

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

вычислительной техники

Кафедра

МОиПЭВМ

Направление 09.03.04 «Программная инженерия»
ПОДГОТОВКИ

БАКАЛАВРСКАЯ РАБОТА

на тему

**Разработка системы автоматического доступа в
помещение с использованием метода распознавания лиц**

Студент

(подпись, дата)

Бонич Иван Александрович

(ФИО полностью)

Руководитель

(подпись, дата)

Попова Н. А.

(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.

(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

(подпись)

Козлов А.Ю.

(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

Попова Н.А.

(фамилия, инициалы)

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»

Заведующий кафедрой

_____ А.Ю. Козлов

«__» _____ 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Бонич Иван Александрович гр. 19ВП2 факультета
<Фамилия, имя, отчество полностью>

вычислительной техники

2. Тема работы Разработка системы автоматического доступа в
помещение с использованием метода распознавания лиц

Тема утверждена приказом ПГУ № _____ от " " апреля 2023 года

2. Руководитель работы Попова Наталья Александровна

3. Задание на работу (назначение разработки, исходные данные и т.п.)

1) *Назначение: система автоматического доступа с использованием метода
Распознавания лиц*

2) *Среда выполнения: операционная система Windows 10*

Языковые средства разработки: C++, HTML, CSS.

Инструментальные средства: MS Visual Studio 2022, Arduino IDE 2.1.0, Draw.io.

*Технологии разработки: ООП, принципы SOLID, шаблоны проектирования,
клиент-серверная архитектура API.*

3) *Основные функции приложения:*

а) контролировать и управлять доступом к помещениям;

б) создание, редактирование и удаление профилей пользователей в системе;
в) система автоматически отправляет уведомления или предупреждения при попытках несанкционированного доступа;

4. Перечень подлежащих разработке вопросов:

- 1) Анализ предметной области и постановка задачи;*
- 2) Проектирование;*
- 3) Реализация;*
- 4) Контроль качества программного обеспечения;*
- 5) План разработки бюджета;*
- 6) Оформление пояснительной записки.*

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	12.04.2023 – 16.04.2023	
2	Проектирование программных средств	20%	17.04.2023 – 26.04.2023	
3	Реализация программных средств	30%	27.04.2023 – 11.05.2023	
4	Тестирование программных средств	20%	12.05.2023 – 21.05.2023	
5	Оформление пояснительной записки	20%	22.05.2023 – 01.06.2023	

Дата выдачи задания «22» декабря 2022г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить

Декан ФВТ

(подпись, дата) _____

Л.Р. Фионова

Реферат

Пояснительная записка содержит 59 листов, 19 рисунков, 7 таблиц, 5 использованных источников и 3 приложения.

СКУД, КЛИЕНТ, СЕРВЕР, ВЕБ-ИНТЕРФЕЙС, UML.

Объект исследования – система контроля управлением доступа, основанная на технологии распознавания лиц.

Цель выпускной квалификационной работы – создание эффективной, надежной и простой в использовании системы автоматического контроля доступа в помещение, которая использует методы распознавания лиц для идентификации и проверки пользователей. Эта система должна обеспечить безопасность и контроль доступа к различным объектам и помещениям, снизив при этом необходимость вручную управлять процессами контроля и идентификации. Система должна предоставлять возможность управлять процессом доступа, регистрировать новых пользователей, настраивать уровни доступа и отслеживать все попытки доступа.

Технологии разработки – разработка приложения проводилась на языке программирования C++ в среде Microsoft Visual Studio 2022 и Arduino IDE. Веб-интерфейс разработан на языке HTML и CSS.

Среда исполнения – веб-браузер.

Результаты работы – в ходе выполнения выпускной квалификационной работы были созданы клиентская и серверная часть системы контроля управления доступом. Все требования к разработке приложения соблюдены.

					ПГУ 09.03.04– 8БР192.04 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата	Системы автоматического доступа в помещение с использованием метода распознавания лиц. Пояснительная записка	Лит.	Лист	Листов	
Разраб.		Бонич И.А.							
Пров.		Попова Н.А.					4	59	
Н. Контр.		Такташкин Д.В				Группа 19ВП2			
Утв.		Козлов А.Ю							

Содержание

Введение	7
1 Анализ СКУД и выделение требований.....	9
1.1 Анализ предметной области.....	9
1.2 Обзор продуктов-аналогов	10
1.2.1 SecurOS FaceX от ISS	10
1.2.2 AnyVision	11
1.2.3 Face-Intelligence.....	12
1.2.4 Vocord FaceControl.....	13
1.3 Сравнительный анализ и критериальная оценка аналогов	14
1.4 Исследование вариантов использования	18
1.5 Анализ требований.....	20
1.6 Постановка задачи.....	20
2 Проектирование приложения.....	22
2.1 Архитектура приложения	22
2.2 Проектирование программных средств	23
2.3 Проектирование интерфейса.....	26
3 Реализация приложения.....	28
4 Контроль качества программного обеспечения	32
4.1 Выбор режима тестирования.....	32
4.2 Тестирование серверной части приложения	33
4.3 Тестирование клиентской части приложения	34
4.3 Планирование разработки и оценка бюджета.....	35
Заключение.....	39
Список использованных источников.....	40
Приложение А. Глоссарий.....	41

Приложение Б. Листинг программы.....	43
Приложение В. Результаты тестирования.....	57

Введение

Системы контроля и управления доступом на основе распознавания лиц действительно становятся все более популярными. В современном мире, где безопасность является приоритетом, автоматическое распознавание лиц становится все более важным инструментом. СКУД используется в самых разных областях, начиная от офисных помещений и заканчивая аэропортами. Системы контроля доступа, основанные на распознавании лиц, предлагают уникальные возможности для усиления безопасности зданий и обеспечения конфиденциальности информации.

Актуальность этой темы обусловлена быстрым развитием технологий и постоянно увеличивающимся спросом на более надежные и удобные методы контроля доступа. В эпоху пандемии COVID-19 бесконтактные технологии получили особую актуальность. Системы на основе распознавания лиц позволяют обеспечивать контроль доступа без физического контакта, что повышает уровень гигиеничности и безопасности. Современные алгоритмы распознавания лиц способны обрабатывать большое количество данных быстро и с высокой степенью точности, что позволяет обеспечивать быстрый и надежный контроль доступа. Системы на основе распознавания лиц предлагают удобство для пользователей, поскольку не требуют носить специальные карты доступа или запоминать пароли. В некоторых отраслях, таких как здравоохранение, финансовый или государственный сектор, могут быть установлены законодательные требования к использованию биометрических систем контроля доступа для обеспечения безопасности данных и защиты от несанкционированного доступа. Поскольку технология распознавания лиц продолжает развиваться и становится все более доступной, барьеры для внедрения таких систем снижаются.

Таким образом, задача разработки системы автоматического доступа в помещение с использованием метода распознавания лиц является актуальной.

Целью выпускной квалификационной работы является разработка системы автоматического доступа в помещение на основе технологии распознавания лиц, которая обеспечивает надежность и безопасность, а также простоту использования.

Объект исследования – средства автоматизации контроля доступа с использованием метода распознавания лиц.

Предмет исследования – система автоматического доступа в помещение.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- провести анализ требований;
- выполнить проектирование программных средств, структур данных и интерфейса;
- реализовать систему автоматического доступа в помещение;
- отладить и протестировать полученное приложение;
- составить план разработки и оценить бюджет разработки.

В работе использованы методы сравнительного анализа, методы проектирования, разработки и тестирования программного обеспечения.

1 Анализ СКУД и выделение требований

1.1 Анализ предметной области

Для проведения анализа требований необходимо предварительно проанализировать предметную область.

Благодаря быстрому развитию технологий, в частности в области машинного обучения и искусственного интеллекта, на сегодняшний день появилось множество новых систем контроля и управления доступом, особенно тех, которые используют биометрические технологии, включая распознавание лиц.

Разнообразие СКУД на рынке сегодня позволяет организациям выбирать системы, которые наиболее соответствуют их уникальным требованиям в плане безопасности, удобства использования, стоимости и других факторов. В то же время, с ростом использования биометрических систем контроля доступа возрастает и необходимость в защите биометрических данных. Организации должны следить за тем, чтобы системы, которые они выбирают, соответствовали стандартам безопасности и приватности данных.

Исходя из этого, в приложении должны быть представлены следующие основные функции:

- система должна иметь возможность распознавать лица пользователей с высокой точностью;
- система должна быстро обрабатывать запросы на доступ;
- система должна иметь возможность хранить и управлять базой данных лиц, а также должна иметь функции, которые позволяют добавлять, удалять или изменять данные;
- система должна иметь механизмы для обработки исключений, или попыток обхода системы;
- интерфейс системы должен быть понятным и простым для использования.

С точки зрения удобства использования, веб-интерфейс является актуальным вариантом. Веб-интерфейс доступен с любого компьютера или

устройства с подключением к Интернету. Он совместим со всеми основными операционными системами и браузерами, обновления и улучшения могут быть легко внедрены через веб-интерфейс, без необходимости устанавливать новое программное обеспечение на каждом устройстве пользователя.

1.2 Обзор продуктов-аналогов

В результате анализа программных продуктов по автоматизации контроля доступа были определены следующие наиболее близкие решения:

- SecurOS FaceX от ISS;
- AnyVision;
- Face-Intellect от AxxonSoft;
- Vocord FaceControl.

Эти решения получают высокие оценки от пользователей и пользуются большой популярностью.

Все эти системы предлагают различные функции, включая мониторинг в реальном времени, оповещения, интеграцию с другими системами безопасности, и интеллектуальный поиск.

1.2.1 SecurOS FaceX от ISS

SecurOS предоставляет универсальные инструменты для обеспечения безопасности и мониторинга, которые могут быть адаптированы к широкому спектру задач, от управления видеонаблюдением до сложных систем контроля доступа и анализа поведения. Этот продукт может быть интегрирован с различными устройствами и системами, включая системы контроля доступа, системы пожарной безопасности, системы биометрического контроля и многое другое.

Интерфейс приложения представлен на рисунке 1.

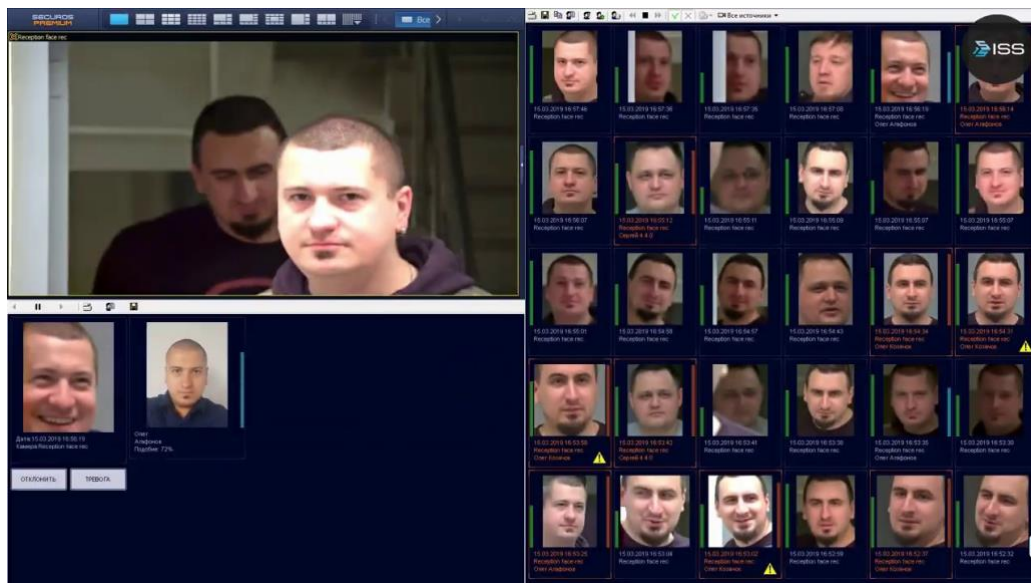


Рисунок 1 – Интерфейс приложения «SecurOS FaceX»

Преимущества:

- высокая точность и скорость распознавания лиц;
- универсальность: может быть интегрирована с другими системами безопасности.

Недостатки:

- возможна сложность использования без предварительного обучения;
- система может быть дорогостоящей для небольших организаций.

SecurOS FaceX представляет собой модуль распознавания лиц, который может быть интегрирован с SecurOS для идентификации лиц на видеоматериале в реальном времени. Этот модуль может быть использован для различных целей, включая контроль доступа, поиск лиц в массовых сборищах людей и идентификацию подозрительных лиц.

1.2.2 AnyVision

AnyVision – это компания по искусственному интеллекту, которая специализируется на компьютерном зрении. Она известна своими технологиями в области распознавания лиц и других типов биометрического идентификационного программного обеспечения.

Интерфейс приложения представлен на рисунке 2.

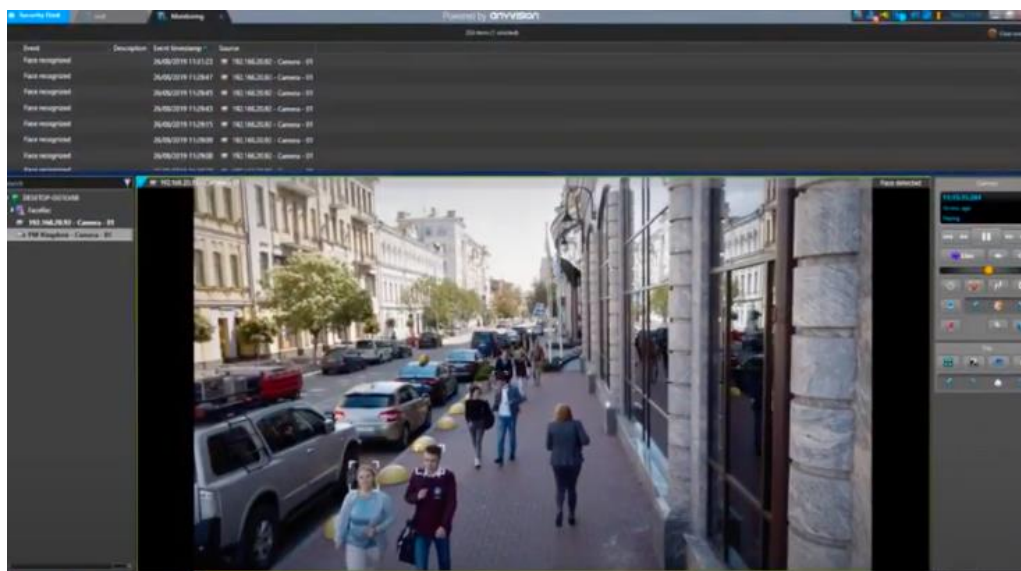


Рисунок 2 – Интерфейс приложения «AnyVision»

Преимущества:

- использует AI и глубокое обучение для распознавания лиц, что обеспечивает высокую точность;
- масштабируемость: подходит как для небольших, так и для крупных организаций.

Недостатки:

- возможные проблемы с конфиденциальностью и безопасностью данных;
- требует мощного оборудования для оптимальной работы.

Система может быть интегрирована с различными подсистемами и таким образом помогает обеспечить эффективную защиту объекта благодаря построению комплексной системы, включающей различные уровни охраны.

1.2.3 Face-Intelligence

Face-Intelligence – система идентификации и верификации человека, основанная на распознавании лица. Система может быть интегрирована с различными подсистемами и таким образом помогает обеспечить эффективную защиту объекта благодаря построению комплексной системы, включающей различные уровни охраны.

Интерфейс приложения представлен на рисунке 3.

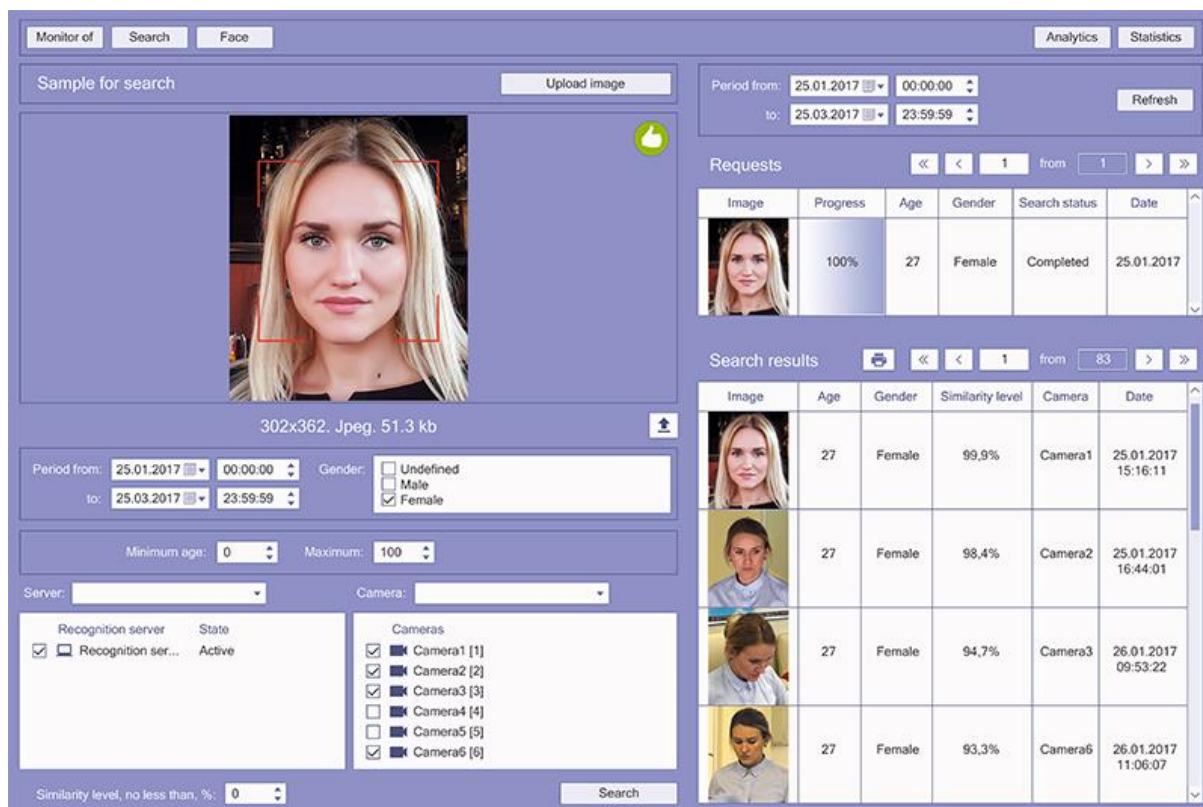


Рисунок 3 – Интерфейс приложения «Face-Интеллект»

Преимущества:

- возможность обнаруживать и распознавать лица в толпе.

Недостатки:

- возможность ложных срабатываний;
- требует значительного объема хранилища данных для работы с большими базами данных.

Face-Интеллект применяется для обеспечения безопасности в местах массового скопления людей – вокзалах, аэропортах, метрополитене и спорткомплексах и т. д.

1.2.4 Vocord FaceControl

Vocord FaceControl – система некооперативной идентификации человека по лицу. Некооперативность означает, что системе не требуется «сотрудничество» человека, чтобы верно его распознать.

Интерфейс приложения представлен на рисунке 4.



Рисунок 4 – Интерфейс приложения «Vocord FaceControl»

Преимущества:

- некооперативность;
- высокая скорость обработки данных.

Недостатки:

- поддерживает ограниченное количество форматов видео;
- высокая цена продукта.

Поэтому VOCORD FaceControl применяется для распознавания лиц в местах массового пребывания людей – на вокзалах, в метро и аэропортах, на стадионах, в концертных залах и в других публичных местах, где людей нельзя заставить взаимодействовать с системой биометрической идентификации.

1.3 Сравнительный анализ и критериальная оценка аналогов

Все четыре системы обладают высокой производительностью и способностью работать в реальном времени, хотя AnyVision и Face-Intelligence могут потребовать мощного оборудования для достижения максимальной производительности. Надежность может зависеть от специфических условий

использования, но в целом все четыре системы обладают высокой степенью надежности.

Для составления объективной оценки оформим таблицу, в которой будут отражены критерии значимости. Данная информация представлена в таблице 1.

Таблица 1 – Выделение критериев значимости

№	Название критерия	Коэффициент значимости
Функциональные возможности		
1	Идентификация и аутентификация пользователей	1
2	Управление через веб-интерфейс	0.75
3	Возможность контролировать доступ к определенным зонам	0.75
4	Наличие форм регистрации и авторизации	1
Технические возможности		
5	Возможность использования на слабом устройстве	0.75
6	Работа в реальном времени	1
Необязательные функциональные возможности		
7	Бесплатность продукта	1
8	Распознавание номерных знаков автомобилей	0.5
9	Тепловое сканирование и обнаружение масок	0.5
Эргономика		
10	Наличие интуитивно понятного интерфейса	1

Для оценки приведенных выше критериев используется двухбалльная система оценки, в которой значения изменяются от 0 до 2, где 0 – «полное несоответствие», 1 – «частичное соответствие», 2 – «полное соответствие» требованиям. В таблице 2 приведены более подробные характеристики оценки.

Таблица 2 – Шкала баллов

Балл	Характеристика оценки
0	функция отсутствует, реализовать ее невозможно
1	похожие, но отличные от требуемых, функции находятся в стадии разработки
2	функция уже реализована

Для выставления оценок необходимо подробнее рассмотреть каждое приложение.

Оценка проводится по следующей формуле:

$$O_{\text{пп}} = \sum_{i=1}^n (Z_i \cdot K_i),$$

где $O_{\text{пп}}$ – интегральная оценка программного продукта;

n – количество критериев сравнения заявленных продуктов согласно таблице критериев;

Z_i – значение выполнения критерия (2 – выполнен, 0 – не выполнен);

K_i – коэффициент значимости критерия сравнения (от 0 до 1).

На основании полученной в ходе анализа информации составим таблицу, в которой будут наглядно видны преимущества и недостатки каждого продукта и общая их оценка. Результат отображен в таблице 3.

Таблица 3 – Критериальная оценка программных продуктов

№	Критерий	Название программного продукта			
		«SecurOS FaceX»	«AnyVision»	«Face-Intellect»	«Vocord FaceControl»
1	Идентификация и аутентификация пользователей	2	2	2	2
2	Управление через веб-интерфейс	1	1	1	1
3	Возможность контролировать доступ к определенным зонам	1	1	0	0
4	Наличие форм регистрации и авторизации	1	2	2	1
5	Возможность использования на слабом устройстве	0	0	0	0
6	Работа в реальном времени	2	2	2	2
7	Бесплатность продукта	0	0	0	0
8	Распознавание номерных знаков автомобилей	1	2	0	0
9	Тепловое сканирование и обнаружение масок	1	1	1	1
10	Наличие интуитивно понятного интерфейса	0	0	2	1
	Интегральная оценка	7,5	10	8,25	6,75

На основании полученных оценок была построена столбчатая диаграмма, отображенная на рисунке 5.

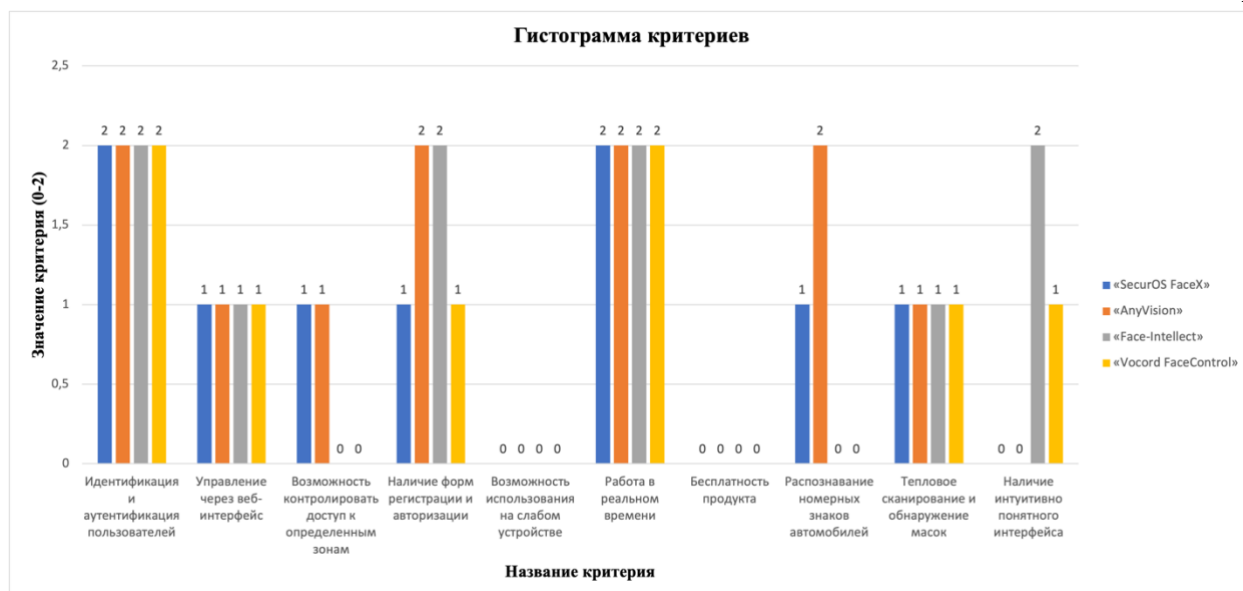


Рисунок 5 – Диаграмма оценок по критериям

Данная диаграмма представляет собой данные в виде разноцветных столбцов. Значение цвета указано в легенде.

На рисунке 6 отображена столбчатая диаграмма итоговых оценок.

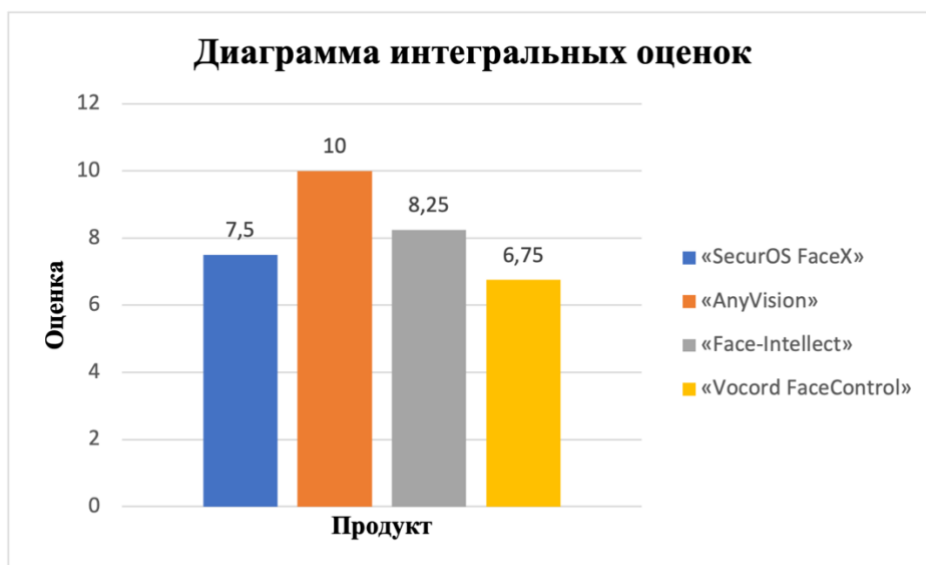


Рисунок 6 – Столбчатая диаграмма интегральных оценок

Выбор конкретного решения для системы контроля доступа на основе распознавания лиц зависит от множества факторов, включая требования к производительности, надежности, функциональности, бюджета и технической инфраструктуры [1]. При выборе системы важно учесть также и долгосрочную перспективу: как система будет масштабироваться, какие затраты будут связаны

с поддержкой и обновлением системы, насколько гибко система может адаптироваться к изменяющимся требованиям и условиям использования.

Подводя итог сравнения, можно сделать вывод, что ни один из рассмотренных сервисов не отвечает всем критериям в полной мере. Каждый из них имеет свои сильные и слабые стороны, а потому выбор конкретного решения должен зависеть от конкретных требований и задач.

1.4 Исследование вариантов использования

Варианты использования определяют, как система должна взаимодействовать с пользователями и что система должна делать для достижения определенных целей.

На основе анализа аналогов была сформирована диаграмма вариантов использования, представленная на рисунке 7.

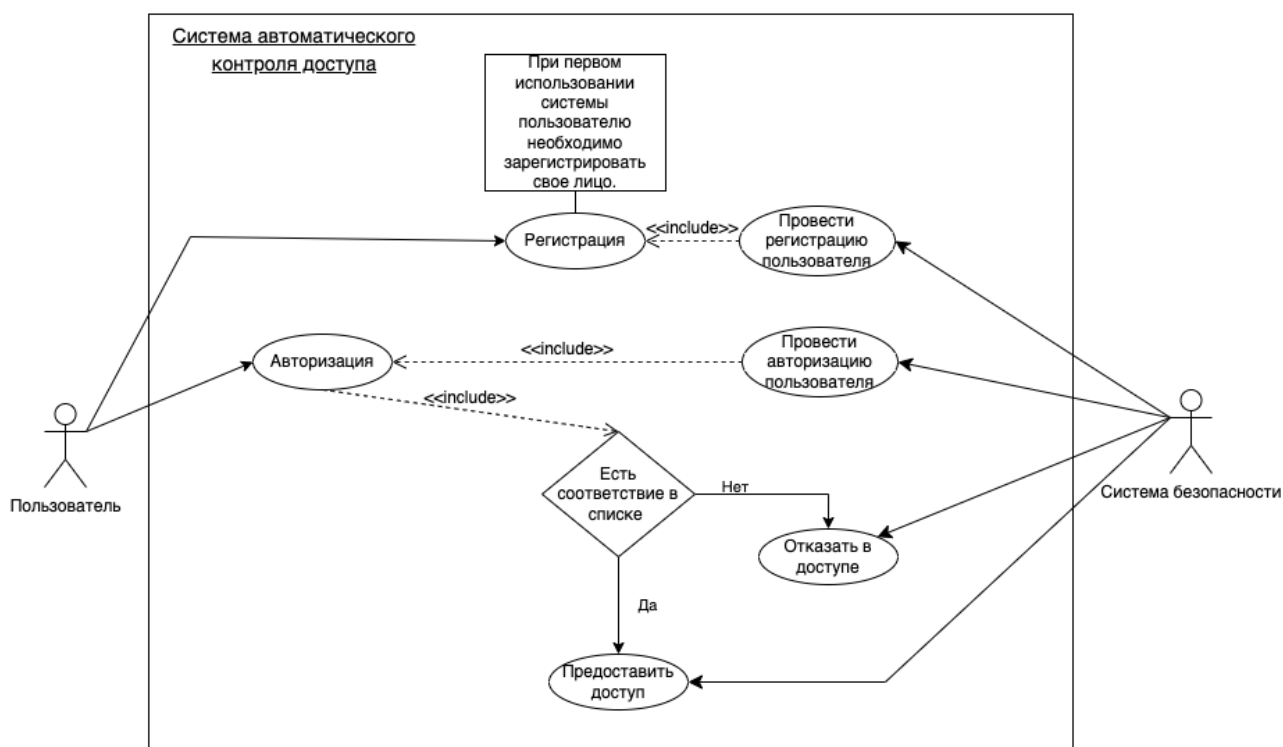


Рисунок 7 – Диаграмма вариантов использования для системы автоматического контроля доступа

Для основных вариантов использования были составлены спецификации. На рисунке 8 представлена спецификации варианта использования «Авторизоваться».

Название: Авторизация
ID: 1
Краткое описание: Система идентифицирует и аутентифицирует пользователя на основе данных о его лице, полученных с помощью технологии распознавания лиц.
Главные актеры: Пользователь, Система Безопасности.
Второстепенные актеры: Нет.
Предусловие: Пользователь уже зарегистрирован в системе контроля доступа и у системы есть данные для сравнения.
Основной поток: <ol style="list-style-type: none"> 1. Пользователь подходит к точке доступа. 2. Система автоматически начинает процесс сканирования лица пользователя. 3. Система сравнивает сканированное лицо с базой данных. Если совпадение найдено, система аутентифицирует пользователя и предоставляет доступ.
Постусловие: Пользователь успешно идентифицирован и аутентифицирован в системе.
Альтернативные потоки: Пользователь не был найден в базе, система выводит сообщение о необходимости повторить попытку входа.

Рисунок 8 – Спецификация варианта использования «Авторизоваться»

На рисунке 9 приведено описание основного порядка выполнения ролями «Пользователь» и «Система Безопасности» варианта использования «Регистрация».

Название: Регистрация
ID: 2
Краткое описание: Пользователь регистрируется в системе, предоставляя необходимую информацию, включая биометрические данные (изображение лица).
Главные актеры: Пользователь, Система Безопасности.
Второстепенные актеры: Нет.
Предусловие: Пользователь еще не зарегистрирован в системе.
Основной поток: <ol style="list-style-type: none"> 1. Пользователь подходит к точке доступа 2. Пользователь вводит требуемую информацию, включая личные данные (имя, контактная информация и т. д.). 3. Пользователь вступает в поле зрения камеры для сканирования лица. 4. Система сканирует лицо пользователя и сохраняет данные в базу данных. Система подтверждает успешную регистрацию пользователя.
Постусловие: Пользователь зарегистрирован в системе и его биометрические данные сохранены в базе данных.
Альтернативные потоки: <ol style="list-style-type: none"> 1. Если система не может правильно сканировать лицо (из-за плохого освещения, технических сбоев и т.д.), она может запросить повторное сканирование. Если пользователь не может предоставить требуемую информацию, регистрация может быть отклонена.

Рисунок 9 – Спецификация варианта использования «Зарегистрироваться»

Исследование вариантов использования помогает лучше понять, как должна функционировать система, и позволяет формализовать требования к функциональности системы.

1.5 Анализ требований

Для создания успешного приложения для системы контроля и управления доступом, основанного на распознавании лиц, необходимо внимательно проанализировать и сформулировать требования, которые должны быть удовлетворены. Эти требования можно разделить на две категории: функциональные и нефункциональные требования.

Проанализировав полученные варианты использования и составив спецификации для основных из них, сформируем функциональные требования.

Система должна:

- обеспечивать аутентификацию и идентификацию пользователей на основе их биометрических данных (лица);
- обеспечивать возможность регистрации и управления посетителями;
- сохранять функциональность и синхронизироваться с сервером после восстановления сети;
- обеспечивать безопасность и конфиденциальность данных пользователей.

Нефункциональные требования определяют, как система должна работать. Система должна обеспечивать быструю обработку запросов и распознавание лиц в реальном времени. Это требует эффективного использования ресурсов и оптимизации алгоритмов.

Для корректной работы приложения системы контроля и управления доступом с использованием распознавания лиц требуется устройство с доступом в интернет. Отдельных требований к мощности системы не предоставляется.

1.6 Постановка задачи

Разрабатываемое приложение должно использовать технологию распознавания лиц для идентификации пользователей и предоставления им доступа и соответствовать приведенным выше требованиям. Благодаря использованию распознавания лиц, система будет способна отслеживать и

контролировать доступ на основе уникальных биометрических характеристик каждого пользователя.

Приложение будет разработано таким образом, чтобы удовлетворять всем приведенным выше требованиям. Оно будет обеспечивать удобство использования для конечных пользователей, высокую точность идентификации, а также надежность и устойчивость к возможным попыткам обхода системы. Кроме того, будет уделено внимание обеспечению конфиденциальности и защиты персональных данных пользователей. Важно подчеркнуть, что успешная реализация этих требований потребует комплексного подхода к разработке, включая тщательное планирование, проектирование и тестирование системы. Это поможет гарантировать, что конечный продукт будет не только функциональным и эффективным, но и безопасным и надежным для использования в реальных условиях.

В таблице 4 описаны категории пользователей информационной системы.

Таблица 4 – Категория пользователей информационной системы

№	Категория пользователей	Наименование приложения	Фрагмент структуры предметной области	Информационные задачи
1	Администратор	Система управления СКУД	Управление базой данных пользователей; управление правилами доступа	Создание и редактирование пользователей; установка и изменение правил доступа
2	Пользователь	Приложение СКУД	Регистрация в системе; доступ к рабочим зонам	Регистрация в системе; получение доступа к определенным зонам

Это позволяет обеспечить безопасность и эффективное управление системой. Так, пользователь из категории «Администратор» обладает наивысшими правами в системе. Ему доступно создание, редактирование и удаление пользователей в системе. Администратор устанавливает и изменяет правила доступа, определяющие, кто, когда и куда может получить доступ.

Пользователь из категории «Пользователь» имеет ограниченные права в системе. Ему доступна регистрация в системе, доступно получение доступа к определенным зонам.

2 Проектирование приложения

2.1 Архитектура приложения

Архитектура приложения определяет его структуру и описывает, как его различные компоненты взаимодействуют друг с другом. Продуманная архитектура позволяет обеспечить устойчивость, масштабируемость, эффективность и безопасность приложения, упрощая его разработку, тестирование и поддержку. В рамках этой работы, архитектура приложения была разработана, учитывая принципы чистой архитектуры и принципы проектирования SOLID Приложение состоит из пользовательского интерфейса, сервер приложения и сервера базы данных.

Пользовательский интерфейс обеспечивает взаимодействие пользователя с системой. Пользовательский интерфейс позволяет пользователю вводить данные, просматривать информацию и управлять системой. Сервер приложений в контексте системы контроля и управления доступом обрабатывает все запросы, полученные от пользователей через пользовательский интерфейс. Это включает аутентификацию и авторизацию пользователей, обработку запросов на доступ и обработку других пользовательских действий [2]. Также сервер приложений взаимодействует с сервером базы данных для хранения и извлечения данных. Это включает сохранение профилей пользователей, а также извлечение этой информации для обработки запросов. Сервер приложений может управлять устройствами контроля доступа, такими как электронные замки и турникеты.

Таким образом, в результате работы над проектом была разработана масштабируемая и гибкая архитектура приложения, которая позволяет легко добавлять новые функции и адаптироваться к изменяющимся требованиям.

Разработанная архитектура СКУД обеспечивает возможность легкого добавления новых модулей и сервисов. Например, если в будущем потребуется внедрить дополнительные методы биометрической идентификации, архитектура системы позволит это сделать без значительных трудозатрат и перестроения всего приложения.

Кроме того, ваша архитектура предусматривает возможность горизонтального масштабирования. Это значит, что при увеличении нагрузки (например, при подключении новых пользователей или устройств) можно просто добавить дополнительные сервера в систему, что позволит обеспечить надлежащую производительность и стабильность работы.

На рисунке 10 изображена схема взаимодействия пользователя с пользовательским интерфейсом, с сервером приложения и сервером базы данных.

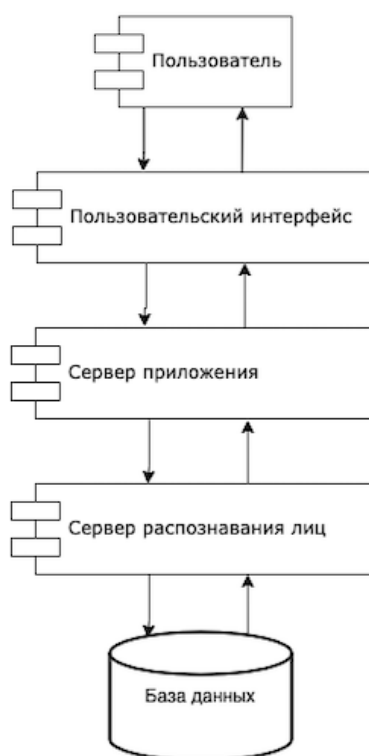


Рисунок 10 – Схема архитектуры разрабатываемого приложения

Как видно из представленного рисунка, каждый компонент обменивается данными с другим, за счет чего система реагирует на действия пользователя и отправляет необходимую информацию.

2.2 Проектирование программных средств

Проектирование программных средств – это критически важный этап в процессе разработки программного обеспечения. На этом этапе осуществляется преобразование требований, собранных на этапе анализа, в структуру программной системы.

Диаграмма деятельности в нотации UML предоставляет возможность представить поведение системы в виде последовательности действий, позволяя легко визуализировать и анализировать процессы и взаимодействие между объектами в системе.

Для варианта использования «Запрос доступа в помещение» была построена диаграмма деятельности. Полученный результат представлен на рисунке 11.

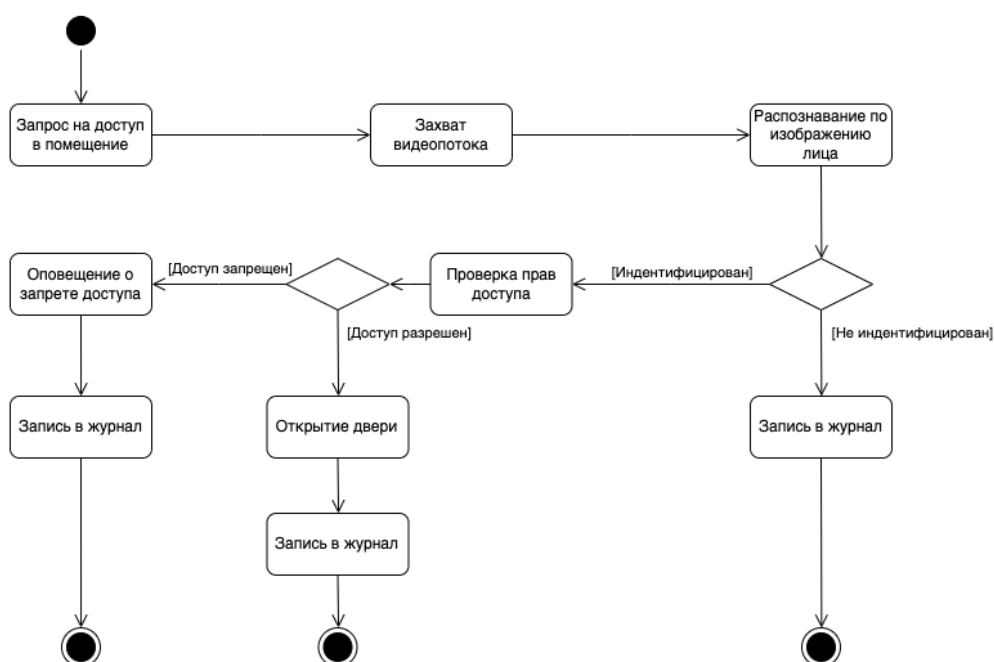


Рисунок 11 – Диаграмма деятельности для варианта использования «Запрос доступа в помещение»

Запуск процесса может произойти, когда пользователь подходит к двери и пытается получить доступ. После того, как пользователь идентифицирован, система проверяет его права доступа. Если у пользователя есть права на доступ в данное помещение, процесс продолжается. Если проверка прошла успешно, система отправляет сигнал для открытия двери. После того, как дверь открыта, пользователь входит в помещение. Процесс завершается.

Для варианта использования «Запрос доступа в помещение» была построена диаграмма деятельности. Полученный результат представлен на рисунке 12.

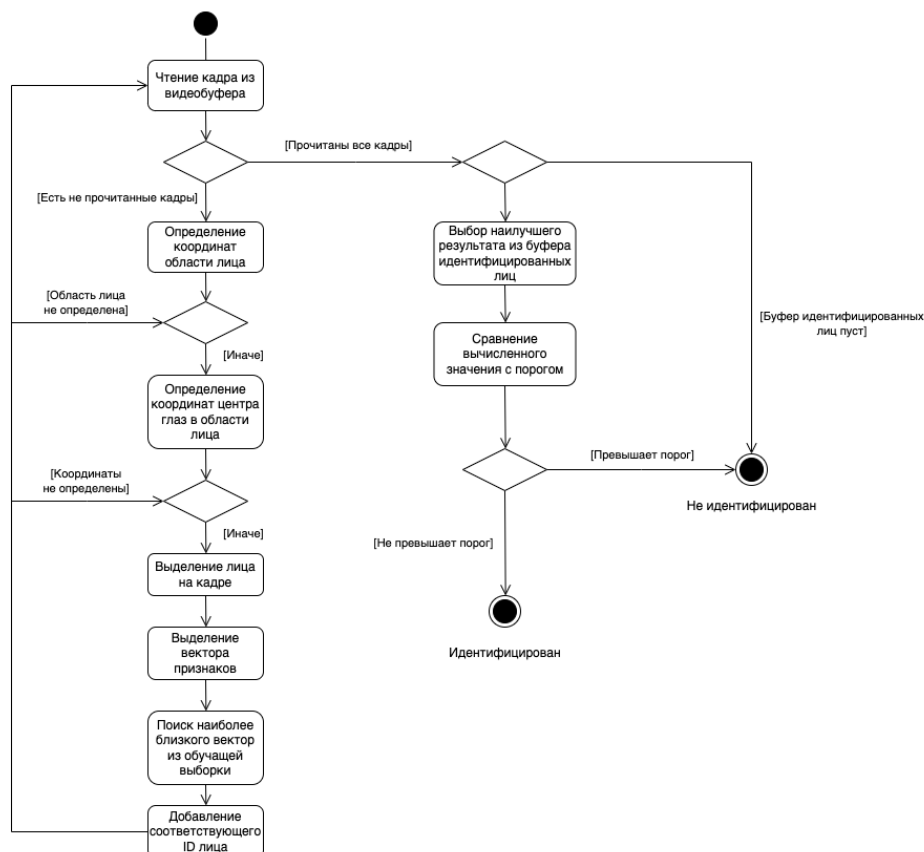


Рисунок 12 – Диаграмма деятельности для варианта использования
«Распознавание лица»

Запуск процесса может произойти, когда пользователь подходит к двери и пытается получить доступ. После того, как пользователь идентифицирован, система проверяет его права доступа. Если у пользователя есть права на доступ в данное помещение, процесс продолжается. Если проверка прошла успешно, система отправляет сигнал для открытия двери. После того, как дверь открыта, пользователь входит в помещение. Процесс завершается.

Захват изображения лица. Это первый шаг в процессе распознавания лица. Затем происходит захваченного изображения, чтобы извлечь ключевые характеристики лица. Система сравнивает извлеченные характеристики лица с базой данных хранящихся профилей лиц. Система примет решение о разрешении или отклонении доступа, принятое на основе результатов сравнения.

2.3 Проектирование интерфейса

Проектирование интерфейса – это задача, основанная на разработке визуального представления и структуры интерактивных элементов продукта (например, веб-сайта, мобильного приложения, программного обеспечения), с целью обеспечения максимального удобства и эффективности взаимодействия пользователя с продуктом.

Проектирование интерфейса включает в себя понимание потребностей, поведения и ожиданий целевой аудитории, определение структуры, расположения элементов управления и навигационных схем для обеспечения удобства использования. Затем прототипы тестируются на пользователях, чтобы определить, насколько удобен интерфейс для использования. На основании обратной связи интерфейс может быть доработан и улучшен.

Интерфейс приложения должен быть дружелюбным, интуитивным и привлекательным для пользователя. Интерфейс должен реализовывать выделенные требования в полной мере. Интерфейс должен быть организован так, чтобы облегчить пользователю взаимодействие с приложением. Это означает, что элементы управления должны быть легко доступны, их функции должны быть понятны из контекста, а ошибка ввода пользователя должна вызывать понятные и информативные сообщения об ошибках. Интуитивность интерфейса означает, что пользователь может понять, как использовать приложение, без необходимости обращения к руководствам или справке. Дизайн интерфейса должен быть приятным и профессиональным, с хорошим сочетанием цветов, шрифтов и изображений.

Все функции и возможности, которые были определены в процессе анализа требований, должны быть реализованы в интерфейсе приложения. Пользователь должен иметь возможность выполнить все задачи, для которых приложение было разработано, через доступные в интерфейсе инструменты и функции.

Цель – сделать продукт удобным и понятным для конечного пользователя.

Для правильной реализации интерфейса необходимо составить макеты. Макет страницы СКУД представлен на рисунке 13.

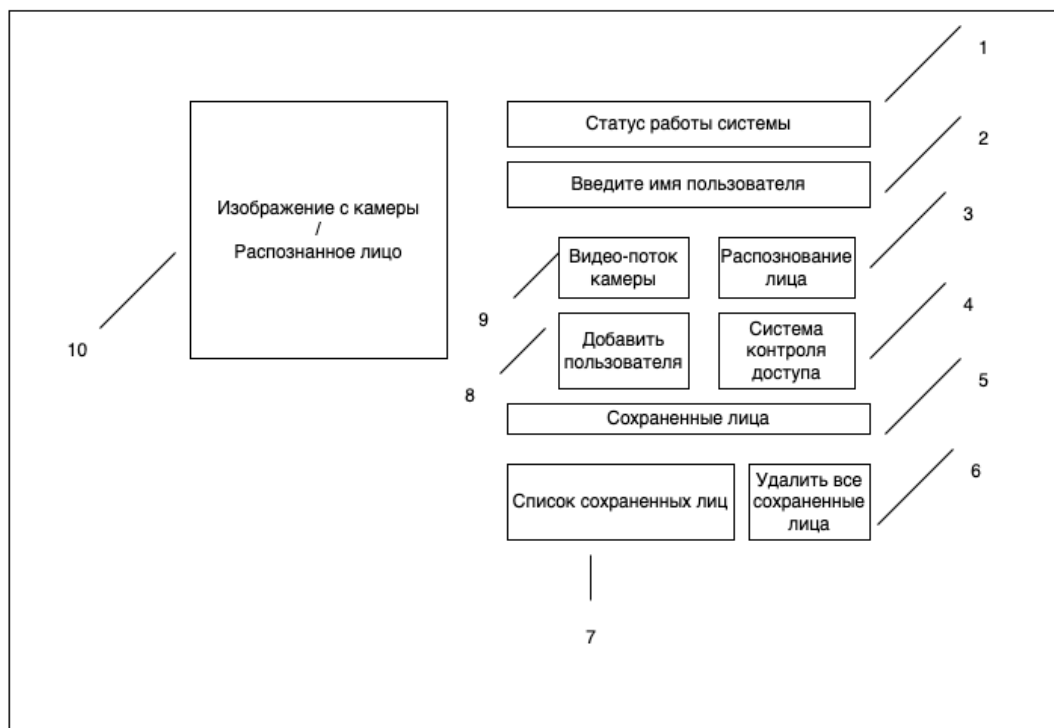


Рисунок 13 – Макет интерфейса СКУД страницы

На рисунке выделены несколько пронумерованных элементов. Элемент 1 представляет собой кнопку статуса работы системы. Элемент 2 представляет собой поле для ввода имени пользователя. Элемент 3 на рисунке представляет собой кнопку, которая используется для распознавания лица. Элемент 4 – кнопка, которая используется для получения доступа к системе контроля доступа. Элемент 5 – текстовое поле. Элемент 6 – кнопка, которая используется для удаления всех сохраненных лиц в системе. Элемент 7 – поле, которое используется для вывода всех сохраненных лиц в системе. Элемент 8 – кнопка, которая используется для добавления нового пользователя в систему. Элемент 9 – кнопка, которая используется для перевода поля камеры в режим просмотра потокового видео. Элемент 10 представляет собой поле для вывода изображения с камеры или режима распознавания лиц.

3 Реализация приложения

Сервер приложения играет центральную роль в архитектуре веб-приложения. Он обрабатывает входящие запросы от клиентов (обычно браузеров или других веб-приложений), выполняет необходимую бизнес-логику и возвращает ответы.

Использование архитектуры API (Application Programming Interface) означает, что взаимодействие между клиентами и сервером структурировано и стандартизировано. API определяет, какие запросы клиенты могут отправлять серверу и какие ответы они могут ожидать получить.

URL (Uniform Resource Locator) используются для определения конкретных ресурсов или функций, доступных через API. Например, URL /users может обращаться к ресурсу «пользователи», а URL /users/123 может обращаться к конкретному пользователю с идентификатором 123.

HTTP методы (GET, POST, PUT, DELETE) определяют, какое действие должно быть выполнено для указанного ресурса. Например, метод GET используется для извлечения данных, метод POST – для создания новых данных, метод PUT – для обновления существующих данных, а метод DELETE – для удаления данных.

Все вместе, эти элементы (сервер приложения, архитектура API, URL и HTTP методы) формируют основу для взаимодействия между клиентами и сервером в веб-приложении. Эти компоненты совместно работают, чтобы обеспечить гладкое взаимодействие между клиентом и сервером в веб-приложении [3].

Сервер приложения служит как основной обработчик запросов, выполняя бизнес-логику и возвращая ответы на клиентские запросы.

Архитектура API определяет конкретные способы взаимодействия с сервером, обозначая форматы запросов и ответов.

URL используются для идентификации конкретных ресурсов в приложении. Они организуют структуру приложения, позволяя пользователям и разработчикам навигировать и манипулировать его ресурсами.

HTTP методы (GET, POST, PUT, DELETE) являются стандартными способами взаимодействия с ресурсами. Они обозначают тип операции, которую нужно выполнить с ресурсом (например, извлечь данные, создать новые данные, обновить существующие данные или удалить данные).

Эта организованная структура обеспечивает четкую, последовательную и эффективную среду для разработки и использования веб-приложения.

Предоставляемые методы представлены в таблице 5.

Таблица 5 – Спецификация API приложения

№	Назначение сервиса	Метод	URI сервиса	Класс контроллера
1	Регистрация нового пользователя	POST	/api/auth/register	AuthController
2	Авторизация пользователя	POST	/api/auth/login	AuthController
3	Получение списка всех пользователей	GET	/api/users	UserController
4	Получение информации о конкретном пользователе	GET	/api/users/{id}	UserController
5	Обновление информации о конкретном пользователе	PUT	/api/users/{id}	UserController
6	Удаление пользователя	DELETE	/api/users/{id}	UserController
7	Получение списка всех правил доступа	GET	/api/access-rules	AccessController
8	Создание нового правила доступа	POST	/api/access-rules	AccessController
9	Обновление правила доступа	PUT	/api/access-rules/{id}	AccessController
10	Удаление правила доступа	DELETE	/api/access-rules/{id}	AccessController
11	Получение списка всех событий	GET	/api/logs	LogController
12	Получение конкретного события	GET	/api/logs/{id}	LogController

Диаграмма компонентов является частью UML (Unified Modeling Language), которая используется для визуализации структуры системы и показывает программные компоненты, связи между ними и их взаимодействия.

В контексте сервера приложения, диаграмма компонентов может использоваться для иллюстрации компонентов, связей, интерфейсов. Это могут быть модули, классы, объекты или другие элементы системы, которые выполняют определенные функции. Например, в приложении могут быть компоненты для управления пользователями, обработки запросов и управления данными. Например, компонент для обработки запросов может взаимодействовать с компонентом управления пользователями для аутентификации запроса.

Диаграмма компонентов сервера приложения полезна, поскольку она позволяет разработчикам и стейкхолдерам понять структуру и взаимодействия системы на высоком уровне. Это может быть полезно при проектировании системы, внесении изменений и устранении неполадок.

На рисунке 14 представлена диаграмма компонентов сервера приложения.

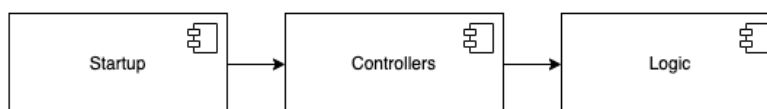


Рисунок 14 – Диаграмма компонентов сервера приложения

Ниже представлено описание каждого компонента:

- компонент Startup – отвечает за логику запуска сервера в сети; благодаря нему приложение может получать доступ по URL;
- компонент Controllers – промежуточный компонент, предназначенный для связи с логикой приложения; сюда включены контроллеры;
- компонент Logic – реализует логику приложения, а именно доступ к базе данных.

Для корректной работы мобильного приложения в связке с сервером и базой данных, необходимо правильно развернуть все компоненты.

Диаграмма развертывания в UML (Unified Modeling Language) обычно используется для визуализации физической архитектуры и развертывания программного обеспечения. Это один из видов структурных диаграмм в UML, которые показывают, как система будет установлена и функционировать на аппаратном обеспечении. Узел представляет собой физическую машину, на которой выполняется часть программного обеспечения[3]. Это может быть сервер, на котором работает ваше веб-приложение, или база данных, на которой хранятся данные. Артефакты представляют собой физические информационные единицы, которые используются и создаются в процессе разработки

программного обеспечения. Это может быть исполняемый файл, скрипт, документация и т. д. Зависимости показывают, как различные узлы и артефакты связаны друг с другом. Например, ваше веб-приложение может зависеть от базы данных.

Для этого была разработана диаграмма развертывания, представленная на рисунке 15.

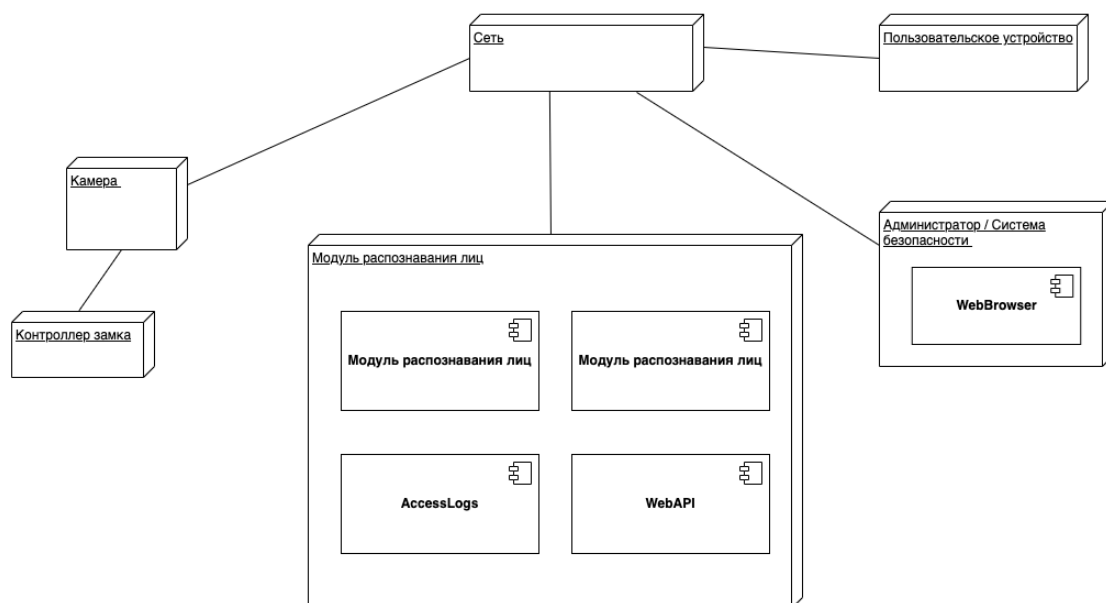


Рисунок 15 – Диаграмма развертывания приложения

Диаграмма развертывания приложения полезна, поскольку она позволяет лучше понять, как и где будет выполняться ваше программное обеспечение. Это особенно полезно при планировании инфраструктуры и развертывания, поскольку она показывает, какие ресурсы будут использоваться и как они будут взаимодействовать друг с другом.

4 Контроль качества программного обеспечения

4.1 Выбор режима тестирования

Тестирование программного обеспечения – это процесс оценки системы или ее компонентов с целью определить, соответствует ли она заданным требованиям. Этот процесс включает в себя запуск программного обеспечения в различных сценариях и условиях для выявления ошибок, проблем с производительностью или несоответствий предварительно определенным стандартам и требованиям.

Существуют различные методы и уровни тестирования, включая модульное тестирование, интеграционное тестирование, системное тестирование и приемочное тестирование. Для разработанного приложения было выбрано функциональное тестирование методом черного ящика, так как оно позволяет рассмотреть продукт от лица пользователя. Функциональное тестирование позволяет проверить поведение системы на соответствие ожидаемым результатам.

Были рассмотрены следующие виды тестирования:

- функциональное тестирование;
- нефункциональное тестирование.

Функциональное тестирование направлено на проверку соответствия функционала программного продукта заранее определённым требованиям. Данный метод тестирования является одним из самых применяемых, так как позволяет наиболее полно проверить программу на соответствие требованиям.

Цель функционального тестирования – гарантировать, что разработанный продукт функционирует в соответствии с требованиями и ожиданиями.

Нефункциональное тестирование – это вид тестирования программного обеспечения, при котором проверяются аспекты системы, не связанные напрямую с определенными пользовательскими действиями или функциями. Эти аспекты включают в себя производительность, надежность, устойчивость, удобство использования, совместимость и безопасность системы.

Было принято решение провести функциональное тестирование, так как оно сосредоточено на проверке того, выполняет ли система свои предполагаемые функции и задачи. Этот вид тестирования важен, так как он напрямую относится к требованиям и ожиданиям пользователей от системы.

4.2 Тестирование серверной части приложения

Для основных вариантов использования были проведены функциональные тесты. Тестирование серверной части приложения – это процесс, который проверяет работоспособность и надежность функционала, который исполняется на стороне сервера. В случае обнаружения ошибок, их исправляют и проводят повторное тестирование.

Тестирование серверной части приложения необходимо, чтобы гарантировать, что она будет работать корректно и надежно в реальных условиях. Это помогает предотвратить проблемы, которые могут возникнуть у пользователей, и обеспечивает высокое качество продукта.

Серверная часть приложения играет ключевую роль в обеспечении работоспособности системы, поэтому ее тестирование необходимо для обеспечения качества продукта.

Функциональные тесты серверной части приложения фокусируются на проверке корректности выполнения функций, которые предоставляет сервер.

Назначение, описание, ожидаемые и полученные результаты приведены в таблице 6.

Таблица 6 – Функциональные тесты серверной части приложения

№ теста	Назначение	Описание	Входные данные	Ожидаемый результат	Актуальный результат
1	Распознавание лиц	Проверка работы системы распознавания лиц	Лицо пользователя	Успешное распознавание лица	Пройден (рисунок В.1)
2	Управление доступом	Проверка системы управления доступом	Запрос доступа от определенного пользователя	Предоставление доступа	Пройден (рисунок В.2)
3	Регистрация активности	Проверка системы регистрации активности	Действия пользователя в системе	Успешное логирование и хранение данных о действиях пользователя	Пройден (рисунок В.3)

Продолжение таблицы 6

№ теста	Назначение	Описание	Входные данные	Ожидаемый результат	Актуальный результат
4	Регистрация нового пользователя	Проверка возможности регистрации нового пользователя в системе	Лицо пользователя	Успешная регистрация и возможность входа в систему	Пройден (рисунок В.4)
5	Удаление пользователя	Проверка возможности удаления пользователя из системы	Идентификатор пользователя, которого следует удалить.	Учетная запись пользователя успешно удалена из системы	Пройден (рисунок В.5)

Эти тесты убеждают, что серверное приложение выполняет все требуемые операции и возвращает ожидаемые результаты.

4.3 Тестирование клиентской части приложения

Для основных функций, представленных на диаграмме вариантов использования, были проведены функциональные тесты. Тестирование клиентской части приложения включает в себя проверку пользовательского интерфейса и взаимодействия пользователя с системой. Это может включать проверку того, что страницы загружаются корректно, элементы интерфейса отображаются и функционируют как ожидается, и данные, введенные пользователем, обрабатываются корректно.

Их перечень и описание представлены в таблице 7.

Таблица 7 – Перечень тестов клиентской части

№ теста	Назначение	Описание	Входные данные	Ожидаемый результат	Актуальный результат
1	Вход в систему	Пользователь пытается войти в систему	Лицо пользователя	Пользователь успешно входит в систему, если лицо распознано и идентифицировано	Пройден
2	Выход из системы	Пользователь хочет выйти из системы.	Пользователь уже вошел в систему.	Пользователь успешно выходит из системы, и его сессия завершается.	Пройден
3	Запрос доступа	Пользователь подает заявку на доступ к определенному помещению	Идентификационные данные пользователя	Запрос на доступ успешно отправлен и появляется соответствующее уведомление	Пройден

Исходя из результатов, полученных в ходе тестирования, можно сделать вывод, что функциональные требования реализованы в полной мере, в том числе и обработка исключительных ситуаций.

4.3 Планирование разработки и оценка бюджета

Планирование разработки и оценка бюджета – это две важные фазы любого проекта разработки программного обеспечения. Они позволяют определить порядок выполнения задач, распределить ресурсы и оценить затраты на проект.

Планирование разработки программного обеспечения – это ключевой этап в жизненном цикле разработки программного обеспечения (Software Development Life Cycle, SDLC). Он включает в себя определение требований к проекту, оценку затрат и ресурсов, а также разработку подробного плана работы.

Существуют различные методологии планирования, такие как Agile, Waterfall и Scrum, каждая из которых имеет свои особенности. Однако, вне зависимости от выбранной методологии, планирование разработки обычно включает в себя следующие шаги [4].

Определение требований. Этот шаг включает в себя определение функциональных и нефункциональных требований к программному обеспечению. Это важный этап, который позволяет разработчикам понять, что именно должно быть реализовано.

Оценка. На основе требований проекта, разработчики оценивают необходимые для выполнения работы ресурсы. Оценка может включать в себя затраты на разработку, тестирование, поддержку и т. д.

Разработка плана. На основе определенных требований и оценок, разрабатывается план работы. Он включает в себя расписание выполнения задач, ресурсы, которые будут использоваться, и т. д.

Оценка бюджета важна для определения общих затрат на проект. Она может включать затраты на разработку, тестирование, поддержку, а также другие затраты, связанные с проектом, например, затраты на обучение персонала, лицензии на программное обеспечение и т. д.

Таким образом, планирование разработки и оценка бюджета – это важные этапы в жизненном цикле разработки программного обеспечения, которые помогают обеспечить успешное выполнение проекта.

При планировании разработки было выделено 23 задачи. Список задач с общей информацией (длительность, дата начала, дата окончания и др.) представлены на рисунке 16.

	i	Название задачи	Длительность	Начало	Окончание	Предшественник	Названия ресурсов
1		Начало проекта	0 дней	Пн 03.04.23	Пн 03.04.23		
2		Анализ требований	9 дней	Пн 03.04.23	Чт 13.04.23		
3		Изучение предметной области	5 дней	Пн 03.04.23	Пт 07.04.23	1	Компьютер[1];Программист
4		Анализ аналогичных приложений	4 дней	Пн 10.04.23	Чт 13.04.23	3	Компьютер[1];Программист
5		Анализ требований завершен	0 дней	Чт 13.04.23	Чт 13.04.23	4	
6		Выбор средств	5 дней	Пн 10.04.23	Пт 14.04.23		
7		Выбор программных средств	3 дней	Пн 10.04.23	Ср 12.04.23	3	Программист ;Компьютер[1]
8		Выбор аппаратных средств	2 дней	Чт 13.04.23	Пт 14.04.23	7	Программист ;Компьютер[1]
9		Завершение выбора средств	0 дней	Пт 14.04.23	Пт 14.04.23	8	
10		Проектирование приложения	10 дней	Пн 17.04.23	Пт 28.04.23		
11		Проектирование пользовательского интерфейса	5 дней	Пн 17.04.23	Пт 21.04.23	9	Программист ;Компьютер[1]
12		Проектирование реализуемых функций	5 дней	Пн 24.04.23	Пт 28.04.23	11	Программист ;Компьютер[1]
13		Завершение проектирования приложения	0 дней	Пт 28.04.23	Пт 28.04.23	12	
14		Реализация приложения	16 дней	Пн 01.05.23	Пн 22.05.23		
15		Разработка пользовательского интерфейса	9 дней	Пн 01.05.23	Чт 11.05.23	13	Компьютер[1];Программист
16		Разработка реализуемых функций	6,22 дней	Пт 12.05.23	Пн 22.05.23	15	Компьютер[1];Программист
17		Завершение реализации приложения	0 дней	Пн 22.05.23	Пн 22.05.23	16	
18		Тестирование	4 дней	Пн 22.05.23	Пт 26.05.23		
19		Модульное тестирование	1,8 дней	Пн 22.05.23	Ср 24.05.23	17	Тестировщик;Компьютер[1]
20		Функциональное тестирование	1,5 дней	Пн 22.05.23	Вт 23.05.23	17	Компьютер[1];Тестировщик
21		Нагрузочное тестирование	1,33 дней	Ср 24.05.23	Чт 25.05.23	19;20	Компьютер[1];Тестировщик
22		Тестирование завершено	0 дней	Чт 25.05.23	Чт 25.05.23	21	
23		Конец проекта	0 дней	Чт 25.05.23	Чт 25.05.23	22	

Рисунок 16 – Список задач проектирования

Для того чтобы успешно выполнить проект, необходимо использовать определенные ресурсы. Список ресурсов, их тип, краткое название, максимальное количество единиц, стандартная ставка, сверхурочная ставка, назначение и некоторые другие параметры, не представляющие особой важности, представлены на рисунке 17.

	i	Название ресурса	Тип	Краткое название	Макс. единиц	Стандартная ставка	Ставка :сверхурочны:	Затраты на исполыз.	Начисление	Базовый календарь
1		Программист	Трудовой	П	200%	р.42 000,00/мес	р.300,00/ч	р.0,00	Пропорциональное	Стандартный
2		Тестировщик	Трудовой	Т	200%	р.39 000,00/мес	р.200,00/ч	р.0,00	Пропорциональное	Стандартный
3		Компьютер	Материальный	К		р.0,00		р.0,00	Пропорциональное	
4		Интернет	Затраты	И					Пропорциональное	

Рисунок 17 – Список ресурсов проекта

После назначения задач была получена полноценная диаграмма Ганта (рисунок 18). Диаграмма Ганта – это инструмент, который позволяет эффективно планировать, координировать и отслеживать конкретные задачи в рамках

проекта. Она показывает все задачи, которые необходимо выполнить, а также время, которое потребуется для их выполнения.

Каждая задача представлена в виде горизонтальной полосы, длина которой соответствует времени, необходимому для выполнения этой задачи. Задачи располагаются вертикально в том порядке, в котором они должны быть выполнены, с самыми ранними задачами вверху диаграммы [5].

Диаграмма Ганта также может показывать зависимости между задачами. Если выполнение одной задачи не может начаться, пока не будет выполнена другая задача, эти две задачи связаны линией на диаграмме.

Использование диаграммы Ганта позволило наглядно представить план разработки, определить критический путь и эффективно распределить ресурсы. Это обеспечивает реалистичность планирования и контроль над выполнением задач, что является критически важным для успешного завершения проекта.

На ней наглядно видно связи между задачами, используемые ресурсы и даты начала и окончания.

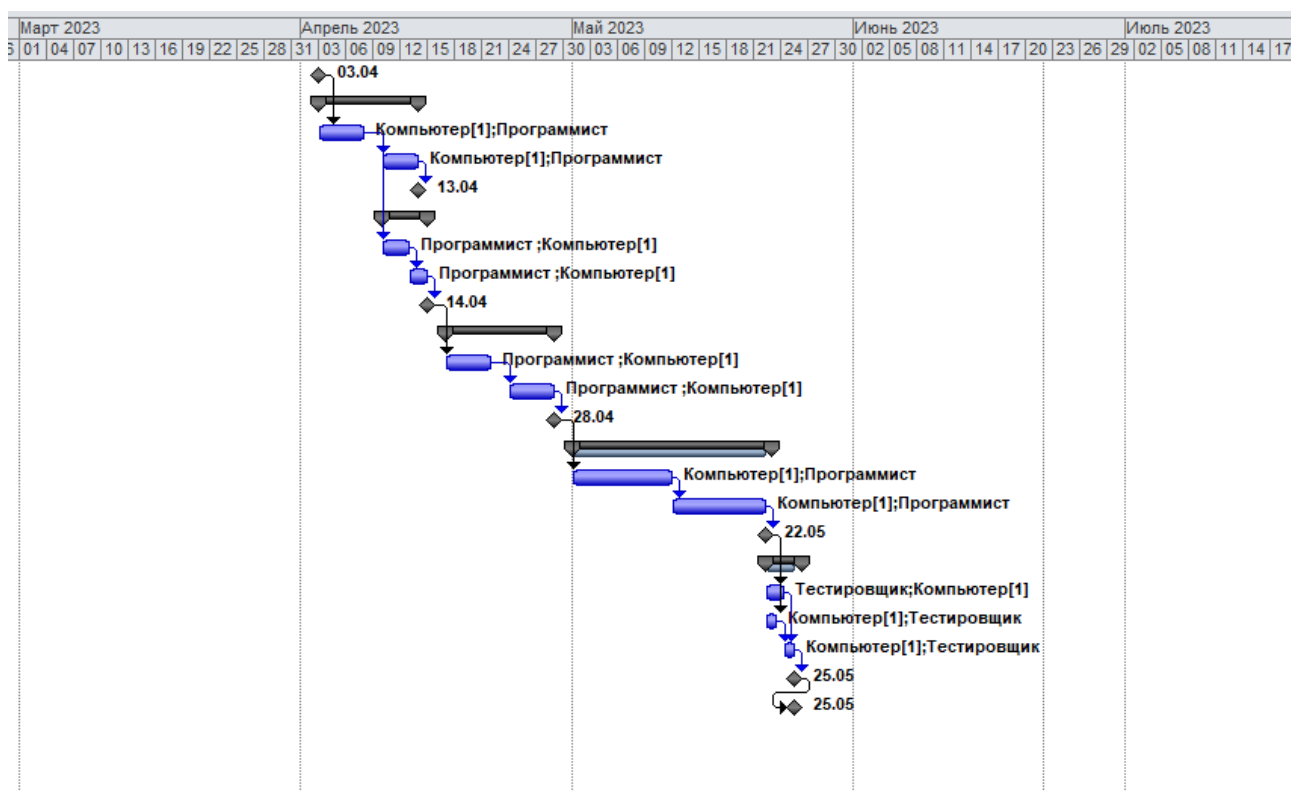


Рисунок 18– Список ресурсов проекта

После составления проекта, определения дат и используемых ресурсов, был создан отчет, представленный на рисунке 19. Отчет отражает общую информацию о проекте, такую как дата начала, дата окончания, трудозатраты на разработку по часам, материальные затраты на ресурсы и общее количество ресурсов.

MSProj11			
на дату Чт 22.06.23			
Даты			
Начало:	Пн 03.04.23	Окончание:	Пт 26.05.23
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
Длительность			
Текущий план:	39,22 дней	Оставшиеся:	39,22 дней
Базовый план:	0 дней	Фактическая:	0 дней
Отклонение:	39,22 дней	Процент завершения:	0%
Трудозатраты			
Текущий план:	350,8 ч	Оставшиеся:	350,8 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	350,8 ч	Процент завершения:	0%
Затраты			
Текущий план:	р.91 391,75	Оставшиеся:	р.91 391,75
Базовый план:	р.0,00	Фактические:	р.0,00
Отклонение:	р.91 391,75		
Состояние задачи		Состояние ресурса	
Еще не начатые задачи:	23	Трудовые ресурсы:	3
Выполняющиеся задачи:	0	Трудовые ресурсы с прев. доступ:	0
Завершенные задачи:	0	Материальные ресурсы:	1
Итого задач:	23	Итого ресурсов:	4

Рисунок 19 – отчет по проекту

В данном отчете представлен ход работы по составлению плана и оценке затрат на реализацию проекта по производству нового вида продукции, разработан сетевой график работ, график Ганта, календарный план и смета проекта.

Заключение

Результатом выполнения выпускной квалификационной работы является разработка и реализация системы контроля и управления доступом. Это приложение позволяет управлять доступом к зданиям и помещениям, используя автоматическое распознавание лиц.

Система была разработана с использованием современных технологий и подходов. Приложение было тщательно протестировано и доказало свою надежность и эффективность.

В ходе выполнения этапа анализа требований была обоснована актуальность разрабатываемого приложения. Рассмотрены приложения-аналоги и осуществлен их обзор, выделены преимущества и недостатки. На основании этих критериев проведена оценка. Результатом стало получение интегральных оценок каждого аналога. Также выявлены функциональных требования и проведен их анализ.

В процессе проектирования приложения системы контроля и управления доступом (СКУД) на основе распознавания лиц была определена его архитектура. Архитектура приложения – это общая организационная структура системы и её компонентов, описывающая их функциональность и взаимодействие друг с другом.

Приложение было протестировано двумя видами тестирования. Серверная часть проверена с помощью модульного тестирования, а клиентская – с помощью функционального тестирования. В результате все тесты были пройдены.

Был составлен план и проведена оценка бюджета. В ходе выполнения данного этапа выявлены временные и финансовые затраты на разработку и необходимые ресурсы. Результатом стала диаграмма Ганта и отчет.

Таким образом, поставленная по реализации СКУД с использованием распознавания лиц задача выполнена в полной мере.

Список использованных источников

1. Ворона, В.А. Системы контроля и управления доступом / В.А. Ворона, В.А. Тихонов. – М.: ГЛТ, 2011. – 272 с.
2. Современные биометрические методы идентификации. [Электронный ресурс]. URL: <https://habrahabr.ru/post/126144/> (дата обращения: 23.06.2023).
3. Как работает Face ID в iPhone X. [Электронный ресурс]. URL: <https://habr.com/ru/post/472948/> (дата обращения: 23.06.2023).
4. Дейт, К. Дж. Введение в системы баз данных, 8–е издание / К. Дж. Дейт; пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 1328 с.
5. Кашкаров А.П. Системы безопасности и устройства кодового доступа. Просто о сложном. М.: ДМК Пресс, 2014. 109 с.

Приложение А
Глоссарий
(справочное)

СКУД (Система Контроля и Управления Доступом) – это комплексное решение, предназначенное для контроля доступа на определенную территорию или в здание, а также для управления правами доступа.

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

UML (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Распознавание лиц: Технология, позволяющая идентифицировать или верифицировать человека по его лицу.

Сервер приложения: Сервер, на котором работает приложение и обслуживает запросы клиентов.

Клиентское устройство: Устройство, используемое для доступа к веб-интерфейсу системы, например, компьютер, планшет или смартфон.

Веб-интерфейс: Пользовательский интерфейс, доступный через веб-браузер.

Приложение Б
Листинг программы
(рекомендуемое)

Листинг Б.1 – Файл Fr_flash.h

```

#pragma once

#ifdef __cplusplus
extern "C"
{
#endif

#include "fr_forward.h"

#define FR_FLASH_TYPE 32
#define FR_FLASH_SUBTYPE 32
#define FR_FLASH_PARTITION_NAME "fr"
#define FR_FLASH_INFO_FLAG 12138

    /
    * @brief Produce face id according to the input aligned face, and save it
    to dest_id and flash.
    *
    * @param l Face id list
    * @param aligned_face An aligned face
    * @return -2 Flash partition not found
    * @return 0 Enrollment finish
    * @return >=1 The left piece of aligned faces should be
    input
    */
    int8_t enroll_face_id_to_flash(face_id_list *l,
                                   dl_matrix3du_t *aligned_face);

    /
    * @brief Produce face id according to the input aligned face, and save the
    id-name pairs to dest_id and flash.
    *
    * @param l Face id list
    * @param new_id An aligned face
    * @param name name corresponding to face id
    * @return -2 Flash partition not found
    * @return 0 Enrollment finish
    * @return >=1 The left piece of aligned faces should be
    input
    */
    int8_t enroll_face_id_to_flash_with_name(face_id_name_list *l,
                                              dl_matrix3d_t *new_id,
                                              char *name);

    /
    * @brief Read the enrolled face IDs from the flash.
    *
    * @param l Face id list
    * @return int8_t The number of IDs remaining in flash
    */
    int8_t read_face_id_from_flash(face_id_list *l);

    /
    * @brief Read the enrolled face IDs and their corresponding names from the
    flash.
    *
    * @param l Face id list
    * @return int8_t The number of IDs remaining in flash
    */
    int8_t read_face_id_from_flash_with_name(face_id_name_list *l);

    /
    * @brief Delete the enrolled face IDs in the flash.
    *
    * @param l Face id list

```

```

    * @return int8_t          The number of IDs remaining in flash
    */
    int8_t delete_face_id_in_flash(face_id_list *l);

    /
    * @brief Delete the enrolled face ID corresponding to the name in the
    flash.
    *
    * @param l                Face id list
    * @param name             The name that needs to be deleted
    * @return int8_t          The number of IDs remaining in flash
    */
    int8_t delete_face_id_in_flash_with_name(face_id_name_list *l, char *name);

    /**
    * @brief Delete all the enrolled face IDs and names paris in the flash.
    *
    * @param l                Face id list
    */
    void delete_face_all_in_flash_with_name(face_id_name_list *l);

#ifdef __cplusplus
}
#endif

```

Листинг Б.2 – Файл Fr_forward.h

```

#pragma once

#ifdef __cplusplus
extern "C"
{
#endif

#include "image_util.h"
#include "dl_lib_matrix3d.h"
#include "frmn.h"

#define FACE_WIDTH 56
#define FACE_HEIGHT 56
#define FACE_ID_SIZE 512
#define FACE_REC_THRESHOLD 0.55

#define LEFT_EYE_X 0
#define LEFT_EYE_Y 1
#define RIGHT_EYE_X 6
#define RIGHT_EYE_Y 7
#define NOSE_X 4
#define NOSE_Y 5
#define LEFT_MOUTH_X 2
#define LEFT_MOUTH_Y 3
#define RIGHT_MOUTH_X 8
#define RIGHT_MOUTH_Y 9

#define EYE_DIST_SET 16.5f
#define NOSE_EYE_RATIO_THRES_MIN 0.49f
#define NOSE_EYE_RATIO_THRES_MAX 2.04f

#define ENROLL_NAME_LEN 16
    typedef struct tag_face_id_node
    {
        struct tag_face_id_node *next;          /*!< next face id node */
        char id_name[ENROLL_NAME_LEN];          /*!< name corresponding to the
face id */
    }

```

```

        dl_matrix3d_t *id_vec;                                /*!< face id */
    } face_id_node;

typedef struct
{
    face_id_node *head;    /*!< head pointer of the id list */
    face_id_node *tail;    /*!< tail pointer of the id list */
    uint8_t count;         /*!< number of enrolled ids */
    uint8_t confirm_times; /*!< images needed for one enrolling */
} face_id_name_list;

typedef struct
{
    uint8_t head;          /*!< head index of the id list */
    uint8_t tail;          /*!< tail index of the id list */
    uint8_t count;         /*!< number of enrolled ids */
    uint8_t size;          /*!< max len of id list */
    uint8_t confirm_times; /*!< images needed for one enrolling */
    dl_matrix3d_t id_list; /*!< stores face id vectors */
} face_id_list;

/
* @brief Initialize face id list.
*
* @param l          Face id list
* @param size       Size of list, one list contains one vector
* @param confirm_times Enroll times for one id
*/
void face_id_init(face_id_list *l, uint8_t size, uint8_t confirm_times);

/
* @brief Initialize face id list with name.
*
* @param l          Face id list
* @param size       Size of list, one list contains one vector
* @param confirm_times Enroll times for one id
*/
void face_id_name_init(face_id_name_list *l, uint8_t size, uint8_t
confirm_times);

/
* @brief Alloc memory for aligned face.
*
* @return dl_matrix3du_t*          Size: 1xFACE_WIDTHxFACE_HEIGHTx3
*/
dl_matrix3du_t *aligned_face_alloc();

/ @{ */
/
* @brief Align detected face to average face according to landmark.
*
* @param onet_boxes    Output of MTMN with box and landmark
* @param src           Image matrix, rgb888 format
* @param dest          Output image
* @return ESP_OK       Input face is good for recognition
* @return ESP_FAIL     Input face is not good for recognition
*/
int8_t align_face_rot(box_array_t *onet_boxes,
                    dl_matrix3du_t *src,
                    dl_matrix3du_t *dest);

int8_t align_face_sim(box_array_t *onet_boxes,
                    dl_matrix3du_t *src,
                    dl_matrix3du_t *dest);

```

```

inline int8_t align_face(box_array_t *onet_boxes,
                        dl_matrix3du_t *src,
                        dl_matrix3du_t *dest)
{
    return align_face_sim(onet_boxes, src, dest);
}
/*@*/

/
* @brief Run the face recognition model to get the face feature
*
* @param aligned_face      A 56x56x3 image, the variable need to do
align_face first
* @return face_id          A 512 vector, size (1, 1, 1, 512)
*/
dl_matrix3d_t *get_face_id(dl_matrix3du_t *aligned_face);

```

Листинг Б.3 – Файл Fd_forward.h

```

#pragma once

#ifdef __cplusplus
extern "C"
{
#endif

#include "image_util.h"
#include "dl_lib_matrix3d.h"
#include "mtmn.h"

typedef enum
{
    FAST = 0,           /*!< fast resize type */
    NORMAL = 1,         /*!< normal resize type */
} mtmn_resize_type;

typedef struct
{
    float score;         /*!< score threshold for filter candidates by
score */
    float nms;           /*!< nms threshold for nms process */
    int candidate_number; /*!< candidate number limitation for each net */
} threshold_config_t;

typedef struct
{
    int w;               /*!< net width */
    int h;               /*!< net height */
    threshold_config_t threshold; /*!< threshold of net */
} net_config_t;

typedef struct
{
    float min_face;      /*!< The minimum size of a detectable
face */
    float pyramid;       /*!< The scale of the gradient scaling
for the input images */
    int pyramid_times;   /*!< The pyramid resizing times */
    threshold_config_t p_threshold; /*!< The thresholds for P-Net. For
details, see the definition of threshold_config_t */
    threshold_config_t r_threshold; /*!< The thresholds for R-Net. For
details, see the definition of threshold_config_t */
    threshold_config_t o_threshold; /*!< The thresholds for O-Net. For
details, see the definition of threshold_config_t */
}

```

```

        mtmn_resize_type type;                /*!< The image resize type. 'pyramid'
will lose efficacy, when 'type'==FAST. */
    } mtmn_config_t;

/
* @brief Get the initial MTMN model configuration
*
* @return mtmn_config_t      MTMN configuration
*/
static inline mtmn_config_t mtmn_init_config()
{
    mtmn_config_t mtmn_config;
    mtmn_config.type = FAST;
    mtmn_config.min_face = 80;
    mtmn_config.pyramid = 0.707;
    mtmn_config.pyramid_times = 4;
    mtmn_config.p_threshold.score = 0.6;
    mtmn_config.p_threshold.nms = 0.7;
    mtmn_config.p_threshold.candidate_number = 20;
    mtmn_config.r_threshold.score = 0.7;
    mtmn_config.r_threshold.nms = 0.7;
    mtmn_config.r_threshold.candidate_number = 10;
    mtmn_config.o_threshold.score = 0.7;
    mtmn_config.o_threshold.nms = 0.7;
    mtmn_config.o_threshold.candidate_number = 1;

    return mtmn_config;
}

/
* @brief Do MTMN face detection, return box and landmark infomation.
*
* @param image_matrix      Image matrix, rgb888 format
* @param config            Configuration of MTMN i.e. score threshold, nms
threshold, candidate number threshold, pyramid, min face size
* @return box_array_t*     A list of boxes and score.
*/
box_array_t *face_detect(dl_matrix3du_t *image_matrix,
                        mtmn_config_t *config);

#ifdef __cplusplus
}
#endif

```

Листинг Б.4 – Файл Camera_pins.h

```

#ifdef CAMERA_MODEL_WROVER_KIT
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM     21
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       19
#define Y4_GPIO_NUM       18
#define Y3_GPIO_NUM       5
#define Y2_GPIO_NUM       4
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

```

```

#elif defined(CAMERA_MODEL_ESP_EYE)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM     4
#define SIOD_GPIO_NUM    18
#define SIOC_GPIO_NUM     23

#define Y9_GPIO_NUM       36
#define Y8_GPIO_NUM       37
#define Y7_GPIO_NUM       38
#define Y6_GPIO_NUM       39
#define Y5_GPIO_NUM       35
#define Y4_GPIO_NUM       14
#define Y3_GPIO_NUM       13
#define Y2_GPIO_NUM       34
#define VSYNC_GPIO_NUM    5
#define HREF_GPIO_NUM     27
#define PCLK_GPIO_NUM     25

#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   15
#define XCLK_GPIO_NUM     27
#define SIOD_GPIO_NUM    25
#define SIOC_GPIO_NUM     23

#define Y9_GPIO_NUM       19
#define Y8_GPIO_NUM       36
#define Y7_GPIO_NUM       18
#define Y6_GPIO_NUM       39
#define Y5_GPIO_NUM        5
#define Y4_GPIO_NUM       34
#define Y3_GPIO_NUM       35
#define Y2_GPIO_NUM       32
#define VSYNC_GPIO_NUM    22
#define HREF_GPIO_NUM     26
#define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_M5STACK_WIDE)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   15
#define XCLK_GPIO_NUM     27
#define SIOD_GPIO_NUM    22
#define SIOC_GPIO_NUM     23

#define Y9_GPIO_NUM       19
#define Y8_GPIO_NUM       36
#define Y7_GPIO_NUM       18
#define Y6_GPIO_NUM       39
#define Y5_GPIO_NUM        5
#define Y4_GPIO_NUM       34
#define Y3_GPIO_NUM       35
#define Y2_GPIO_NUM       32
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     26
#define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_AI_THINKER)
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35

```

```

#define Y8_GPIO_NUM      34
#define Y7_GPIO_NUM      39
#define Y6_GPIO_NUM      36
#define Y5_GPIO_NUM      21
#define Y4_GPIO_NUM      19
#define Y3_GPIO_NUM      18
#define Y2_GPIO_NUM       5
#define VSYNC_GPIO_NUM   25
#define HREF_GPIO_NUM    23
#define PCLK_GPIO_NUM    22

#else
#error "Camera model not selected"
#endif

```

Листинг Б.5 – Файл Esp_camera.h

```

#pragma once

#include "esp_err.h"
#include "driver/ledc.h"
#include "sensor.h"
#include "sys/time.h"

#ifdef __cplusplus
extern "C" {
#endif

/
 * @brief Configuration structure for camera initialization
 */
typedef struct {
    int pin_pwdn;           /*!< GPIO pin for camera power down line */
    int pin_reset;         /*!< GPIO pin for camera reset line */
    int pin_xclk;          /*!< GPIO pin for camera XCLK line */
    int pin_sscb_sda;      /*!< GPIO pin for camera SDA line */
    int pin_sscb_scl;      /*!< GPIO pin for camera SCL line */
    int pin_d7;            /*!< GPIO pin for camera D7 line */
    int pin_d6;            /*!< GPIO pin for camera D6 line */
    int pin_d5;            /*!< GPIO pin for camera D5 line */
    int pin_d4;            /*!< GPIO pin for camera D4 line */
    int pin_d3;            /*!< GPIO pin for camera D3 line */
    int pin_d2;            /*!< GPIO pin for camera D2 line */
    int pin_d1;            /*!< GPIO pin for camera D1 line */
    int pin_d0;            /*!< GPIO pin for camera D0 line */
    int pin_vsync;         /*!< GPIO pin for camera VSYNC line */
    int pin_href;          /*!< GPIO pin for camera HREF line */
    int pin_pclk;          /*!< GPIO pin for camera PCLK line */

    int xclk_freq_hz;      /*!< Frequency of XCLK signal, in Hz. Either
20KHz or 10KHz for OV2640 double FPS (Experimental) */

    ledc_timer_t ledc_timer; /*!< LEDC timer to be used for generating
XCLK */
    ledc_channel_t ledc_channel; /*!< LEDC channel to be used for generating
XCLK */

    pixformat_t pixel_format; /*!< Format of the pixel data: PIXFORMAT_ +
YUV422|GRAYSCALE|RGB565|JPEG */
    framesize_t frame_size; /*!< Size of the output image: FRAMESIZE_ +
QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA */

    int jpeg_quality;      /*!< Quality of JPEG output. 0-63 lower
means higher quality */

```

```

    size_t fb_count;                /*!< Number of frame buffers to be
allocated. If more than one, then each frame will be acquired (double speed) */
} camera_config_t;

/
 * @brief Data structure of camera frame buffer
 */
typedef struct {
    uint8_t * buf;                  /*!< Pointer to the pixel data */
    size_t len;                     /*!< Length of the buffer in bytes */
    size_t width;                   /*!< Width of the buffer in pixels */
    size_t height;                  /*!< Height of the buffer in pixels */
    pixformat_t format;             /*!< Format of the pixel data */
    struct timeval timestamp;        /*!< Timestamp since boot of the first DMA
buffer of the frame */
} camera_fb_t;

#define ESP_ERR_CAMERA_BASE 0x20000
#define ESP_ERR_CAMERA_NOT_DETECTED (ESP_ERR_CAMERA_BASE + 1)
#define ESP_ERR_CAMERA_FAILED_TO_SET_FRAME_SIZE (ESP_ERR_CAMERA_BASE + 2)
#define ESP_ERR_CAMERA_FAILED_TO_SET_OUT_FORMAT (ESP_ERR_CAMERA_BASE + 3)
#define ESP_ERR_CAMERA_NOT_SUPPORTED (ESP_ERR_CAMERA_BASE + 4)

/
 * @brief Initialize the camera driver
 *
 * @note call camera_probe before calling this function
 *
 * This function detects and configures camera over I2C interface,
 * allocates framebuffer and DMA buffers,
 * initializes parallel I2S input, and sets up DMA descriptors.
 *
 * Currently this function can only be called once and there is
 * no way to de-initialize this module.
 *
 * @param config Camera configuration parameters
 *
 * @return ESP_OK on success
 */
esp_err_t esp_camera_init(const camera_config_t* config);

/
 * @brief Deinitialize the camera driver
 *
 * @return
 *     - ESP_OK on success
 *     - ESP_ERR_INVALID_STATE if the driver hasn't been initialized yet
 */
esp_err_t esp_camera_deinit();

/
 * @brief Obtain pointer to a frame buffer.
 *
 * @return pointer to the frame buffer
 */
camera_fb_t* esp_camera_fb_get();

/
 * @brief Return the frame buffer to be reused again.
 *
 * @param fb Pointer to the frame buffer
 */
void esp_camera_fb_return(camera_fb_t * fb);

/

```

```

* @brief Get a pointer to the image sensor control structure
*
* @return pointer to the sensor
*/
sensor_t * esp_camera_sensor_get();

/
* @brief Save camera settings to non-volatile-storage (NVS)
*
* @param key    A unique nvs key name for the camera settings
*/
esp_err_t esp_camera_save_to_nvs(const char *key);

/**
* @brief Load camera settings from non-volatile-storage (NVS)
*
* @param key    A unique nvs key name for the camera settings
*/
esp_err_t esp_camera_load_from_nvs(const char *key);

#ifdef __cplusplus
}
#endif

#include "img_converters.h"

```

Листинг Б.6 – Файл `Img_converters.h`

```

#ifndef _IMG_CONVERTERS_H_
#define _IMG_CONVERTERS_H_

#ifdef __cplusplus
extern "C" {
#endif

#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
#include "esp_camera.h"

typedef size_t (* jpg_out_cb)(void * arg, size_t index, const void* data, size_t len);

/
* @brief Convert image buffer to JPEG
*
* @param src          Source buffer in RGB565, RGB888, YUYV or GRAYSCALE format
* @param src_len      Length in bytes of the source buffer
* @param width        Width in pixels of the source image
* @param height       Height in pixels of the source image
* @param format       Format of the source image
* @param quality      JPEG quality of the resulting image
* @param cb           Callback to be called to write the bytes of the output JPEG
* @param arg          Pointer to be passed to the callback
*
* @return true on success
*/
bool fmt2jpg_cb(uint8_t *src, size_t src_len, uint16_t width, uint16_t height,
pixformat_t format, uint8_t quality, jpg_out_cb cb, void * arg);

/
* @brief Convert camera frame buffer to JPEG
*
* @param fb           Source camera frame buffer

```

```

* @param quality    JPEG quality of the resulting image
* @param cp         Callback to be called to write the bytes of the output JPEG
* @param arg        Pointer to be passed to the callback
*
* @return true on success
*/
bool frame2jpg_cb(camera_fb_t * fb, uint8_t quality, jpg_out_cb cb, void * arg);

/
* @brief Convert image buffer to JPEG buffer
*
* @param src        Source buffer in RGB565, RGB888, YUYV or GRAYSCALE format
* @param src_len    Length in bytes of the source buffer
* @param width      Width in pixels of the source image
* @param height     Height in pixels of the source image
* @param format      Format of the source image
* @param quality    JPEG quality of the resulting image
* @param out        Pointer to be populated with the address of the resulting
buffer.
*                  You MUST free the pointer once you are done with it.
* @param out_len    Pointer to be populated with the length of the output buffer
*
* @return true on success
*/
bool fmt2jpg(uint8_t *src, size_t src_len, uint16_t width, uint16_t height,
pixformat_t format, uint8_t quality, uint8_t out, size_t * out_len);

/
* @brief Convert camera frame buffer to JPEG buffer
*
* @param fb         Source camera frame buffer
* @param quality    JPEG quality of the resulting image
* @param out        Pointer to be populated with the address of the resulting
buffer
* @param out_len    Pointer to be populated with the length of the output buffer
*
* @return true on success
*/
bool frame2jpg(camera_fb_t * fb, uint8_t quality, uint8_t out, size_t *
out_len);

/
* @brief Convert image buffer to BMP buffer
*
* @param src        Source buffer in JPEG, RGB565, RGB888, YUYV or GRAYSCALE
format
* @param src_len    Length in bytes of the source buffer
* @param width      Width in pixels of the source image
* @param height     Height in pixels of the source image
* @param format      Format of the source image
* @param out        Pointer to be populated with the address of the resulting
buffer
* @param out_len    Pointer to be populated with the length of the output buffer
*
* @return true on success
*/
bool fmt2bmp(uint8_t *src, size_t src_len, uint16_t width, uint16_t height,
pixformat_t format, uint8_t out, size_t * out_len);

/
* @brief Convert camera frame buffer to BMP buffer
*
* @param fb         Source camera frame buffer
* @param out        Pointer to be populated with the address of the resulting
buffer

```

```

* @param out_len    Pointer to be populated with the length of the output buffer
*
* @return true on success
*/
bool frame2bmp(camera_fb_t * fb, uint8_t out, size_t * out_len);

/**
* @brief Convert image buffer to RGB888 buffer (used for face detection)
*
* @param src        Source buffer in JPEG, RGB565, RGB888, YUYV or GRAYSCALE
format
* @param src_len    Length in bytes of the source buffer
* @param format     Format of the source image
* @param rgb_buf    Pointer to the output buffer (width * height * 3)
*
* @return true on success
*/
bool fmt2rgb888(const uint8_t *src_buf, size_t src_len, pixformat_t format,
uint8_t * rgb_buf);

#ifdef __cplusplus
}
#endif

#endif /* _IMG_CONVERTERS_H_ */

```

Листинг Б.7 – Файл Esp_http_server.h

```

/
* @brief Function prototype for opening a session.
*
* Called immediately after the socket was opened to set up the send/recv
functions and
* other parameters of the socket.
*
* @param[in] hd      server instance
* @param[in] sockfd  session socket file descriptor
* @return
* - ESP_OK : On success
* - Any value other than ESP_OK will signal the server to close the socket
immediately
*/
typedef esp_err_t (*httpd_open_func_t)(httpd_handle_t hd, int sockfd);

/
* @brief Function prototype for closing a session.
*
* @note It's possible that the socket descriptor is invalid at this point,
the function
* is called for all terminated sessions. Ensure proper handling of
return codes.
*
* @param[in] hd      server instance
* @param[in] sockfd  session socket file descriptor
*/
typedef void (*httpd_close_func_t)(httpd_handle_t hd, int sockfd);

/
* @brief Function prototype for URI matching.
*
* @param[in] reference_uri  URI/template with respect to which the other URI
is matched
* @param[in] uri_to_match   URI/template being matched to the reference
URI/template

```

```

    * @param[in] match_upto      For specifying the actual length of `uri_to_match`
up to
    *                          which the matching algorithm is to be applied (The
maximum
    *                          value is `strlen(uri_to_match)`, independent of
the length
    *                          of `reference_uri`)
    * @return true on match
    */
typedef bool (*httpd_uri_match_func_t)(const char *reference_uri,
                                       const char *uri_to_match,
                                       size_t match_upto);

/
* @brief   HTTP Server Configuration Structure
*
* @note    Use HTTPD_DEFAULT_CONFIG() to initialize the configuration
*          to a default value and then modify only those fields that are
*          specifically determined by the use case.
*/
typedef struct httpd_config {
    unsigned    task_priority;      /*!< Priority of FreeRTOS task which runs
the server */
    size_t      stack_size;        /*!< The maximum stack size allowed for the
server task */

    /
    * TCP Port number for receiving and transmitting HTTP traffic
    */
    uint16_t     server_port;

    /
    * UDP Port number for asynchronously exchanging control signals
    * between various components of the server
    */
    uint16_t     ctrl_port;

    uint16_t     max_open_sockets; /*!< Max number of sockets/clients connected
at any time*/
    uint16_t     max_uri_handlers; /*!< Maximum allowed uri handlers */
    uint16_t     max_resp_headers; /*!< Maximum allowed additional headers in
HTTP response */
    uint16_t     backlog_conn;     /*!< Number of backlog connections */
    bool         lru_purge_enable; /*!< Purge "Least Recently Used" connection
*/
    uint16_t     recv_wait_timeout; /*!< Timeout for recv function (in
seconds)*/
    uint16_t     send_wait_timeout; /*!< Timeout for send function (in
seconds)*/

```

Листинг Б.8 – Файл FaceDoorEntry.cpp

```

out_res.face_id = get_face_id(aligned_face);
    last_detected_millis = millis();
    if (g_state == START_DETECT) {
        client.send("FACE DETECTED");
    }

    if (g_state == START_ENROLL)
    {
        int left_sample_face = do_enrollment(&st_face_list,
out_res.face_id);
        char enrolling_message[64];
        sprintf(enrolling_message, "SAMPLE NUMBER %d FOR %s",
ENROLL_CONFIRM_TIMES - left_sample_face, st_name.enroll_name);

```

```

        client.send(enrolling_message);
        if (left_sample_face == 0)
        {
            ESP_LOGI(TAG, "Enrolled Face ID: %s", st_face_list.tail->id_name);
            g_state = START_STREAM;
            char captured_message[64];
            sprintf(captured_message, "FACE CAPTURED FOR %s",
st_face_list.tail->id_name);
            client.send(captured_message);
            send_face_list(client);
        }
    }

    if (g_state == START_RECOGNITION && (st_face_list.count > 0))
    {
        face_id_node *f = recognize_face_with_name(&st_face_list,
out_res.face_id);
        if (f)
        {
            char recognised_message[64];
            sprintf(recognised_message, "DOOR OPEN FOR %s", f->id_name);
            open_door(client);
            client.send(recognised_message);
        }
        else
        {
            client.send("FACE NOT RECOGNISED");
        }
    }
    dl_matrix3d_free(out_res.face_id);
}

else
{
    if (g_state != START_DETECT) {
        client.send("NO FACE DETECTED");
    }
}

    if (g_state == START_DETECT && millis() - last_detected_millis > 500) { //
Detecting but no face detected
        client.send("DETECTING");
    }

}

    client.sendBinary((const char *)fb->buf, fb->len);

    esp_camera_fb_return(fb);
    fb = NULL;
}
}

```

Приложение В
Результаты тестирования
(рекомендуемое)

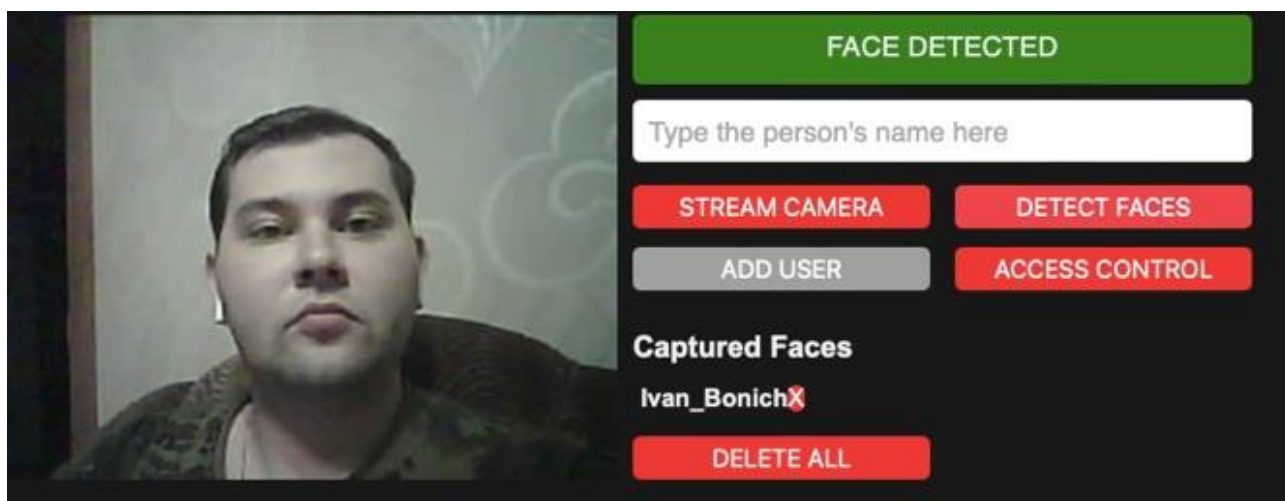


Рисунок В.1 – Результат теста №1

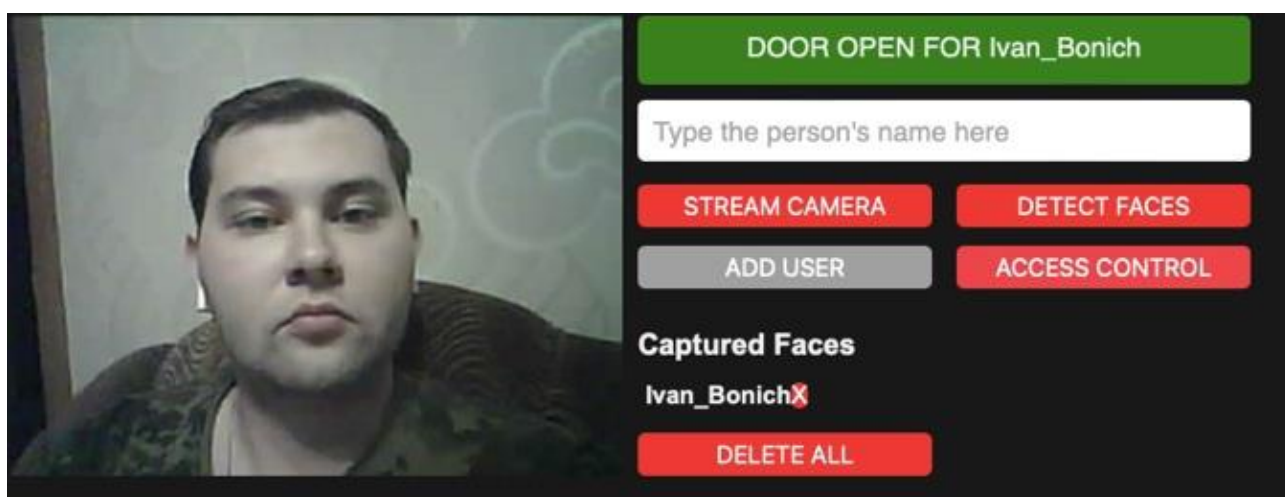


Рисунок В.2 – Результат теста №2

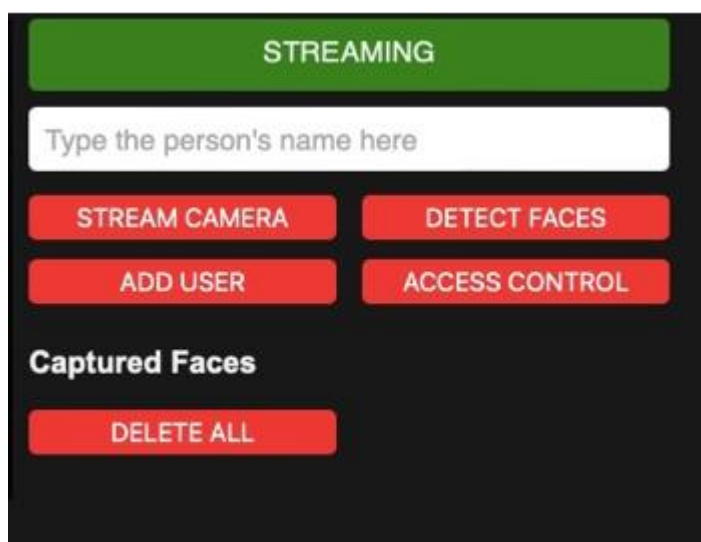


Рисунок В.3 – Результат теста №3

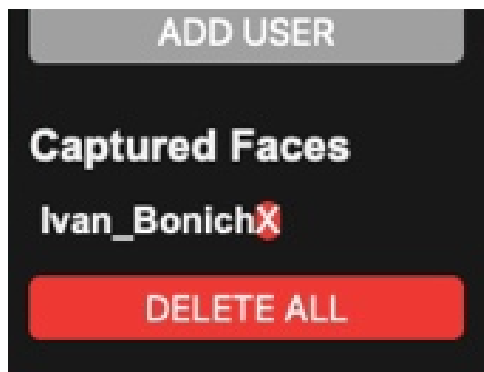


Рисунок В.4 – Результат теста №4

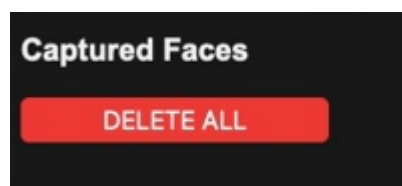


Рисунок В.5 – Результат теста №5