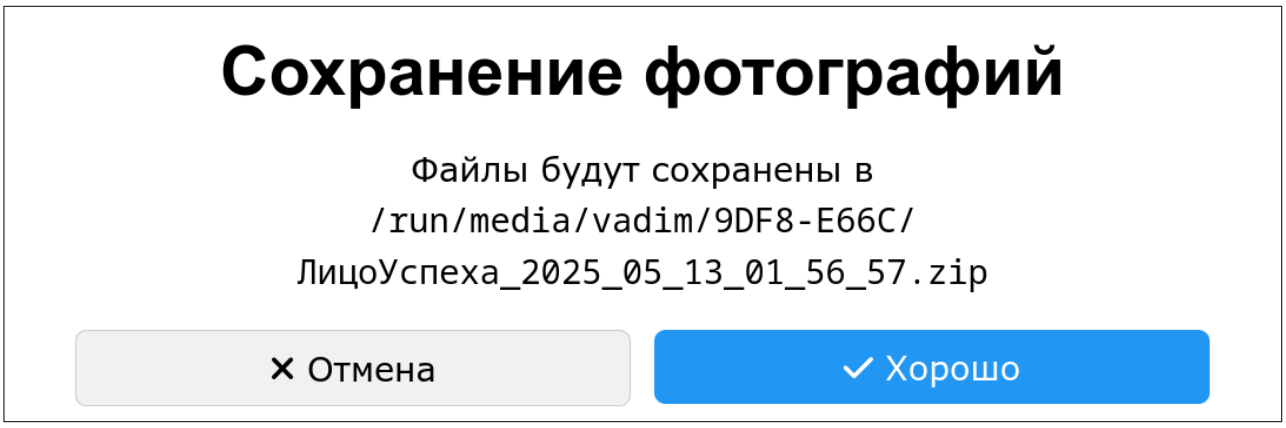

Рисунок 3.7 — Предупреждение о сохранении фотографий

После подтверждения сохранения, система передаёт ID фотографий, которые нужно сохранить, формирует zip-архив, который скачивается и сохраняется на внешнем хранилище (рис. 3.8). Код, отвечающий за формирование zip-архива приведён в листинге Ж.8.

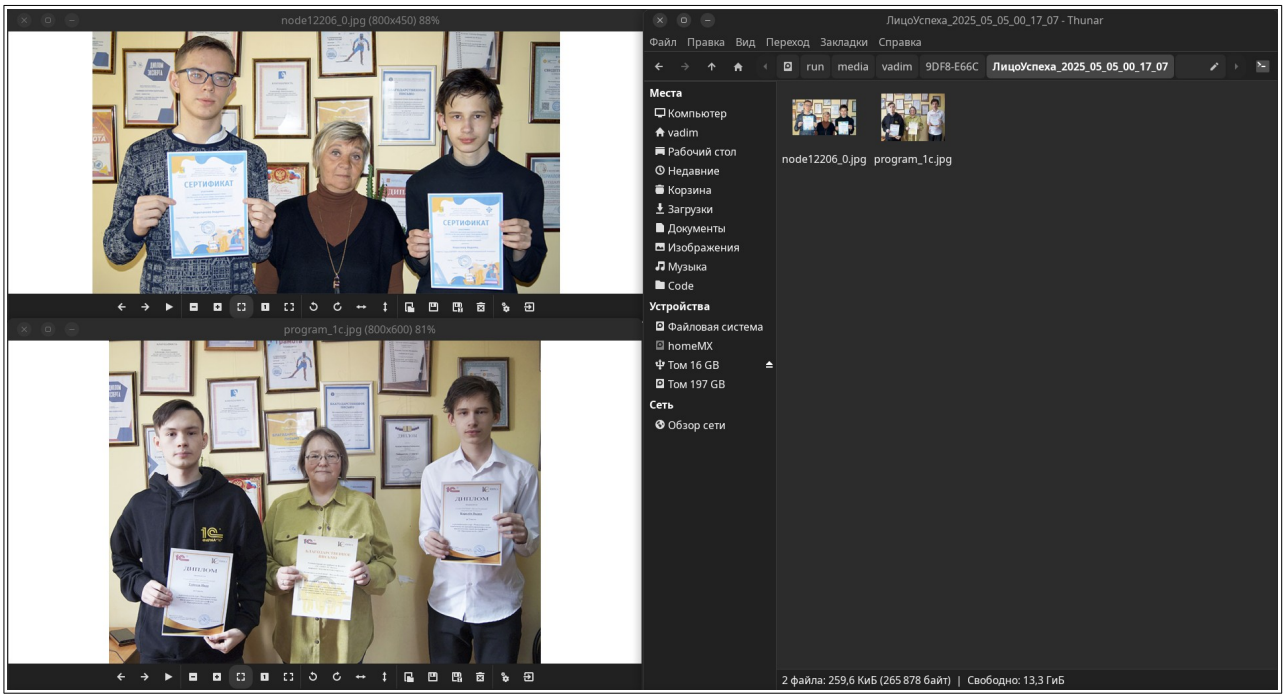

Рисунок 3.8 — Пример сохранения на внешнем носителе

4 Анализ разработанной информационной системы, выявление достоинств и недостатков

После завершения первой итерации проекта, была создана тестовая выборка фотографий, загруженных с сайта Вятско-Полянского механического техникума [7] в количестве 1000 файлов. Файлы были обработаны системой, проведено несколько тестов для проверки работоспособности системы. Тесты оказались успешными, для автора результат содержал 7 фотографий.

Разработанная система имеет как достоинства, так и недостатки.

Список достоинств:

- Точное распознавание лиц на фотографиях;
- Быстрая обработка данных;
- Гибкость в настройке;
- Простой интерфейс.

Список недостатков:

- Обязательно подключение USB-накопителя;
- Нет возможности отправить результат на электронную почту;
- Громоздкая архитектура проекта.

Использованная в проекте библиотека распознавания лиц позволяет не только распознавать лица, но и выявлять черты лица и выделять прямоугольник, который был определён как совпавшее лицо. В будущем возможно применение этого выделения как опция для пользователей.

Система может быть использована не только в образвательных учереждениях (школах, техникумах), но и в любой другой организации, ведущей медиатеку. Степень уникальности решения является довольно высокой, так как альтернативы предлагают обработку только в облаке, а так же не ведут базу данных лиц для распознавания.

5 Разработка методики внедрения и сопровождения информационной системы

После разработки первой версии программного продукта, были разработаны руководство пользователя в соответствии с ГОСТ 19.505-79 [8] (приложение Г) и руководство администратора в соответствии с ГОСТ 34.201-2020 [9] (приложение Д).

Руководство пользователя описывает порядок работы с системой, а так же описания ошибок, которые выдаются системой и шаги для их исправления.

Руководство администратора описывает каким образом система работает, как её запускать, заполнять данными, а так же описания ошибок и шаги по исправлению.

5.1 Программа резервного копирования и восстановления

5.1.1 Общие положения

Настоящий регламент проведения резервного копирования (восстановления) программ и данных, хранящихся на автоматизированных рабочих местах и серверах разработан с целью:

- Определения порядка резервирования данных для последующего восстановления работоспособности информационной системы персональных данных организации при полной или частичной потере информации, вызванной сбоями или отказами аппаратного или программного обеспечения, ошибками пользователей;
- Определения порядка восстановления информации в случае возникновения такой необходимости;
- Упорядочения работы должностных лиц, связанной с резервным копированием и восстановлением информации.

В настоящем документе регламентируются действия при выполнении следующих мероприятий:

- Резервное копирование;
- Контроль резервного копирования;
- Хранение резервных копий;
- Полное или частичное восстановление данных и приложений.

Резервному копированию подлежит информация следующих основных категорий:

- Данные распознавания лиц на фотографиях;
- Связи данных лиц с фотографиями.

5.1.2 Порядок резервного копирования

Система резервного копирования должна обеспечивать производительность, достаточную для сохранения информации, в установленные сроки и с заданной периодичностью. Резервное копирование должно производиться каждый день, ровно в 00:00 по Московскому времени.

5.1.3 Контроль результатов резервного копирования:

Контроль результатов всех процедур резервного копирования осуществляется системным администратором.

На протяжении периода времени, когда система резервного копирования находится в аварийном состоянии, должно осуществляться ежедневное копирование информации, подлежащей резервированию, с использованием средств файловых систем.

5.1.4 Восстановление информации из резервной копии

В случае необходимости, восстановление данных из резервных копий производится самостоятельно. Делается это следующим образом: определить используемый файл базы данных .sqlite3, затем заменить файл резервной копией.

Перечень резервируемой информации:

- Копируемый ресурс: данные информационной системы;
- Метод копирования: бэкап данных с помощью средств планировщика ОС;
- Тип копирования: полное копирование;
- Тип носителя: внешнее файловое хранилище;
- Периодичность смены носителя: согласно технической документации установленного заводом изготовителем, либо после диагностики диска;
- Раписания копирования: раз в неделю;

- Срок хранения: 3 месяца;

- Первичность проверки резервных копий: раз в месяц.

Методика настройки резервного копирования на сервере, приведена в приложении Д.

5.2 План мероприятий по внедрению и сопровождению информационной системы

Предлагаемый и рекомендуемый план мероприятий по внедрению информационной системы описан в таблице 5.1.

Таблица 5.1 — Рекомендуемый план мероприятий по внедрению

№	Содержание работ	Оптимальное время на выполнение этапа
1	Установить операционную систему на серверную часть проекта, все зависимости проекта в соответствии с README	1 день
2	Настроить веб-сервер на использование протокола Hyper Text Markup Protocol Secure (HTTPS)	1 день
3	Установить операционную систему на клиентскую часть проекта, установить браузер	1 день
4	Предоставить клиенту доступ к серверной части проекта	2-3 дня
5	Загрузить изображения медиатеки в систему	3-4 дня
6	Протестировать корректность работы	1 день

При эксплуатации системы, компоненты, её составляющие должны быть обновлены до самой последней версии, не ломающей совместимость с существующим кодом. В случае если несовместимое обновление необходимо установить обязательно (например, обновление устраняет серьёзную проблему безопасности), необходимо обновить код в соответствии с установленной версией компонента.

При обновлении компонента системы, необходимо обновить версию в файле README репозитория проекта.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы было проведено исследование существующих систем, разработан алгоритм работы, описан бизнес-процесс. На основании проектирования была разработана, внедрена и успешно протестирована система.

Разработка системы проходила в соответствии с требованиями системы и предполагаемыми желаниями будущих пользователей.

Система открыта для будущих улучшений, правок и добавления нового функционала.

Достоинства разработанной системы:

- API доступно для использования другими системами;
- Интерфейс простой и понятный;
- Точное распознавание лиц на фотографиях;
- Быстрая обработка данных;
- Гибкость в настройке.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Сбор изображений студентов и сотрудников | МИЭМ Wiki [Электронный ресурс] — Режим доступа: https://wiki.miem.hse.ru/Projects/394/docs/photo-collecting-output. — Заглавие с экрана. — (Дата обращения: 02.05.2025).

2 Фото — Google Фото [Электронный ресурс] — Режим доступа: https://photos.google.com/. — Заглавие с экрана. — (Дата обращения: 02.05.2025).

3 Яндекс Диск [Электронный ресурс] — Режим доступа: https://360.yandex.ru/disk/. — Заглавие с экрана. — (Дата обращения: 02.05.2025).

4 ageitgey/face_recognition: The world's simplest facial recognition api for Python and the command line [Электронный ресурс] — Режим доступа: https://github.com/ageitgey/face_recognition. — Заглавие с экрана. — (Дата обращения: 02.05.2025).

5 Vue.js - The Progressive JavaScript Framework | Vue.js [Электронный ресурс] — Режим доступа: https://vuejs.org/. — Заглавие с экрана. — (Дата обращения: 02.05.2025).

6 User guide — pyudev 0.21.0 documentation [Электронный ресурс] — Режим доступа: https://pyudev.readthedocs.io/en/latest/guide.html#id4. — Заглавие с экрана. — (Дата обращения: 04.05.2025).

7 Вятско-Полянский механический техникум | Ваш шанс! Ваш успех! Ваше будущее! [Электронный ресурс] — Режим доступа: https://www.vpmt.ru/. — Заглавие с экрана. — (Дата обращения: 15.05.2025).

8 ГОСТ 19.505-79. Руководство оператора. Требования к содержанию и оформлению. Технические требования. — Введ. 1980-01-01.— М.: Государственный комитет СССР по стандартам, 2010.— 3 с.

9 ГОСТ 34.201-2020. Межгосударственный стандарт. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем. Введ. 2022-01-01 - Издательство стандартов, 2020 - 12 с.

ПРИЛОЖЕНИЕ А

(обязательное)

Диаграмма основного бизнес-процесса

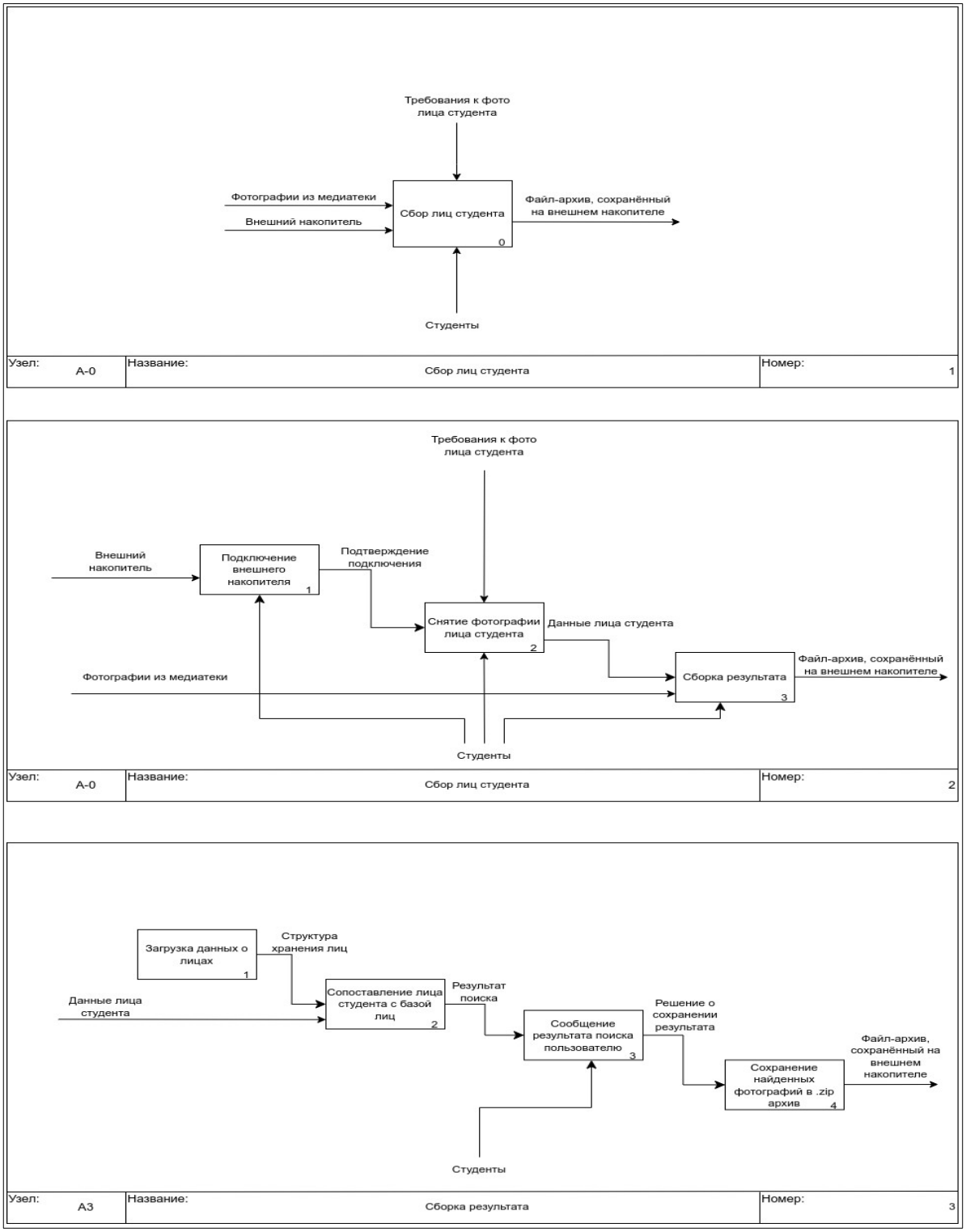

Рисунок А.1 — Диаграмма основного бизнес-процесса

ПРИЛОЖЕНИЕ Б

(обязательное)

Диаграмма связей сущностей

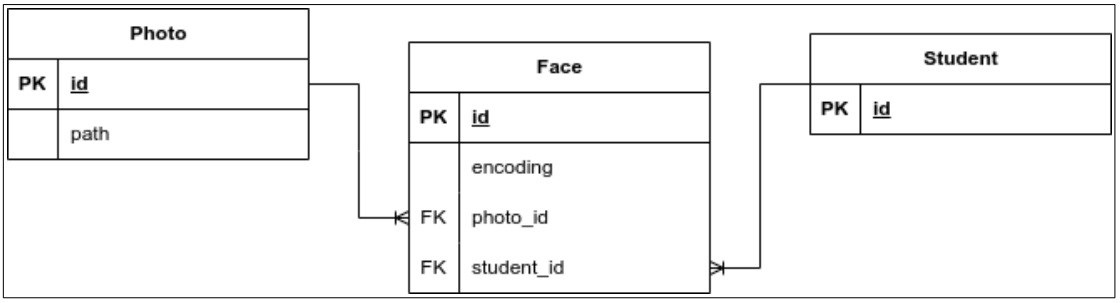

Рисунок Б.1 — Диаграмма связей сущностей

ПРИЛОЖЕНИЕ В

(обязательное)

Словарь данных

Таблица В.1 — Словарь данных сущности Photo

Название поля	Тип данных	Пояснение	Ограничения
id	int	Уникальный идентификатор фотографии.	PRIMARY KEY, NOT NULL
path	text	Путь к фотографии в медиатеке.	UNIQUE

Таблица В.2 — Словарь данных сущности Student

Название поля	Тип данных	Пояснение	Ограничения
id	int	Уникальный анонимный идентификатор студента.	PRIMARY KEY, NOT NULL

Таблица В.3 — Словарь данных сущности Face

Название поля	Тип данных	Пояснение	Ограничения
id	int	Уникальный идентификатор лица.	PRIMARY KEY, NOT NULL
encoding	blob	Сериализованные данные лица. Исходные данные генерируются функцией face_encodings.	
photo_id	int	Ссылается на фотографию, на которой запечатлено это лицо.	FOREIGN KEY
student_id	int	Ссылается на студента, которому, как считает система, принадлежит данное лицо.	FOREIGN KEY

ПРИЛОЖЕНИЕ Г

(обязательное)

Руководство пользователя

ИНФОРМАЦИОННАЯ СИСТЕМА «ЛИЦО УСПЕХА»
МОДУЛЬ «ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ»

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ
ВЕРСИЯ 1.0

2025

1 Введение

1.1 Область применения

Областью применения Модуля является взаимодействие пользователя с системой.

1.2 Краткое описание возможностей

Информационная система «Лицо успеха» предназначена для обработки медиатеки техникума, поиска и идентификации лиц, найденных на фотографиях.

В Модуле реализованы следующие возможности:

- Снятие фотографии пользователя;
- Автоматический поиск подключенных устройств внешнего хранения;
- Сохранение результата работы системы на внешнее хранилище.

1.3 Уровень подготовки пользователя

Пользователь должен иметь базовые навыки работы с компьютером.

1.4 Перечень эксплуатационной документации, с которой необходимо ознакомиться пользователю

- Настоящее Руководство.

2 Назначение и условия применения

Модуль «Интерфейс пользователя» в составе системы «Лицо успеха» предназначен для предоставления возможности пользователю взаимодействовать с системой. В модуле реализована возможность снятия фотографии и сохранения результата работы системы.

Работа с Модулем доступна всем пользователям. Работа с Модулем доступна всегда, когда есть необходимость в получении или подаче информации.

3 Подготовка к работе

Для работы с системой «Лицо успеха» необходимо следующее аппаратное обеспечение:

• Внешнее хранилище данных, подключаемое по USB.

Система, на которой запущен интерфейс пользователя, должна иметь сенсорный экран и оборудована USB-камерой для возможности снятия фотографий.

4 Описание операций

4.1 Выполняемые функции и задачи

Модуль «Интерфейс пользователя» в составе системы «Лицо успеха» выполняет задачи, приведённые в таблице ниже:

Таблица Г.1 — Описание функций и задач

Задачи	Описание
Снятие фотографии	Пользователю предоставляется возможность снять фотографию своего лица, с живым предпросмотром.
Просмотр результата работы системы	Пользователю предоставляется возможность просмотреть результат работы системы. Отображается до 6 фотографий.
Сохранение результата работы системы	Пользователю предоставляется возможность сохранить результат работы системы на внешнее хранилище. Хранилище должно быть подключено перед началом работы системы.

4.2 Описание операций технологического процесса обработки данных, необходимых для выполнения задач

Ниже приведено описание пользовательских операций для выполнения каждой из задач.

Задача: «Снятие фотографии»

Действия для выполнения:

1 Нажать на кнопку «Начать» (рис. Г.1);

2 Оставить своё лицо в кадре;

3 Нажать на кнопку «Сфотографироваться» (рис. Г.2);

Рисунок Г.1 — Кнопка «Начать»

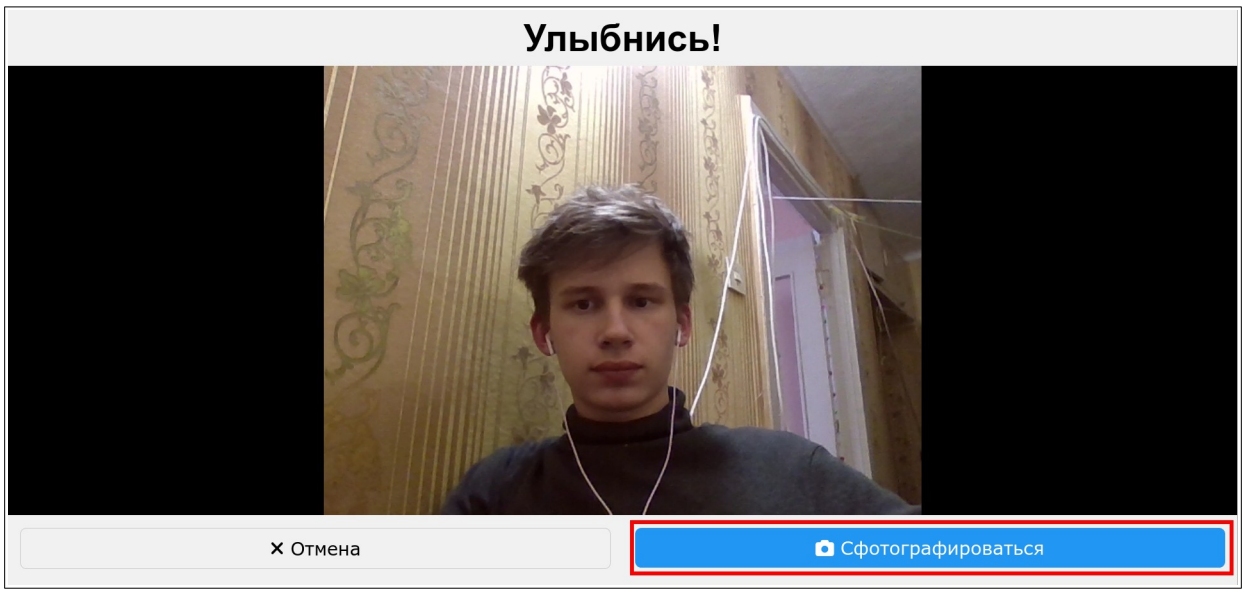

Рисунок Г.2 — Кнопка «Сфотографироваться»

Задача: «Просмотр результата работы системы»

Действия для выполнения:

1 Сфотографироваться;

2 Подождать от 2 до 10 секунд;

3 Просмотреть результат (рис. Г.3).

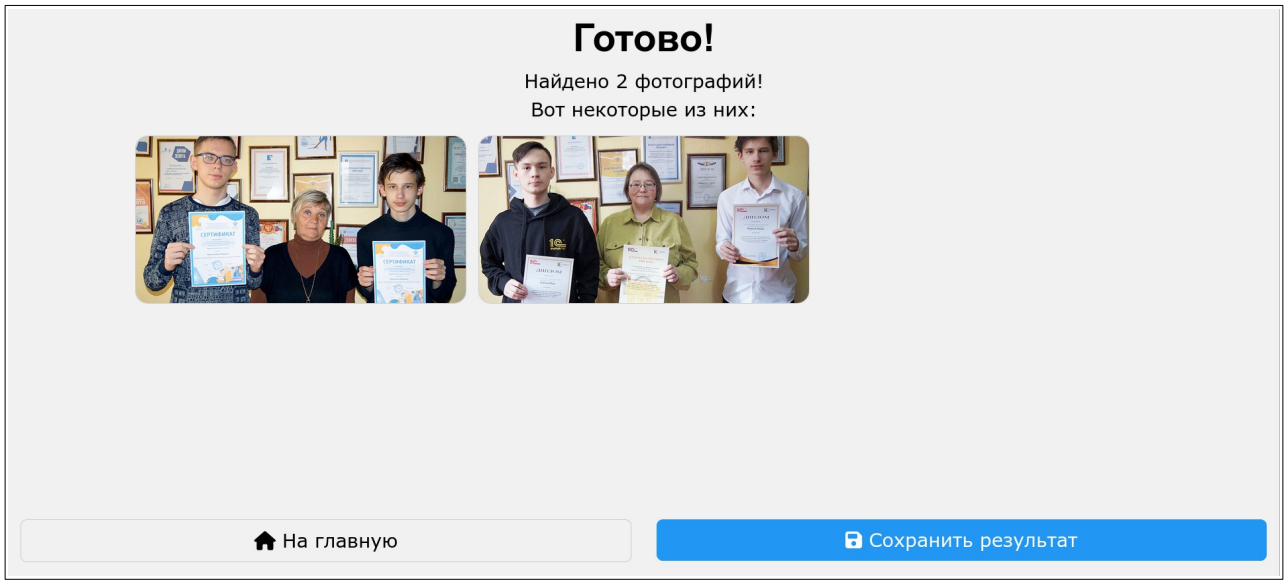

Рисунок Г.3 — Страница просмотра результата

Задача «Сохранение результата работы системы»

Действия для выполнения:

1 Подключить внешнее хранилище;

2 Сфотографироваться;

3 Нажать на кнопку «Сохранить результат»;

4 Подтвердить местоположение файла;

5 Отключить внешнее хранилище.

5 Аварийные ситуации

В случае возникновения ошибок при работе системы «Лицо успеха», не описанных ниже в данном разделе, необходимо обращаться к сотруднику подразделения технической поддержки, либо к ответственному Администратору системы «Лицо успеха».

Таблица Г.2 — Описание аварийных ситуаций

Ошибка	Описание ошибки	Требуемые действия пользователя при возникновении ошибки
USB хранилище не подключено	Пользователь пытается начать работу с системой, не подключив внешнее хранилище данных.	Подключить в терминал внешнее хранилище данных (флешка, ssd накопитель)
На фотографии не обнаружено ни одного лица	Системе не удалось распознать ни одного лица на сделанной фотографии.	Принять другую позу лица, снять любые предметы одежды, закрывающие лицо.
На фотографии обнаружено больше одного лица	Система распознала на фотографии больше одного лица.	Оставьте в кадре только одно лицо.
Нет связанных с тобой фотографий	Система распознала пользователя, но не нашла ни одной фотографии, на котором запечатлено его лицо.	-

ПРИЛОЖЕНИЕ Д

(обязательное)

Руководство администратора

ИНФОРМАЦИОННАЯ СИСТЕМА «ЛИЦО УСПЕХА»
МОДУЛЬ «НЕЙРОДВИЖОК»

РУКОВОДСТВО АДМИНИСТРАТОРА
ВЕРСИЯ 1.0

2025

Аннотация

Документ представляет руководство по сбору и обработке фотографий в медиатеке техникума с использованием системы распознавания лиц. Описаны основные возможности системы, такие как идентификация людей, ведение базы данных и развертывание HTTP API. Уровень подготовки пользователей требует базовых навыков работы с компьютером.

В разделе «Подготовка системы» указаны требования к программному обеспечению и шаги по настройке, включая сбор ресурсов интерфейса, настройку SSL, загрузку фотографий и запуск нейродвижка. Раздел "Аварийные ситуации" содержит информацию о возможных ошибках и рекомендации по их устранению. Документ служит справочным материалом для пользователей и администраторов системы.

1 Введение

Область применения: сбор фотографий медиатеки техникума.

Краткое описание возможностей:

- Распознавание лиц на фотографиях;
- Идентификация людей на снимках, формирование выборки фотографий по снимку пользователя;
- Ведение базы данных;
- Развертывание HTTP API для взаимодействия с системой.

Уровень подготовки пользователей: от пользователей требуются базовые знания работы с компьютером.

Перечень эксплуатационной информации:

- фреймворк для разработки api на языке программирования python - uvicorn (https://www.uvicorn.org/)
- веб-сервер для развертывания интерфейса пользователя — apache (https://apache.org/)
- установка ssl сертификата для apache (https://losst.pro/ustanovka-ssl-sertifikata-apache-ot-lets-encrypt)

2 Подготовка системы к работе

Система была протестирована, работает корректно в следующем окружении (таблица Д.1).

Таблица Д.1 — Версии компонентов

Компонент	Версия
Операционная система	Ubuntu Linux, ядро 6.8.0-60-generic
Разрядность процессора	x64
Веб-сервер	Apache 2.4.58
Версия python	3.12.3
Версия npm	9.2.0
Пакеты ОС	build-essential python3-venv libopenblas-dev liblapack-dev net-tools
Веб-браузер	Firefox 137.0.2

Версии пакетов python и node.js приведены репозиториях проекта и находятся по пути src/face-api/requirements.txt и src/web/package-lock.json соответственно.

Для запуска системы, требуется выполнить следующие действия:

1 Собрать все ресурсы пользовательского интерфейса в пакет с помощью команды npm run build, выгрузить пакет на веб-сервер. Пакет представляет собой папку (обычно создаётся под названием dist после выполнения команды).

2 Так как интерфейс системы запускается как сайт, то подключение должно быть защищено, так как некоторые возможности системы недоступны без безопасного контекста исполнения. Для создания защищённого соединения, потребуется настройка apache для работы с SSL сертификатами.

3 Загрузить фотографии в систему. За загрузку отвечает скрипт по пути src/face-api/scan-dir.py. Скрипт принимает как аргументы путь к базе данных (sqlite файл) и путь к папке, в которой хранятся фотографии. Скрипт рекурсивно собирает фотографии, распознаёт лица и загружает данные в систему.

4 Запустить нейродвижок. Команда для запуска нейродвижка должна быть похоже на что-то подобное:

uvicorn api:app --host 0.0.0.0 --port 8000 --ssl-keyfile <ПУТЬ К ФАЙЛУ .key> --ssl-certfile <ПУТЬ К ФАЙЛУ .pem>

5 Запустить файловое API на клиенте. Файловое API требуется для сохранения результата работы системы. Скрипт запуска расположен по пути src/file-api/start.sh.

Для проверки работы системы, необходимо открыть адрес, на котором развёрнут веб-сервер. Для проверки подключения внешнего хранилища, нужно нажать на кнопку «Начать». Если хранилище подключено, система покажет страницу снятия фотографии. Если система выдаёт результат слишком долго (больше 10 сек.), вероятнее всего, возникла проблема при запуске нейродвижка.

Резервное копирование должно осуществляться через программу cron.

В программу нужно ввести следующую запись (Каждый день в 2 часа ночи выполнять скрипт backup_sqlite.sh. Скрипт должен иметь права на выполнение.):

0 2 * * * /path/to/your/backup_sqlite.sh

В скрипт backup_sqlite поместить следующее:

```
#!/bin/bash
DATABASE="/путь/к/вашей/бд.db"
BACKUP_DIR="/путь/к/вашей/бэкап/папке"
DATE=$(date +"%Y-%m-%d")
BACKUP_FILE="$BACKUP_DIR/backup_$DATE.db"
cp "$DATABASE" "$BACKUP_FILE"
find "$BACKUP_DIR" -name "backup_*.db" -type f -mtime +7 -exec rm {} \;
```

Скрипт копирует файл базы данных в отдельную папку и добавляет к имени текущую дату. Затем скрипт удаляет файлы базы данных, старше чем 7 дней от текущей даты.

3 Аварийные ситуации

В процессе эксплуатации системы, может произойти несколько видов аварийных ситуаций. Описания и шаги к исправлению описаны в таблице Д.2.

Таблица Д.2 — Описание аварийных ситуаций

Описание ошибки	Шаги к исправлению
При запросе проверки подключения USB хранилища, консоль браузера выдаёт сетевую ошибку.	1 Проверить статус работы файлового API; 2 Проверить адрес файлого API, указанный в коде интерфейса пользователя. Исправить, если отличается от текущего.
При запросе к API нейродвижка, консоль браузера выдаёт сетевую ошибку.	1 Проверить статус работы API нейродвижка; 2 Проверить адрес API нейродвижка, исправить если отличается от текущего; 3 Убедиться, что API работает по протоколу HTTPS. Если нет, то настроить сертификаты HTTP веб-сервера, передать их при запуске в API.
Предпросмотр снимка лица не включается.	1 Проверить наличие и подключение USB-камеры; 2 Убедиться, что интерфейс браузера открыт в безопасном контексте (localhost или через протокол HTTPS).
Порт уже занят (при запуске любого API)	1 Определить процесс, использующий занятый порт; 2 Если процесс критичный, запустить API на другом порту, иначе завершить блокирующий процесс.

4 Описание сообщений

В процессе запуска или эксплуатации системы, может быть выдано множество сообщений об ошибках. В таблице Д.2 приведены описания сообщений и рекомендуемые действия при их получении.

Таблица Д.2 — Описание сообщений

Компонент системы	Сообщение	Рекомендуемые действия
scan-dir.py	Файл настроек по указанному пути не найден	Убедиться что по указанному пути файл настроек существует
	Ключ настроек не найден	В файле настроек отсутствует требуемый ключ
	Файл настроек содержит неверный JSON	Исправить форматирование файла настроек
	Не удалось открыть файл базы данных	Убедиться что путь к базе данных существует и есть права на его запись и чтение
	Каталог не найден	Убедиться что путь к папке с фотографиями действительно существует
	Не обнаружено новых изображений	- (не обработано ни одной отографии потому что новых изображений не найдено)
api.py	Started server process	- (api успешно запущено)
	Finished server process	- (api успешно выключено)
Интерфейс пользователя	На фотографии не обнаружено ни одного лица	Попробовать сфотографироваться, убрав с лица закрывающие его предметы
	На фотографии обнаружено больше одного лица	Оставить в кадре только одно лицо
	Нет связанных с тобой фотографий	- (фотографий не найдено)

ПРИЛОЖЕНИЕ Ж

(обязательное)

Листинги кода программного продукта

Листинг Ж.1 — Разметка начальной страницы

```
<script setup>
// Стартовая страница

import { createApp, h, ref }    from "vue"
import { useRouter }            from 'vue-router'
import { createConfirmDialog }  from 'vuejs-confirm-dialog'

import Button           from "../components/Button.vue"
import Heading          from "../components/Heading.vue"
import Step             from "../components/Step.vue"
import UsbNotConnected  from "../components/Modals/UsbNotConnected.vue"

import config                   from "../config.js"
import { checkExternalStorage}  from "../api.js"

const router = useRouter();

function start() {
    checkExternalStorage().then(function(data) {
        const isOk = data.ok;

        if (isOk) {
            router.push('/Camera');
            return;
        }

        // USB не подключен
        const { reveal, onConfirm, onCancel } = createConfirmDialog(
            UsbNotConnected
        );
        reveal();
        onConfirm(() => {
            // pass
        });
        onCancel(() => {
            // pass
```

```
            });
        });
    }
    </script>

    <template>
        <div id="mainPage" class="page-main">
            <Heading text="Лицо успеха"/>
            <p class="w3-center w3-xlarge">
                    Сфотографируйся, а программа найдёт все фотографии
техникума, на которых ты присутствуешь!
            </p>
            <div class="steps">
                <Step
                    number="1"
                    text="Вставь флешку"
                    color="w3-red"
                    border-color="w3-border-pink"/>
                <Step
                    number="2"
                    text="Сфотографируйся"
                    color="w3-amber"
                    border-color="w3-border-orange"/>
                <Step
                    number="3"
                    text="Сохрани фотографии"
                    color="w3-yellow"
                    border-color="w3-border-amber"/>
            </div>

            <div class="w3-row">
                <div class="w3-container w3-col s12">
                    <Button
                        class="w3-block"
                        text="Начать"
                        stylename="primary"
                        icon="play"
                        @click="start"/>
                </div>
            </div>
        </div>
    </template>

    <style scoped>
```

```
#mainPage {
    grid-template-rows: auto auto auto 10vh;
}

.steps {
    margin: auto;
}

#modals {
    position: absolute;
    height: 100vh;
    min-height: 100vh;
    max-height: 100vh;
    width: 100vw;
    left: 0;
    top: 0;
    display: none;
}
</style>
```

Листинг Ж.2 — api для использования в JavaScript

```
// Связь с backend API, в Javascript функциях
import config from "./config.js";

////////////// API ЛИЦ ////////////
// Посылает запрос на API лиц
function doFaceFetch(path, params = {}) {
    params.headers = {
        'Accept': 'application/json',
        'Content-Type': 'application/json'
    };
    return fetch(config.faceApiUrl + path, params);
}

// Отправляет лицо на обработку
export async function sendFace(base64image) {
    const response = await doFaceFetch("/accept-face", {
        method: 'post',
        body: JSON.stringify({base64image: base64image})
    });
    return await response.json();
}
```

```
// Посылает запрос на генерацию ZIP на сервере
export async function getZip(imageIds) {
    const response = await doFaceFetch("/zip", {
        method: 'post',
        body: JSON.stringify({imageIds: imageIds})
    });
    return await response.blob();
}

////////////// API ФАЙЛОВОЙ СИСТЕМЫ //////////////

// Посылает запрос на API лиц
function doFSFetch(path, params = {}) {
    params.headers = {
        'Accept': 'application/json',
        'Content-Type': 'application/json'
    };
    return fetch(config.filesystemApiUrl + path, params);
}

// Запрашивает у сервера генерацию имени файла для сохранения
export async function getFilenameToSave() {
    if (config.disableUSBChecks) {
        return {
            ok: true,
            path: "C:\\face_of_success\\result.zip"
        };
    }
    const response = await doFSFetch("/get-name-to-save");
    const jsonData = await response.json();
    return jsonData;
}

// Запрашивает у сервера сохранение фотографий
export async function requestSaving(zipBlob, path) {
    if (config.disableUSBChecks) {
        console.log("Сохранение файлов отключено");
        return;
    }
    const fd = new FormData();
    fd.append("zipfile", zipBlob);
    fd.append("savepath", path);

    await fetch(config.filesystemApiUrl + "/save", {
```

```
            method: 'post',
            body: fd
        });
    }

    // Проверяет, подключено ли внешнее устройство хранения
    export async function checkExternalStorage() {
        debugger;
        if (config.disableUSBChecks) {
            return {ok: true};
        }
           const response = await fetch(config.filesystemApiUrl + "/usb-
connected");
        const jsonData = await response.json();
        return jsonData;
    }
```

Листинг Ж.3 — Модальное окно отображения ошибки о том, что USB-устройство не подключено.

```
    <script setup>
    import Heading  from "../Heading.vue"
    import Button   from "../Button.vue"

    const props = defineProps({filename: String});
    const emit  = defineEmits(['confirm', 'cancel'])
    </script>

    <template>
        <div class="w3-modal" style="display: block">
            <div class="w3-modal-content w3-animate-top">
                <div class="w3-container">
                    <div id="layout">
                        <Heading text="Хранилище не подключено"/>
                        <p class="w3-center w3-xlarge">
                                Необходимо вставить в киоск флеш-карту или
другое USB-хранилище.
                        </p>
                        <img class="w3-padding" src="/img/usb.png"/>
                                    <Button  icon="check"  text="Хорошо"
@click="emit('confirm')" stylename="secondary"/>
                    </div>
                </div>
            </div>
```

```
        </div>
    </template>

    <style scoped>
    #layout
    {
        display: grid;
        grid-template-rows: auto auto 256px 10vh;
    }

    img
    {
        width: auto;
        margin: auto;
        height: 100%;
    }
    </style>
```

Листинг Ж.4 — Разметка страницы снятия фотографии

```
    <script setup>
    // Страница снятия фотографии
    // Помощь и немного кода:
    //
https://developer.mozilla.org/ru/docs/Web/API/Media_Capture_and_Streams_API/
Taking_still_photos

    import { onMounted, ref }   from "vue"
    import { useRouter }        from 'vue-router'

    import Button       from "../components/Button.vue"
    import NavButton    from "../components/NavButton.vue"
    import Heading      from "../components/Heading.vue"

    import config       from "../config.js"
    import { storage }  from '../storage.js'

    const video     = ref(null);
    const canvas    = ref(null);
    const router    = useRouter();

    function takePhoto() {
        const ctx           = canvas.value.getContext("2d");
        canvas.value.width  = video.value.videoWidth;
```

```
            canvas.value.height = video.value.videoHeight;
            ctx.drawImage(video.value, 0, 0);

                                        storage.photoData                          =
canvas.value.toDataURL("image/png").split(',')[1];

            router.push('/Searching');
        }

        onMounted(() => {
            navigator.mediaDevices
                .getUserMedia({ video: true, audio: false })
                .then((stream) => {
                    video.value.srcObject = stream;
                    video.value.play();
                })
                .catch((err) => {
                    console.error(`An error occurred: ${err}`);
                });
        })
        </script>

        <template>
            <div id="cameraPage" class="page-main">
                <Heading text="Улыбнись!"/>

                <div id="videoWrapper" class="w3-center w3-margin-bottom">
                    <video ref="video">Видеопоток недоступен</video>
                </div>

                <div class="w3-row">
                    <div class="w3-container w3-col s6">
                        <NavButton
                            class="w3-block"
                            text="Отмена"
                            icon="xmark"
                            target="Main"
                            stylename="secondary"/>
                    </div>
                    <div class="w3-container w3-col s6">
                        <Button
                            class="w3-block"
                            icon="camera"
                            text="Сфотографироваться"
```

```
                    stylename="primary"
                    @click="takePhoto"/>
            </div>
        </div>
    </div>

    <canvas ref="canvas" class="w3-hide"></canvas>
</template>

<style scoped>
#cameraPage
{
    grid-template-rows: auto 1fr 10vh;
}

#videoWrapper
{
    background-color: black;
}

#videoWrapper > video
{
    height: 100%;
    margin: auto;
}
</style>
```

Листинг Ж.5 — Код принятия фотографии от интерфейса

```
# accept-face.py
# Загружает изображение, возвращает список из id студентов, которые
были
# найдены в БД
# Выводит ответ в стандартный вывод в формате JSON
# Пример ошибки
#{
#    "ok": false,
#    "description": "Перефотографируйтесь"
#}
# Пример успеха
#{
#    "ok": true,
#    "photoIds": [1, 2, 3, 4]
#}
```

```
import logging
import face_recognition
import numpy as np
import FaceCoding.FaceStorage as FaceStorage

# Выводит ошибку в std
def fail(message):
    return {"ok": False, "description": message}

def success(photoIds):
    output = []
    for item in photoIds:
        output.append(item[0])
    obj = {"ok": True, "photoIds": output}
    return obj

def process(dbManager, faceEncoding):
    # Хранилище
    storage = FaceStorage.FaceStorage()
    allFaces = dbManager.getStudentFaces()
    storage.fill(allFaces)

    # Узнаём что это за студент
    result = storage.compare(faceEncoding)
    if not result.found:
        return fail("Нет связанных с тобой фотографий")

    # Собираем все фотографии, которые есть с этим студентом
    photoIds = dbManager.getPhotosByStudentId(result.foundStudentId)
    return success(photoIds)

def accept(filename, db):
    # Загрузка фотографии
    image = face_recognition.load_image_file(filename)

    # Поиск лиц
    faceEncodings = face_recognition.face_encodings(image, model='cnn')
    logging.info("Найдено лиц: {}".format(len(faceEncodings)))
    if (len(faceEncodings) == 0):
        return fail("На фотографии не обнаружено ни одного лица")
    if (len(faceEncodings) > 1):
        return fail("На фотографии обнаружено больше одного лица")
    faceEncoding = faceEncodings[0]
```

```
        return process(db, faceEncoding)
```

Листинг Ж.6 — Код класса обработчика распознавания фотографий

```
# face-storage.py
import face_recognition
import numpy as np
import io

class FaceSearchResult:
    def __init__(self, found, foundStudentId=None):
        self.found = found
        self.foundStudentId = foundStudentId

# Класс для хранения кодировок лиц и студентов в памяти
class FaceStorage:
    def __init__(self):
        self.faces = []
        self.ids = []

    # Добавляет соответствие лица и студента
    # array - numpy массив
    # studentId - число, id студента
    def add(self, array, studentId):
        self.faces.append(array)
        self.ids.append(studentId)

                # Заполняет лица. allFaces - результат работы
database.DB.getStudentFaces
    def fill(self, allFaces):
        for item in allFaces:
            byte_io = io.BytesIO(item[1])
            self.add(np.load(byte_io), item[0])

    def compare(self, encoding):
         # Список из булевых значений, где matches[i] будет True, если
faces,
        # и соответственно ids[i] будет похоже на studentFace
        matches = face_recognition.compare_faces(
            self.faces,
            encoding,
            tolerance=0.5
        )
```

```
            if not True in matches:
                # Совпадений не найдено
                return FaceSearchResult(False)

            # Узнаём какие индексы совпали
             matchedIndexes = [i for (i, matched) in enumerate(matches) if
matched]

            # {student id: сколько совпадений}
            counts = {}
            for i in matchedIndexes:
                studentId = self.ids[i]
                counts[studentId] = counts.get(studentId, 0) + 1

            # Выбор студента с наибольшим количеством совпадений
            finalStudentId = max(counts, key=counts.get)

            return FaceSearchResult(True, finalStudentId)
```

Листинг Ж.7 — Код страницы просмотра результата

```
<script setup>
// Страница результата
import Button           from "../components/Button.vue"
import NavButton        from "../components/NavButton.vue"
import Heading          from "../components/Heading.vue"
import ResultImage      from "../components/ResultImage.vue"
import ShowFilename     from "../components/Modals/ShowFilename.vue"

import { onMounted, createApp, h }  from "vue"
import { createConfirmDialog }      from 'vuejs-confirm-dialog'
import { useRouter }                from 'vue-router'
import { storage }                  from '../storage.js'
import { getFilenameToSave }        from '../api.js'

const router = useRouter();

async function start() {
    const response = await getFilenameToSave();

    const { reveal, onConfirm, onCancel } = createConfirmDialog(
        ShowFilename,
        {filename: response.path}
    );
```

```
            reveal();
            onConfirm(() => {
                storage.savePath = response.path;
                router.push("/Saving");
            });
            onCancel(() => {
                console.log('pass');
            });
        }
        </script>

        <template>
            <div id="resultsPage" class="page-main">
                <Heading text="Готово!"/>
                            <p  class="no-margin  w3-center  w3-xlarge">Найдено
{{storage.foundFaces.length}} фотографий!<br/>Вот некоторые из них:</p>

                <div class="images">
                    <ResultImage
                        v-for="imageId in storage.foundFaces.slice(0,6)"
                        :imageId="imageId"
                        :key="imageId"/>
                </div>

                <div class="w3-row">
                    <div class="w3-container w3-col s6">
                        <NavButton
                            class="w3-block"
                            icon="house"
                            text="На главную"
                            target="Main"
                            stylename="secondary"/>
                    </div>
                    <div class="w3-container w3-col s6">
                        <Button
                            class="w3-block"
                            icon="save"
                            text="Сохранить результат"
                            stylename="primary"
                            @click="start"/>
                    </div>
                </div>
            </div>
            <div id="modals"></div>
```

```
</template>
<style scoped>
#resultsPage {
    grid-template-rows: auto auto minmax(0, 1fr) 10vh;
}

.images {
    width: 80%;
    margin-left: auto;
    margin-right: auto;
    margin-top: 1rem;
    margin-bottom: 1rem;
    display: grid;
    grid-template-columns: repeat(3, minmax(0, 1fr));
    grid-template-rows: 1fr 1fr 1em;
    grid-gap: 1rem;
}

.no-margin {
    margin: 0;
}
</style>
```

Листинг Ж.8 — Код упаковки фотографий в zip-архив

```
# Возвращает zip файл
@app.post("/zip")
async def createZip(r: ZipRequest):
    db = getDb()
    photoPaths = db.getPathsByIds(r.imageIds)
    db.close()

    # Сохранение изображения во временный файл
    # Генерируем путь к файлу
    tmpDir = tempfile.gettempdir()
    zipPath = os.path.join(tmpDir, 'bundle.zip')

    # Запись ZIP файла
    with zipfile.ZipFile(zipPath, "w") as z:
        for row in photoPaths:
            z.write(row[0], os.path.basename(row[0]))

    # Возвращаем на клиент zip файл
    return FileResponse(zipPath)
```

ПРИЛОЖЕНИЕ К

(обязательное)

Программный продукт

ДП.011.09.02.07.000.ПП