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 «Мобильное приложение фотохостинга»

Исполнитель
студент 3 курса 8 группы _____ А. С. Новиченко
подпись, дата

Руководитель
ассистент _____ Н. И. Уласевич
должность, учен. степень, ученое звание подпись, дата

Допущен(а) к защите _____ _____
_____ _____
_____ _____
дата, подпись

Курсовой проект защищен с оценкой _____

Руководитель _____ _____ Н. И. Уласевич
подпись дата инициалы и фамилия

Минск 2024

Содержани

Введение

В современном цифровом мире, где визуальное содержание играет ключевую роль в нашей повседневной коммуникации, мобильные приложения фотохостинга становятся важными платформами для обмена изображениями, а также для творчества, вдохновения и социального взаимодействия. Текущая эпоха характеризуется стремительным развитием технологий и ростом спроса на качественные инструменты для создания, хранения и обмена фотографиями. В этом контексте разработка мобильного приложения фотохостинга является актуальной и перспективной областью программного обеспечения.

Целью данного курсового проекта является изучение и практическое применение принципов разработки мобильных приложений фотохостинга с использованием современных технологий. Проект направлен на создание инновационного и удобного в использовании приложения, которое позволит пользователям легко загружать, организовывать и делиться своими фотографиями, а также находить вдохновение в творчестве других участников сообщества. В результате успешной реализации проекта ожидается создание полнофункционального мобильного приложения, способного удовлетворить потребности современного пользователя в области визуального контента и социального взаимодействия.

Разработка такого приложения имеет высокую практическую значимость. Оно позволит пользователям не только хранить и организовывать свои фотографии, но и взаимодействовать с другими, делиться своими работами и впечатлениями. Это способствует укреплению социального взаимодействия и развитию творческих навыков.

1 Анализ требований к программному средству

1. Регистрация и авторизация пользователя:

– пользователь может зарегистрироваться в приложении, создав свой учетный профиль;

– пользователь имеет возможность проходить авторизацию, используя ранее созданный аккаунт.

2. Добавление изображения:

– пользователь может добавлять новые изображения в приложение.

3. Удаление изображения:

– пользователь имеет возможность удалять изображения.

4. Просмотр изображения:

– пользователь может просматривать доступные изображения в приложении.

5. Поиск изображения:

– пользователь имеет возможность выполнять поиск изображений по ключевым словам.

6. Добавление комментария:

– пользователь может добавлять комментарии к изображениям.

7. Удаление комментария:

– пользователь может удалять комментарии.

8. Добавление изображений в избранное

– пользователь может добавлять понравившиеся изображения в список избранных для последующего просмотра.

9. Удаление изображений из избранного

– пользователь может удалять изображения из списка избранных.

1.1 Аналитический обзор аналогов

Перед началом разработки мобильного приложения для фотохостинга проведен анализ существующих аналогичных продуктов на рынке. Данный этап необходим для изучения особенностей конкурентов и выявления их функциональных возможностей. Поиск аналогов осуществлялся с помощью изучения соответствующей литературы и анализа существующих приложений фотохостинга.

1.1.1 Аналог «Pinterest»

Pinterest [1] — социальный интернет-сервис, фотохостинг, позволяющий пользователям добавлять в режиме онлайн изображения, помещать их в тематические коллекции и делиться ими с другими пользователями.

Гости площадки ограничены в возможностях: они не могут просматривать всю ленту и сохранять контент. Чтобы использовать все функции Pinterest, пользователям нужно зарегистрироваться (рис. 1.1).

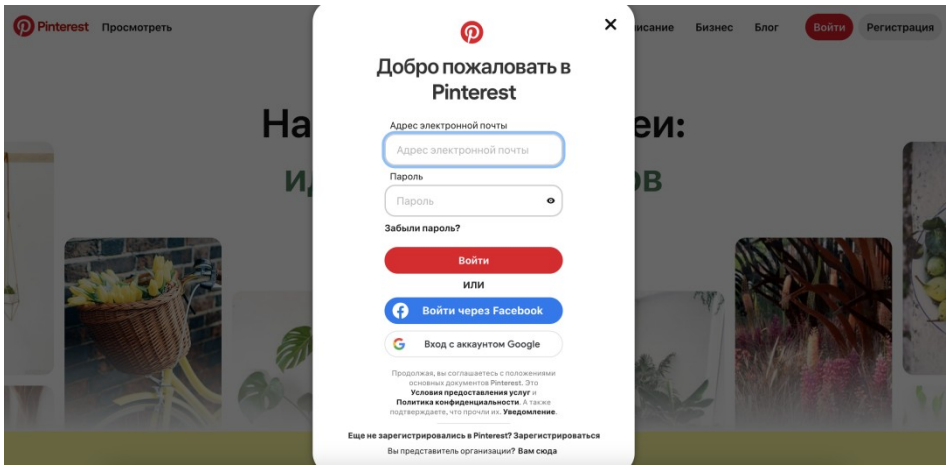


Рисунок 1.1 – Форма регистрации

Следующий шаг после регистрации — создание пина. Для этого нужно нажать на иконку, на которой изображен плюс, и выбрать «Создать пин» (рис. 1.2).

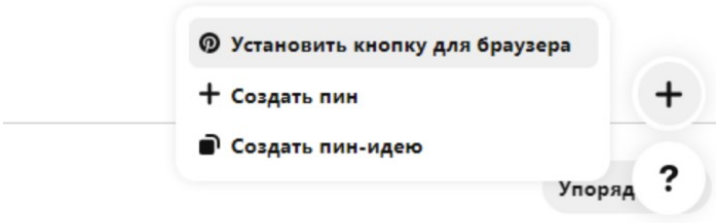


Рисунок 1.2 – Создание пина

Чтобы сделать пин, нужно выбрать публикацию, навести на нее курсор мыши и нажать кнопку «Сохранить» (рис. 1.3). При нажатии открывается окно для добавления изображения, заголовка, подписи, ссылки и доски.

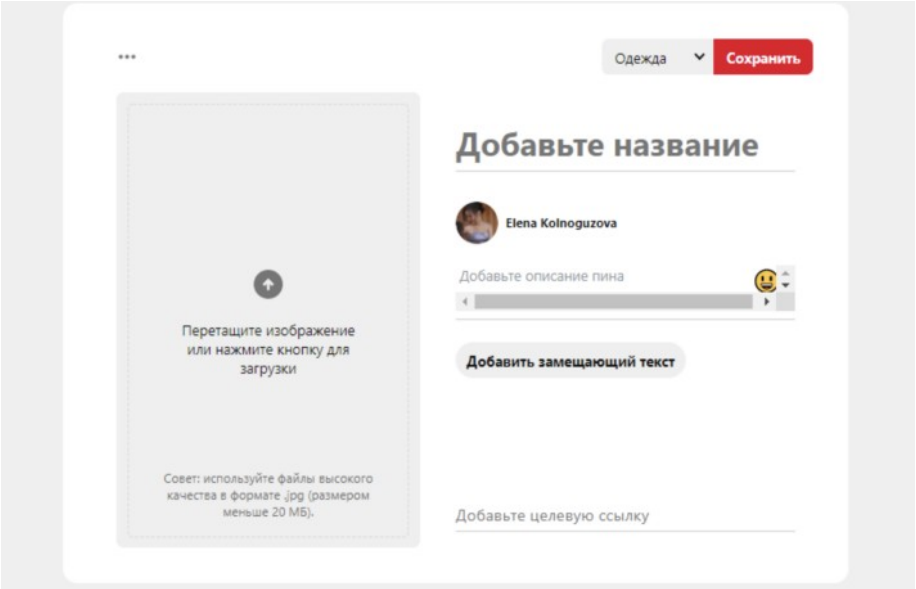


Рисунок 1.3 – Окно изображения

На сайте можно искать изображения, добавлять их к себе, удалять, создавать пины, а так же оставлять комментарии.

1.1.2 Аналог «Designspiration»

Designspiration [2] ориентирован на тех, кто увлекается дизайном. В его новостной ленте можно найти все виды дизайна — как раз для тех, кто занят в творческой индустрии. Здесь можно открывать для себя новых художников, делиться творческими идеями и любимыми работами, найденными на просторах интернета. Одним из безусловных преимуществ сайта является динамический поиск, который обязательно приносит результаты, даже если вы ищете что-то более «нишевое». Здесь тоже можно сохранять понравившиеся файлы и делать из них подборки, во многом похожие на доски в Pinterest.

На главной странице сайта располагаются удобные инструменты, такие как поле поиска для быстрого поиска нужных изображений, иконка аккаунта для доступа к персональным настройкам, возможности создания своих собственных коллекций, а также иконка с многоточием, через которую доступна дополнительная информация, такая как "О нас" и возможность получения Pro-аккаунта. Все это представлено на рисунке 1.4.

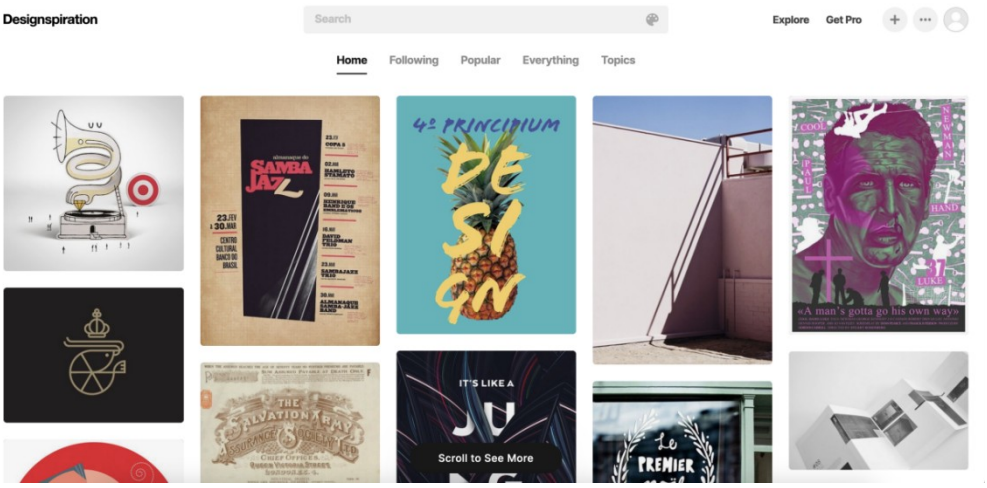


Рисунок 1.4 - Главная страница

При нажатии на любое изображение на сайте Designspiration, открывается окно с увеличенным изображением, где пользователь может взаимодействовать с ним различными способами (рис 1.5). В этом окне доступна кнопка "Сохранить в коллекцию", которая позволяет добавить изображение в уже существующую коллекцию или создать новую. Помимо этого, есть возможность оставить заметки или комментарии к изображению, чтобы поделиться своими мыслями или заметками с другими пользователями или просто сделать пометки для себя. Если у пользователя еще нет коллекции, он может создать ее непосредственно из этого окна, чтобы сохранить изображение в удобной для себя категории.

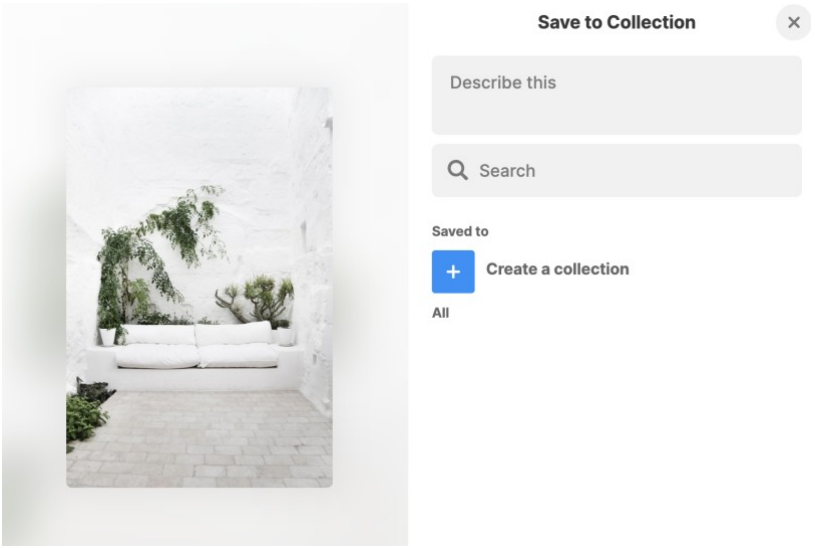


Рисунок 1.5 - Интерактивный просмотр

В профиле доступны сохранённые пины пользователя. При просмотре конкретного сохранённого пина, отображаются кружки с цветовой гаммой, присутствующей на изображении (рис 1.6). Нажатие на любой цвет перенаправляет пользователя к изображениям схожих тональностей.

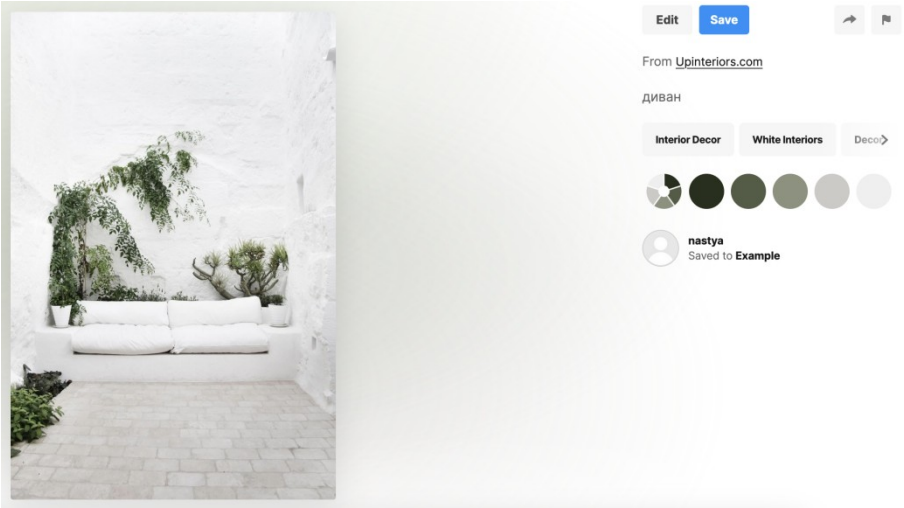


Рисунок 1.6 - Сохраненный пин с функцией цветового анализа

На Designspiration отсутствует возможность обмена пинами между пользователями, в отличие от Pinterest. Пинами можно поделиться только на Twitter или Facebook. Также не предусмотрена функция оставления комментариев под пинами.

1.1.3 Аналог «Dribbble»

На сайте Dribbble [3] можно найти вдохновляющие изображения от талантливых авторов. Здесь представлены разнообразные работы, отражающие различные стили и идеи. Можно легко найти интересующие проекты благодаря сортировке по популярности и использованию различных фильтров. В шапке сайта присутствует удобный поиск для быстрого доступа к нужным работам, а также иконка аккаунта для управления личным профилем (рис 1.7).

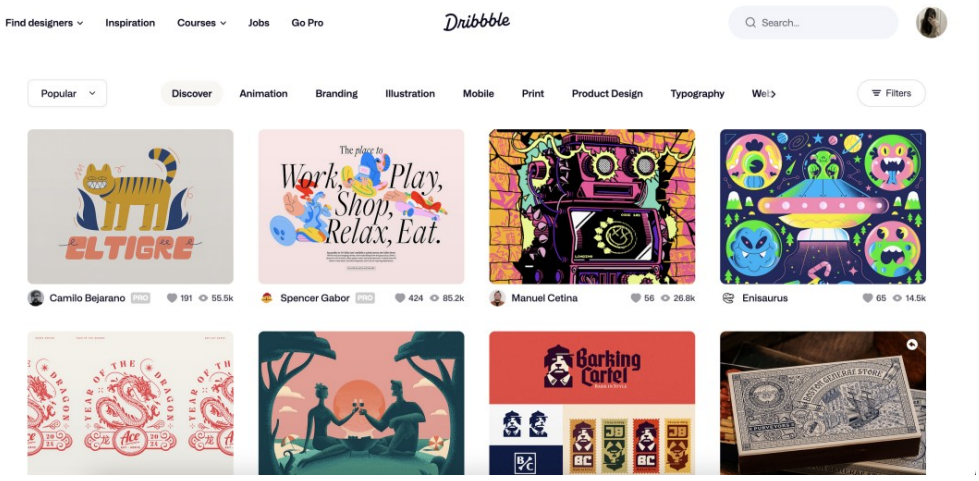


Рисунок 1.7 - Главная страница сайта Dribbble

При переходе на любое изображение на платформе Dribbble открывается модальное окно с увеличенным изображением, предоставляя более его детальный обзор (рис 1.8). Пользователь может добавить изображение в свою коллекцию

"Понравившиеся" или "Избранное", чтобы легко вернуться к нему в будущем. Кроме того, доступна функция обратной связи с автором изображения.

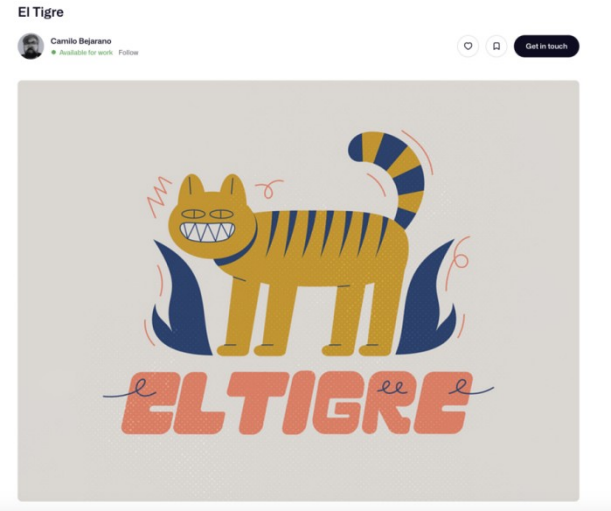


Рисунок 1.8 - Модальное окно изображения

При просмотре изображения на Dribbble, можно добавить его в коллекцию, но для этого сначала необходимо создать ее, если еще не была создана (рис. 1.9). Кроме того, есть возможность поделиться мыслями об изображении.

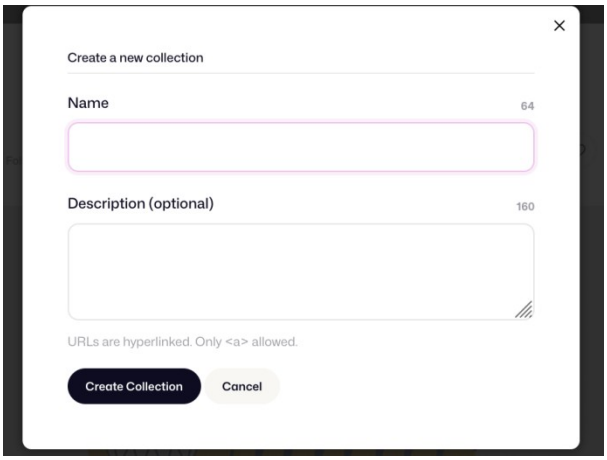


Рисунок 1.9 - Создание новой коллекции

На Dribbble отсутствует возможность обмена пинами между пользователями. Также не предусмотрена функция оставления комментариев под пинами авторов.

1.2 Вывод

В данной главе был проведены обзоры веб-сайтов, предназначенных для поиска изображения, добавления их и т.д.. Исследование возможностей данного приложения послужило основой для определения требований к разрабатываемому приложению и постановки основной задачи проекта.

2 Проектирование программного средства

2.1 Проектирование базы данных

База данных состоит из 4 таблиц: «Users», «Images», «Comments», «Favorites».

Схема базы данных (далее БД) представлена на рисунке 2.1.

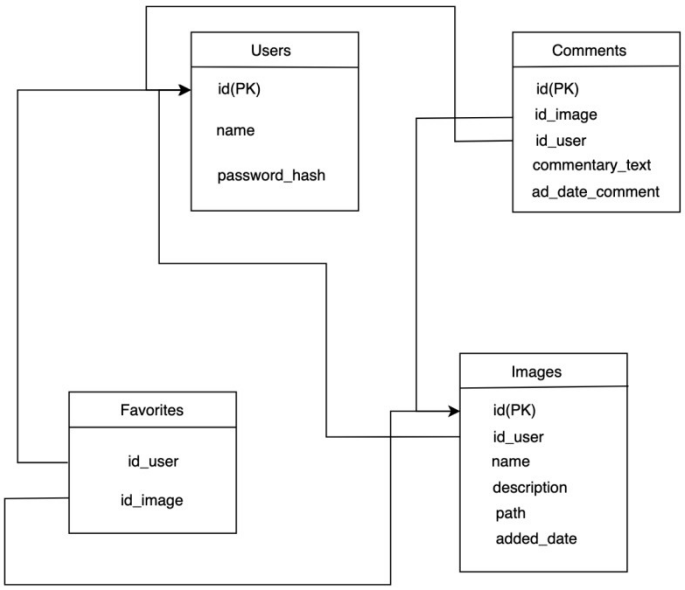


Рисунок 2.1 – Схема базы данных

Таблица «Пользователи»:

– ID (идентификатор пользователя, первичный ключ);

– имя;

– хэш пароля.

Эта таблица содержит информацию о зарегистрированных пользователях мобильного приложения фотохостинга. Каждый пользователь имеет уникальный идентификатор (ID), который является первичным ключом таблицы. Для идентификации пользователей также используется их имя, электронная почта и хэш пароля для безопасного хранения и проверки их учетных данных. Дата регистрации фиксирует момент регистрации каждого пользователя в системе.

Таблица «Изображения»:

– ID (идентификатор изображения, первичный ключ);

– ID пользователя (внешний ключ, связывающий с таблицей «Пользователи»);

– название;

– описание;

– путь к изображению;

– дата добавления.

Эта таблица содержит информацию об изображениях, загруженных пользователями. Каждое изображение имеет уникальный идентификатор (ID), который служит первичным ключом таблицы. ID пользователя связывает каждое изображение с конкретным пользователем, который загрузил его. Название и описание позволяют пользователям описывать и идентифицировать изображения, а путь к изображению указывает на его расположение. Дата добавления фиксирует момент, когда изображение было загружено в приложение.

Связь между таблицами «Пользователи» и «Изображения» позволяет каждому пользователю управлять своими загруженными изображениями, а также обеспечивает безопасность и целостность данных.

Таблица «Комментарии»:

– ID (идентификатор комментария, первичный ключ);

– ID изображения (внешний ключ, связывающий с таблицей «Изображения»);

– ID пользователя (внешний ключ, связывающий с таблицей «Пользователи»);

– текст комментария;

– дата добавления комментария.

Эта таблица содержит комментарии, оставленные пользователями к изображениям. Каждый комментарий имеет уникальный идентификатор (ID), который является первичным ключом таблицы. ID изображения и ID пользователя это внешние ключи, связывающие комментарий с конкретным изображением и пользователем, который его оставил. Текст комментария хранит содержание комментария, а дата добавления комментария фиксирует момент его создания.

Таблица «Избранное»:

– ID пользователя (внешний ключ, связывающий с таблицей «Пользователи»);

– ID изображения (внешний ключ, связывающий с таблицей «Изображения»).

Эта таблица предназначена для хранения изображений, добавленных пользователями в избранное. Каждая запись в таблице содержит ID пользователя и ID изображения, образующие связь между пользователем и изображением. Такая структура данных позволяет пользователям отслеживать и управлять своими избранными изображениями.

2.2 Проектирование мобильного приложения

На данном этапе был разработан макет приложения, создание которого позволило более точно определить функции, которые будут доступны в приложении.

На рисунке 2.2 представлены макеты экранов приложения. В ходе разработки дизайн конечного приложения будет немного изменен.

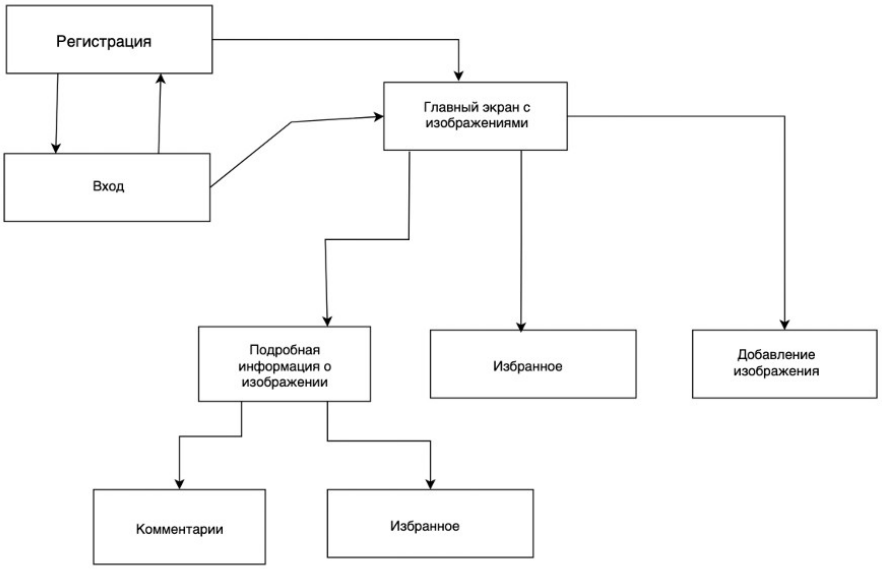


Рисунок 2.2 – Схема взаимодействия между экранами мобильного приложения

Также на этапе проектирования мобильного приложения была разработана диаграмма вариантов использования. Целью разработки диаграммы является описание функционала мобильного приложения, который будет доступен каждой группе пользователей. Диаграмма представлена на рисунке 2.3.

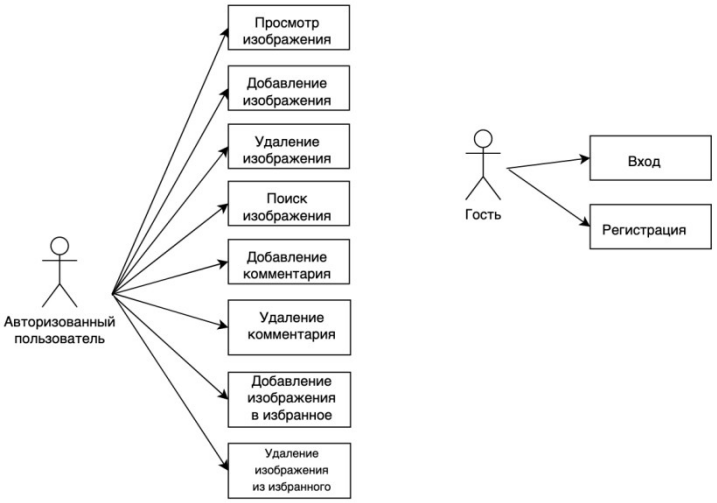


Рисунок 2.3 – Диаграмма вариантов использования

2.3 Описание средств разработки

Flutter [4], разработанный компанией Google, представляет собой открытую платформу для создания мобильных приложений. Он использует язык

программирования Dart и фреймворк Flutter, обеспечивая возможность разработки кроссплатформенных приложений, которые могут быть запущены как на устройствах Android, так и на iOS. Одним из главных преимуществ Flutter является возможность написания единого кода для различных платформ, что ускоряет процесс разработки и сокращает издержки. Flutter также позволяет разработчикам создавать высокопроизводительные и интуитивно понятные мобильные приложения, используя привычные инструменты и методы разработки.

SQLite [5] — это компактная система управления реляционными базами данных, широко применяемая в мобильной разработке с использованием Flutter. Она обеспечивает эффективное хранение и управление данными приложения, включая операции создания, чтения, обновления и удаления записей. SQLite идеально подходит для приложений с небольшими и средними объемами данных, которые требуют локального хранения.

2.4 Вывод

Проектирование мобильного приложения фотохостинга включало создание макетов и диаграммы вариантов использования, что позволило определить основные функции и пользовательский интерфейс. Связь между таблицами базы данных «Пользователи», «Изображения», «Комментарии» и «Избранное» обеспечивает эффективное управление данными, поддерживая целостность и взаимодействие между пользователями.

Для разработки приложения был выбран фреймворк Flutter, который позволяет создавать кроссплатформенные приложения для Android и iOS. Это ускоряет процесс разработки и снижает затраты. SQLite была выбрана для управления данными, обеспечивая эффективное и надежное хранение и обработку данных, подходящее для приложений с небольшими и средними объемами данных.

В целом, выбранные инструменты и методы разработки обеспечивают создание функционального и удобного мобильного приложения фотохостинга, удовлетворяющего потребности пользователей в управлении и обмене изображениями.

3 Реализация приложения

Реализация программного продукта означает превращение абстрактной концепции или детального плана, заложенных в процессе проектирования, в живой, функциональный продукт. Этот этап включает в себя конкретные шаги по написанию кода, настройке баз данных и другие технические моменты, необходимые для того, чтобы концепция стала реальностью. В процессе реализации разработчики превращают абстрактные идеи в конкретные алгоритмы, структуры данных и пользовательские интерфейсы, что в конечном итоге позволяет создать работающее приложение или программу.

3.1 Разработка мобильного приложения

Мобильное приложение содержит в себе несколько папок. Каждая из перечисленных папок в мобильном приложении выполняет определенную функцию:

- Registration: содержит классы, связанные с регистрацией пользователей.
- Authorization: содержит классы, связанные с авторизацией пользователей.
- Gallery: содержит классы, отвечающие за отображение галереи изображений.
- AddPhoto: содержит классы, отвечающие за добавление новых фотографий.
- ViewImage: содержит классы, отвечающие за отображение выбранного изображения.
- Favorites: содержит классы, отвечающие за работу с избранными изображениями.

3.2 Разработка локальной базы данных

Для взаимодействия с локальной базой данных используется библиотека sqflite. Sqflite – это популярная библиотека Dart, которая позволяет разработчикам Flutter работать с базами данных SQLite. Она предоставляет простой и удобный API для создания, чтения, обновления и удаления данных.

Метод, представленный в листинге 3.1 создаёт таблицы `Users`, `Images`, `Comments`, и `Favorites` с нужными полями и связями между ними.

```
Future<void> _createDb(Database db, int version) async {
  await db.execute('''
    CREATE TABLE Users (
      ID INTEGER PRIMARY KEY,
      Name TEXT,
      PasswordHash TEXT
    )
  ''');

  await db.execute('''
```

```
    CREATE TABLE Images (
      ID INTEGER PRIMARY KEY,
      UserID INTEGER,
      Name TEXT,
      Description TEXT,
      ImagePath TEXT,
      DateAdded TEXT,
      FOREIGN KEY (UserID) REFERENCES Users(ID)
    )
  ''');

  await db.execute('''
    CREATE TABLE Comments (
      ID INTEGER PRIMARY KEY,
      ImageID INTEGER,
      UserID INTEGER,
      Text TEXT,
      DateAdded TEXT,
      FOREIGN KEY (ImageID) REFERENCES Images(ID),
      FOREIGN KEY (UserID) REFERENCES Users(ID)
    )
  ''');

  await db.execute('''
    CREATE TABLE Favorites (
      UserID INTEGER,
      ImageID INTEGER,
      FOREIGN KEY (UserID) REFERENCES Users(ID),
      FOREIGN KEY (ImageID) REFERENCES Images(ID),
      PRIMARY KEY (UserID, ImageID)
    )
  ''');
}
```

Листинг 3.1 — создание базы данных

Остальной скрипт будет рассмотрен в приложении А.

3.3 Обеспечение безопасности информационной системы

Для обеспечения безопасности данных и предотвращения несанкционированного доступа к аккаунтам пользователей в мобильном приложении используется хеширование паролей. Для этой цели была выбрана библиотека `crypto`, которая предоставляет набор хеширующих функций для языка Dart. В состав библиотеки входят алгоритмы SHA-1, SHA-224, SHA-256 и MD5.

В данном приложении для хеширования паролей используется алгоритм MD5. Алгоритм MD5 (англ. Message-Digest Algorithm 5) – это криптографическая хеш-функция, которая на входе принимает данные произвольной длины и выдаёт хеш размером 128 бит. Хотя MD5 считается менее безопасным по сравнению с

более современными алгоритмами, его применение в локальных приложениях, где данные не передаются через открытые сети, всё же может быть оправданным.

Процесс хеширования паролей осуществляется на стороне мобильного приложения. Это важно, так как передача паролей в незашифрованном виде через сеть может привести к их перехвату злоумышленниками.

Ниже в листинге 3.2 представлен код, который реализует хеширование паролей с использованием MD5 в Dart.

```
String _generatePasswordHash(String password) {

  var bytes = utf8.encode(password);
  var digest = md5.convert(bytes);
  return digest.toString();
}
```

Листинг 3.2 — хеширование паролей

3.4 Вывод

Данный раздел представляет собой описание процесса реализации программного средства, начиная с разработки его структуры и функциональности до обеспечения безопасности информационной системы.

Во-первых, описывается разработка мобильного приложения, которое организовано в виде нескольких функциональных папок. Каждая из этих папок выполняет определенные функции, такие как регистрация пользователей, авторизация, управление галереей изображений и работа с избранными фотографиями.

Затем описывается разработка локальной базы данных с использованием библиотеки sqflite, которая предоставляет удобный API для работы с базами данных SQLite в среде Dart и Flutter. Методы создают необходимые таблицы с соответствующими полями и связями между ними для хранения пользовательских данных, изображений, комментариев и избранных элементов.

Наконец, обеспечивается безопасность информационной системы путем хеширования паролей пользователей. Для этой цели используется библиотека crypto с алгоритмом MD5, который применяется на стороне мобильного приложения. Это важно для защиты конфиденциальности пользовательских данных и предотвращения несанкционированного доступа к аккаунтам.

4 Тестирование приложения

Тестирование мобильных приложений играет ключевую роль в обеспечении их качества и производительности. Оно помогает выявить и устранить потенциальные сбои, гарантировать стабильную работу без ошибок и улучшить общее восприятие пользователями. Это способствует повышению уровня доверия к приложению и увеличению числа загрузок.

На рисунке 4.1 представлена форма для регистрации. Здесь показано, что при попытке создать пользователя с именем, которое уже существует в базе данных, будет выдано сообщение об ошибке, указывающее, что пользователь с таким именем уже зарегистрирован.

Также продемонстрирована страница авторизации. При вводе некорректного логина или пароля отображается сообщение об ошибке, информирующее пользователя о неправильных данных для входа.

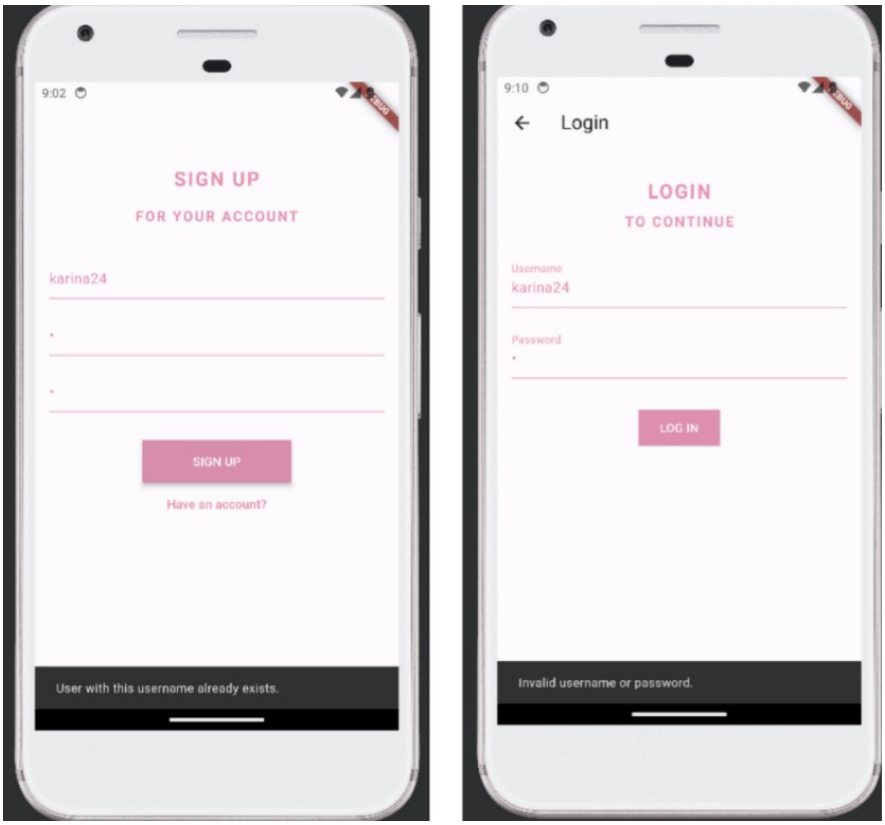


Рисунок 4.1 — Экраны регистрации и авторизации

На следующем экране представлена главная страница приложения (рис 4.2). В верхней части находится строка поиска, в которую пользователь может ввести запрос. По нажатию на кнопку с изображением лупы осуществляется поиск.

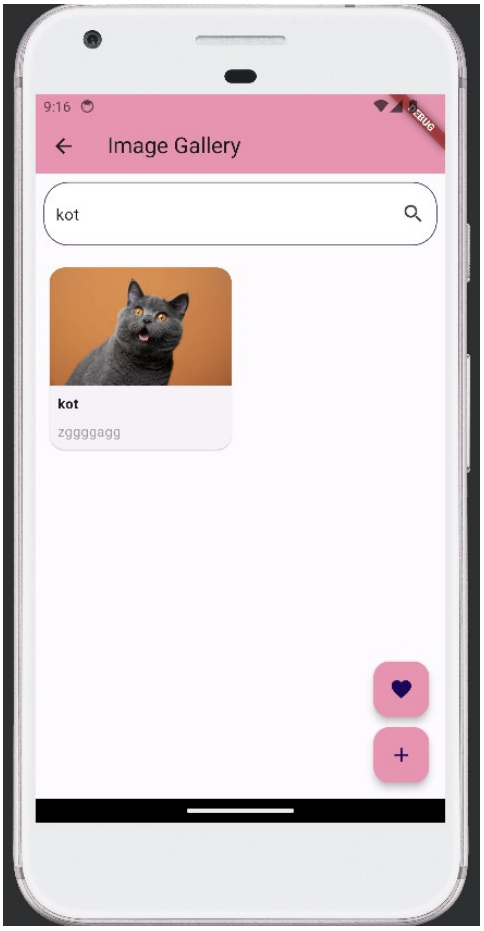


Рисунок 4.2 – Результат поиска

Таким образом, приложение предоставляет пользователю удобный интерфейс для взаимодействия с базой данных и поиска необходимой информации.

4.1 Вывод

Тщательное тестирование является неотъемлемой частью разработки прогр аммных продуктов, включая мобильные приложения. Этот этап имеет критическое значение, поскольку позволяет выявлять и устранять ошибки до выпуска продукта, гарантируя высокое качество и надежность конечного решения. В контексте мобил ьных приложений, тестирование охватывает широкий спектр аспектов - от проверк и функциональности и производительности до оценки безопасности и совместимос ти

5 Руководство по использованию

5.1 Руководство пользователя мобильного приложения

При входе в приложение пользователь попадает на страницу регистрации, н а ней же внизу будет предложено «Sign Up» ддя регистрации и «Have an account?» для пользователей уже имеющих аккаунт, что можно увидеть на рисунке 5.1.

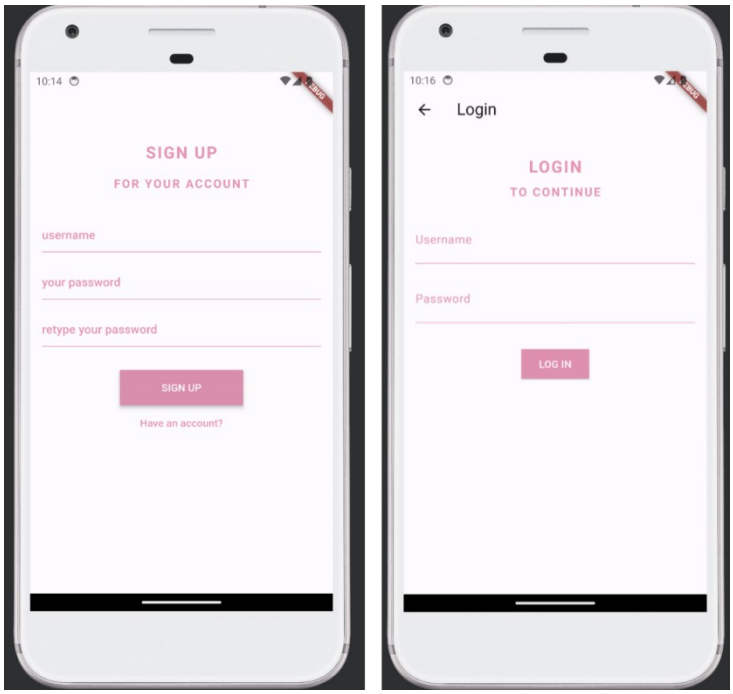


Рисунок 5.1 – Страница входа в приложение

Важно отметить, что при вводе пароля символы заменяются на точки, что обеспечивает скрытие пароля от посторонних глаз и повышает безопасность ввода (рис 5.2).

Рисунок 5.2 – Скрытие пароля

На рисунке 5.3 представлена главная страница приложения, на которую пользователь попадает после ввода правильного логина и пароля.

В верхней части экрана находится строка поиска, в которую можно ввести запрос. Поиск осуществляется по нажатию на значок лупы, расположенный справа от строки ввода.

Ниже строки поиска отображается галерея изображений, где каждое изображение снабжено заголовком и описанием. В правом нижнем углу экрана присутствуют две кнопки: кнопка в виде сердца для работы с избранными изображениями и кнопка с плюсом для добавления новых фотографий.

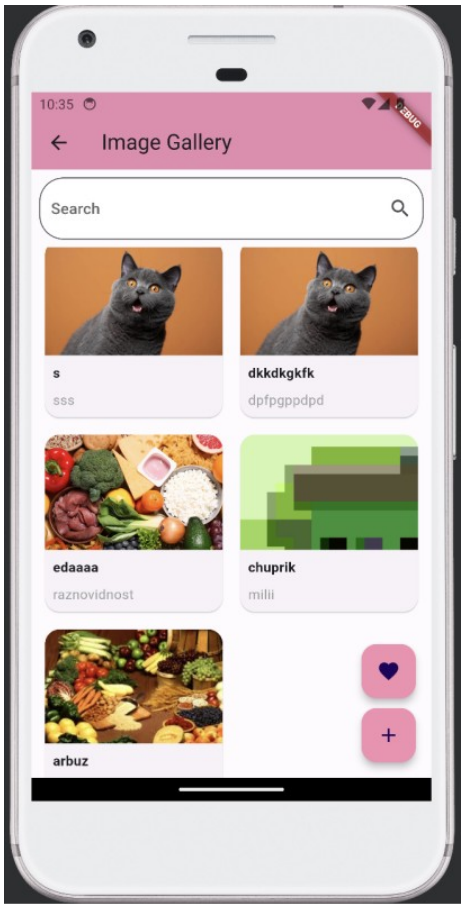


Рисунок 5.3 – Главная страница

На рисунке 5.4 изображено визуальное представление поиска. Он показывает, как вводится ключевое слово или фраза, как система анализирует базу данных изображений, и как результаты поиска выводятся пользователю.

Функция поиска изображений по ключевым словам работает следующим образом: когда пользователь вводит определенное слово или фразу в поисковую строку, система проверяет свою базу данных изображений на наличие совпадений с этими ключевыми словами. Если изображение содержит в своем описании указанные ключевые слова, оно считается соответствующим запросу пользователя.

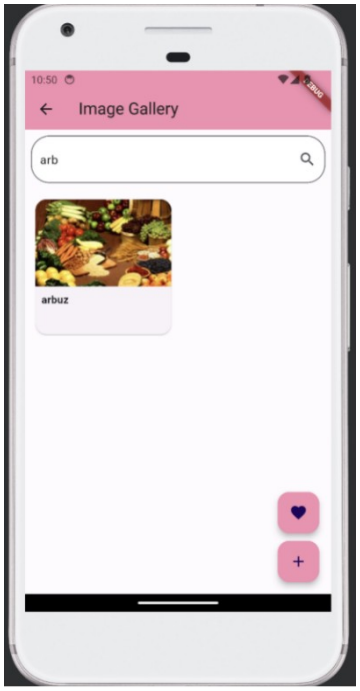
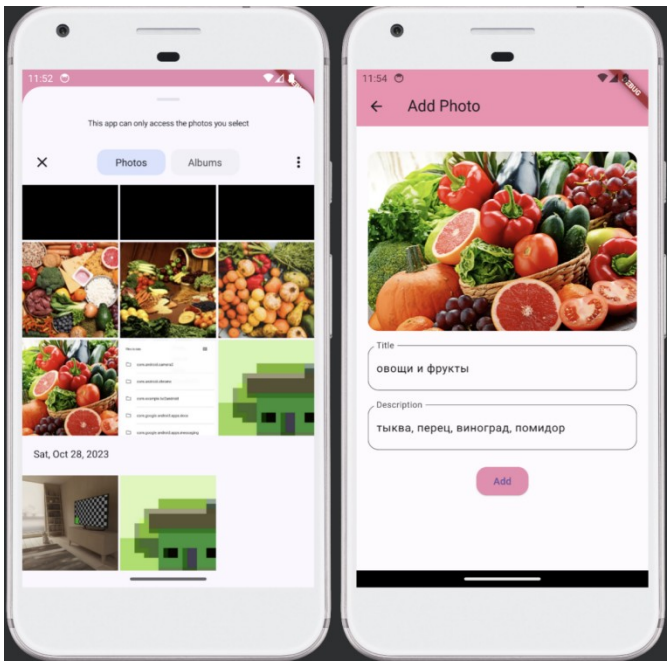


Рисунок 5.4 – Поиск на главной странице

При нажатии на иконку с плюсом появляется окно с выбором картинки из галереи (рис 5.5). После выбора изображения пользователь автоматически перенаправляется на новую страницу, где ему предлагается указать название и описание картинки (рис 5.6). После ввода информации пользователь должен нажать кнопку «Add», чтобы завершить процесс.



Рисунки 5.5 и 5.6 – Добавление изображения

При нажатии на изображение на главном экране происходит переход на новую страницу (рис 5.7), где пользователь может просматривать изображение в увеличенном виде. На этой странице также отображается название и описание изображения, которые можно прочитать.

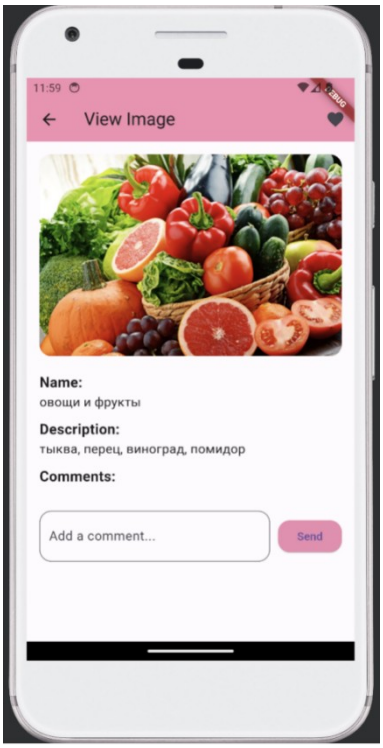


Рисунок 5.7 – Добавление изображения

На странице, где пользователь просматривает изображение, также есть функция написания комментариев. Пользователь может оставить свой комментарий под изображением (рис 5.8). Кроме того, пользователь имеет возможность удалять только свои собственные комментарии, обеспечивая тем самым контроль над своим контентом.

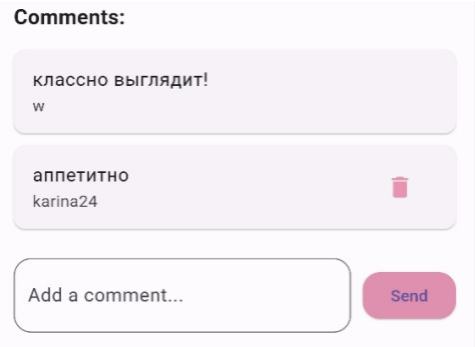


Рисунок 5.8 – Комментарии

Также расположено сердечко в верхней части. При нажатии на него изображение добавляется в список избранных.

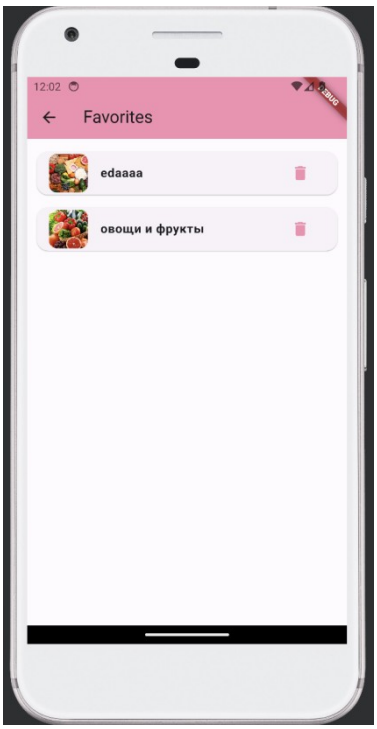


Рисунок 5.9 – Избранное

На главной странице пользователь может нажать на иконку с сердцем, расположенную сбоку внизу, чтобы перейти на страницу с избранными изображениями. Здесь отображаются все изображения, добавленные в избранное (рис 5.9). Пользователь также имеет возможность удалить изображение из избранного, нажав на значок корзины рядом с ним.

5.2 Вывод

Мобильное приложение предоставляет удобный интерфейс для работы с изображениями. Пользователи могут зарегистрироваться или войти, просматривать галерею, добавлять новые фотографии и искать изображения по ключевым словам. Возможность добавления комментариев и управления избранными изображениями добавляет социальный аспект и удобство использования. Приложение обеспечивает безопасность и контроль над контентом.

Заключение

В рамках разработки мобильного приложения для фотохостинга были задействованы передовые технологии, включая мощный фреймворк Flutter, признанный своей эффективностью в создании кроссплатформенных приложений. Это решение обеспечило не только высокую производительность, но и плавную работу приложения на различных устройствах, от смартфонов до планшетов.

Неотъемлемой частью функционала приложения стала использование локальной базы данных SQLite, что позволило эффективно организовать хранение и управление данными о изображениях и аккаунтах пользователей. Благодаря этому пользователи могут быстро получать доступ к информации и сохранять свои данные между сеансами использования приложения, что повышает удобство его использования.

Созданное приложение предлагает обширный набор возможностей, начиная от регистрации и авторизации пользователей, и заканчивая возможностью просмотра, добавления и удаления изображений. Пользователи могут легко находить интересующие их фотографии с помощью поиска по ключевым словам и оставлять комментарии к изображениям, что создает дополнительную социальную составляющую приложения.

Особое внимание уделяется функционалу добавления изображений в избранное и управлению ими. Это предоставляет пользователям возможность организовывать свою коллекцию фотографий и удобно находить их в будущем, подчеркивая индивидуальные предпочтения пользователей.

В итоге, разработанное приложение является мощным инструментом для обмена и управления изображениями, сочетающим в себе высокий уровень функциональности и удобства использования. Это воплощение передовых технологий и эффективной организации данных, что делает его незаменимым помощником в области фотографии и обмена контентом.

Список использованных источников

[1] Pinterest [Электронный ресурс]. – Режим доступа: https://www.pinterest.com/– Дата доступа: 03.03.2024

[2] Designspiration [Электронный ресурс]. – Режим доступа: https://www.designspiration.com/ – Дата доступа: 03.03.2024

[3] Dribbble [Электронный ресурс]. – Электронные данные. – Режим доступа: https://dribbble.com/ – Дата доступа: 03.03.2024;

[4] Flutter [Электронный ресурс]. – Режим доступа: https://docs.flutter.dev/ – Дата доступа: 08.04.2024

[5] sqflite [Электронный ресурс]. – Режим доступа: https://pub.dev/packages/sqflite – Дата доступа: 12.04.2024

Приложение А

```
class DatabaseHelper {
  static Database? _database;
  Future<Database> get database async {
    if (_database != null) return _database!;
    String databasesPath = await getDatabasesPath();
    String path = join(databasesPath, 'kursach.db');
    _database = await openDatabase(path, version: 1, onCreate: _creat
eDb);
    return _database!;
  }
  Future<List<Map<String, dynamic>>> getAllImages() async {
    final Database db = await database;
    final List<Map<String, dynamic>> images = await db.query('image
s');
    return images;
  }
Future<int> insertUser(Map<String, dynamic> row) async {
    Database db = await database;
    return await db.insert('Users', row);
  }
  Future<List<Map<String, dynamic>>> getAllUsers() async {
    Database db = await database;
    return await db.query('Users');
  }
Future<Map<String, dynamic>?> getUserByUsername(String username) asyn
c {
  Database db = await database;
  List<Map<String, dynamic>> users = await db.query(
    'Users',
    where: 'Name = ?',
    whereArgs: [username],
  );
  return users.isNotEmpty ? users.first : null;
}
  Future<int> insertImage(Map<String, dynamic> image) async {
    Database db = await database;
    return await db.insert('Images', image);
  }
  Future<List<Map<String, dynamic>>> getImagesByUserId(int userId) as
ync {
    Database db = await database;
    List<Map<String, dynamic>> images = await db.query(
      'Images',
      where: 'UserID = ?',
      whereArgs: [userId],
    );
    return images;
  }
  Future<void> deleteImageByPath(String imagePath) async {
    Database db = await database;
    await db.delete(
      'Images',
```

```
      where: 'ImagePath = ?',
      whereArgs: [imagePath],
    );
  }
// Метод для получения информации об изображении по его пути
  Future<Map<String, dynamic>> getImageInfoByPath(String imagePath) a
sync {
    final db = await database;
    var result = await db.query('Images', where: 'ImagePath = ?', whe
reArgs: [imagePath]);
    if (result.isNotEmpty) {
      return result.first;
    } else {
      throw Exception('Image not found in database');
    }
  }
  Future<void> insertFavorite(Map<String, dynamic> favorite) async {
    final db = await database;
    await db.insert('Favorites', favorite);
  }
  Future<List<Map<String, dynamic>>> getFavorites(int userId) async {
    final db = await database;
    return await db.query('Favorites', where: 'UserID = ?', whereArgs:
 [userId]);
  }
  Future<void> deleteFavorite(int userId, int imageId) async {
    final db = await database;
    await db.delete('Favorites', where: 'UserID = ? AND ImageID = ?',
whereArgs: [userId, imageId]);
  }
  Future<Map<String, dynamic>?> getImageInfoById(int imageId) async {
    final db = await database;
    var result = await db.query('Images', where: 'ID = ?', whereArgs:
[imageId]);
    return result.isNotEmpty ? result.first : null;
  }
  Future<void> insertComment(Map<String, dynamic> comment) async {
    final db = await database;
    await db.insert('Comments', comment);
  }
  Future<List<Map<String, dynamic>>> getCommentsByImageId(int imageI
d) async {
    final db = await database;
    final comments = await db.rawQuery('''
      SELECT Comments.*, Users.Name as Username FROM Comments
      INNER JOIN Users ON Comments.UserID = Users.ID
      WHERE Comments.ImageID = ?
    ''', [imageId]);
    return comments;
  }
  Future<void> deleteComment(int commentId) async {
    final db = await database;
```

```
    await db.delete('Comments', where: 'ID = ?', whereArgs: [commentI
d]);
  }
}
Future<void> _registerUser(BuildContext context) async {
    String username = usernameController.text.trim();
    String password = passwordController.text.trim();
    String retypePassword = retypePasswordController.text.trim();
    if (username.isNotEmpty &&
        password.isNotEmpty &&
        password == retypePassword) {
      String passwordHash = _generatePasswordHash(password);
// Проверка наличия пользователя с таким именем
      Map<String, dynamic>? existingUser = await dbHelper.getUserByUs
ername(
          username);
      if (existingUser != null) {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text('User with this username already exists.'),
          ),
        );
        return;
      }
      Map<String, dynamic> user = {
        'Name': username,
        'PasswordHash': passwordHash,
      };
      int id = await dbHelper.insertUser(user);
      print('User registered with ID: $id');
      await _displayUsers();
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => AuthorizationScreen
()),
      );
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('Please fill in all fields correctly.'),
        ),
      );
    }
  }
  Future<void> _displayUsers() async {
    List<Map<String, dynamic>> users = await dbHelper.getAllUsers();
    print('All users:');
    for (Map<String, dynamic> user in users) {
      print(
          'ID: ${user['ID']}, Name: ${user['Name']}, Password: ${user
['PasswordHash']}');
  }
}
```