Министерство образования и науки РФ  
Ростовский Государственный Экономический Университет (РИНХ)  
Факультет Компьютерных Технологий и Информационной   
Безопасности

Индивидуальная работа по дисциплине

«Инженерное программирование»

На тему “Мобильное приложение для отчета успеваемости студентов”

Выполняли студенты 1-го курса,   
группы ПРИ-311:

Дробот Илья и Бородинов Иван.

Ростов-на-Дону 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ……………………………………………………………...……..3-4

1. Актуальность тематики исследования……………………………….5
2. Обоснование выбора стека программирования (включая ЯП)…….6-9
3. Описание теории……………………………………………………10-21
4. Детальность алгоритма решения (блок-схема)…………………...21-22
5. Календарное планирование этапов разработки (trello доска)…….…22
6. Демонстрация работоспособности программного решения….……..23

ЗАКЛЮЧЕНИЕ………………………………………………………………….23

СПИСОК ЛИТЕРАТУРЫ………….………………………………………...23-24

ВВЕДЕНИЕ

Сегодня мы живем в век технологий, которые не стоят на месте. Система образования развивается и вместе с ней подлежат развитию и разработки для поддержания курса точного и современного обучения студентов.

Современные технологии прогрессировали до такой степени, что благодаря их помощи мы можем позволить себе осуществлять свою учебную деятельность не только в школах/вузах, но и за пределами учебных заведений (домашнее обучение, дистанционное обучение); также они значительно облегчают жизнь и работу преподавателей, имея надёжную опору в качестве «маленького помощника» в учебных хлопотах.

Такие сайты, приложения и прочие вспомогательные ресурсы для вузов уже являются неотъемлемой частью учебной жизни, которые могут размещаться даже в наших смартфонах, которые у всех под рукой и каждый имеет доступ к нему и находящимся приложениям в нем.

На данный момент времени создано не так много сайтов и прикладных компьютерных программ в помощь представителям учебной стези и это огромное поле для исследований, нужно дальше развиваться в этой сфере, чтобы процесс обучения не был заторможен в силу возможных возникновений проблем с технологической стороны.

По многочисленным жалобам от преподавателей нашего университета на уже имеющуюся систему управления обучения, мы с нашим преподавателем-наставником пришли к выводу, что необходимо создать более удобное приложение, которое не будет иметь таких существенных недостатков, как, например, платформа Moodle. Одна из нерешённых задач этой платформы, приносящая неимоверный объём мороки, – отсутствие пластичной таблицы оценивания студентов по баллам. Создавая её вручную, преподаватели тратят подавляющее количество своего времени на энергоёмкий процесс, но без этого обходиться ещё сложнее. Так было положено начало разработки нашего нового мобильного приложения для отчёта успеваемости студентов.

Из-за колоссального объёма функционала данное приложение ещё находится в стадии активной доработки некоторых моментов (которые будут указаны в дальнейшем), но оно имеет очень большие и многообещающие перспективы на будущее. Разработка этого приложения будет продолжена и после её презентации преподавателю. В дальнейшем в наш план включён выход обновлений с новыми функциями, системой уведомлений, добавлением полноценной таблицы, улучшением дизайна, выгрузкой домашних работ и множеством других мелких функций, которые будут с каждым разом модернизировать данный продукт и в конечном счёте выйдет полноценное и удобное приложение, обеспечивающее и студентов, и преподавателей всеми необходимыми ресурсами для благополучной и продуктивной учебной работы.

1. Актуальность тематики исследования.

Наш всем известный сайт do.rsue.ru, построенный на платформе Moodle, имеет недостатки в качестве отсутствия таблицы Excel для выставления баллов и функции оповещения преподавателей о выполненном домашнем задании, что является огромной и нерешённой проблемой на сегодняшний день. В нашем университете представленное нами приложение поспособствовало бы упрощению и сортировки процесса оценивания знаний студентов, что является актуальным на сегодняшний день, так как электронный ресурс в наше время – самое удобное изобретение, в котором можно быстро и доступно размещать всё что угодно, в том числе и информацию об учёбе/работе.

В связи с текущими событиями и повесткой дня, импортозамещение является одним из самых востребованных родов деятельности, поэтому отечественные разработки находятся в огромном приоритете.

1. Обоснование выбора стека программирования (включая ЯП).

Изображение выглядит как текст, монитор, телефон, мобильный телефон

Автоматически созданное описаниеМы выбрали фреймворк Flutter, который использует язык программирования Dart. Выбрав Flutter, нам не придется писать отдельно на языке Kotlin, а затем отдельно на языке Swift, так как фреймворк и ЯП позволяет писать кроссплатформенные приложения. Flutter имеет быстрое развитие и обладает многими преимуществами перед другими фреймворками. Один из них: Hot Reload - фича, позволяющая моментально переносить все изменения из кода на запущенный эмулятор или подключенное устройство.

Графический рисунок 1 – фича Hot Reload  
Источник - [Flutter — новый взгляд на кроссплатформенную разработку / Хабр (habr.com)](https://habr.com/ru/company/google/blog/426701/?hl=ru_RU&fl=ru,en&)

Для тестирования и написания кода мы отдали своё предпочтение такой интегрированной среде разработки (IDE), как Android Studio. Данная среда разработки имеет в себе встроенный эмулятор, как и для Android (для Windows, Linux), так и для IOS (MacOS). Но всё же, в самом начале, во время установки эмулятора и фреймворка Flutter мы столкнулись с рядом проблем. Докачивание библиотек (SDK) для работы с эмулятором происходило, но установка не была завершена. Нам постоянно выводило ошибку, связанную с драйвером HAXM.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Ошибка установки драйвера HAXM

Мы изрядно старались понять нашу ошибку в течение долгого времени, но затем всё-таки нашли истинную причину проблемы: её фикс происходит с помощью меню биоса и в нем нужно включить такую настройку как Virtualization Technology.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Virtualization Technology в меню BIOS

После того, как мы это сделали, эмулятор запустился, но появилась другая проблема. На компьютере у одного студента-разработчика приложения не работал интернет внутри эмулятора, когда на компьютере другого студента всё работало исправно. Мы очень долго пытались решить данную проблему и выяснили, что в самом эмуляторе нужно вручную настроить APN для подключения к интернету. Нужно найти раздел APN (он находится в разделе Mobile Data). Создать новый APN и настроить его, согласно настройкам вашей сети.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 4 – Меню настроек APN

Не лишним будет выключить Брандмауэр в вашей операционной системе. Он может блокировать трафик из эмулятора.   
После правильной настройки интернет начал работать исправно.

Для базы данных мы использовали Google Firebase. С одной стороны, она намного проще MongoDB. У нее более упрощенная система авторизации пользователей. Встроенная шифровка паролей позволяет тратить меньше времени на кодинг данной функции.  
Firebase имеет внутри себя две разные базы данных: Firestore Database и Realtime Database. Мы используем первую, так как она легче в понимании и более “подточена” под разработку мобильных приложений.

Но также она имеет и свои минусы. Один из них: новизна данной БД. Очень мало информации в документации.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Документация по Правилам в БД Firestore

Источник - [Security Rules language  |  Firebase Documentation (google.com)](https://firebase.google.com/docs/rules/rules-language?hl=en#firestore)

Один из главных минусов – использование языка Common Expression Language (CEL), который построен на командах match и allow. Соответственно, привязка JSON файлов не представляется возможной.

1. Описание теории

Повторимся, что наше приложение использует фреймворк Flutter, язык программирования Dart и базу данных Firebase. Всё это принадлежит корпорации Google.   
Перед началом разработки мы составили план для оптимизации рабочего процесса. Нашей отправной точкой была концепция кроссплатформенного приложения. Первой целью было познакомиться с фреймворком и языком программирования. Было просмотрено внушительное количество материалов, курсов и гайдов, чтобы углубиться в синтаксис и возможности языка. Далее, мы продумали и записали основную структуру приложения, поставили свои первые цели для реализации основных функций приложения.

Первым делом, мы подключили базу данных Firestore в наш проект. Нужно перейти на их главный сайт firebase.google.com и зарегистрироваться. Это можно сделать с помощью своего Google аккаунта. Нажимаем на кнопку “Get Started” после чего мы попадаем в меню разработчика, где мы можем создать проект.

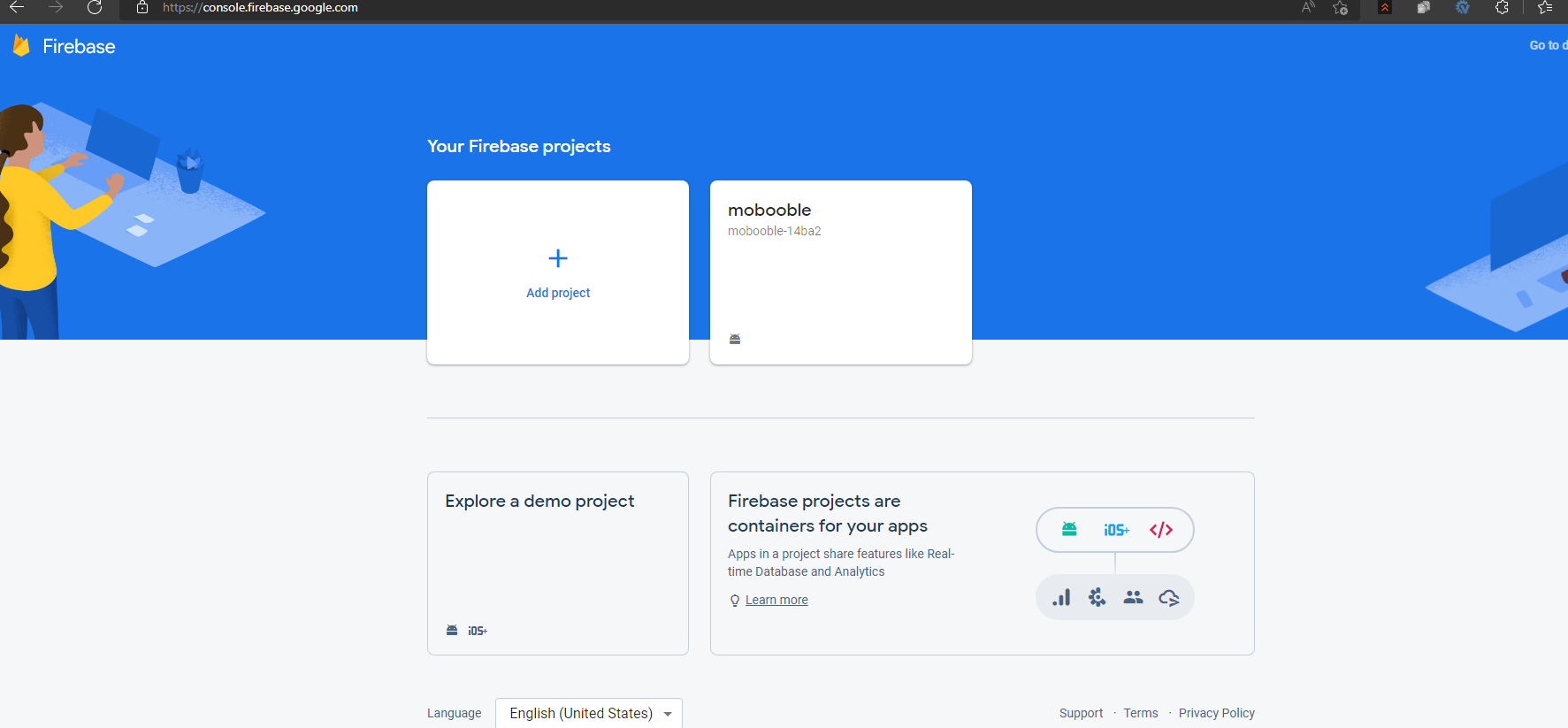


Рисунок 6 – Консоль меню разработчика в Firebase

Источник - [Firebase console (google.com)](https://console.firebase.google.com/)

Вводим название проекта, Выбираем, использовать ли аналитику от Google (мы от него отказались в силу ранней стадии разработки приложения). Нам нужно прописать взаимодействие базы данных с нашим приложением. Для этого нужно нажать иконку Android.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Создание проекта в Firebase

Источник - [Firebase console (google.com)](https://console.firebase.google.com/)

В первой строке нам нужно указать Package name. Его мы можем получить из папки Android/app/src/profile/AndroidManifest.xml

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – Название Package name AndroidManifest.xml

Копируем туда название проекта в кавычках. Во второй строке можно указать опциональное название приложения. Нажимаем на Register App.

Следующее, что нам нужно сделать, это скачать json файл. Его нужно добавить в папку android/app/

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – Скачивание файла google-services.json

Источник - [Firebase console (google.com)](https://console.firebase.google.com/)

Далее нам нужно подключить библиотеки от Firebase.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 – Подключение библиотек в проект build.gradle

Источник - [Firebase console (google.com)](https://console.firebase.google.com/)

В файлы build.gradle нам нужно добавить строчку в dependencies

classpath 'com.google.gms:google-services:4.3.10'

Для ios подключение базы данных происходит аналогичным способом.

Далее, для создание базы данных, мы переходим в Firestore Database, указываем локацию сервера бд и нажимаем Next.

Теперь, нам нужно подключить библиотеки базы данных в другой файл в проекте. Он находится на пути Android/pubspec.yaml

В строчку dependencies мы прописываем следующее:

firebase\_core:  
firebase\_auth: ^0.15.5+2  
cloud\_firestore: ^0.13.4  
cloud\_firestore\_web:  
firebase\_core\_web:  
firebase\_auth\_web:

Все эти библиотеки служат для работы с БД в самом коде приложения. Ядро БД, авторизация и сам firestore.

Стоит помнить, что приложение было задумано для конкретного университета. Регистрация пользователей происходит вручную админом в БД (в меню authentication) путём привязки почты и созданием пароля. В приложении лишь доступна только авторизация.

Итак, первый шаг выполнен. Теперь непосредственно к коду.

Чтобы код был более читаемым, мы использовали принципы ООП.

Первым делом, мы имеем единственный файл main.dart. Он является отправной точкой в запуске приложения.

Самое первое, что нужно подключить, это библиотека material.dart.

import 'package:flutter/material.dart';

Она необходима, для подключения виджетов. Очень важно упомянуть, что Flutter построен на системе «виджет в виджете». То-есть, когда мы разрабатываем что-либо, мы определяем структуру экранов(виджетов).

Widget build(BuildContext context) {  
 return SafeArea(  
 child: Scaffold(  
 appBar: AppBar(  
 title: Text('текст'),  
 ),  
 body: Container(  
 child: Center(  
 child: Text('центрированный текст'),  
 ),  
 )  
 ),  
 );  
}

В данном примере мы используем целых семь виджетов, все они вместе наглядно показывают структуру самого фреймворка.

Схематично это выглядит так:

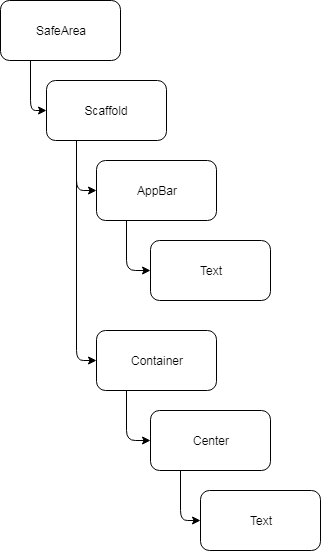


Рисунок 11 – блок-схема для пояснения принципа виджетов

Вернёмся к нашему коду.

void main() => runApp(MoboobleApp());

Данная функция служит для запуска приложения на устройстве. Она вызывает класс MoboobleApp(). Данный класс имеет в себе параметры цветовой темы приложения и он вызывает метод home: который ссылается на класс LandingPage()

home: LandingPage()

Для класса LandingPage() был создан отдельный dart файл (screens/landing.dart). В этом файле возвращается состояние пользователя isLoggedIn (авторизован или не авторизован). Была создана страница с авторизацией (screens/auth.dart), в которой вводится логин и пароль зарегистрированного пользователя. Если пароль введён неверно, то всплывает ошибка:

Fluttertoast.*showToast*(  
 msg: "Не получается войти. Пожалуйста, проверьте логин или пароль!",

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12 – Использование библиотеки FlutterToast

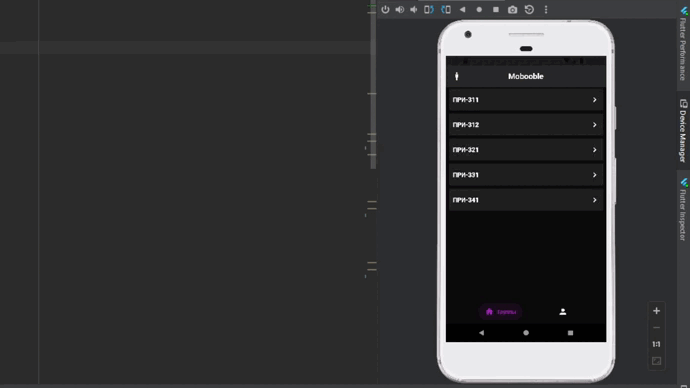
Для вывода ошибок мы использовали библиотеку FlutterToast.

Если же пароль верный, то пользователь получает свой уникальный userid, который не удаляется из памяти при его закрытии и пользователя кидает на HomePage(). При повторном входе в приложение проверка isLoggedIn выполняется снова но юзеру не нужно снова авторизовываться, так как он сделал это ранее.

Теперь мы более развёрнуто расскажем о HomePage() (screens/home.dart). Этот файл является связующим звеном между «искусственными страницами».

Почему же они «искусственные»? Дело в том, что класс HomePage() является стейтфулл виджетом.

class HomePage extends StatefulWidget

это значит, что в данном классе мы можем менять “состояние” страницы, вызывая различные классы со своими собственными функциями (например, список групп или профиль пользователя). То-есть создается иллюзия того, что при нажатии кнопки “профиль” меняется страница, но меняется лишь состояние самого HomePage(). 

Графический рисунок 13 – изменение состояния страницы

В самом Flutter конечно же есть такой класс как BottomNavigationBar, который отвечает за навигацию в нижней части экрана. Но мы нашли отличную библиотеку от одного из пользователей, которая называется salomon\_bottom\_bar. Два его главных преимущества: первая – красивая анимация с переходом, вторая – перейти на эту библиотеку не составило труда, так как семантика классического BottomNavigationBar`а сохраняется полностью идентичной.

Стоит немного рассказать о том, как же меняется то самое состояние с помощью кнопок. В salomon\_bottom\_bar есть свойство onTap которое меняет переменную \_currentIndex = i в значениях от 0 до 1 (где 0 – это Groupslist(), а где 1 – это Profile())

body: \_currentIndex == 0 ? Groupslist() : Profile() , //проверка на "состояние" и вызов соответствующего метода  
bottomNavigationBar: SalomonBottomBar(  
currentIndex: \_currentIndex,  
onTap: (i) => setState(() => \_currentIndex = i), //при нажатии состояние меняется от 0 до 1

Переменная \_currentIndex инициализирована заранее:

var \_currentIndex = 0;

Метод Groupslist() (components/groups.dart)   
Этот dart файл отвечает за вывод групп. Но в настоящий момент мы пока создали только заглушки. Объяснять эту часть особо не имеет смысла, но вкратце мы просто создали массив групп со свойством title и оформили их в «контейнер-карточку» списком.

return Card(  
 elevation: 2.0,  
 margin: EdgeInsets.symmetric(horizontal: 8, vertical:4),  
 child: Container(  
 decoration: BoxDecoration(color: Color.fromRGBO(30, 30, 30, 1)),  
 child: ListTile(  
 contentPadding: EdgeInsets.symmetric(horizontal: 10),  
 title: Text(groups[i].title, style: TextStyle(color: Theme.*of*(context).textTheme.subtitle1.color, fontWeight: FontWeight.*bold*)),  
 trailing: Icon(Icons.*keyboard\_arrow\_right*, color: Theme.*of*(context).textTheme.subtitle1.color),  
 )  
 )  
); //карточка групп(ы)

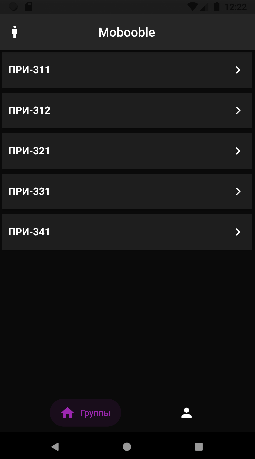


Рисунок 14 – метод Groupslist()

В данном файле по закомментированной части кода видно, что мы уже начали разрабатывать систему добавления групп(включая студентов в ней, добавление к ним баллов). К сожалению, она еще не готова, так как на данный момент нам не хватило времени и немного знаний. Мы собираемся довести эту часть до рабочего состояния в ближайшее время. Сейчас это наш самый главный приоритет.

Метод Profile() (components/profile.dart)   
Этот dart файл отвечает за информацию о профиле и имеет в себе кнопку выхода из аккаунта. Парсинг информации на данный момент не реализован. Кнопка выхода реализована стандартным для Flutter ElevatedButton. Он имеет в себе свойство onPressed, которое вызывает функцию AuthService().logOut() выхода из аккаунта.

onPressed: () => AuthService().logOut(),

Изображение выглядит как текст, монитор, электроника, снимок экрана

Автоматически созданное описание

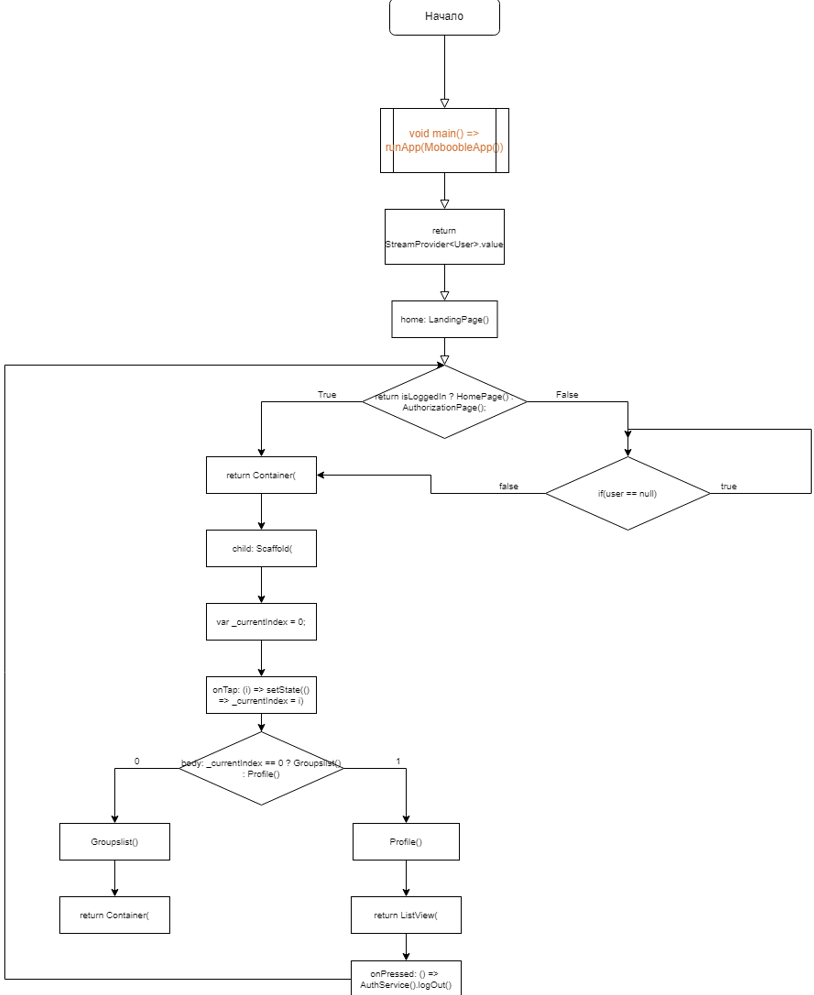
Рисунок 15 – Метод Profile()

Вернёмся к странице входа (screens/auth.dart)

Документация Flutter + Firebase (раздел аутентификации) достаточно обширна и довольно сильно помогла нам при реализации данного проекта. Почти весь функционал данной страницы был написан с помощью той же документации.  
Для начала следует запомнить, что Dart — однопоточный а Flutter использует Dart. Этот язык исполняет одновременно одну и только одну инструкцию. Последовательно. Это значит, что выполнение одной инструкции не может быть прервано другой инструкцией. В свою очередь, Flutter позволяет обойти эту проблему путём асинхронных методов.   
Экземпляр Future представляет собой задачу, которая выполняется асинхронно и завершается (успешно или с ошибкой) когда-то в будущем.

1. Детальность алгоритма решения (блок-схема)

Файл блок-схемы находится в репозитории. И тут:



1. Календарное планирование этапов разработки (trello доска)

<https://trello.com/invite/b/FushJGzo/c8daaab449876d221afe8b8e1b0ad7e4/разработка>

1. Демонстрация работоспособности программного решения

[salituder/Mobooble (github.com)](https://github.com/salituder/Mobooble)

Демонстрация работоспособности полностью показана в описании теории.

ЗАКЛЮЧЕНИЕ

В заключение хочется сказать, что на самом деле это не конец проекта. Как уже говорилось, у него много перспектив и есть над чем поработать. Много над чем. Flutter нам очень интересен, хоть и до сих пор вызывает постоянные трудности в его изучении и освоении. Новизна фреймворка очень сильно даёт о себе знать. Стоит отметить, что мы писали не на самой новой версии флаттера, а откатывали её назад. Причина в том, что в новой версии большая часть курсов и гайдов, которые мы смотрели, просто неактуальна. Но несмотря на это, мы будем развиваться дальше.

СПИСОК ЛИТЕРАТУРЫ

1. Flutter — новый взгляд на кроссплатформенную разработку <https://habr.com/ru/company/google/blog/426701/?hl=ru_RU&fl=ru,en&>
2. Flutter. Асинхронность и параллельность <https://habr.com/ru/post/497278/?>
3. Widget Catalog <https://docs.flutter.dev/development/ui/widgets>
4. Уроки Flutter и Dart с нуля / #1 – Разработка мобильных приложений... <https://www.youtube.com/watch?v=YPRaA6KhyXc&list=PL0lO_mIqDDFWqpFH_ZxzfgDqsK9LHNxRY>
5. Add Firebase to your Flutter app  |  Firebase Documentation <https://firebase.google.com/docs/flutter/setup?platform=ios>
6. Security Rules language  |  Firebase Documentation <https://firebase.google.com/docs/rules/rules-language?hl=en#firestore>
7. Full Flutter Tutorial For Beginners: 2021 <https://www.youtube.com/watch?v=b-NS_5za44c>
8. Dart Programming in 4 hours | Full beginners tutorial <https://www.youtube.com/watch?v=5xlVP04905w>
9. Flutter Tutorial for Beginners <https://youtube.com/playlist?list=PL4cUxeGkcC9jLYyp2Aoh6hcWuxFDX6PBJ>
10. Flutter official <https://www.youtube.com/channel/UCwXdFgeE9KYzlDdR7TG9cMw>
11. Flutter уроки для начинающих на русском <https://youtube.com/playlist?list=PLyaYkfwvXhRKjYAIO4_J_IcHtAXUR_1ci>
12. Flutter Forum <https://flutterforum.co/>