МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-98 01 03 “Программное обеспечение информационной

безопасности мобильных систем”

Специализация Инженер-программист

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

Приложение для проведения олимпиад (соревнований) по базам данных

Выполнил студент Барабанов Дмитрий Сергеевич

(Ф.И.О.)

Руководитель проекта ассистент Уласевич Н. И. (учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой ст. преподаватель Блинова Е. А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: ассистент Уласевич Н. И. (учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: ассистент Уласевич Н. И. (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

2024, Минск

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc167095803)

[1 Постановка задачи и обзор аналогичных решений 4](#_Toc167095804)

[1.1 Постановка задачи 4](#_Toc167095805)

[1.2 Обзор аналогичных решений 4](#_Toc167095806)

[1.2.1 Заметки (Apple) 4](#_Toc167095807)

[1.2.2 ЛидерТаск 5](#_Toc167095808)

[1.2.3 Notion 6](#_Toc167095809)

[1.3 Аналогичность и различие программных средств 7](#_Toc167095810)

[2. Проектирование приложения 8](#_Toc167095811)

[2.1.Диаграмма вариантов использования 8](#_Toc167095812)

[2.2.Архитектура 9](#_Toc167095813)

[2.3.Проектирование базы данных 10](#_Toc167095814)

[2.4.Выводы по главе «Проектирование приложения» 11](#_Toc167095815)

[3.Реализация приложения 12](#_Toc167095816)

[3.1.Разработка приложения 12](#_Toc167095817)

[3.1.1 Хранение данных 12](#_Toc167095818)

[3.1.2 Разработка пользовательского интерфейса 15](#_Toc167095819)

[3.1.3 Разработка авторизации 19](#_Toc167095820)

[3.2.Выводы по главе «Разработка приложения» 20](#_Toc167095821)

[4. Тестирование приложения 22](#_Toc167095822)

[5. Руководство пользователя 23](#_Toc167095823)

[5.1.Мобильное приложение 23](#_Toc167095824)

[5.2. Выводы по главе «Руководство пользователя» 26](#_Toc167095825)

[Заключение 27](#_Toc167095826)

[Список использованных источников 28](#_Toc167095827)

[ПРИЛОЖЕНИЕ А. Код страницы «KeepDetailScreen» 29](#_Toc167095828)

# **Введение**

Каждый день в нашей голове миллион мыслей и идей, вещей, которые было бы неплохо помнить и иметь к ним доступ под рукой, но что же может нам помочь? Ответ прост – любой находящийся под рукой гаджет. В рамках курсового проекта по данной дисциплине было разработано приложение заметок, позволяющее взаимодействовать с ним на любом удобном вам устройстве будь то смартфон, планшет, ноутбук или персональный компьютер с доступ в интернет и без.

Приложение получило название ThinkSync, что означает конкатенацию слов, описывающих его функционал. Приложение в реальном времени синхронизирует заметки созданные в рамках одной учетной записи между всеми устройствами, на которых совершена авторизация с использованием данной учетной записи.

В качестве архитектуры приложение использует архитектуру клиент-сервер. Клиент приложения написан на кроссплатформенном фреймворке Flutter, что позволяет ему работать на широком списке платформ без необходимости глобального переноса кодовой базы с платформы на платформу. Данный фреймворк позволяет сократить внушительную часть времени и расходов, а также ускоряет поддержку и обновление приложения.  
Курсовой проект состоит из девяти разделов: введение, содержание, обзор аналогов, проектирование программного продукта, программная реализация продукта, тестирование, руководство по использованию, заключение и список литературных источников.

# **1 Постановка задачи и обзор аналогичных решений**

1.1 Постановка задачи

В рамках курсового проекта необходимо создать мобильное приложение для платформы Android которое будет представлять собой приложение заметок. Результатом данного проекта будет функциональное и интуитивно понятное приложение для создания, управления заметками с использованием фреймворка Flutter. Приложение должно предоставлять пользователям следующий функционал:

* создание заметок;
* редактирование заметок;
* удаление заметок;
* хранение заметок в локальной базе данных;
* синхронизация заметок через облачное хранилище;
* аутентификация пользователя.
  1. **Обзор аналогичных решений**

В данном разделе будет произведен обзор и анализ аналогичных

решений, которые уже представлены другими разработчиками и пользуются спросом среди пользователей. Данный раздел поможет нам определить ключевые функции, которые должны присутствовать в нашем приложении для того чтобы полностью удовлетворять пользовательские потребности и предоставлять лучший опыт взаимодействия пользователя с приложением. Также раздел позволит нам определиться с примерным интерфейсов будущего приложения.

**1.2.1 Заметки (Apple)**

Данное приложение было разработано для всех операционных систем поставляющихся с продуктами компании Apple. Приложение имеет следующие возможности:

* добавление заметок;
* удаление заметок;
* изменение заметок;
* возможность сканирования документов;
* возможность пометки документов различными цветамиж
* синхронизация данных между устройствами с одним Apple ID;
* поддержка вставки информации различных типов данных.

Единственный недостаток приложения – работа только на устройствах производства компании Apple, что не дает возможности использования приложения в гибкой экосистеме пользователя. Также к недостаткам можно добавить излишнюю функциональность приложения.

Интерфейс приложения (рис. 1.1) простой и понятный, адаптируется под пользовательскую тему и не нагружен различными кнопками. Пользователю сразу становится понятно где расположены необходимые функции.

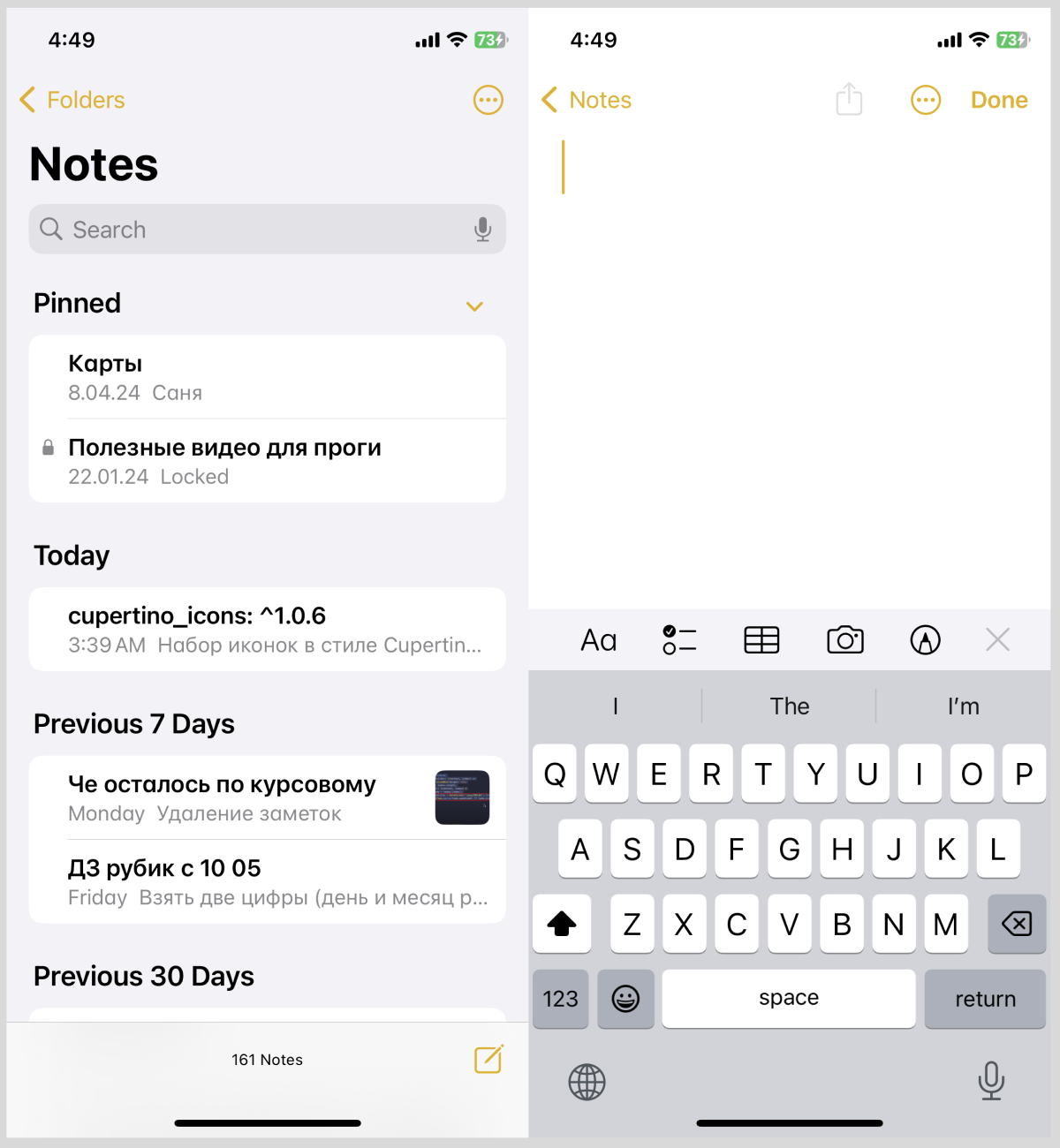


Рисунок 1.1 – Apple Notes

**1.2.2** **ЛидерТаск**

ЛидерТаск — отечественное кроссплатформенное приложение для ведения личных заметок и планов, [канбан](https://www.cnews.ru/book/Kanban_-_%D0%9A%D0%B0%D0%BD%D0%B1%D0%B0%D0%BD_-_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BE%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8_%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%B0_%D0%B8_%D1%81%D0%BD%D0%B0%D0%B1%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F_%D1%82%D0%BE%D1%87%D0%BD%D0%BE_%D0%B2_%D1%81%D1%80%D0%BE%D0%BA_)-досок и бизнес-проектов. Приложение работает на всех мобильных устройствах и ПК, в том числе и автономно.

Функционал, предоставляемый приложением:

* составления списка дел на каждый день;
* создание цикличных событий;
* работа с контактами;
* установка приоритетов;
* добавление подзадач и делегирование задач;
* настраивать метки, цвета и доступ;
* создавать напоминания и уведомления;
* синхронизировать данные.

Интерфейс приложения представлен на рисунке 1.2 ниже:

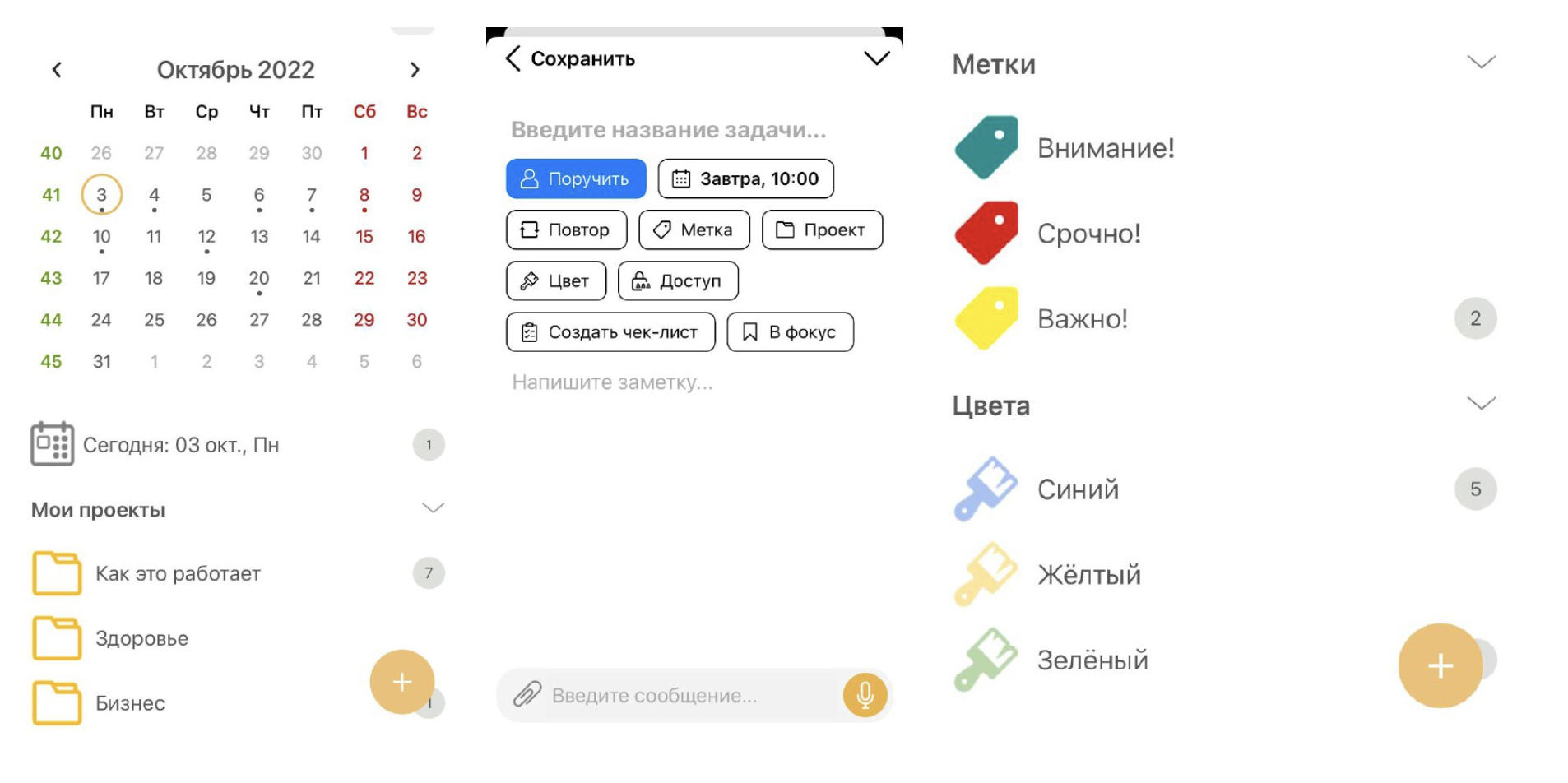


Рисунок 1.2 - Интерфейс приложения ЛидерТаск

Дизайн приложения оставляет желать лучшего, нет такой интуитивности как в приложении Apple Заметки, также отсутствует создание учетной записи в бесплатной версии, однако есть и премиум-доступ, который стоит от 2650 российских рублей в год. С ним увеличивается количество фильтров для поиска и инструментов для оформления записей. Подписка также дает возможность хранить данные в облаке и автоматически синхронизировать информацию на всех устройствах.

**1.2.3 Notion**

Приложение пользуется огромной популярностью среди пользователей различных платформ. Целевая аудитория приложения – как одиночные пользователи, так и команды различных размеров. В приложении имеются как платный так и бесплатный тариф, которые подойдут различным группам пользователей. Приложение предоставляет огромное количество возможностей, основными из которых является:

* создание и организация задач;
* создание страниц и подстраниц;
* создание канбан-досок;
* cоздание диаграм для планирования задач;
* коллаборация;
* интеграция с внешними календарями;
* проектный менеджмент;
* мультимедийная интеграция;
* интеграции с внешними сервисами;
* кроссплатформенность.

Приложение имеет весьма удобный и интуитивный интерфейс (рис. 1.3).

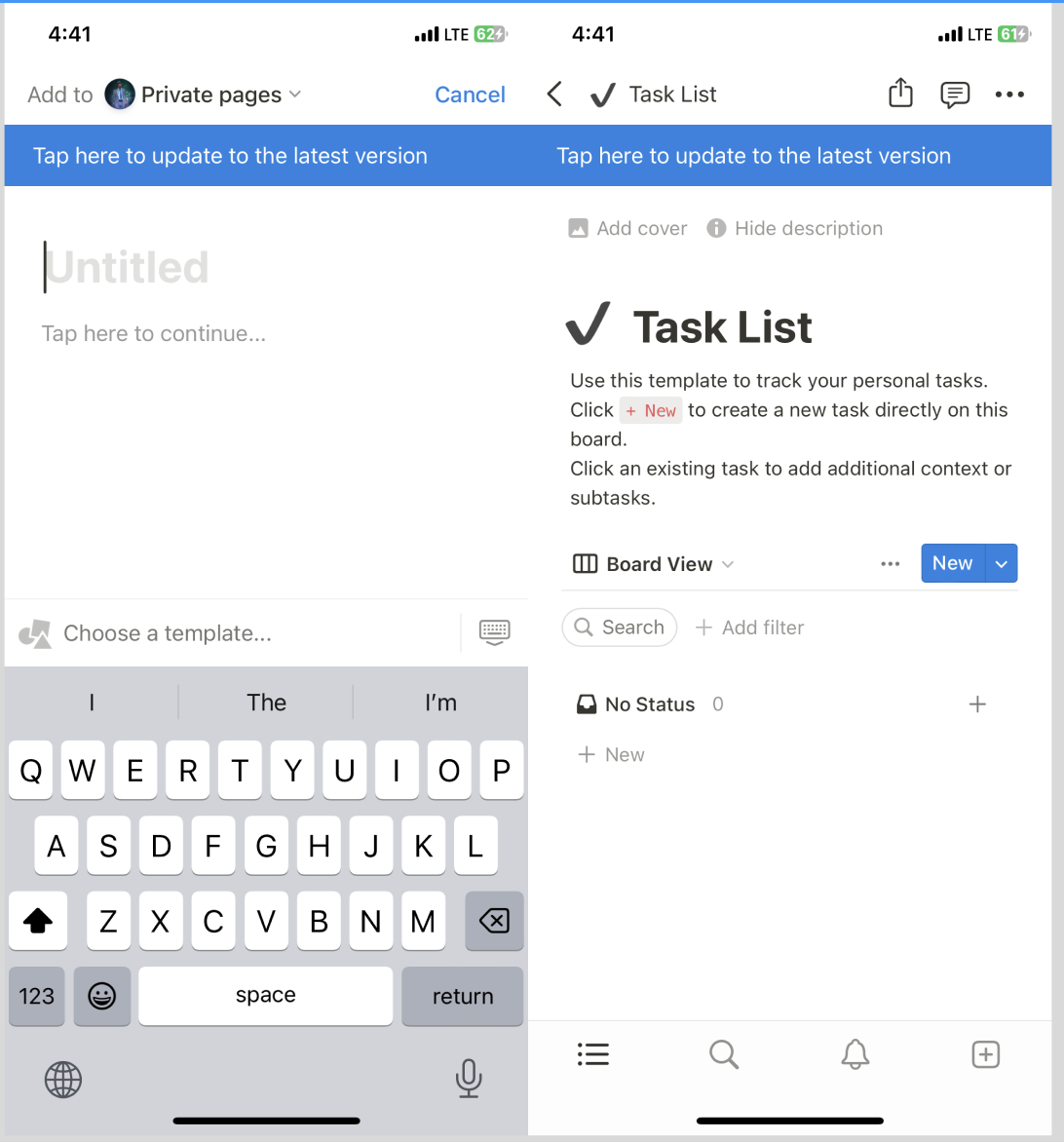


Рисунок 1.3 – Приложение Notion

**1.3 Аналогичность и различие программных средств**

Схожесть всех программных средств заключается в том, что все они имеют возможность создания учетной записи или же работают на ее основе. Также некоторых из них имеют платные функции, что совершенно неуместно в данной категории приложений.

Главные отличия приложений в их функциональности и стоимости. Так, например, в приложениях ЛидерТаск и Notion есть платные функции, в отличии от приложения Apple Заметки. Однако, есть моменты в которых функционал приложений перегружен, и переводит их из категории “приложений для заметок” в “органайзеры”.

Анализируя выше приведенные примеры и определив их категории, можно сделать вывод что приложение заметок должно включать в себя следующий функционал:

* создание заметок;
* редактирование заметок;
* удаление заметок;
* хранение заметок в локальной базе данных;
* синхронизация заметок через облачное хранилище;
* аутентификация пользователя.

Это минимальный достаточный функционал, который позволит приложению быть конкурентноспособным на рынке.

**2. Проектирование приложения**

Проектирование приложения – самая важная часть разработки, ведь именно на этапе проектирования выявляются как мелкие огрехи, так и крупные недоработки, исправление которых на этапе его развёртывания будет крайне затруднительно и накладно. Важность проектирования в данной курсовой работе очень высока ещё и потому, что разрабатываться будут три компонента: мобильное приложение, серверная часть, а также веб-приложение. Необходимо продумать то, как они будут без помех взаимодействовать друг с другом.

**2.1.Диаграмма вариантов использования**

Диаграмма вариантов использования (Use Case Diagram) служит для визуализации функциональности системы с точки зрения ее пользователей. В рамках данной системы выделяется всего одна роль: пользователь. В свою очередь данная роль обладает определенными функциями.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация сервисов или функций, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий.

Участники могут принимать участие в проводимых олимпиадах, регистрируясь для участия в мероприятии. Они также имеют функцию отправки выполненных заданий на проверку системе, включая выбор конкретного задания и загрузку своих решений. После завершения олимпиады участники могут просматривать свои результаты.

Диаграмма варианта использования для пользователя представлена на рисунке 2.1.

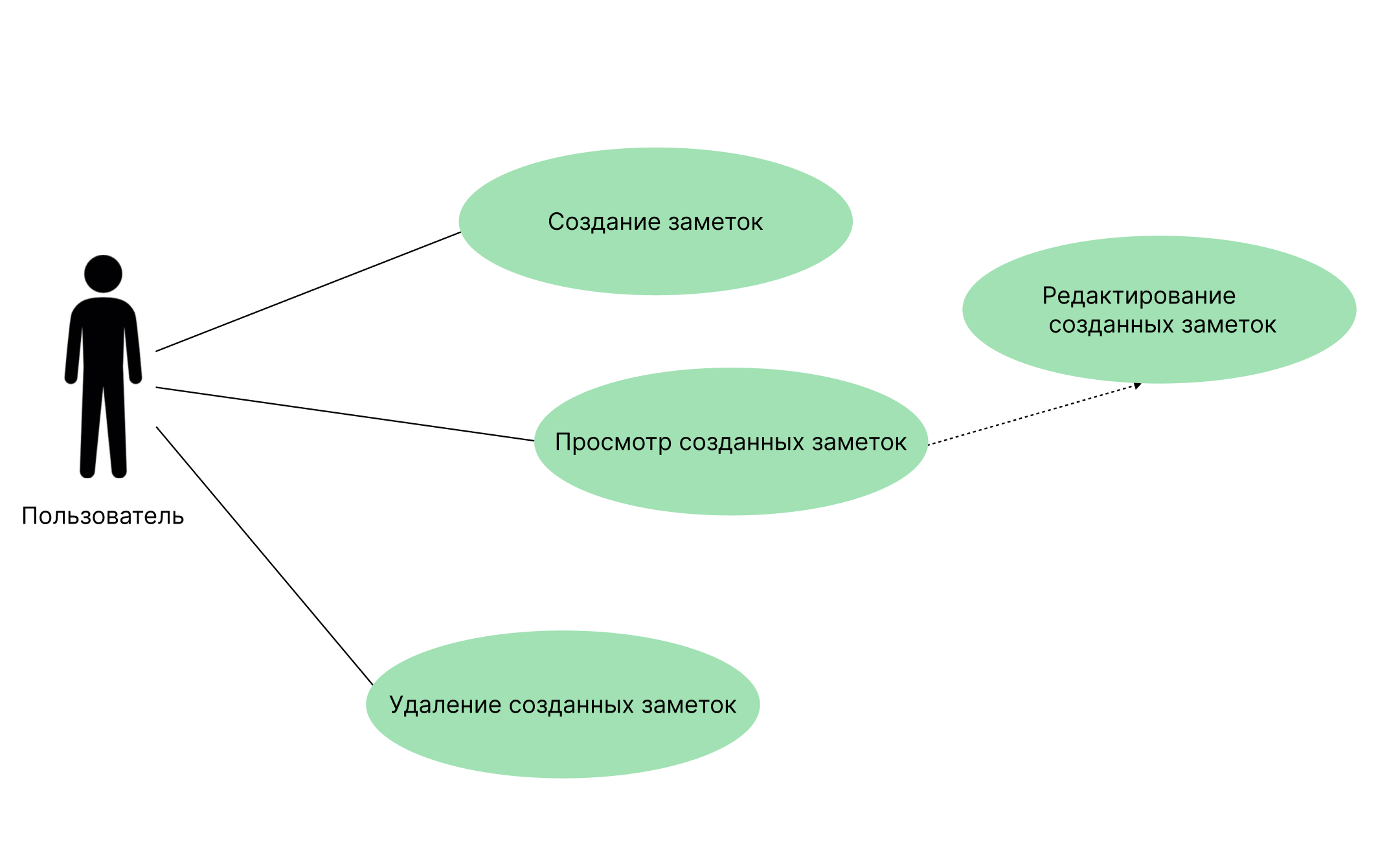


Рисунок 2.1 – Диаграмма вариантов использования

**2.2.Архитектура**

Клиент представляет собой приложение, написанное с помощью фреймворка Flutter для операционной системы Android. Однако, при небольших усилиях проект можно будет собрать и для таких платформ как IOS, Windows, MacOS и Linux. В связи с тем, что основная функция приложения – хранение данных, в формате определенном темой, было принято решение хранить данные, в данном случае заметки, распределённо. Данное решение было выбрано для обеспечения работы приложения без подключения к интернету. Заметки будут хранится на устройстве – в локальной базе данных, в данном случае – SQLite. А также основным хранилищем данных будет выступать удалённый сервер, состояние которого и будет являться источником истины.

На роль удалённого хранилища был выбран сервис, разработанный и поддерживаемый компанией Google, а именно – Firebase. Учитывая, что компания Google также разработала и на данный момент поддерживает сам фреймворк Flutter, на котором написано приложение, а также предоставляет удобные средства интеграции со своими сервисами, выбор базы данных был очевиден.

Также Firebase будет предоставлять аутентификацию для пользователей, основным способом авторизации будет Google аккаунт.

**2.3.Проектирование базы данных**

Перед началом разработки программного продукта необходимо провести проектирование базы данных. На основе сформулированных требований к будущему приложению определяется, какая информация должна храниться в базе данных, и выявляются взаимосвязи между данными. Эта структура данных отображается в модели данных предметной области, которая является фундаментальной основой для любой базы данных.

Изучение предметной области и составленный впоследствии функционал показали, что данному приложению достаточно одной таблицы, которая позволяет полностью охватить обозначенный функционал (рис. 2.2). Этой сущностью является таблица **«**KEEPS**»** которая содержит в себе заметки и всю информацию о них.

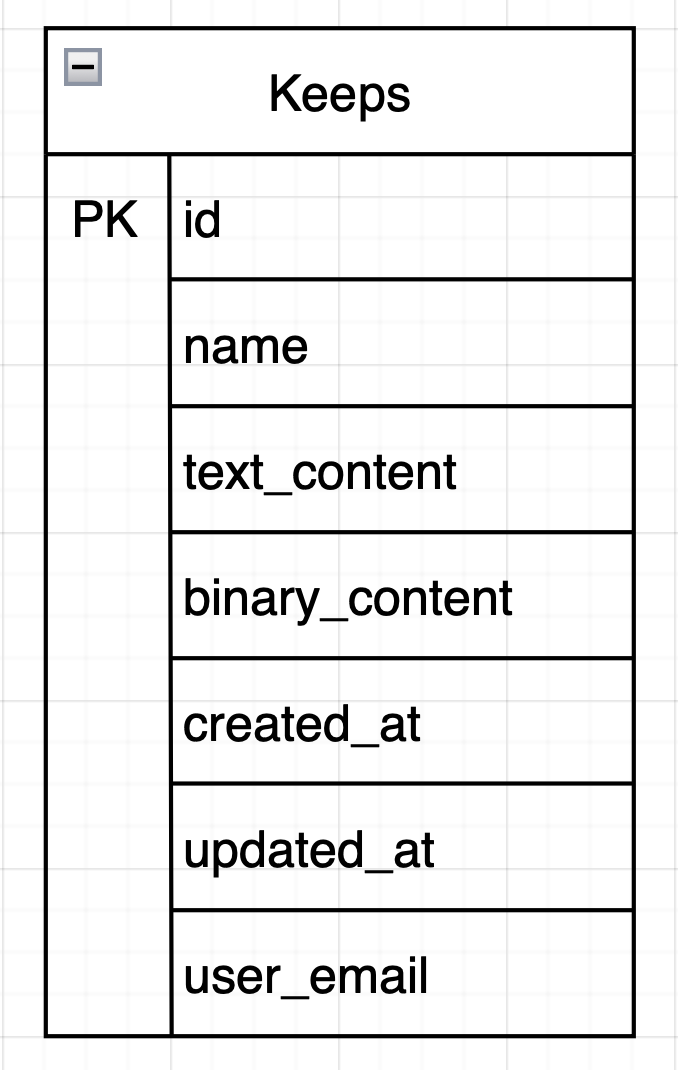


Рисунок 2.2 – Схема таблицы Keeps

Таблица содержит в себе следующие поля:

* id – поле, идентифицирующее запись в таблице базы данных;
* name – имя заметки, оно же является заголовком;
* text\_content – основное поле, содержащее тело заметки;
* binary\_content – поле которое может хранить файлы, в том числе фото , видео и иные форматы в бинарном виде;
* created\_at – хранит в себе дату, и время создания заметки;
* updated\_at – поле, хранящее в себе время последнего изменения заметки, при создании заметки оно идентично значению сreated\_at.

Перечисленных выше таблиц

Ввиду того, что Firebase позволяет нам разграничивать данные между пользователями, таблица пользователей в проекте будет только ухудшать производительность.

**2.4.Выводы по главе «Проектирование приложения»**

В данном разделе произведено проектирование функциональности системы с использованием диаграмм вариантов использования (Use Case Diagrams). Выделена ключевая роль – пользователь, которая обладает определенным функционалом.

Для пользователя приложения представлена диаграмма, охватывающая основные функциональные возможности, такие как регистрация, авторизация, создание заметок, редактирование заметок и удаление заметок.

В разделе также представлена архитектура приложения, охватывающая клиентскую и серверную части, базы данных, а также используемые технологии, такие как Flutter, SQLite и Firebase, который предоставляет сервисы регистрации, авторизации и хранения данных.

В целом, представленная концепция дает четкое представление о функциональности будущего приложения, его архитектуре и структуре базы данных.

**3.Реализация приложения**

Разработка приложений под Android имеет свою специфику и во многом отличается от разработки для других платформ. Изначально в SDK идут только самые базовые компоненты, которые очень сложно использовать для того, чтобы реализовать требуемый функционал, однако существует большое множество библиотек и фреймворков, предоставляемых в том числе и разработчиками Google.

Самые главные сторонних компонентов, которые использовались, это:

* firebase\_core – основная библиотека для инициализации и настройки Firebase в нашем Flutter-приложении;
* firebase\_auth – библиотека для аутентификации пользователей с использованием Firebase Aunthenticatoin;
* firebase\_storage – библиотека для получения и обработки push-уведомлений;
* Cloud\_firestore – библиотека для работы с базой данных Firestore, позволяющая хранить и синхронизировать данные в реальном времени;
* sqflite – библиотека для работы с SQLite базой данных, позволяющая хранить данные локально на устройстве;
* intl – библиотека для локализации, позволяющая работать с датами, числами и текстами на разных языках;
* flutter\_slidable – библиотека, позволяющая создавать свайповые элементы в списках, такие как слайдер для удаления элемента;
* flutter\_spinkit – набор виджетов для создания различных идикаторов загрузки в приложении.
* Material и Cupertino – библиотеки, предоставляющая компоненты из Material UI и Cupertino;

Языком программирования был выбран Dart.

**3.1.Разработка приложения**

**3.1.1 Хранение данных**

Для хранения данных используются две базы данных, одна – для хранения данных локально, вторая – для хранения данных на удаленном сервере. Список использующихся баз данных;

* SQLite – база данных для хранения данных внутри приложения и при отсутствии подключения к интернету;
* Firebase Firestore – база данных, входящая в состав Firebase, расположена на серверах Google.

Для работы с встроенной базой данных SQLite, которая располагается на устройстве используется библиотека sqflite, при первом запуске приложения создается пустая таблица в которую в процессе использования приложения будут сохранятся данные. Фрагмент кода с инициализацией базы данных представлен в листинге 3.1. При инициализации проверяется подключение устройства к интернету, если оно есть – происходит загрузка данных из Firestore в локальную базу данных SQLite, в таком случае после установки приложения и авторизации мы получаем список заметок, который создавал раньше пользователь под кредами которого мы зашли в приложение. Также стоит отметить, что пользователь имеет доступ и получает на устройство только те заметки, которые он создал, проверка выполняется по логину. Если в поле user\_email находится значение, не совпадающее с логином вошедшего в приложение пользователя – доступ к таким заметкам он не получит.

Future<void> createTable(Database database) async {  
 await database.execute(  
 """  
 CREATE TABLE IF NOT EXISTS $tableName (  
 "id" INTEGER PRIMARY KEY AUTOINCREMENT,  
 "name" TEXT NOT NULL,  
 "text\_content" TEXT,  
 "binary\_content" BLOB,  
 "created\_at" TEXT NOT NULL,  
 "updated\_at" TEXT NOT NULL,  
 "user\_email" TEXT NOT NULL  
 );  
 """  
 );  
  
 List<Keep> keeps = await \_firebaseService.fetchKeepsFromFirebase();  
  
 // Вставьте данные в SQLite  
 for (Keep keep in keeps) {  
 await database.insert(  
 tableName,  
 {  
 'name': keep.name,  
 'text\_content': keep.textContent,  
 'binary\_content': keep.binaryContent,  
 'created\_at': keep.createdAt,  
 'updated\_at': keep.updatedAt,  
 'user\_email': keep.userEmail,  
 },  
 conflictAlgorithm: ConflictAlgorithm.ignore,  
 );  
 }  
}

Листинг 3.1 – Инициализация базы данных

Для обеспечения синхронизации локальной базы данных с удаленной базой данных Firestore в приложении используются два отдельных сервиса для определения наличия подключения к интернету, и для самой синхронизации с удалённой базой данных:

* NetworkService – сервис, определяющий наличия подключения к интернету у устройства в данный момент;
* FirebaseService– база данных, входящая в состав Firebase, расположена на серверах Google.

FirebaseService позволяет приложению получать данные из Firestore, загружать их в Firestore, удалять и изменять. Код, реализующий методы получения данных из Firestore и добавления их туда с помощью FirebaseService показан в листинге 3.2 ниже.

class FirebaseService {  
 final CollectionReference keepsCollection = FirebaseFirestore.instance.collection('keeps');  
 Future<List<Keep>> fetchKeepsFromFirebase() async {  
 SharedPreferences prefs = await SharedPreferences.getInstance();  
 String? userEmail = prefs.getString('user');  
 print(userEmail);  
 QuerySnapshot snapshot = await keepsCollection.where('user\_email', isEqualTo: userEmail).get();  
 print("blablabla" + snapshot.toString());  
 return snapshot.docs.map((doc) => Keep.fromFirestore(doc)).toList();  
 }  
 Future<void> addKeepToFirebase(Keep keep) async {  
 await keepsCollection.add(keep.toFirestore());  
 }  
 Future<void> updateKeepInFirebaseByCreatedAt(String createdAt, Map<String, dynamic> data) async {  
 QuerySnapshot querySnapshot = await keepsCollection.where('created\_at', isEqualTo: createdAt).get();  
  
 if (querySnapshot.docs.isNotEmpty) {  
 // Предполагаем, что 'created\_at' является уникальным  
 String docId = querySnapshot.docs.first.id;  
 await keepsCollection.doc(docId).update(data);  
 } else {  
 throw Exception('Document with created\_at = $createdAt not found');  
 }  
 }

Листинг 3.2 – Реализация FirebaseService

Как говорилось ранее, для корректной работы приложения используется NetworkService.

Сервис выполняет проверку подключения приложения к интернету чтобы дать приложению понять, сможет-ли оно синхронизироваться с Firestore и избежать появления ошибок. Код реализации сервиса предоставлен в листинге 3.3.

class NetworkService {  
 Future<bool> hasInternetConnection() async {  
 try {  
 return await InternetConnectionChecker().hasConnection;  
 } catch (e) {  
 print('Ошибка при проверке подключения: $e');  
 return false;  
 }  
 }  
}

Листинг 3.2 – Реализация NetworkService

**3.1.2 Разработка пользовательского интерфейса**

Рассмотрим разработку мобильного приложения. Поскольку было принято разрабатывать приложение при помощи языка программированияd Dart и фреймворка Flutter – приложение можно скомпилировать под различные системы с минимальными изменениями, однако, в рамках курсового проекта приложение собрано для мобильных устройств под управлением операционной системой Android.Пользователям доступны следующие функции:

* Аутентификация по адресу электронной почты и паролю;
* Создание заметок;
* Просмотр заметокж
* Изменение заметок;
* Удаление заметок.

Проект разбит на папки, в которых находятся страницы, сервисы, модели и сервисы. Каждая страница находится с отдельной папке для упрощения их расширения, как пример – написание дополнительного виджета или модуля, который используется только на этой странице. Такая файловая структура была позаимствована из веб-разработки приложений, которые для последующего масштабирония разбиваются подобным образом. Такая организация файлов помогает быстрее ориентироваться в приложении а также упрощает понимание проекта, если он будет передан другому разработчику для доработки. В общаем и целом, структура не типична для мобильной разработки, однако, весьма эффективна.

Файловая структура приложения представлена на рисунке 3.3 ниже.

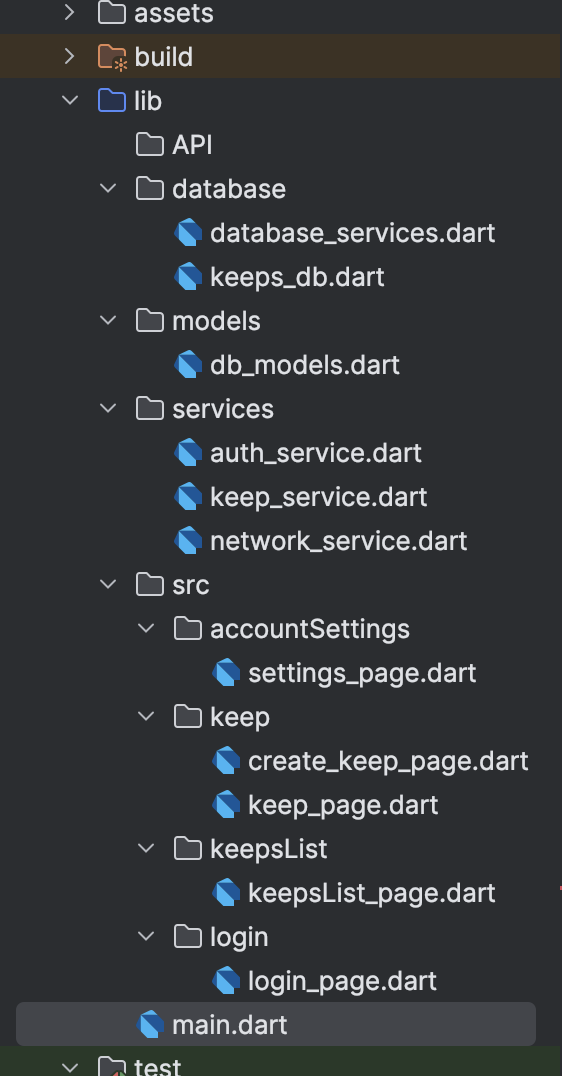


Рисунок 3.3 – Файловая структура приложения

Точкой входа в приложение является файл main.dart, ключевые части которого будут представлены в листингах ниже. Данный файл – ключевой для приложения, здесь определяются параметры подключения к сервису Firebase, осуществляется аутентификация пользователя определяются маршруты страниц, производится работа с Shared Preferences и инициализируется база данных, если приложение запускается первый раз и не имеет готового экземпляра базы данных с таблицей.

Главной функцией приложения является функция main, в ней происходят все перечисленные выше действия кроме определения маршрутов страниц, ее реализация представлена в листинге 3.4.

void main() async {  
 WidgetsFlutterBinding.ensureInitialized();  
 await Firebase.initializeApp(options:  
 FirebaseOptions(  
 apiKey: 'AIzaSyDAwVaR8kUW7CdEVxhzWuLHmVommLvoyAo',  
 appId: '1:109977764447:android:9e01cf4a3de1259cd611ea',  
 messagingSenderId: '109977764447',  
 storageBucket: 'think-sync-project.appspot.com',  
 projectId: 'think-sync-project')  
 );  
  
 final authService = AuthService();  
  
 SharedPreferences prefs = await SharedPreferences.getInstance();  
 final user = await authService.getCurrentUser();  
 prefs.setString('userEmail', user?.email ?? '');  
  
 runApp(MyApp(authService: authService));  
}

Листинг 3.4 – Функция main

Также в данном файле определяются все маршруты страниц приложения и задаётся главный маршрут, который пользователь наблюдает при запуске приложения. Происходит это всё в классе MyApp код которого представлен в листинге 3.5 ниже. Приложение включает в себя следующие маршруты страниц:

* / - страница LoginScreen, показывается пользователю первой и служит страницей для авторизации пользователя и предоставляет ему возможность перейти на страницу регистрации;
* /register – страница регистрации, доступная из LoginScreen;
* /home – страница со всеми заметками, на ней отображаются все заметки пользователя;
* /createKeep – страница создания заметки, на нее можно перейти из страницы /home, позволяет пользователю создать заметку и созранить ее в локальную и удалённую базы данных.

class MyApp extends StatelessWidget {  
  
 final AuthService authService;  
  
 const MyApp({super.key, required this.authService});  
  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 title: 'ThinkSync',  
 theme: ThemeData(  
 useMaterial3: true,  
 ),  
 initialRoute: '/',  
 routes: {  
 '/': (context) => LoginScreen(authService: authService),  
 '/home': (context) => KeepListScreen(authService: authService),  
 '/createKeep':(context) => CreateKeepScreen(authService: authService),  
 },  
 );  
 }  
}

Листинг 3.4 – Класс MyApp

Также в приложении имеется страница просмотра и изменения заметки, однако в силу специфичной передачи параметров в нее переход к данной странице осуществляется посредством указания конкретного виджета с передаваемыми ему параметрами.

Код отрисовки данных в приложении однотипен, для упрощения работы используется библиотека MaterialUI которая поставляется вместе с фреймворком Flutter и каждую страницу отдельно мы не будем здесь рассматривать. Коротко говоря – MaterialUI используется практически во всех стандартных приложениях операционной системы Android и ее использование позволяет приложению не выделятся на фоне системных приложений что позволяет пользователю думать, что он использует приложение, предоставленное самой системой. В листинге 3.6 представлена часть кода страницы **«**KeepDetailScreen**»** которая служит своеобразной формой для создания заметок , полный код можно увидеть в приложении А.

Form(  
 key: \_formKey,  
 child: Column(  
 children: <Widget>[  
 TextFormField(  
 controller: \_nameController,  
 decoration: InputDecoration(  
 border: InputBorder.none,  
 contentPadding: EdgeInsets.symmetric(vertical: 0.0, horizontal: 0.0),  
 ),  
 style: TextStyle(  
 fontSize: 24,  
 fontWeight: FontWeight.w700,  
 color: Colors.black,  
 ),  
 ),  
 TextFormField(  
 controller: \_contentController,  
 maxLines: null,  
 decoration: InputDecoration(  
 border: InputBorder.none,  
 contentPadding: EdgeInsets.symmetric(vertical: 0.0, horizontal: 0.0),  
 ),  
 style: TextStyle(  
 fontSize: 18,  
 fontWeight: FontWeight.w400,  
 color: Colors.black87,  
 ),  
 ),

Листинг 3.5 – Фрагмент страницы KeepDetailScreen

**3.1.3 Разработка авторизации**

Было принято решение посвятить авторизации целый раздел.

Авторизация – одна из ключевых составляющих любого приложения которое имеет доступ к интернету. В данном случае она выступает гарантом того, что пользователь отправляет данные на удалённый сервер без возможности их получения другим пользователем. Достигается это путём загрузки в приложение из удалённой базы данных только тех данных, которые были созданы пользователем, который их запрашивает. При получении данных из Firestore приложение отправляет логин пользователя сервису, а тот в свою очередь возвращает только те данные, которые помечены данным логином, таким образом на устройстве не оказывается данных, созданных другим пользователем. Обеспечение авторизации ложится на AuthService, часть реализации которого представлена в листинге 3.6 ниже.

class AuthService {  
 final FirebaseAuth \_auth;  
  
 AuthService({FirebaseAuth? auth})  
 : \_auth = auth ?? FirebaseAuth.instance;  
  
 Future<User?> signInWithEmailAndPassword(String email, String password) async {  
 try {  
 final UserCredential result = await \_auth.signInWithEmailAndPassword(  
 email: email,  
 password: password,  
 );  
 return result.user;  
 } catch (e) {  
 print('Error signing in: $e');  
 rethrow;  
 }  
 }

Листинг 3.6 – Реализация авторизации в приложении

Также данных класс предоставляет возможность регистрации пользователя используя электронную почту и пароль с помощью метода registerUserWithEmailAndPassword, который продемонстрирован в листинге 3.7. Данный метод используется внутри страницы RegisterScreen. Он позволяет пользователю зарегистрироваться в системе, если он не имеет учётной записи.

Future<User?> registerWithEmailAndPassword(String email, String password) async {  
 try {  
 final UserCredential result = await \_auth.createUserWithEmailAndPassword(  
 email: email,  
 password: password,  
 );  
 return result.user;  
 } catch (e) {  
 print('Error registering: $e');  
 rethrow;  
 }  
}

Листинг 3.7 – Реализация регистрации нового пользователя

Еще одним методом относящимся к AuthService является getCurrentUser. Он позволяет приложению получать внутрь себя полную информацию о пользователе под которым осуществлен вход. Данная информация используется в перечисленных выше сервисах для запроса данных конкретного пользователя. Реализация метода показана в листинге 3.8 ниже.

Future<User?> getCurrentUser() async {  
 try {  
 final User? currentUser = \_auth.currentUser;  
 return currentUser;  
 } catch (e) {  
 print('Error getting current user: $e');  
 rethrow;  
 }  
}

Листинг 3.8 – Метод getCurrentUser

**3.2.Выводы по главе «Разработка приложения»**

В данной главе мы достаточно подробно исследовали процесс создания различных частей приложения начиная с реализации хранения данных в приложении и заканчивая сервисом авторизации и регистрации нового пользователя.

Базы данных были успешно созданы синхронизированы с помощью библиотеки sqflite и сервисов NetworkService и FirebaseService.

Пользовательский интерфейс был успешно разработан с помощью языка Dart, фреймворка Flutter и библиотеки MaterialUI, которая позволила нашему приложению не выделятся на фоне системных приложений.

Аутентификация и регистрация пользователя была произведена с помощью сервиса AuthService со стороны нашего приложения и сервиса Firebase Authentication со стороны Firebase.

В результате весь заранее спроектированный функционал был успешно реализован, и приложение готово к следующим этапам доработки. Этот этап разработки подчеркивает нашу способность создавать интегрированные и эффективные решения, сочетая в себе передовые технологии в области серверной и клиентской разработки.

Кроме того, были уделены усилия созданию дружественного и интуитивно понятного пользовательского интерфейса, что способствует приятному взаимодействию пользователя с приложением. Важным аспектом является также возможность быстрой сборки приложения под различные платформы благодаря фреймворку Flutter, что несомненно расширяет охват аудитории.

**4. Тестирование приложения**

Тестирование программного обеспечения играет ключевую роль в обеспечении высокого качества программных продуктов. Оно направлено на проверку соответствия фактических результатов работы программы ожидаемым. Главная цель тестирования - убедиться, что программа соответствует заявленным требованиям, выявить и исправить ошибки до того, как они окажутся замечены конечными пользователями.

Существующие методы тестирования, такие как модульное тестирование, интеграционное тестирование, системное тестирование и приемочное тестирование, несмотря на свою эффективность, не гарантируют полного устранения всех дефектов. Тем не менее, они предоставляют важные инструменты для выявления и исправления большинства проблем в процессе разработки.

Тестировщики создают тест-кейсы, которые включают в себя детальные описания, последовательность действий и ожидаемые результаты. Эти тест-кейсы позволяют систематически проверять каждый аспект программы, начиная от отдельных модулей и заканчивая полным функционалом. Отслеживание выявленных ошибок происходит при помощи баг-трекинговых систем, где каждый баг получает уникальный идентификатор, и его статус отслеживается до момента устранения.

Для отображения тестируемых элементов составлена таблица 4.1 тест-кейсов.

Таблица 4.1 – Описание функциональных тест-кейсов

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Статус |
| 1 | 2 | 3 |
| Вход в учетную запись | Перенаправление на домашнюю страницу c заметками | Успешно |
| Переход на страницу заметки | Перенаправление на страницу заметки | Успешно |
| Переход на страницу создания заметки | Перенаправление на страницу создания новой заметки | Успешно |
| Создание заметки | Заметка успешно создана | Успешно |
| Изменение существующей заметки | Заметка успешно изменена | Успешно |
| Удаление заметки | Заметка успешно удалена | Успешно |
| Выход из учетной звписи | Перенаправление на старницу логина и удаление информации о предыдущем пользователе | Успешно |

# **5. Руководство пользователя**

В данном разделе рассмотрим, как и взаимодействовать с мобильным клиентом.

**5.1.Мобильное приложение**

Вход в мобильное приложение происходит с помощью адреса электронной почты и пароля. Для входа необходимо иметь электронную почту и пароль для аккаунта, под которым вы хотите войти. Если таковые имеются – их нужно ввести в соответствующие поля. Экран входа продемонстрирован на рисунке 5.1.

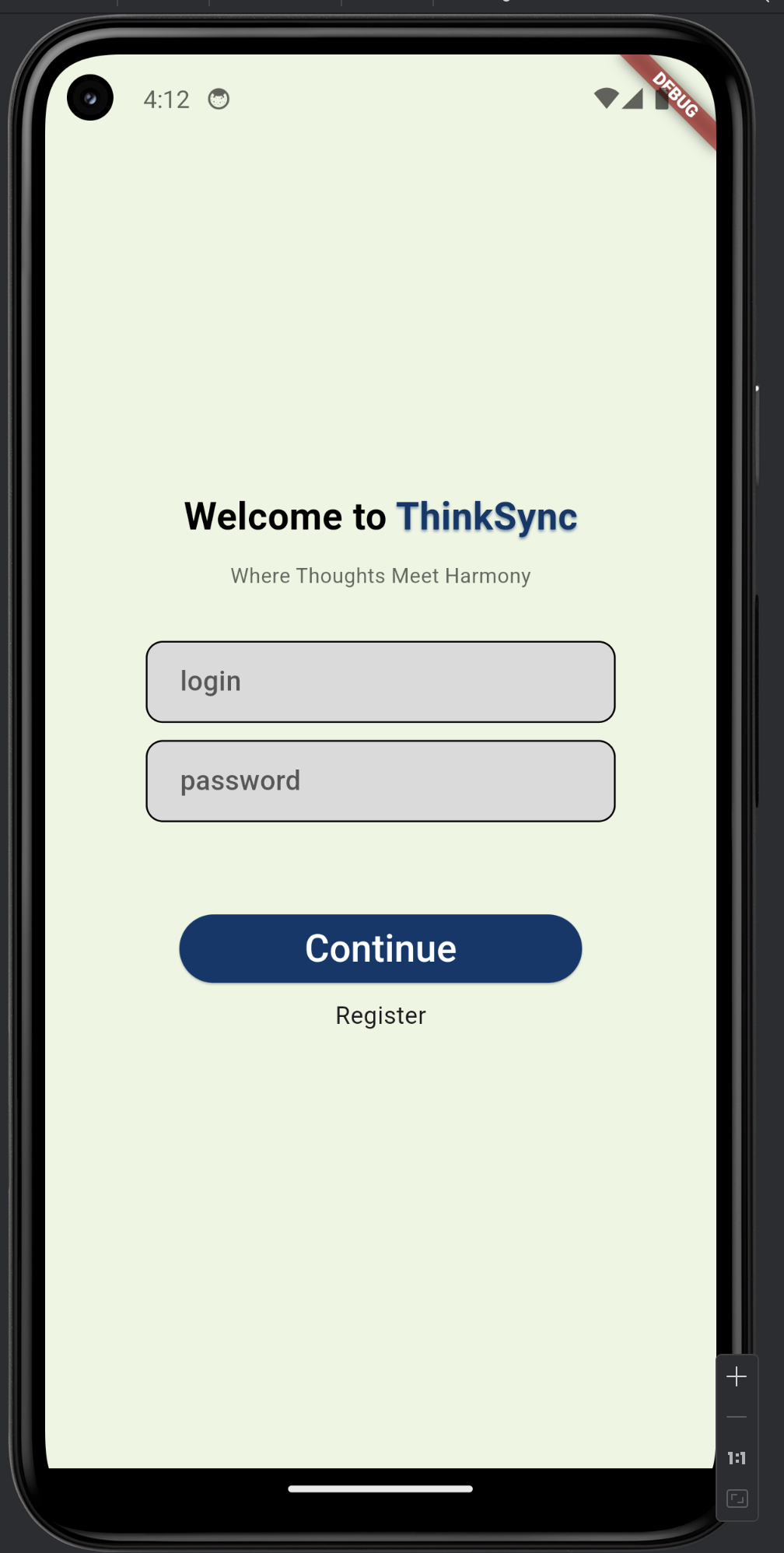


Рисунок 5.2 – Страница аутентификации

При успешной авторизации вас перенаправит на главную страницу. На данной странице вы можете увидеть все заметки созданные с аккаунта под которым вы вошли а также при нажатии на левую правую иконку пользователя вы можете вернуться на страницу аутентификации. Экран входа продемонстрирован на рисунке 5.2.

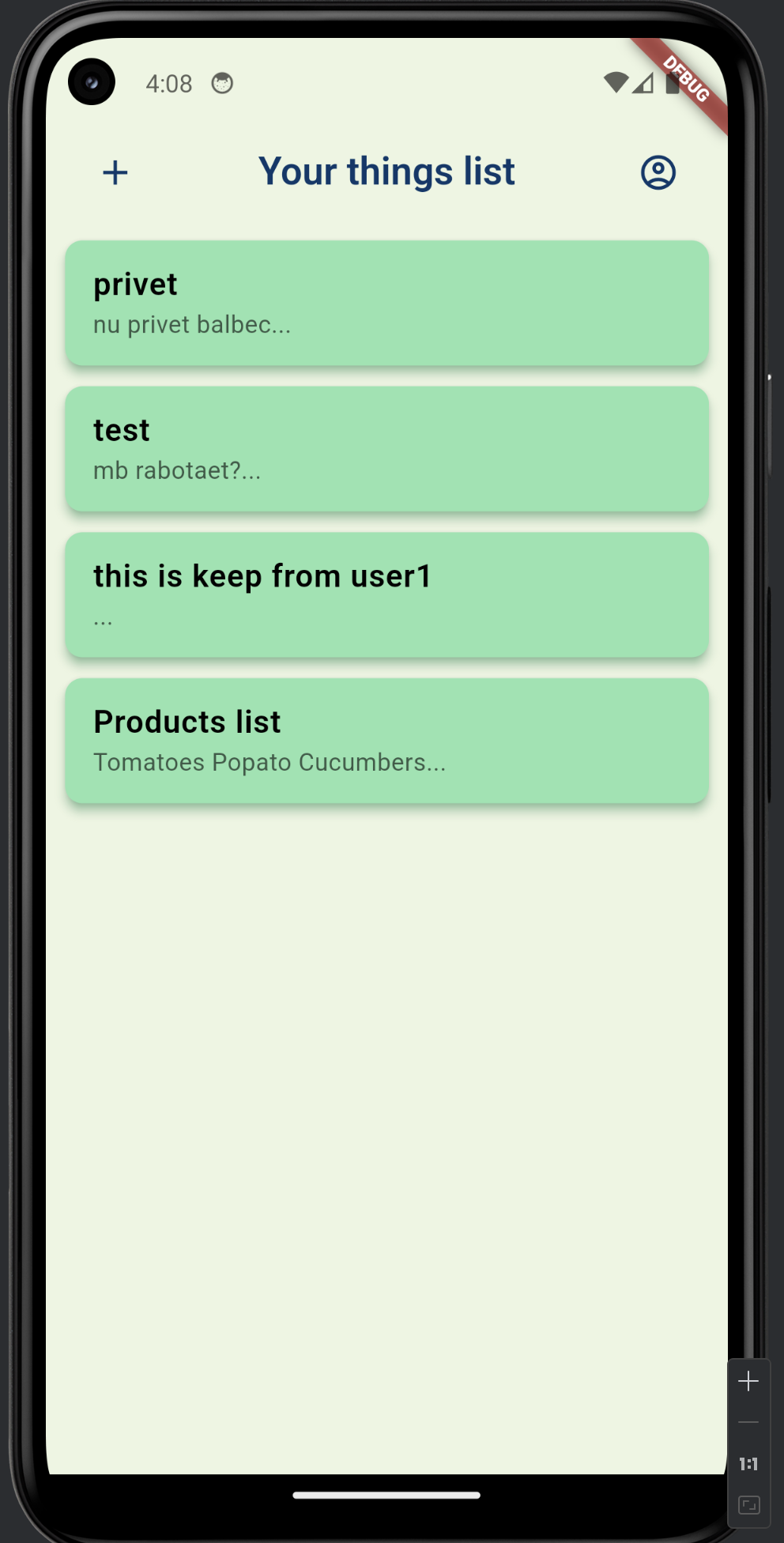


Рисунок 5.2 – Главная страница приложения

Если вы хотите создать новую заметку – вам необходимо нажать на иконку добавления в левом верхнем углу экрана. После нажатия вас направит на страницу создания новой заметки. Здесь вы можете нажать на любую часть верхней половины экрана для вызова клавиатуру и немедленного набора текста заметки. Для сохранения заметки нужно нажать на кнопку в виде стрелки также в левом верхнем углу экрана. Экран входа продемонстрирован на рисунке 5.3.

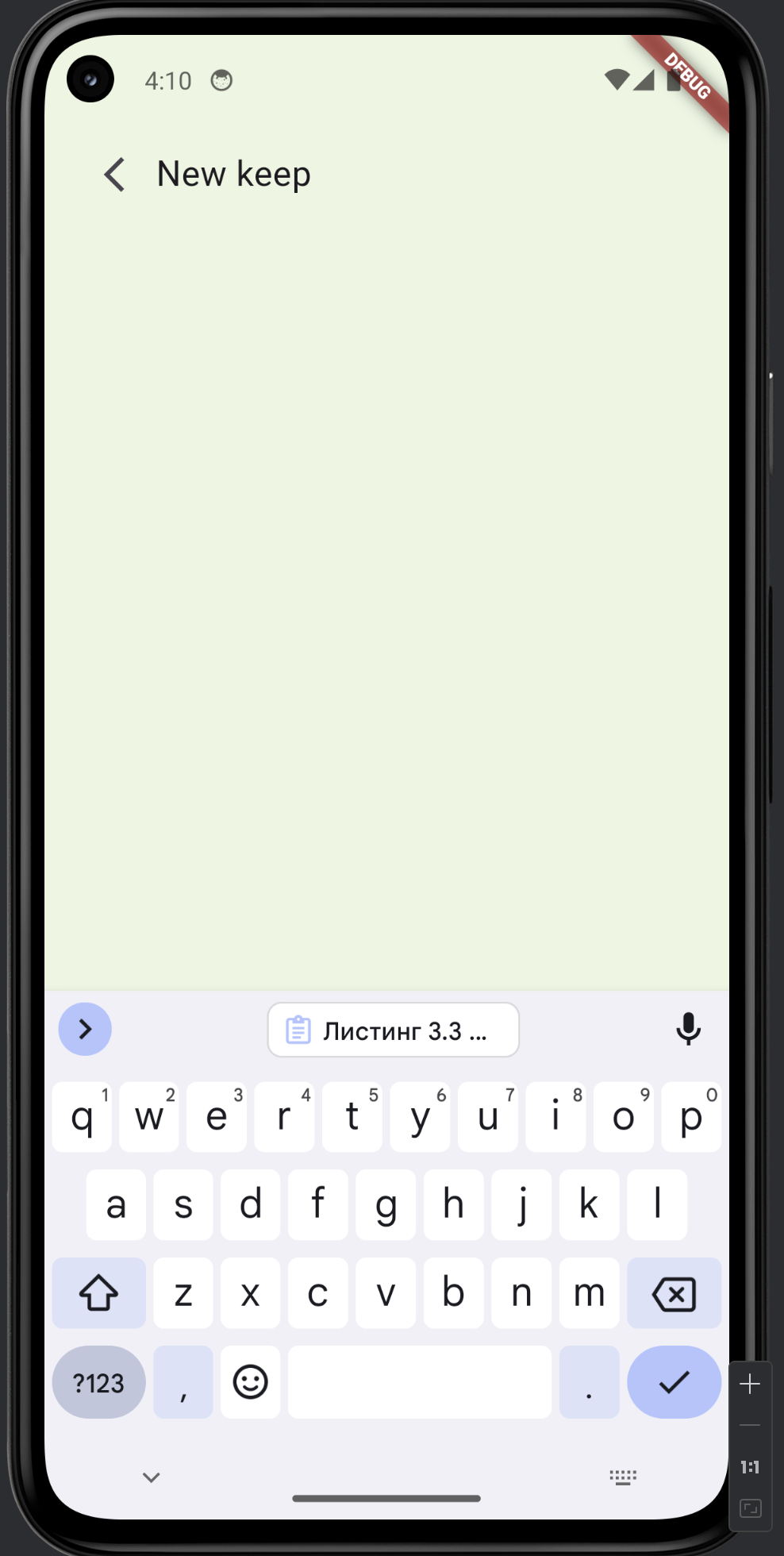


Рисунок 5.3 – Создание новой заметки

Для просмотра полной версии нужной вам уже созданной заметки или ее изменения на главной странице нажмите на нужную заметку. После ее открытия выберите поле которое необходимо изменить и введите в него новые данные. Для сохранения достаточно нажать на кнопу перехода на предыдущий экран. Экран входа продемонстрирован на рисунке 5.4.

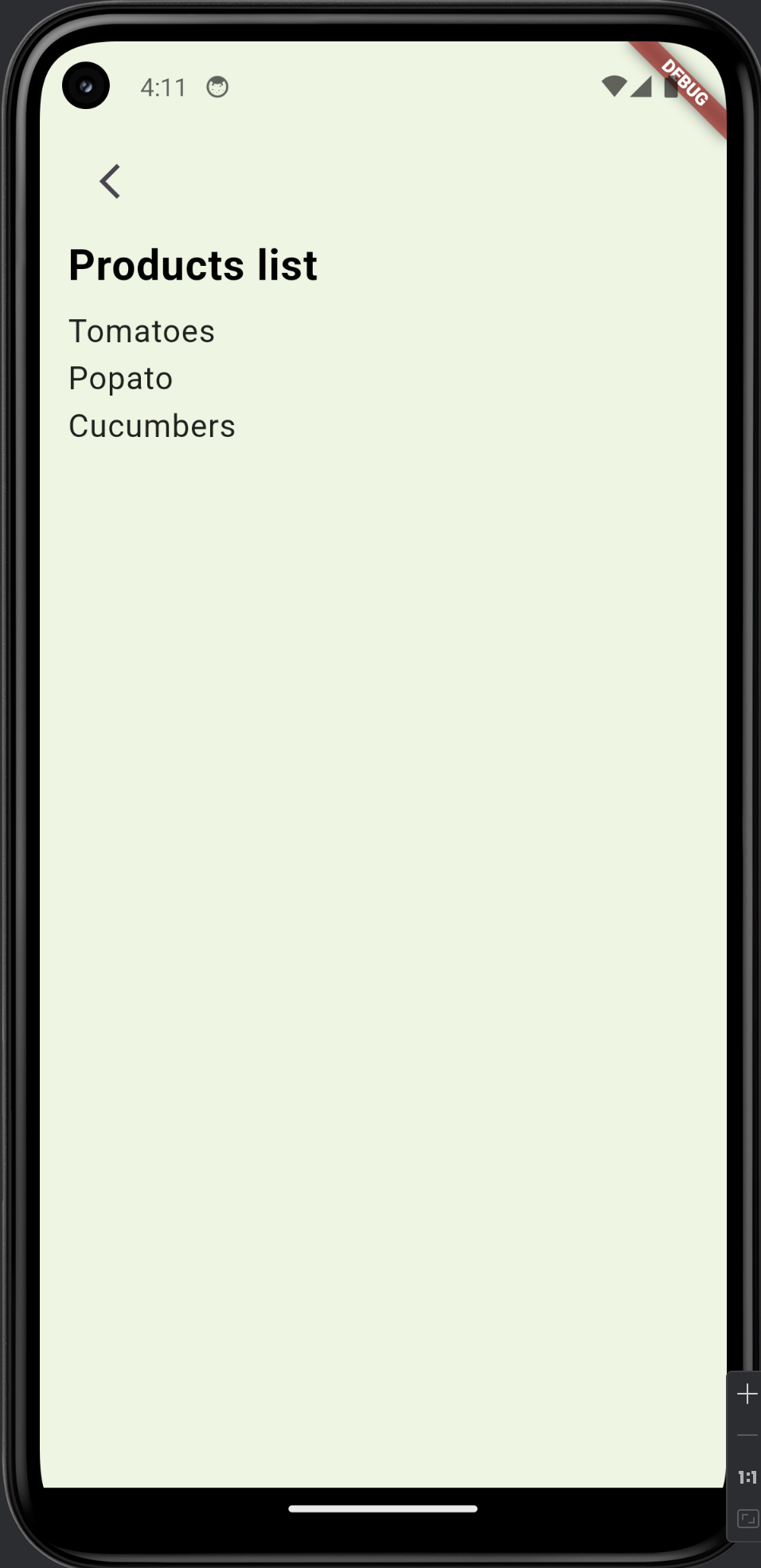


Рисунок 5.4 – Страница существующей заметки

**5.2. Выводы по главе «Руководство пользователя»**

В данной главе представлено подробное описание функционала мобильного приложений, предназначенного для пользователей. Важными аспектами являются удобство и авторизация пользователя при помощи адреса электронной почты и пароля. Также следует отметить простой и понятный интерфейс, выполненный в приятных глазу цветах.

Мобильное приложение обеспечивает пользователям простой вход и навигацию, предоставляя информацию о созданных пользователем заметках. Отдельное внимание уделено процессу загрузки данных, что повышает удобство использования.

# **Заключение**

В ходе выполнения курсового проекта было приложение для проведения создания заметок.

Перед началом разработки был произведен анализ и сравнительный обзор аналогов приложений подобной тематики, и определение функциональных возможностей разрабатываемого приложения.

В результате написания курсового проекта, было разработано приложение на основе таких технологий как Flutter, MaterialUI и Firebase с использованием языка программирования: Dart.

При разработке были выполнены следующие функциональные требования:

* создание заметок;
* редактирование заметок;
* удаление заметок;
* хранение заметок в локальной базе данных;
* синхронизация заметок через облачное хранилище;
* аутентификация пользователя.

Кроме того, в процессе разработки уделено внимание аспектам безопасности приложения. Механизм авторизации и регистрации пользователей реализован с учетом современных стандартов безопасности данных, обеспечивая защиту личной информации пользователей.

Интеграция с базой данных заметок подразумевает эффективное выполнение запросов, что способствует быстрому доступу к информации и обеспечивает плавное взаимодействие с приложением.

Также важным аспектом является возможность расширения функционала приложения в будущем. Архитектура приложения разработана с учетом возможности добавления новых функций и модулей, что обеспечивает его долгосрочную устойчивость и развитие.

Наконец, важно отметить, что разработанное приложение не только соответствует техническим требованиям проекта, но и ориентировано на удовлетворение потребностей конечных пользователей, обеспечивая им удобный и эффективный опыт в использовании приложения для проведения олимпиад по базам данных.

**Список использованных источников**

**1.Flutter Documentation [Электронный ресурс]. – Режим доступа: https://docs.flutter.dev/. – Дата доступа: 05.10.2023.**

**2. Firebase Documentation [Электронный ресурс]. - Режим доступа:** [**https://firebase.google.com/docs**](https://firebase.google.com/docs) **- Дата доступа : 01.03.2024**

**3. Билиотеки для Flutter [Электронный ресурс]. - Режим доступа:** [**https://pub.dev/**](https://pub.dev/) **- Дата доступа : 04.03.2024**

**ПРИЛОЖЕНИЕ А. Код страницы «KeepDetailScreen»**