

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий
Кафедра Информационных систем и технологий
Специальность 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА

по дисциплине «Программирование мобильных систем»
Тема «Приложение для просмотра фотографий и видео»

Исполнитель

студентка 3 курса 7 группы

подпись, дата

Е.В.Конашук

Руководитель

асс.

преподавателя

должность, ученая степень, ученое звание

подпись, дата

Н. И. Уласевич

Допущен(а) к защите _____

дата, подпись

Курсовой проект защищен с оценкой _____

Руководитель _____
подпись

дата

Н. И. Уласевич
инициалы и фамилия

Содержание

Введение	4
1 Постановка задачи и обзор аналогичных решений	5
1.1 Обзор аналогичных решений	5
1.1.1 Instagram	5
1.1.2 Pinterest	6
1.1.3 Snapchat	7
1.2 Постановка задачи	7
1.3 Выводы по разделу	8
2 Анализ требований к программному средству и разработка функциональных требований	9
2.1 Технические средства разработки	9
2.2 Описание разрабатываемой функциональности приложения	9
2.3 Спецификация функциональных требований	10
2.4 Выводы по разделу	10
3 Проектирования приложения	11
3.1 Диаграмма вариантов использования	11
3.2 Проектирование базы данных	12
3.3 Описание информационных объектов и ограничений целостности	13
3.3.1 Таблица Profiles	13
3.3.2 Таблица UserPhotos	13
3.3.3 Таблица UserVideos	14
3.3.4 Таблица Comments	14
3.3.5 Таблица VideoComments	14
3.3.6 Таблица Likes	15
3.3.7 Таблица VideoLikes	15
3.3.8 Таблица Subscriptions	15
3.3.9 Таблица CommentLikes	15
3.3.10 Таблица PhotoCommentLikes	16
3.3.11 Таблица UserNotes	16
3.3.12 Таблица PhotoReports	16
3.3.13 Таблица VideoReports	16
3.4 Анализ решений	17
3.4.1 Выбор библиотек и технологий	17
3.4.2 Выбор средств программирования	18
3.5 Выводы по разделу	18
4 Реализация приложения	19
4.1 Технические средства разработки	19
4.2 Подключение базы данных	19
4.3 Разработка мобильного приложения	20
4.4 Выводы по разделу	25
5 Анализ информационной безопасности приложения	26
5.1 Защита пользовательских данных	26
5.2 Выводы по разделу	26

6 Тестирование приложения	27
6.1 Тестирование клиентской области	27
6.2 Выводы по разделу	28
7 Руководство пользователя	29
7.1 Пользователь.....	29
7.2 Администратор	35
7.3 Выводы по разделу.....	37
Заключение	38
Список литературы.....	39
Приложение А. Диаграмма вариантов использования пользователя.....	40
Приложение Б. Диаграмма вариантов использования администратора.....	41
Приложение В.....	42

Введение

В настоящее время стремительно набирают популярность мобильные приложения, которые предоставляют пользователю все необходимые услуги. Это возможно благодаря стремительно развивающимся технологиям, которые дают доступ пользователю к учебе, развлечениям или просто к нужным услугам. Разработка мобильного приложения для просмотра коротких видеороликов даёт возможность его пользователям отвлечься и всегда быть в курсе современных трендов. Такие видео удобно просматривать в любом месте, где бы не находился пользователь: дом, транспорт, очередь. Данное приложение поможет скоротать время. Алгоритмы персонализации удерживают внимание пользователей, предлагая контент по их интересам.

Кроме того, пользователи имеют возможность не только потреблять контент, но и создавать его. Каждый пользователь может вести свой блог, загружая собственные видеоролики. Это создает сообщество активных участников, которые могут обмениваться идеями, делиться опытом и взаимодействовать друг с другом.

Целью данного курсового проекта является разработка мобильного приложения для просмотра фотографий и видео, где пользователи могут не только просматриваться ленту с фотографиями и видео, но и публиковать свои собственные.. Будут исследованы различные аспекты данного приложения такие как функциональные возможности, способы взаимодействия с базой данных, обработка ошибок и валидация данных, пользовательский интерфейс.

Курсовой проект должен быть завершен выводами по каждому разделу работы и заключением, включающий вывод по проделанной работе.

В конечном итоге, создание мобильного приложения для просмотра коротких видеороликов не только предоставляет пользователям развлекательную платформу, но и создает возможности для самовыражения и взаимодействия, что делает его актуальным и востребованным в современном цифровом мире.

1 Постановка задачи и обзор аналогичных решений

1.1 Обзор аналогичных решений

Одним из первых пунктов создания программного средства является анализ прототипов и литературных источников. На сегодняшний день можно встретить множество программных решений со схожей тематикой. Рассмотрим наиболее популярные и удобные приложения по просмотру фотографий и видео.

1.1.1 Instagram

«Instagram» – это социальная сеть, которая позволяет пользователям делиться фотографиями и видео[1]. Данная социальная сеть позволяет пользователю загружать фото и видео, вести прямые эфиры, обмениваться фотографиями, подписываться на других пользователей. Интерфейс приложения «Instagram» представлен на рисунке 1.1.

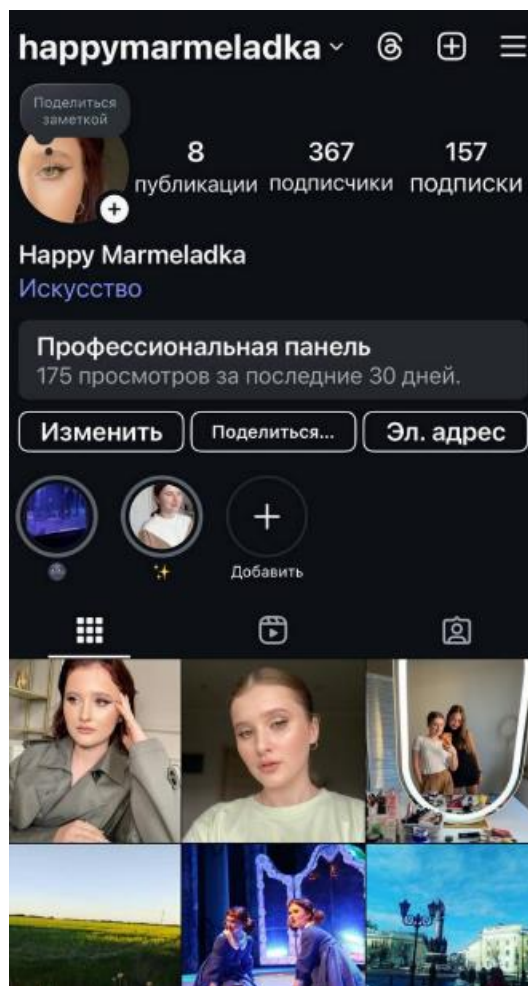


Рисунок 1.1 – Интерфейс приложения «Instagram»

«Instagram» предоставляет пользователю удобный и понятный интерфейс для просмотра фото и видео, а также содержит необходимые функции для отслеживания статистики: количество лайков и комментариев, подписчиков и подписок.

К недостаткам приложения «Instagram» можно отнести: загруженный интерфейс, обилие функций, которые редко используются.

1.1.2 Pinterest

«Pinterest» – второй аналог приложения по просмотру фото и видео[2]. Данное приложение позволяет искать и сохранять изображения, которые потом можно объединить в коллекции.

Важным преимуществом «Pinterest» является кроссплатформенность. Приложение доступно на различных платформах, включая Windows, MacOS, Android и IOS. Программное средство предлагает простой и понятный интерфейс, что делает его доступным для широкого круга пользователей. Минусами «Pinterest» являются проблемы с авторским правом, изображения отображаются не по актуальности, а по алгоритмам, что иногда затрудняет поиск информации.



Рисунок 1.2 – Интерфейс приложения «Pinterest»

«Pinterest» объединяет функции создания тематических досок для организации фото по интересам, удобный поиск по ключевым словам, темам и категориям.

К недостаткам приложения «Pinterest» можно отнести: иногда трудно найти оригинальные источники изображений, так как некоторые доски имеют неактуальные или несуществующие ссылки.

1.1.3 Snapchat

«Snapchat» – третий аналог приложения по просмотру фото и видео[3].

Snapchat предлагает множество фильтров, линз и эффектов, которые делают фотографии и видео более интересными и веселыми. В приложении есть встроенные игры, возможность добавления опросов и вопросов в истории, что способствует взаимодействию с подписчиками. Редакторы видео и фото, позволяющие добавлять текст, рисунки и стикеры, что увеличивает творческие возможности пользователей.

Snapchat регулярно добавляет новые функции и обновления, что делает приложение актуальным и интересным.

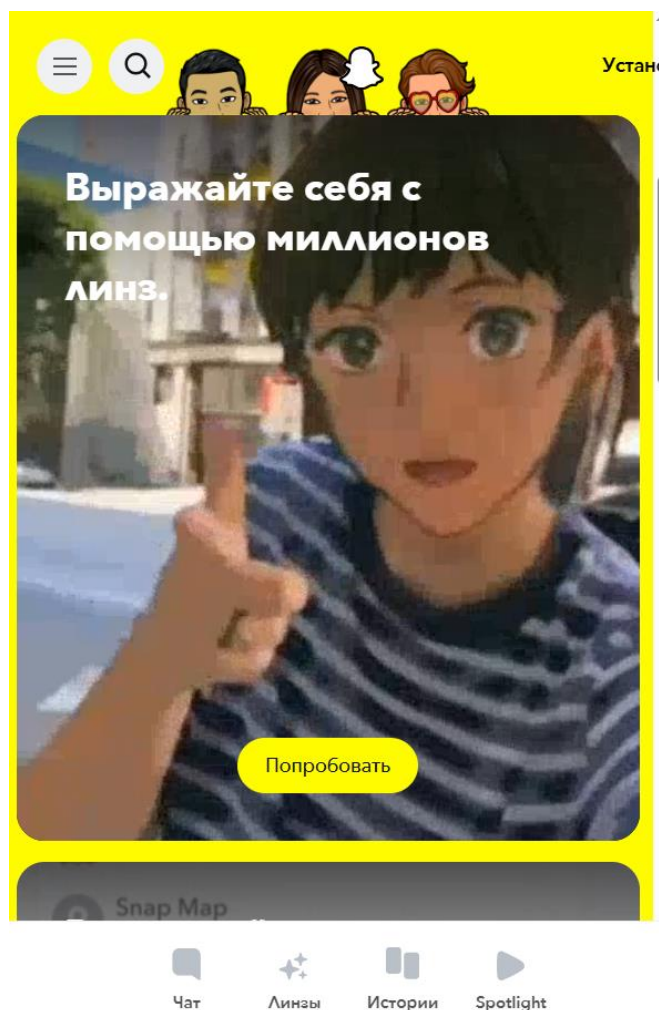


Рисунок 1.3 – Интерфейс приложения «Snapchat»

Из недостатков можно выделить исчезновение фотографий и видео через 24 после публикации.

1.2 Постановка задачи

Основываясь на анализе аналогичных решений из пункта 1.1, проектируемое программное средство для просмотра фото и видео:

1. Понятный интерфейс: пользователи должны иметь доступ к интуитивно понятному и визуально привлекательному интерфейсу, который облегчит публикацию и просмотр фотографий и видео.

2. Гибкость и настраиваемость: пользователи должны иметь возможность подписываться и отписываться на других пользователей.

3. Публикация комментариев: пользователи должны иметь возможность оставлять комментарии под фотографиями и видео.

4. Поиск пользователей: пользователи могут искать пользователей по названию аккаунтов, просматривать комментарии, просмотр подписок и подписчиков других пользователей.

1.3 Выводы по разделу

В данной главе был проведен анализ существующих программных средств для просмотра фото и видео, таких как «Instagram», «Pinterest» и «Snapchat», выявлены их достоинства и недостатки, были сформулированы основные требования к проектируемому программному средству. Разработка нового программного средства для просмотра фото и видео должна основываться на положительном опыте существующих решений и учитывать их недостатки.

Применение современных технологий и подходов в разработке поможет улучшить функциональность и удобство использования.

Разработка нового программного средства должна опираться на положительный опыт существующих решений, таких как быстрая загрузка контента и персонализация, но при этом учитывать их недостатки, чтобы избежать аналогичных проблем.

Таким образом, разработка нового программного средства должна не только восполнять пробелы, выявленные в существующих решениях, но и предлагать инновационные функции, которые сделают просмотр фото и видео более эффективным и безопасным.

Все, описанное в данной главе, важно для создания высококачественного продукта.

2 Анализ требований к программному средству и разработка функциональных требований

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы. Этот раздел начнём с технических средств, которые использовались для разработки программного средства, после чего перейдём к архитектуре системы и диаграмме вариантов использования.

2.1 Технические средства разработки

Для создания «Приложение для просмотра фотографий и видео» были выбраны передовые технологии и инструменты, чтобы обеспечить высокое качество и удобство использования.

В процессе разработки был использован язык программирования Dart, фреймворк Flutter.

Flutter — это платформа с открытым исходным кодом, который разработан и поддерживается Google. Flutter упрощает процесс создания единообразных привлекательных пользовательских интерфейсов для приложения на шести поддерживаемых платформах.

2.2 Описание разрабатываемой функциональности приложения

Программное средство предоставляет пользователю следующие функциональные возможности:

- добавление изображения в профиль;
- публикация комментариев;
- создание профиля пользователя;
- подписка на пользователя;
- добавление видео;
- создание и просмотр заметки;
- просмотр подписок и подписчиков;
- возможность ставить лайки на понравившиеся фотографии и видео;
- возможность комментировать фотографии и видео;
- поиск пользователей.

2.3 Спецификация функциональных требований

Для функциональности приложения необходимо создание базы данных для хранения информации приложения. Подробно о базе данных описано в главе 3.

В «Приложение для просмотра фотографий и видео» при запуске необходимо реализовать регистрацию и авторизацию пользователей для дальнейшего использования приложения. Введенные данные, успешно прошедшие валидацию, заносятся в базу данных, а пользователь получает доступ к приложению. Для регистрации пользователя необходимо придумать имя аккаунта, логин и пароль. Для авторизации входными параметрами являются логин и пароль пользователя, которые содержатся в базе данных.

Таким образом, в ходе работы над этим разделом были сформулированы основные функциональные требования для проектирования программного средства.

2.4 Выводы по разделу

В данной главе была проведена детальная аналитика требований к проектируемому «Приложению для просмотра фотографий и видео». Анализ требований является ключевым этапом разработки, так как он формирует четкие рамки проекта.

Таким образом, сформулированные требования и спецификации в данной главе создают прочную основу для дальнейшей разработки программного средства. Этот этап является важным шагом к достижению целей проекта и его успешной реализации.

Разработка данного приложения направлена на создание удобной платформы для просмотра фотографий и видео, которая позволит пользователям скоротать время, просматривая интересующий их контент.

3 Проектирования приложения

3.1 Диаграмма вариантов использования

Проектирование является критическим этапом в процессе разработки программного обеспечения, который, к сожалению, часто пренебрегают новички в этой области. Обычно начинающие разработчики стремятся держать все детали в голове или, в лучшем случае, записывают некоторые ключевые моменты на листке бумаги. В результате у них отсутствует четкий план действий, что может привести к тому, что проект будет отложен на неопределенное время.

Обычно при проектировании разработчики представляют систему в графическом виде, так как этот формат легко воспринимается человеком. Именно поэтому вместо того, чтобы писать объемные тексты о каждой функции будущей программы, разработчики создают различные диаграммы для описания своих систем. Это помогает помнить, что нужно реализовать в программе, и быстро вводить в курс дела.

Диаграмма вариантов использования (use-case diagram) – это диаграмма, которая описывает, какой функционал разрабатываемой программной системы доступен каждой группе пользователей.

Суть данной диаграммы заключается в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой через так называемые варианты использования. Актером в данном контексте является любой объект, субъект или внешняя система, взаимодействующая с моделируемой системой. Вариант использования, в свою очередь, представляет собой спецификацию сервисов или функций, которые система предоставляет актеру. Иными словами, каждый вариант использования описывает определенный набор действий, выполняемых системой при взаимодействии с актером, при этом не раскрывая деталей внутренней реализации этих действий.

Перед началом разработки программного средства крайне важно четко определить роли актеров, их задачи и полный перечень вариантов использования. Это позволяет сформировать ясное представление о функциональных возможностях системы и ее взаимодействии с пользователями. Для достижения этой цели необходимо построить диаграмму вариантов использования, которая наглядно отображает связи между актерами и функциями системы. Диаграмма вариантов использования для пользователей представлена в приложениях А и Б, а также на рисунке 3.1.

Зарегистрированный пользователь имеет различные функциональные возможности. После авторизации он может создать свой профиль, публиковать фотографии и видео, писать комментарии, подписываться на других пользователей. Администратор может удалять фото и пользователей.

Таким образом, диаграмма вариантов использования служит важным инструментом для проектирования системы, позволяя четко структурировать функциональные требования и определить зоны ответственности каждого актера. Это способствует созданию удобного и эффективного программного средства, отвечающего потребностям всех участников взаимодействия.

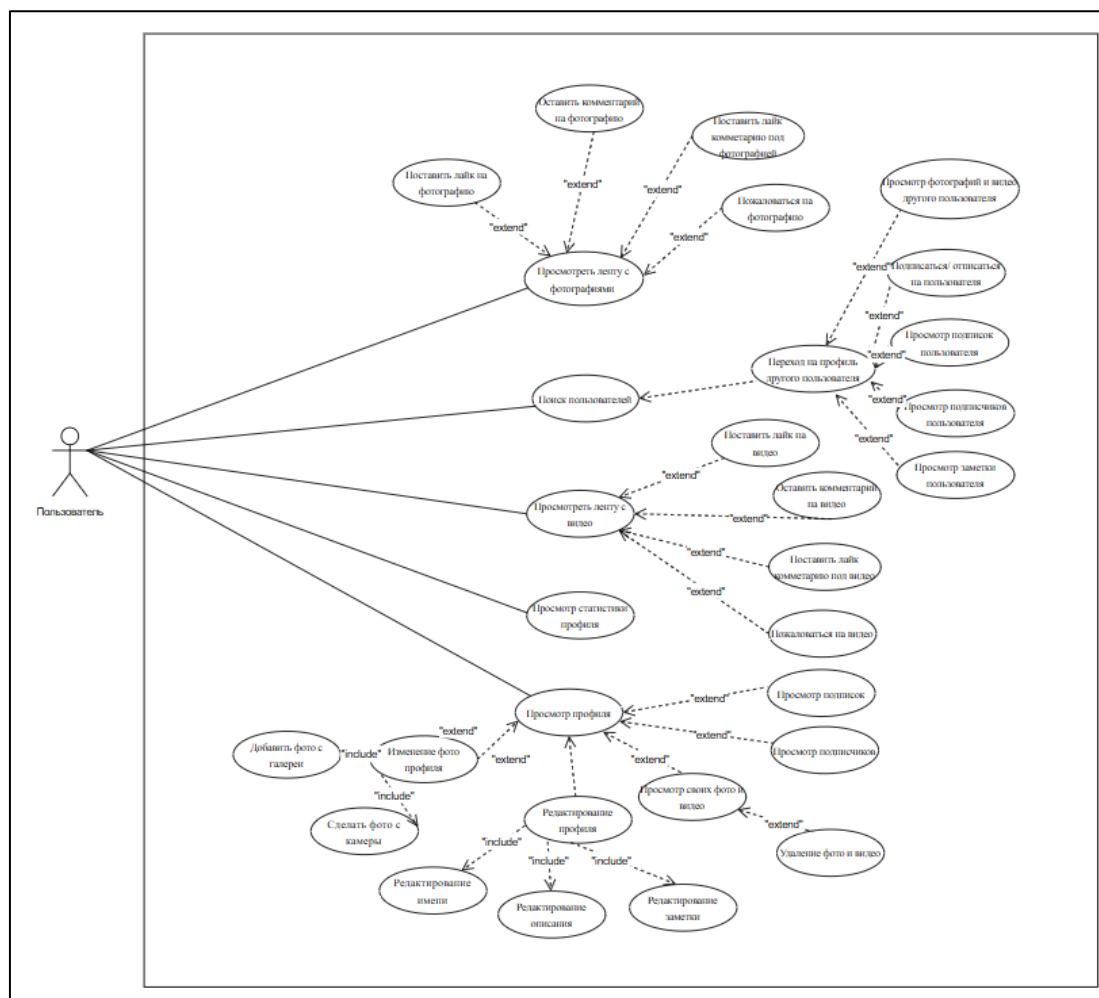


Рисунок 3.1 – Диаграмма вариантов использования

Таким образом были определены границы функциональных возможностей разрабатываемого приложения.

3.2 Проектирование базы данных

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Модель данных играет ключевую роль в проектировании базы данных, так как она определяет структуру хранения данных и способы их обработки.

Для данного курсового проекта была спроектирована база данных, реализована в Supabase. Supabase — это бесплатный аналог Firebase, полифункциональная платформа, объединяющая в себе несколько важных программных решений и упрощающая их реализацию до предельно примитивного уровня, чтобы даже новички в разработке могли спокойно добавить в свои приложения или сайты такие функции, как авторизация, хранилище файлов, обновление контента на сайте в реальном времени и т.п[4].

PostgreSQL в Supabase — это мощный инструмент для разработчиков, сочетающий надежность и гибкость PostgreSQL с удобством платформы Backend-as-a-Service.

Для понимания структуры разрабатываемой базы данных была создана логическая схема, представленная в приложении В.

Созданная в Supabase база данных содержит в себе всего 13 таблиц: Comment_likes, Comments, Likes, Photo_comment_likes, Profiles, Subscriptions, User_notes, User_photos, User_videos, Video_comments, Video_likes, PhotoReport, VideoReport.

3.3 Описание информационных объектов и ограничений целостности

3.3.1 Таблица Profiles

Таблица «Profiles» будет содержать информацию о пользователях. Структура представлена в таблице 3.1.

Таблица 3.1 – Описание таблицы Profiles

Название	Тип данных	Описание
ID	uuid	Уникальный идентификатор профиля (первичный ключ)
UserName	text	Имя пользователя
Email	text	Электронная почта пользователя
Bio	text	Биография пользователя
Avatar_url	text	URL аватара пользователя
Total_posts	integer	Общее количество постов пользователя (по умолчанию 0)
Followers_count	integer	Количество подписчиков (по умолчанию 0)
Subscriptions_count	integer	Количество подписок (по умолчанию 0)
Role	text	Роль пользователя (по умолчанию 'user')

3.3.2 Таблица UserPhotos

Таблица «UserPhotos» будет содержать информацию о фотографиях пользователей. Структура представлена в таблице 3.2.

Таблица 3.2 – Описание таблицы UserPhotos

Название	Тип данных	Описание
ID	uuid	Уникальный идентификатор фотографии (первичный ключ)
UserId	uuid	ID пользователя (внешний ключ к Profiles.Id)
PhotoUrl	text	URL фотографии (обязательное поле)
CreatedAt	timestamp	Время создания записи (по умолчанию текущее время)
PhotoData	text	Описание фотографий
LikesCount	integer	Количество лайков (по умолчанию 0)
CommentsCount	integer	Количество комментариев (по умолчанию 0)

3.3.3 Таблица UserVideos

Таблица «UserVideos» будет содержать информацию о видео пользователя. Структура представлена в таблице 3.3.

Таблица 3.3 – Описание таблицы UserVideos

Название	Тип данных	Описание
ID	uuid	Уникальный идентификатор видео (первичный ключ)
UserId	uuid	ID пользователя (внешний ключ к Profiles.Id)
VideoUrl	text	URL видео (обязательное поле)
CreatedAt	timestamp	Время создания видео (по умолчанию текущее время)
LikesCount	integer	Количество лайков (по умолчанию 0)
CommentsCount	integer	Количество комментариев (по умолчанию 0)
Description	text	Описание фотографий

3.3.4 Таблица Comments

Таблица «Comments» будет содержать информацию о комментариях о фото. Структура представлена в таблице 3.4.

Таблица 3.4 – Описание таблицы Comments

Название	Тип данных	Описание
ID	uuid	Уникальный идентификатор комментария фотографии (первичный ключ)
PhotoId	uuid	ID фотографии (внешний ключ к UserPhotos.Id)
UserId	uuid	ID пользователя (внешний ключ к Profiles.Id)
Content	text	Текст комментария (обязательное поле)
CreatedAt	timestamp	Время создания (по умолчанию текущее время)
LikesCount	integer	Общее количество лайков комментария публикации (по умолчанию 0)
AvatarUrl	text	URL аватара автора комментария
Username	text	Имя пользователя (обязательное поле)

3.3.5 Таблица VideoComments

Таблица «VideoComments» будет содержать информацию о комментариях о видео. Структура представлена в таблице 3.5.

Таблица 3.5 – Описание таблицы VideoComments

Название	Тип данных	Описание
ID	uuid	Уникальный идентификатор комментария видео (первичный ключ)
VideoId	uuid	ID видео (внешний ключ к UserVideos.Id)
UserId	uuid	ID пользователя (внешний ключ к Profiles.Id)
Content	text	Текст комментария (обязательное поле)
CreatedAt	timestamp	Время создания (по умолчанию текущее время)
Username	text	Username

3.3.6 Таблица Likes

Таблица «Likes» будет содержать информацию о лайках фотографий у всех пользователей. Структура представлена в таблице 3.6.

Таблица 3.6 – Описание таблицы Likes

Название	Тип данных	Описание
ID	seria	Уникальный идентификатор лайка (первичный ключ)
PhotoId	uuid	ID фотографии (внешний ключ к UserPhotos.Id)
UserId	uuid	ID пользователя (внешний ключ к Profiles.Id)
CreatedAt	timestamp	Время создания (по умолчанию текущее время)

3.3.7 Таблица VideoLikes

Таблица «VideoLikes» будет содержать информацию о лайках видео. Структура представлена в таблице 3.7.

Таблица 3.7 – Описание таблицы VideoLikes

Название	Тип данных	Описание
ID	seria	Уникальный идентификатор лайка (первичный ключ)
VideoId	uuid	ID видео (внешний ключ к UserVideos.Id)
UserId	uuid	ID пользователя (внешний ключ к Profiles.Id)
CreatedAt	timestamp	Время создания (по умолчанию текущее время)

3.3.8 Таблица Subscriptions

Таблица «Subscriptions» будет содержать информацию о пользователях. Структура представлена в таблице 3.8.

Таблица 3.8 – Описание таблицы Subscriptions

Название	Тип данных	Описание
ID	uuid	Уникальный идентификатор подписки (первичный ключ)
FollowerId	uuid	ID подписчика (внешний ключ к Profiles.Id)
FollowingId	uuid	ID пользователя, на которого подписались (внешний ключ к Profiles.Id)

3.3.9 Таблица CommentLikes

Таблица «CommentLikes» будет содержать информацию о лайках комментария у видео. Структура представлена в таблице 3.9.

Таблица 3.9 – Описание таблицы CommentLikes

Название	Тип данных	Описание
ID	seria	Уникальный идентификатор лайка (первичный ключ)
CommentId	uuid	ID комментария (внешний ключ к UserComments.Id)
UserId	uuid	ID пользователя (внешний ключ к Profiles.Id)
CreatedAt	timestamp	Время создания (по умолчанию текущее время)

3.3.10 Таблица PhotoCommentLikes

Таблица «PhotoCommentLikes» будет содержать информацию о лайков комментариев у фото. Структура представлена в таблице 3.10.

Таблица 3.10 – Описание таблицы PhotoCommentLikes

Название	Тип данных	Описание
ID	serial	Уникальный идентификатор лайка (первичный ключ)
CommentId	uuid	ID комментария (внешний ключ к Comments.Id)
UserId	uuid	ID пользователя (внешний ключ к Profiles.Id)
CreatedAt	timestamp	Время создания (по умолчанию текущее время)

3.3.11 Таблица UserNotes

Таблица «UserNotes» будет содержать информацию о пользователях. Структура представлена в таблице 3.11.

Таблица 3.11 – Описание таблицы UserNotes

Название	Тип данных	Описание
ID	uuid	Уникальный идентификатор заметки (первичный ключ)
UserId	uuid	ID пользователя, создавшего заметку (внешний ключ к Profiles.Id)
NoteText	text	Текст заметки
CreatedAt	timestampz	Время создания (по умолчанию текущее время)

3.3.12 Таблица PhotoReports

Таблица «PhotoReports» будет содержать информацию о жалобах пользователей на фотографии. Структура представлена в таблице 3.12.

Таблица 3.12 – Описание таблицы PhotoReports

Название	Тип данных	Описание
ID	uuid	Уникальный идентификатор жалобы (первичный ключ)
PhotoId	uuid	ID фотографии
ReportedBy	uuid	ID пользователя который пожаловался
Reason	text	Причина жалобы
CreatedId	timestamp	Время написания жалобы

3.3.13 Таблица VideoReports

Таблица «VideoReports» будет содержать информацию о жалобах пользователей на видео. Структура представлена в таблице 3.13.

Таблица 3.13 – Описание таблицы VideoReports

Название	Тип данных	Описание
ID	uuid	Уникальный идентификатор жалобы (первичный ключ)
VideoId	uuid	ID видео
ReportedBy	uuid	ID пользователя который пожаловался
Reason	text	Причина жалобы

Продолжение таблицы 3.13

CreatedId	timestamp	Время написания жалобы
-----------	-----------	------------------------

3.4 Анализ решений

При создании программного средства одним из важных моментов является выбор языка программирования, так как от него зависит множество факторов, таких как скорость создания программы, скорость и удобство тестирования, возможность миграции на другие платформы, возможность быстрого внесения изменений и так далее. Также стоит понимать, что идеального языка программирования нет, так как все разные инструменты и имеют свои преимущества и недостатки при разработке.

3.4.1 Выбор библиотек и технологий

В качестве платформы для разработки приложения был выбран Flutter, который зарекомендовал себя как мощный и универсальный инструмент для создания кроссплатформенных мобильных приложений. Flutter — это бесплатный и открытый набор средств разработки мобильного пользовательского интерфейса, созданный компанией Google и впервые представленный в мае 2017 года [5]. Основное преимущество Flutter заключается в возможности создания высокопроизводительных приложений для различных платформ с использованием единой кодовой базы. Это означает, что разработчик может написать один массив кода, который будет работать как на iOS, так и на Android, что существенно сокращает время и затраты на разработку, тестирование и поддержку двух отдельных приложений.

Для разработки приложений на Flutter используется язык программирования Dart, также разработанный компанией Google и впервые представленный в октябре 2011 года. За последние годы Dart значительно эволюционировал, став современным и удобным языком для создания пользовательских интерфейсов. Dart ориентирован на упрощение разработки фронтенд-компонентов, что делает его идеальным выбором для создания мобильных и веб-приложений. Он обладает простым синтаксисом, поддерживает объектно-ориентированное программирование и предоставляет разработчикам инструменты для создания динамичных и отзывчивых интерфейсов. Кроме того, Dart поддерживает компиляцию как в машинный код (AOT — Ahead of Time), так и в промежуточный код (JIT — Just in Time), что обеспечивает высокую производительность приложений и гибкость в процессе разработки.

Выбор Flutter и Dart для разработки приложения обусловлен их способностью обеспечить высокую производительность, кроссплатформенность и удобство в создании сложных пользовательских интерфейсов. Эти технологии позволяют сократить время выхода продукта на рынок, минимизировать затраты на разработку и обеспечить единообразный пользовательский опыт на разных устройствах. Таким образом, использование Flutter предоставляет надежную основу для создания современного, функционального и конкурентоспособного мобильного приложения, отвечающего требованиям пользователей и бизнеса.

Dart фокусируется на развитии вёрстки веб-страниц; его можно с легкостью использовать для создания мобильных и веб-приложений.

3.4.2 Выбор средств программирования

В качестве среды разработки кода программы была выбрана интегрированная среда разработки Android Studio – это официальная интегрированная среда разработки (IDE) для создания приложений Android. Она предлагает ряд мощных функций и инструментов, которые помогают разработчикам создавать высококачественные приложения для Android. Android Studio предоставляет удобный интерфейс и множество возможностей для облегчения процесса разработки. Среди них: редактор кода, который обеспечивает интеллектуальные функции автодополнения, проверки синтаксиса и рефакторинга кода; инструменты для дизайна интерфейсов, позволяющие визуально создавать и настраивать пользовательские интерфейсы, используя инструмент "Design View"; эмулятор Android, позволяющий тестировать приложения на различных устройствах без необходимости физического устройства.

Для разработки и управления базой данных использовался пакет Supabase. Пакет Supabase — это бесплатный аналог облачных сервисов Google Firebase (БД, аутентификация, хранение файлов, realtime обмен данными, framework для популярных языков). Перечисленные инструменты позволяют разработчикам быстро создавать как простые мобильные приложения и сайты, так и сложные корпоративные системы.

3.5 Выводы по разделу

В завершение данной главы были выполнены следующие задачи: спроектирована логическая схема базы данных, описана UML-диаграмма, определены основные алгоритмы.

Каждый аспект приложений был тщательно проработан с учетом потребностей и ожиданий конечных пользователей, что обеспечивает их полное удовлетворение и комфорт при работе с приложениями. Также было произведено определение общей архитектуры проекта и базы данных. Были описаны технологии для разработки и приведено назначение основных блоков приложений.

4 Реализация приложения

4.1 Технические средства разработки

Для реализации курсового проекта использован язык программирования Dart и фреймворк Flutter.

Flutter - это кросс-платформенный фреймворк с открытым исходным кодом, разработанный компанией Google. Flutter предоставляет обширную библиотеку готовых виджетов, которые можно использовать для построения интерфейса приложения.

Dart - это объектно-ориентированный, открытый язык программирования, разработанный компанией Google. Он используется в качестве основного языка для разработки мобильных приложений с помощью фреймворка Flutter.

Supabase - это облачная платформа разработки приложений, предоставляемая компанией Google.

4.2 Подключение базы данных

Для работы с базой данных Supabase в файл main.dart необходимо написать код, представленный на листинге 4.1.

```
WidgetsFlutterBinding.ensureInitialized();

await Supabase.initialize(
  url: 'https://iymoexinovejjpmizdry.supabase.co', Supabase URL

  anonKey:
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6Im15bW9leGlub3ZlampwbWl6ZHJ5Iiwicm9sZSI6ImFub24iLCJpYXQiOiE3NDE5NjUwNDksImV4cCI6MjA1NzU0MTA0OX0.xNZC0eTLRosBlbMG1RVyQUFq7K3KgxeRS4nBn9NoH8U', );
```

Листинг 4.1 – Подключение базы данных

Перед добавлением вышенаписанного кода в файл pubspec.yaml необходимо добавить библиотеки для работы Supabase.

```
sqflite: ^2.4.2
supabase_flutter: ^2.8.4
uuid: ^4.5.1
```

Листинг 4.2 – Библиотеки в файле pubspec.yaml

При помощи данных библиотек и кода, написанного выше, была подключения база данных Supabase для разрабатываемого приложения.

4.3 Разработка мобильного приложения

Мобильное приложение построено с использованием нескольких папок, каждая из которых выполняет определенную функцию и обеспечивает структурированное и логичное разделение кода.

- Data включает классы, которые описывают данные;
- Instagram содержит компоненты, которые обеспечивают работу с базой данных;

- Screens представляет основные экраны приложения.

Структура проекта представлена на рисунке 4.3.

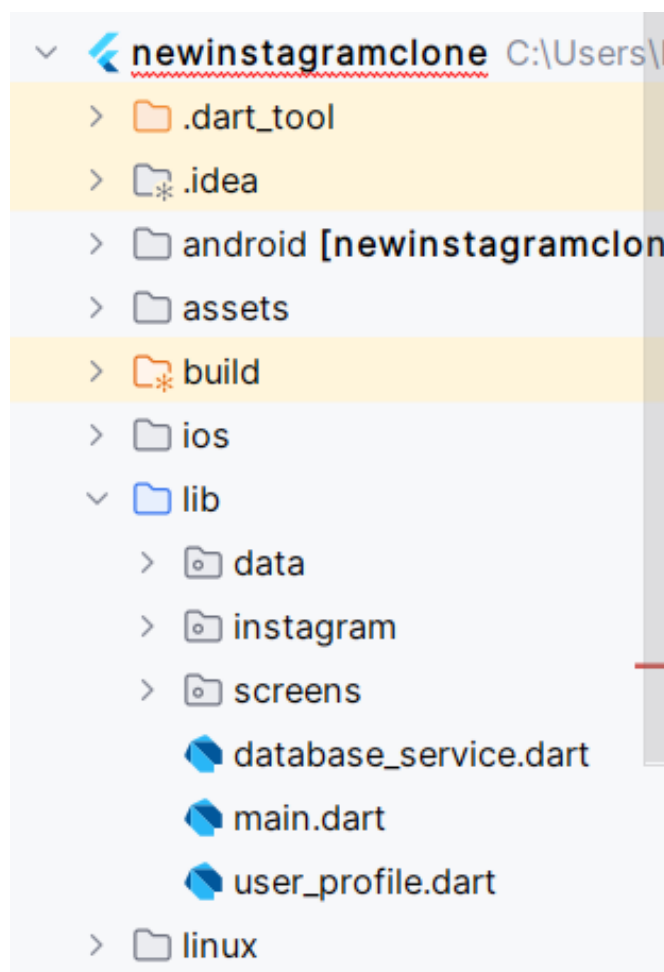


Рисунок 4.3 – Структура проекта

Данная структура позволяет легко ориентироваться в проекте, упрощает процесс разработки и поддержки, а также обеспечивает возможность повторного использования компонентов, что повышает эффективность разработки.

В разработке приложения важную роль играет создание согласованного дизайна, который улучшает пользовательский опыт. Класс `AppThemes` определяет три темы оформления для приложения: `lightTheme`, `darkTheme` и `pinkTheme`. Эти темы, созданные с использованием класса `ThemeData` из пакета `flutter/material.dart`, задают стили для ключевых элементов интерфейса, таких как цвета фона, текста, кнопок, иконок и панели приложений, обеспечивая единообразный вид приложения.

Светлая тема построена на тёплой персиковой палитре с кремовым фоном, который создаёт мягкую и уютную атмосферу. Панель приложений использует нежно-розовый оттенок, что обеспечивает читаемость и элегантность. Такая цветовая схема делает светлую тему идеальной для приложений, стремящихся к дружелюбному и современному дизайну.

Тёмная тема ориентирована на комфорт в условиях низкой освещённости. Она использует палитру с чёрным фоном для основного экрана и тёмно-серым для панели приложений. Текст и иконки выполнены в белом цвете, что гарантирует контраст и удобство чтения. Эта тема подходит для пользователей, предпочитающих тёмный режим для снижения нагрузки на глаза.

Розовая тема представляет собой светлый дизайн с акцентом на яркие розовые оттенки. Фон экрана выполнен в пастельном розовом, а панель приложений, иконки, кнопки и плавающая кнопка действия используют насыщенный розовый цвет. Текст оформлен в тёмно-коричневом оттенке, который гармонично контрастирует с фоном.

В качестве примера в листинге 4.3 представлен код светлой темы оформления.

```
static final lightTheme = ThemeData(
  primarySwatch: Colors.amber,
  brightness: Brightness.light,
  scaffoldBackgroundColor: Color(0xFFF8F1F1),
  appBarTheme: AppBarTheme(
    backgroundColor: Color(0xFFFFE4E1),
    foregroundColor: Colors.black87,
  ),
  textTheme: TextTheme(
    bodyLarge: TextStyle(color: Color(0xFF2F4F4F)),
    bodyMedium: TextStyle(color: Color(0xFF2F4F4F)),
    titleLarge: TextStyle(color: Color(0xFF2F4F4F), fontWeight:
FontWeight.bold),
    titleMedium: TextStyle(color: Color(0xFF2F4F4F)),
    titleSmall: TextStyle(color: Color(0xFF2F4F4F)),
  ),
  iconTheme: IconThemeData(color: Color(0xFFFF7F50)),
  elevatedButtonTheme: ElevatedButtonThemeData(
    style: ElevatedButton.styleFrom(
      backgroundColor: Color(0xFFFF7F50),
      foregroundColor: Colors.white,
    ),
  ),
  floatingActionButtonTheme: FloatingActionButtonThemeData(
    backgroundColor: Color(0xFFFF7F50),
  ),
);
```

Листинг 4.3 – Код светлой темы оформления

Нижняя панель навигации, реализованная через виджет `BottomNavigationBar`, представленный в листинге 4.4. Первый элемент навигации отвечает за переход на главный экран приложения. Он использует иконку `Icons.home` в заполненном виде.

Второй и третий элементы панели предназначены для функций поиска и доступа к видеоконтенту. Четвёртый элемент представляет профиль пользователя и выделяется использованием виджета `CircleAvatar`. Этот элемент отображает аватар пользователя через `NetworkImage`, если в данных `widget.userProfile` доступен URL-адрес аватара. Если же URL отсутствует или `userProfile` равно `null`, вместо изображения показывается иконка `Icons.person` размером 16 пикселей. Такой подход позволяет гибко обрабатывать случаи, когда данные профиля недоступны, сохраняя функциональность панели.

Дизайн панели навигации тесно связан с темами оформления приложения, определёнными в классе `AppThemes`.

```
items: [
  BottomNavigationBarItem(
    icon: Icon(
      _selectedIndex == 0 ? Icons.home : Icons.home_outlined,
    ),
    label: '',
  ),
  BottomNavigationBarItem(
    icon: Icon(
      _selectedIndex == 1 ? Icons.search : Icons.search_outlined,
    ),
    label: '',
  ),
  BottomNavigationBarItem(
    icon: Icon(
      _selectedIndex == 2
        ? Icons.video_library
        : Icons.video_library_outlined,
    ),
    label: '',
  ),
  BottomNavigationBarItem(
    icon: CircleAvatar(
      radius: 14,
      backgroundImage: widget.userProfile?['avatar_url'] != null
        ? NetworkImage(widget.userProfile!['avatar_url'])
        : null,
      child: widget.userProfile?['avatar_url'] == null
        ? const Icon(Icons.person, size: 16)
        : null,
    ),
    label: '',
  ),
],
```

Листинг 4.4 – Пример виджета `BottomNavigationBar`

Создание удобных и функциональных интерфейсов для административных задач является важной частью обеспечения эффективного управления контентом и пользователями. Представленный в листинге 4.5 код реализует администраторскую

панель с использованием виджета `DefaultTabController` и `Scaffold`, предоставляя администратору доступ к трём основным разделам: управление пользователями, жалобы на фотографии и жалобы на видеоконтент.

```
DefaultTabController(
  length: 3,
  child: Scaffold(
    appBar: AppBar(
      title: const Text('Администратор'),
      bottom: TabBar(
        tabs: [
          const Tab(text: 'Пользователи'),
          const Tab(text: 'Жалобы на фото'),
          const Tab(text: 'Жалобы на видео'),
        ],
      ),
    actions: [
      IconButton(
        icon: const Icon(Icons.logout),
        onPressed: () => _confirmLogout(context),
        tooltip: 'Выйти из аккаунта',
      ),
      IconButton(
        icon: const Icon(Icons.refresh),
        onPressed: () => Navigator.pushReplacement(
          context,
          MaterialPageRoute(
            builder: (context) => AdminScreen(themeNotifier:
themeNotifier),
          ),
        ),
        tooltip: 'Обновить',
      ),
    ],
  ),
  body: TabBarView(
    children: [
      _buildUsersTab(context),
      _buildPhotoReportsTab(context),
      _buildVideoReportsTab(context),
    ],
  ),
),
);
```

Листинг 4.5 – Пример виджетов `DefaultTabController` и `Scaffold`

Представленный в листинге 4.6 код функции `deletePhoto` демонстрирует, как можно безопасно удалить фотографию из базы данных Supabase, обновить интерфейс приложения и уведомить пользователя о результате операции. Эта функция, написанная с учётом асинхронного программирования, подчёркивает важность обработки ошибок и интеграции с пользовательским интерфейсом.

При наличии аутентифицированного пользователя функция отправляет запрос на удаление записи из таблицы `user_photos` в Supabase. Запрос использует два условия: совпадение `id` с переданным `photoId` и `user_id` с идентификатором текущего пользователя (`user.id`). Это предотвращает удаление чужих фотографий, добавляя дополнительный уровень защиты. Логирование через `debugPrint` помогает отслеживать процесс, записывая идентификатор фотографии и пользователя в консоль для отладки.

После успешного удаления функция проверяет, смонтирован ли виджет (`mounted`), чтобы избежать ошибок при обновлении состояния неактивного виджета. Затем она вызывает `setState`, удаляя фотографию из локального списка `uploadedPhotos` и пересчитывая общее количество постов (`totalPosts`) как сумму фотографий и видео.

Функция `deletePhoto` принимает параметр `photoId` — идентификатор фотографии, которую нужно удалить.

```
Future<void> deletePhoto(String photoId) async {
  final user = Supabase.instance.client.auth.currentUser;
  if (user == null) {
    debugPrint('Delete photo failed: No authenticated user');
    return;
  }

  debugPrint('Deleting photo: $photoId for user: ${user.id}');
  try {
    await Supabase.instance.client
      .from('user_photos')
      .delete()
      .eq('id', photoId)
      .eq('user_id', user.id);

    if (mounted) {
      setState(() {
        uploadedPhotos.removeWhere((photo) => photo['id'] == photoId);
        totalPosts = uploadedPhotos.length + uploadedVideos.length;
      });
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Фото успешно удалено')),
      );
    }
  } catch (error) {
    debugPrint('Photo deletion error: $error');
    if (mounted) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Ошибка при удалении фото: $error')),
      );
    }
  }
}
```

Листинг 4.6 – Функция `deletePhoto`

Правильная организация файлов помогает легче ориентироваться в проекте и облегчает управление кодом.

4.4 Выводы по разделу

В данной главе был проведен всесторонний анализ процесса разработки всех компонентов приложения, охватывающий ключевые этапы создания программного продукта. На начальном этапе было выполнено тщательное исследование и отбор необходимых технологий для реализации курсового проекта. Этот процесс включал сравнительный анализ различных платформ, фреймворков и инструментов, чтобы выбрать оптимальный стек технологий, обеспечивающий высокую производительность, масштабируемость и удобство разработки. Выбор технологий был обоснован с учетом требований проекта, таких как кроссплатформенность, поддержка современных стандартов и возможность интеграции с внешними сервисами.

Была представлена графическая структура проекта и основные файлы. Обсуждение включало описание архитектуры веб-приложения, взаимодействие с базой данных, а также реализацию пользовательского интерфейса.

Таким образом, данная глава предоставила детальное описание всех этапов разработки приложения — от выбора технологий и проектирования базы данных до реализации серверной логики, клиентского интерфейса и их интеграции в единое программное обеспечение. Особое внимание уделялось соблюдению принципов модульности, что упростило тестирование и дальнейшее масштабирование системы. Описанный процесс разработки позволил не только создать функциональный продукт, но и получить глубокое понимание всех аспектов создания комплексного программного обеспечения. Итогом стало формирование прочной основы для успешной реализации проекта, которая может быть использована для дальнейшего совершенствования приложения или создания аналогичных решений в будущем. Эта работа также подчеркнула важность системного подхода к разработке, который обеспечивает высокое качество и надежность конечного продукта, а также заложила фундамент для потенциального внедрения новых функций и улучшений в будущем.

5 Анализ информационной безопасности приложения

Важным аспектом при разработке мобильного приложения является обеспечение информационной безопасности. Приложение должно предоставлять защиту данных пользователя от несанкционированного доступа и взлома. В ходе анализа информационной безопасности были выявлены следующие аспекты, которым необходимо было уделить особо пристальное внимание защите данных в приложении, то есть должны быть реализованы меры по защите данных пользователей, таких как пароль для доступа к приложению, ограничение прав доступа пользователей, то есть пользователи должны иметь доступ только к той информации, к которой имеют необходимость, администраторы должны иметь доступ к более широкому набору функций. Необходимо четко определить и ограничить возможности каждой роли в приложении.

5.1 Защита пользовательских данных

Supabase обеспечивает безопасную аутентификацию и хранение данных через проверенные механизмы: хеширование паролей с помощью алгоритма PBKDF2-SHA256 (260,000 итераций) для защиты от перебора, поддержку OAuth 2.0 для входа через Google, GitHub и другие провайдеры, а также JWT-токены с цифровой подписью для управления сессиями. Все данные хранятся в PostgreSQL с обязательным шифрованием (SSL/TLS), а встроенная система Row-Level Security (RLS) позволяет гибко настраивать права доступа на уровне отдельных записей в базе. Благодаря открытому исходному коду и возможности самохостинга, Supabase обеспечивает прозрачность и полный контроль над безопасностью данных, включая аудит операций и интеграцию с дополнительными инструментами шифрования (например, pgcrypto).

5.2 Выводы по разделу

В данном разделе курсовой работы был проведён анализ системы информационной безопасности разрабатываемого мобильного приложения. Основное внимание уделено защите пользовательских данных с использованием Supabase Authentication.

Для обеспечения безопасности данных пользователей применяется Supabase Authentication, который предоставляет следующие механизмы защиты: хеширование паролей Supabase использует алгоритм Scrypt, что исключает возможность восстановления пароля из хеша и защищает от атак таблицам, JWT-токены для аутентификации, то есть после успешного входа генерируется защищённый токен (JWT), который содержит информацию о пользователе и имеет ограниченный срок действия.

6 Тестирование приложения

6.1 Тестирование клиентской области

Тестирование мобильных приложений важно и необходимо для обеспечения их качества и производительности. Оно помогает уменьшить риск сбоев приложения, гарантирует отсутствие ошибок и улучшает восприятие пользователей, что в свою очередь способствует увеличению количества загрузок и успешному запуску приложения на рынке.

Для начала протестируем страницу авторизации. Если пользователь не зарегистрирован, ввел неверные данные или не ввел какое-либо поле, то он получит предупреждение, показанное на рисунке 6.1.

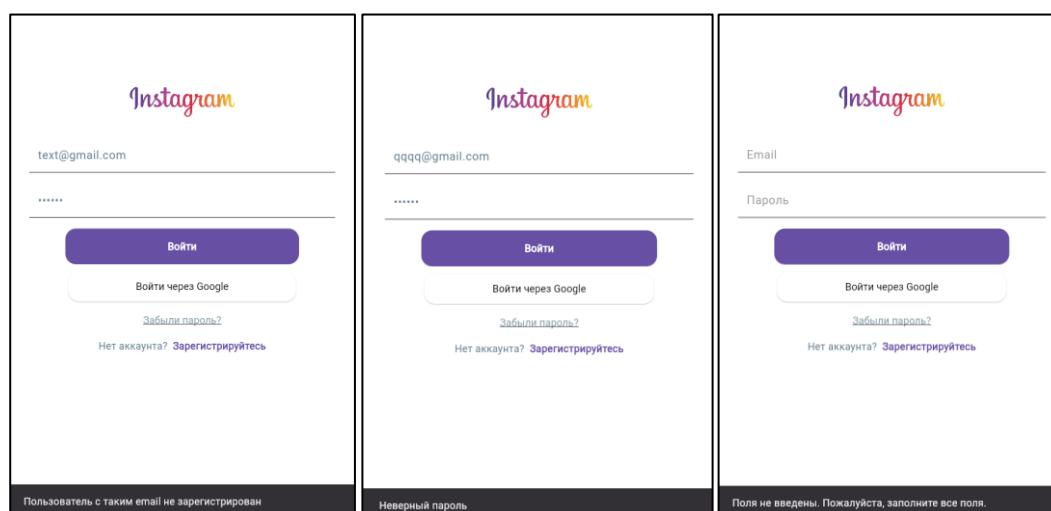


Рисунок 6.1 – Тестирование авторизации

Далее протестируем страницу регистрации. Пользователь может зарегистрироваться с уникальным именем, также ему необходимо повторить введенный пароль. Роль пользователя автоматически считается User. Кроме того, есть проверка на корректный email и пароль. Тестирование продемонстрировано на рисунке 6.2.

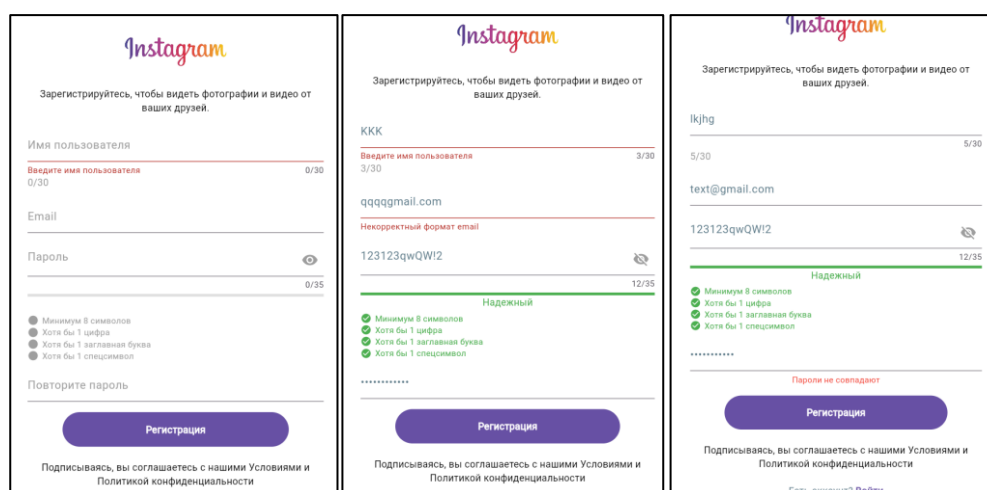


Рисунок 6.2 – Тестирование регистрации

После авторизации или регистрации пользователь попадает на главную страницу с фотографиями и заметками других пользователей.

Также необходимо протестировать выбор тем приложения. На выбор пользователю было предоставлено три темы: светлая, розовая и темная. На рисунке 6.3 представлены темы.

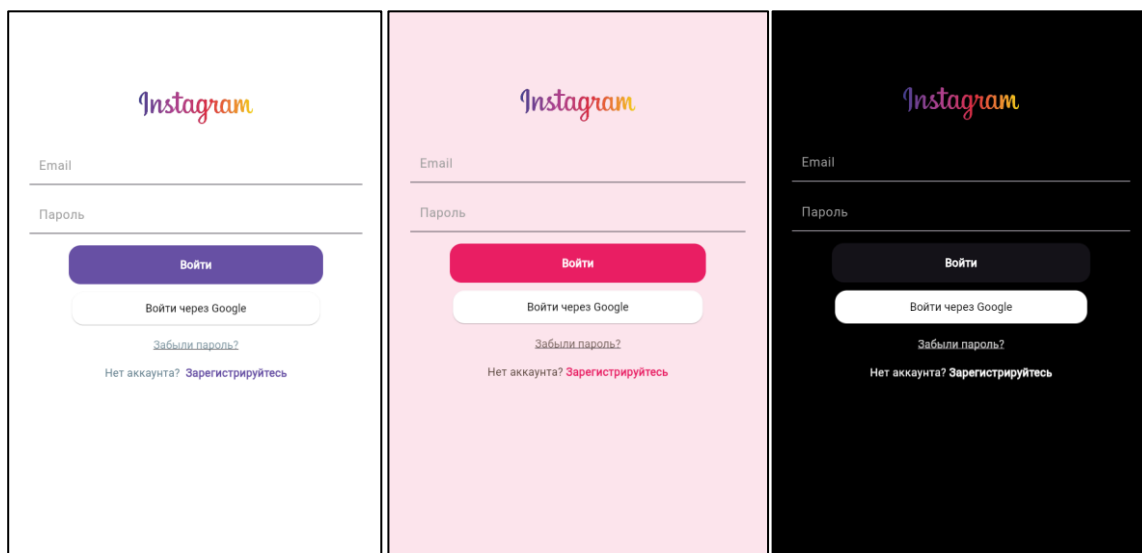


Рисунок 6.3 – Тестирование регистрации

6.2 Выводы по разделу

В данном разделе была описана валидация данных приложения для просмотра фотографий и видео, а также были рассмотрены сценарии регистрации и авторизации, при которых пользователь предоставляет недостаточно данных или эти данные указаны в неверном формате.

Тестирование приложения является неотъемлемой частью процесса разработки, позволяющей обеспечить высокое качество продукта и удовлетворенность пользователей. В рамках данного проекта, благодаря тщательной валидации и обработке ошибок, было создано надежное и удобное в использовании мобильное приложение, которое успешно выполняет свои функции и отвечает потребностям целевой аудитории.

7 Руководство пользователя

В данном разделе описывается руководство для зарегистрированных пользователей, а именно для пользователя и администратора. У каждой роли есть свой функционал, что позволяет эффективно управлять системой и обеспечивает удобство в использовании.

7.1 Пользователь

Для того, чтобы пользователь мог получить доступ к основному функционалу приложения, а именно просмотр видео и фотографий, создание своего собственного контента, взаимодействие с другими пользователями посредством лайков и комментариев ему нужно авторизоваться либо же зарегистрироваться, доступна авторизация через Google. Страница с авторизацией представлена на рисунке 7.1.

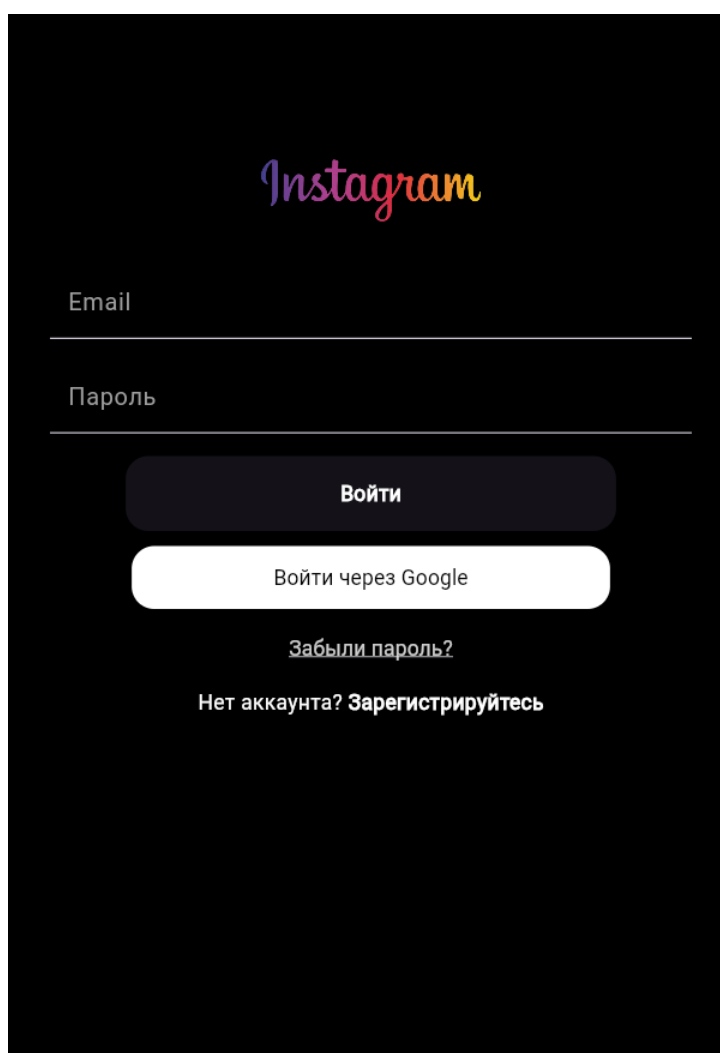


Рисунок 7.1 – Страница авторизации

В случае, если у пользователя еще нет аккаунта, он может нажать на кнопку «Зарегистрируйтесь» и быть перенаправленным на страницу регистрации, которая продемонстрирована на рисунке 7.2. Процесс регистрации состоит из трех этапов, на каждом из которых пользователю необходимо ввести определенные данные.

На первом этапе пользователю нужно ввести свое имя, что позволит создать персонализированный профиль. Далее, на втором этапе, требуется указать электронную почту, которая будет использоваться для входа в систему. На третьем этапе пользователю необходимо придумать надежный пароль. Рекомендуется использовать комбинацию букв, цифр и специальных символов для повышения безопасности. После успешного завершения всех этапов регистрации пользователь получит подтверждение о создании аккаунта и сможет войти в систему, используя указанные логин и пароль.

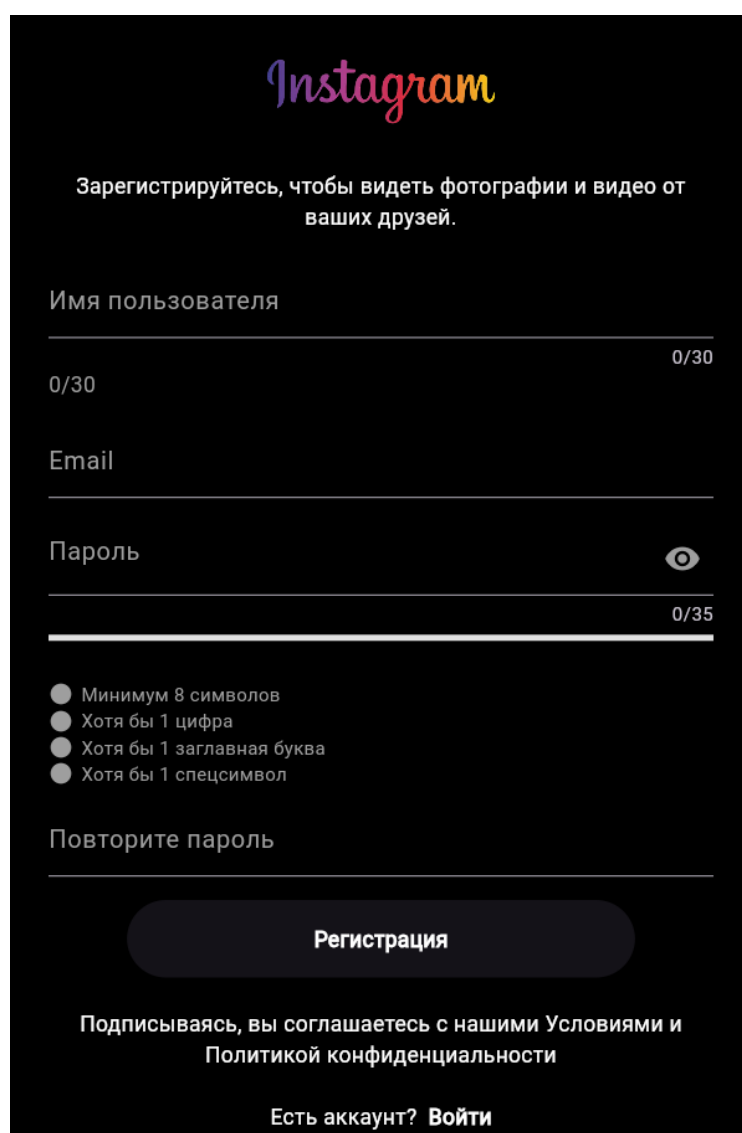
The image shows the Instagram registration interface on a dark background. At the top is the Instagram logo. Below it is a text prompt: "Зарегистрируйтесь, чтобы видеть фотографии и видео от ваших друзей." (Sign up to see photos and videos from your friends). The form consists of four input fields: "Имя пользователя" (Username) with a character count of 0/30, "Email", "Пароль" (Password) with a character count of 0/35 and an eye icon for toggling visibility, and "Повторите пароль" (Repeat password). Below the password field are four requirements listed with circular icons: "Минимум 8 символов" (Minimum 8 characters), "Хотя бы 1 цифра" (At least 1 number), "Хотя бы 1 заглавная буква" (At least 1 uppercase letter), and "Хотя бы 1 спецсимвол" (At least 1 special character). A large "Регистрация" (Sign up) button is centered below the fields. At the bottom, there is a line of text: "Подписываясь, вы соглашаетесь с нашими Условиями и Политикой конфиденциальности" (By signing up, you agree to our Terms and Privacy Policy), followed by the text "Есть аккаунт? Войти" (Have an account? Log in).

Рисунок 7.2 – Страница регистрации

После того, как пользователь зарегистрировался и авторизовался, он получает доступ к функционалу приложения.

При входе в аккаунт пользователь по умолчанию перенаправляется на страницу профиля текущего пользователя, где есть фотографии и видео, число постов, подписчиков и подписок, заметка пользователя, кнопки смены темы и выхода из учетной записи. Данная страница продемонстрирована на рисунке 7.3.

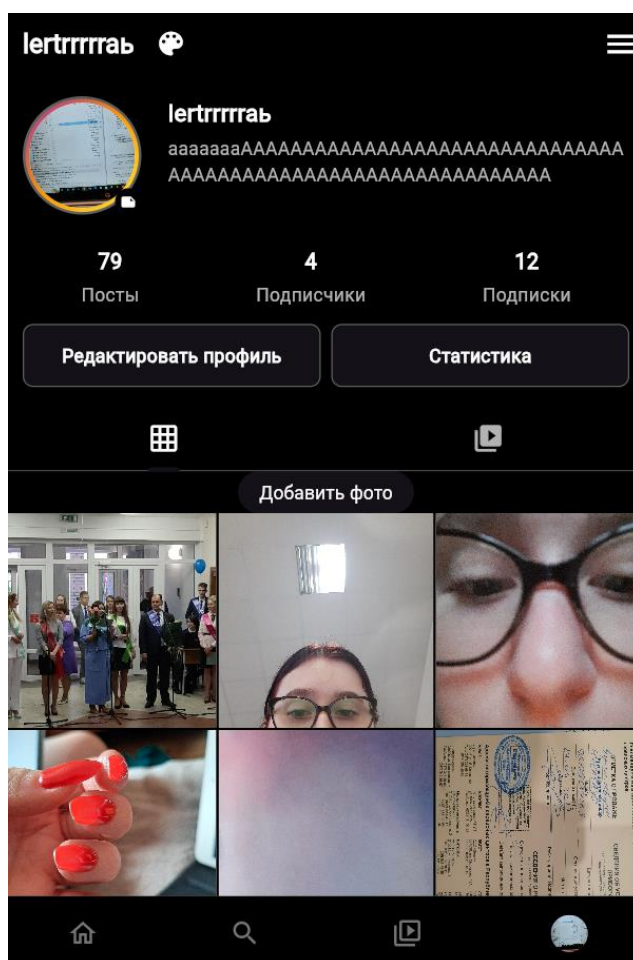


Рисунок 7.3 – Страница профиля текущего пользователя

В профиле пользователь может загрузить фото или видео, написать заметку и редактировать профиль: поменять имя пользователя, описание. Фото представлены на рисунке 7.4.

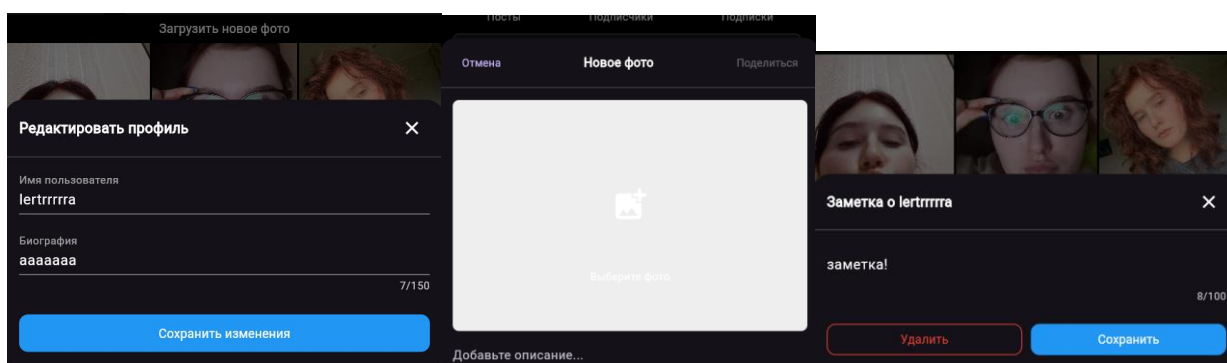


Рисунок 7.4 – Модальные окна изменения профиля, заметки, загрузки фотографии

Пользователь может просмотреть список подписок и подписчиков, при нажатии на пользователя открывается его профиль. Список подписчиков представлен на рисунке 7.5.

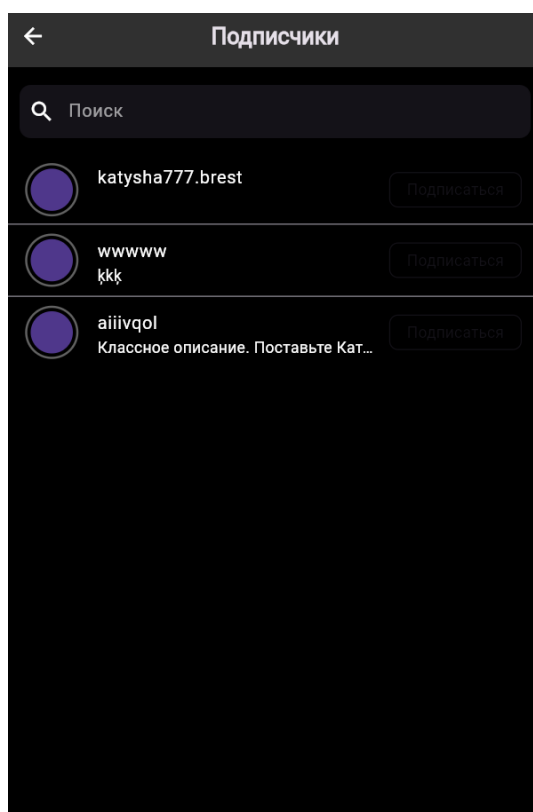


Рисунок 7.5 – Страница просмотра подписчиков

Кроме редактирования профиля и просмотра подписок и подписчиков пользователь может просмотреть статистику своего профиля, увидеть число лайков и комментариев которые он получил, число комментариев, которые он написал и число лайков, которое поставил. Страница со статистикой представлена на рисунке 7.6.

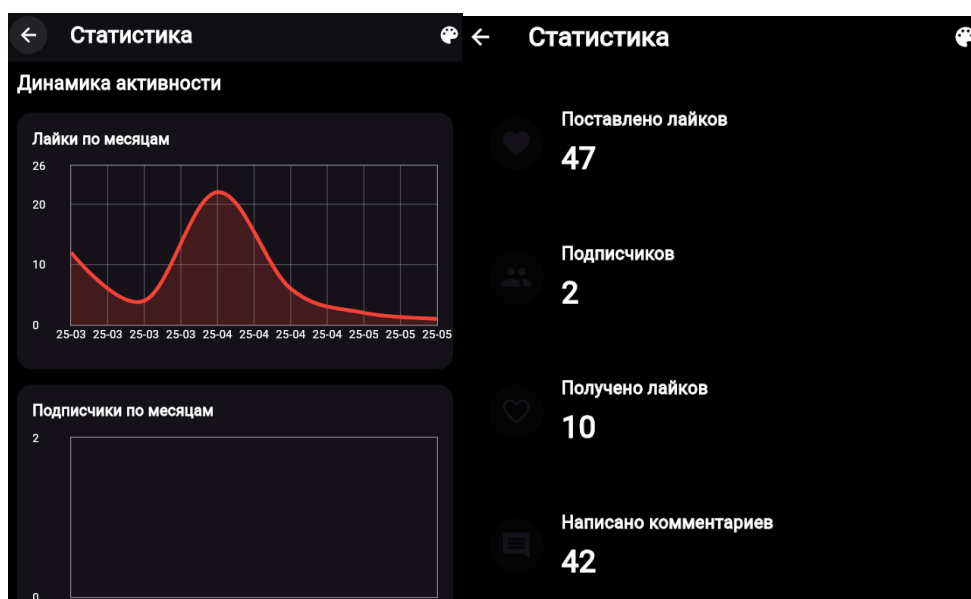


Рисунок 7.6 – Страница просмотра статистики профиля

Далее, нажав на иконку в меню, переходим на страницу с фотографиями и заметками тех пользователей, на которых подписан текущий пользователь.

Фотографии на странице можно лайкать и комментировать. Вид страницы представлен на рисунке 7.7.

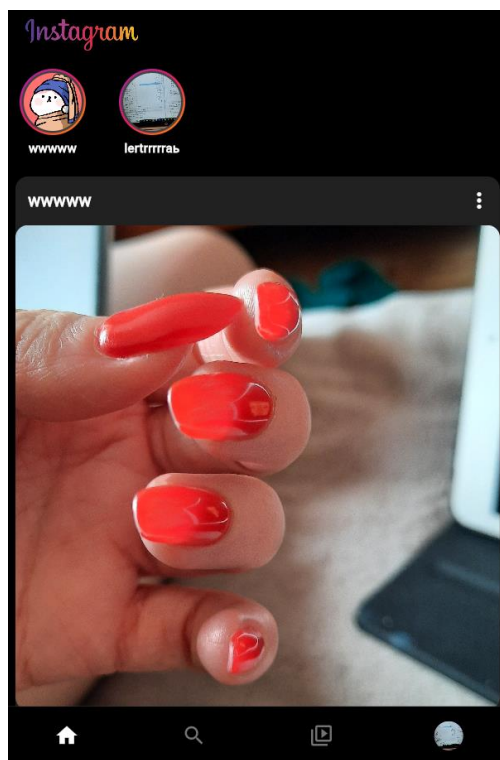


Рисунок 7.7 – Страница с фотографиями

Нажав на заметку открывается заметка и имя профиля. Демонстрация заметки на рисунке 7.8.

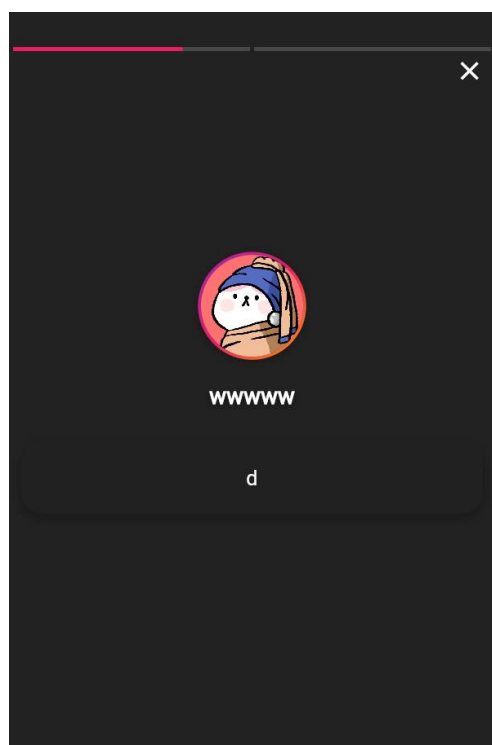


Рисунок 7.8 – Страница с фотографиями

Следующая страница, на которую может зайти пользователь это страница поиска. На данной странице отображаются все фотографии всех пользователей, в строке поиска можно ввести имя пользователя и, нажав на имя пользователя, перейти на его аккаунт. Страница поиска представлена на рисунке 7.9.

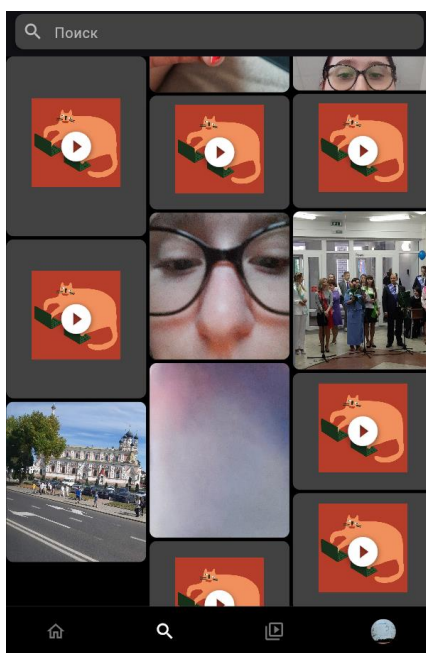


Рисунок 7.9 – Страница поиска

Далее страница с видео пользователей, на которых подписан текущий пользователь, видео можно лайкать и комментировать. Страница представлена на рисунке 7.10.

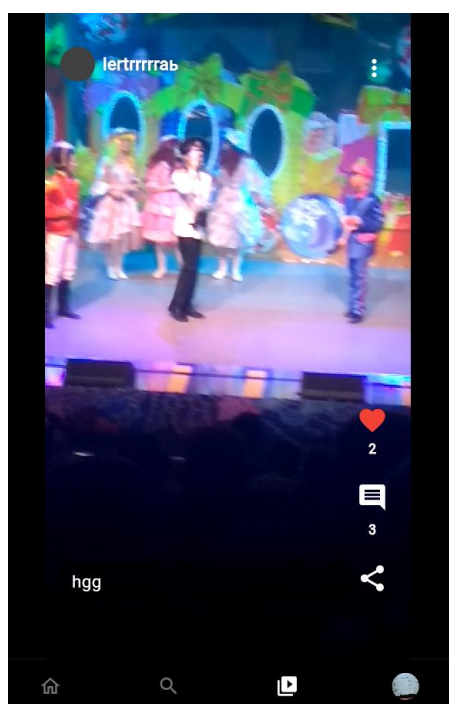


Рисунок 7.10 – Страница просмотра видео

На странице комментариев пользователь может написать комментарий к определенному фото/видео. Страница комментариев представлена на рисунке 7.11.

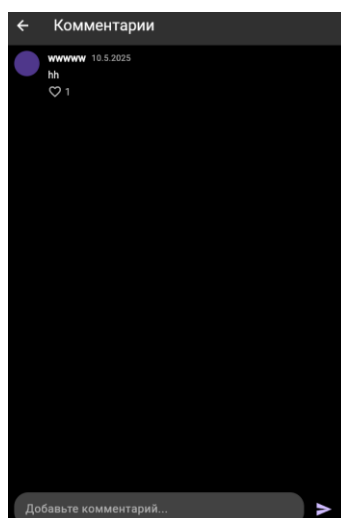


Рисунок 7.11 – Страница комментариев

Согласно данному руководству пользования зарегистрированный пользователь может легко ориентироваться в приложении..

7.2 Администратор

Курсовой проект разрабатывался с использованием фреймворка Flutter, что позволяет приложению быть кроссплатформенным. Для администратора была разработана веб-часть. Чтобы продолжить работу в приложении как администратор, пользователь должен иметь специальную роль и войти в приложение под соответствующей учетной записью. Такой пользователь может удалять профиль, фото, видео. Страница для администратора представлена на рисунке 7.12.

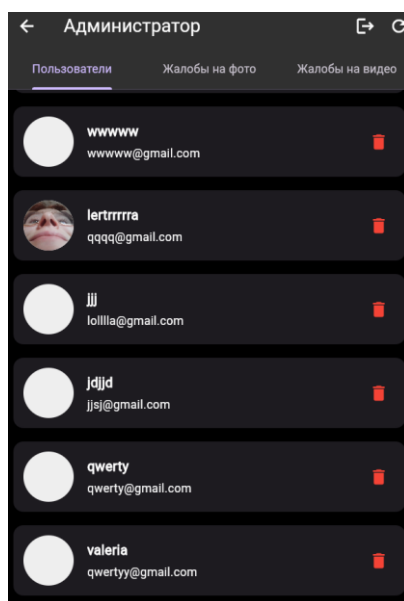


Рисунок 7.12 – Страница администратора

Администратор может переходить на профиль пользователя и удалять фотографии и видео. Профиль пользователя для администратора представлен на рисунке 7.13.



Рисунок 7.13 – Страница профиля для администратора

Администратор, получая жалобы от пользователей на фотографию или видео, может удалить или отклонить жалобу. Страница жалоб представлена на рисунке 7.14.

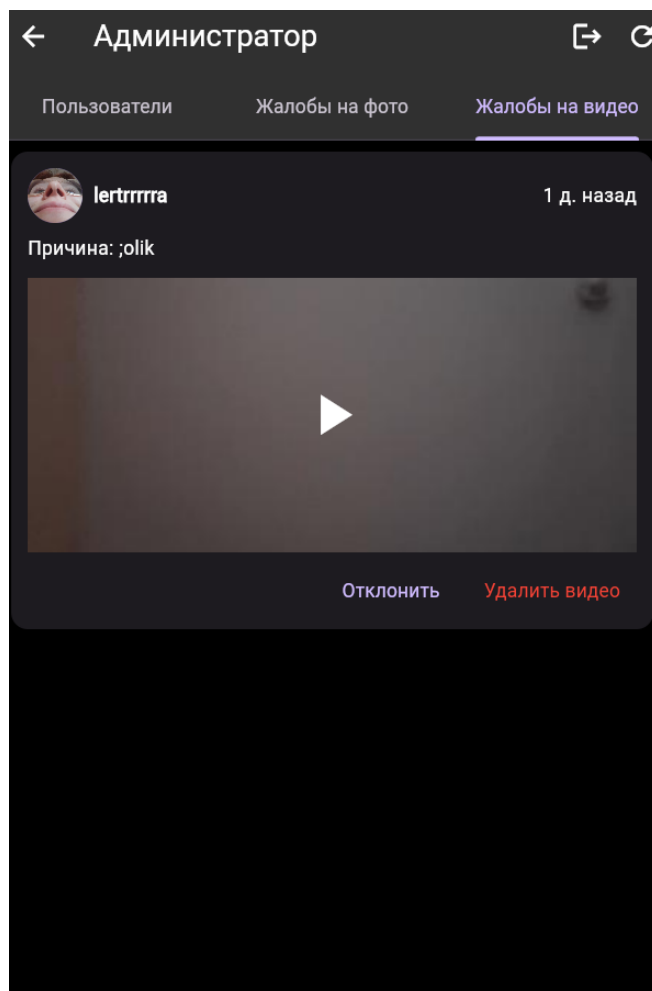


Рисунок 7.14 – Страница жалоб

Для выхода из учетной записи администратору нужно нажать на иконку выхода на главной странице.

7.3 Выводы по разделу

Цель данного раздела заключалась в создании руководства пользователя, в соответствии с разработанным программным продуктом. В разделе представлены основные моменты работы с приложением для пользователя и администратора. В нем содержится подробное описание того, что каждая роль может делать в пределах данного мобильного приложения. Также описан порядок действий для достижения поставленных целей при выполнении работы.

Заключение

В ходе курсового проекта было разработано мобильное приложение для просмотра фотографий и видео.

Перед началом разработки были выявлены функциональные задачи мобильного приложения, рассмотрена актуальность данной темы, а также был произведен анализ аналогов приложений подобной тематики.

Функционально выполнены следующие задачи:

- добавление изображения в профиль;
- публикация комментариев;
- создание профиля пользователя;
- подписка на пользователя;
- добавление видео;
- создание и просмотр заметки;
- просмотр подписок и подписчиков;
- возможность ставить лайки на понравившиеся фотографии и видео;
- возможность комментировать фотографии и видео;
- поиск пользователей.

На следующем этапе разработки было выполнено проектирование, а именно реализована диаграмма вариантов использования (use-case diagram), которая описывает, какой функционал разрабатываемой программной системы доступен каждой группе пользователей. Также, с использованием платформы Supabase, была разработана база данных, состоящая из 6 таблиц. Для данной базы данных также была создана логическая схема, демонстрирующая связи между таблицами.

В качестве технических средств для разработки мобильного приложения использовались бэкенд-решение Supabase и Android Studio как среда разработки. Авторизация и регистрация пользователей осуществляется с помощью встроенных функций, которые предоставляет Supabase.

В ходе разработки мобильного приложения был проведен анализ информационной безопасности разрабатываемого мобильного приложения. Были выявлены ключевые аспекты, которые необходимо учесть при разработке приложения: защита данных, проверка на уязвимости, регулярное обновление мобильного приложения и ограничение прав доступа пользователей.

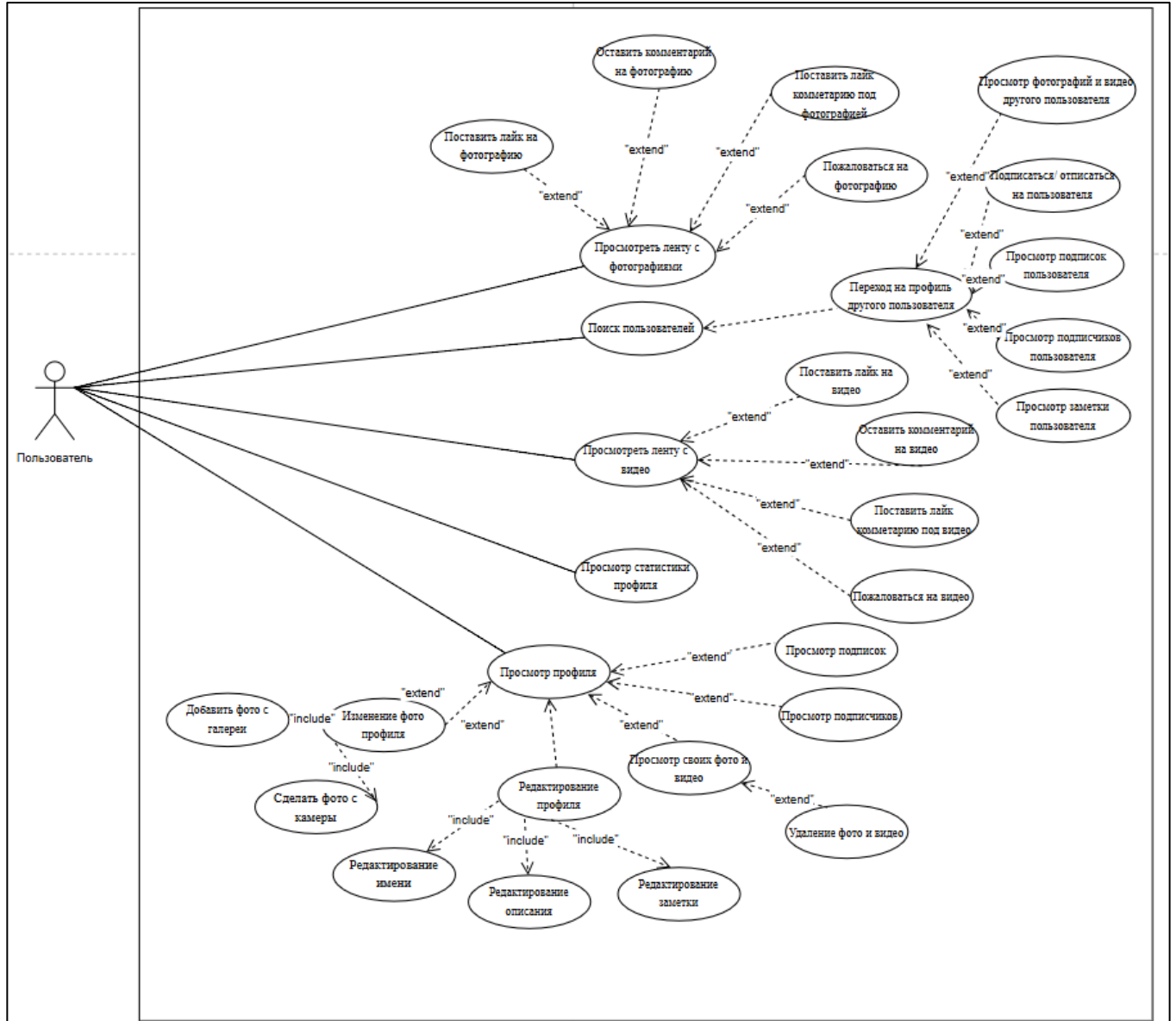
Также было рассмотрено руководство для пользователя и администратора

В ходе курсового проектирования были закреплены, расширены и углублены полученные теоретические знания в области мобильной разработки с использованием фреймворка Flutter и платформы Supabase, приобретены практические навыки самостоятельной работы, выработаны умения применять их при решении конкретных вопросов и задач.

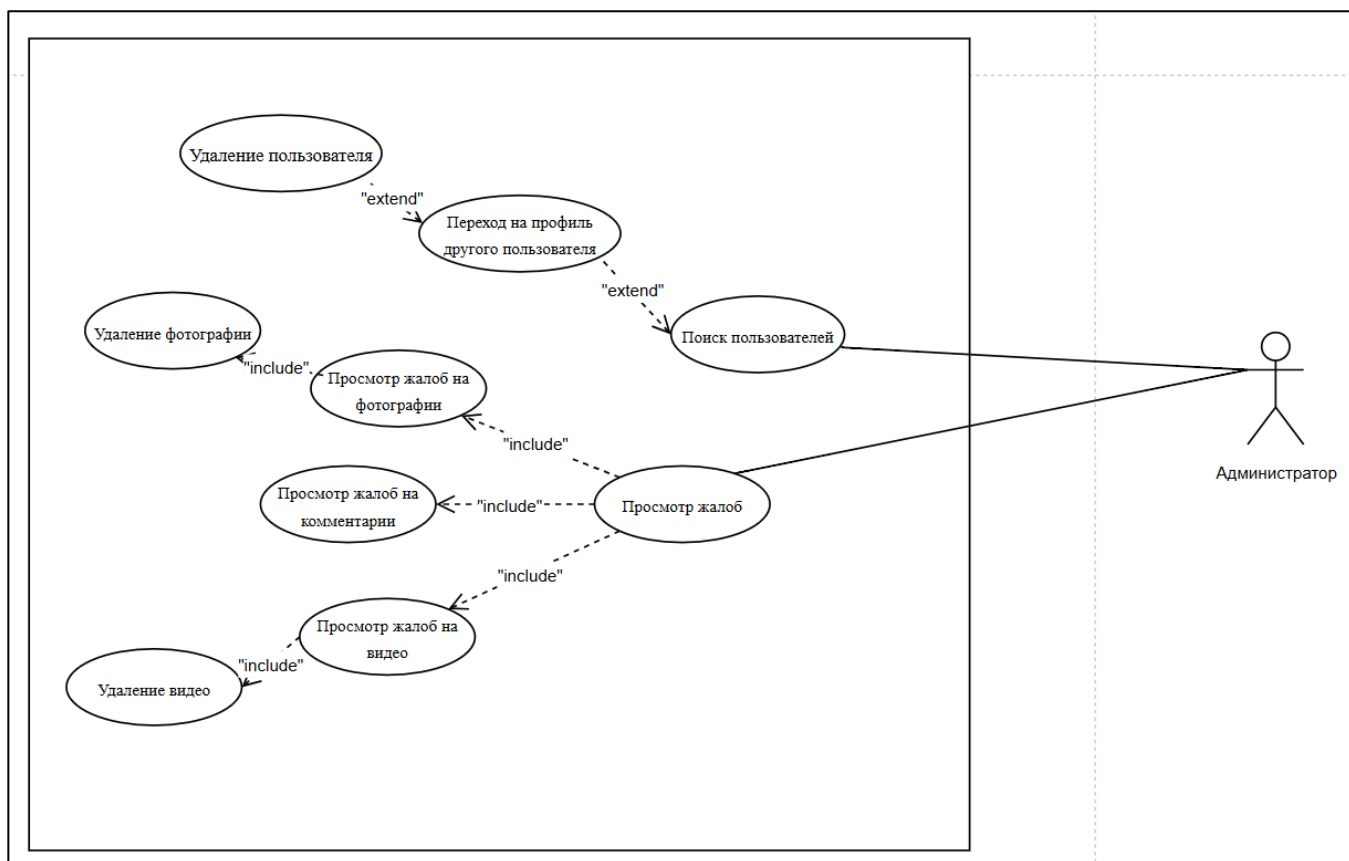
Список литературы

1. Приложение Instagram [Электронный ресурс]. – Режим доступа <https://www.instagram.com/>– Дата доступа: 10.03.2025.
2. Приложение Pinterest [Электронный ресурс]. – Режим доступа <https://www.pinterest.com/>Дата доступа: 10.03.2025.
3. Приложение Snapchat [Электронный ресурс]. – Режим доступа <https://www.snapchat.com/>Дата доступа: 10.03.2025.
4. Сайт о программировании [Электронный ресурс] / Режим доступа: <https://timeweb.com/ru/community/articles/chto-takoe-supabase-i-kak-s-neu-rabotat/>Дата доступа: 22.04.2025.
5. Хабр – Что такое Flutter? [Электронный ресурс] / Режим доступа: <https://habr.com/ru/articles/481326/>Дата доступа:22.04.2025.

Приложение А. Диаграмма вариантов использования пользователя



Приложение Б. Диаграмма вариантов использования администратора



Приложение В

