Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет им Н.И. Лобачевского»

Институт информационных технологий, математики и механики

Отчёт по проектной работе

Приложение под android HomeAssistant

Выполнили:

студентки ИИТММ, гр. 381706-1

Безручко И.Ю.

Кукушкина К.О.

Максимова И.И.

Проверил:

преподаватель каф. МОСТ, ИИТММ

Лебедев И.Г.

Нижний Новгород

2019 г.

Оглавление

[Оглавление 2](#_Toc9505074)

[Введение 4](#_Toc9505075)

[Постановка задачи 5](#_Toc9505076)

[Исследование рынка 6](#_Toc9505077)

[Руководство пользователя 8](#_Toc9505078)

[Руководство программиста 11](#_Toc9505079)

[Описание структуры программы 11](#_Toc9505080)

[Описание структур данных 12](#_Toc9505081)

[Описание алгоритмов. 17](#_Toc9505082)

[Заключение. 18](#_Toc9505083)

[Список литературы 19](#_Toc9505084)

[Приложения 20](#_Toc9505085)

[Event.java 20](#_Toc9505086)

[Group.java 20](#_Toc9505087)

[Message.java 21](#_Toc9505088)

[MyChat.java 22](#_Toc9505089)

[Purchase.java 23](#_Toc9505090)

[User.java 23](#_Toc9505091)

[Accounts.java 25](#_Toc9505092)

[Budget.java 26](#_Toc9505093)

[Calendar.java 30](#_Toc9505094)

[Chat.java 31](#_Toc9505095)

[ChooseGroup.java 34](#_Toc9505096)

[Contacts.java 36](#_Toc9505097)

[DateEvents.java 36](#_Toc9505098)

[EditEvent.java 37](#_Toc9505099)

[Groups.java 38](#_Toc9505100)

[LogOn.java 40](#_Toc9505101)

[MainActivity.java 43](#_Toc9505102)

[Money.java 46](#_Toc9505103)

[MyWishlist.java 47](#_Toc9505104)

[NewGroup.java 49](#_Toc9505105)

[OperationsHistory.java 51](#_Toc9505106)

[Profile.java 52](#_Toc9505107)

[Settings.java 54](#_Toc9505108)

[WelcomePage.java 56](#_Toc9505109)

# Введение

Когда ты – студент технического направления, ты можешь позволить себе сразу две прелести жизни: самостоятельно жить, потому что ты уже студент, и взаимопонимание со стороны современных устройств – потому что ты технарь. В своей работе наша команда представляет проект приложения «HomeAssistant», созданное как для студентов-технарей, так и для тех, кто хочет улучшить свою коммуникабельность в социальной жизни новшествами технологий 21 века.

Данное приложение создавалось под реальные задачи студентов, поэтому несет в себе ряд особенностей, о которых нет представления у крупных компаний, занимающихся разработкой приложений. Обзор таких приложений представлен в главе «Исследование рынка». Мы взяли пример с удачных вариантов и добавили свои новшества для приложений подобного типа.

Цель проекта: написать рабочую версию мобильного многопользовательского приложения.

Для достижения цели были поставлены следующие задачи:

1. Проанализировать рынок мобильных приложений, найти похожие и выявить их преимущества и недостатки.
2. Описать подробное техническое задание.
3. Разработать проект приложения.
4. Реализовать рабочий код.
5. Протестировать приложение на реальных устройствах.

Перед началом работы мы уже подобрали необходимые инструменты. То, почему мы выбрали именно платформу android, а язык программирования - Java можно узнать из таблиц исследований рынка в одноименной главе и описании руководства программиста. О выборе баз данных и прочих сторонних сервисов также можно прочитать из руководства программиста, описание алгоритмов. Те или иные методы и подходы мы выбирали исходя из советов, описанных в выбранной нами литературе. Так как полноценный курс android разработки в ВУЗе на нашем направлении не читается, то книг достаточно много.

Суть приложения зародилась на почве идеи совместного быта двух студенток, поэтому большинство примеров в работе будут рассмотрены именно на этом событии.

# Постановка задачи

Создать многопользовательское приложение под android с возможностью обмена сообщениями авторизованных пользователей, поддержкой календаря, групповых чатов, записи и хранения данных о бюджете и списке покупок. Более подробная информация о каждой из задач указана ниже.

**Данные о пользователе.**

* Реализовать возможность авторизации и входа в приложение по электронной и паролю.
* Требуется создать базу данных, в которой будет хранится идентификатор каждого пользователя, его данные для входа (почтовый адрес и пароль), личные данные (имя, дата рождения, телефон), и номер группы, в которой пользователь состоит.
* Должна быть реализована возможность изменять информацию о пользователе (кроме идентификатора), удалять данные (например, оставить дату рождения пустым).

**Доска бюджета.**

* Создать окно, отражающее пользовательский счет, который хранит в себе сумму, идентификатор пользователя, и название счета.
* Реализовать возможность хранить локально историю операций со счетом.

**Список покупок.**

* Создать список, хранящий в себе название продукта и его ориентировочную сумму.

**Сообщения.**

* Прописать один общий чат для пользователей, состоящих в одной группе. Текст сообщения, отправитель и id группы. должны храниться удаленно.

**Календарь.**

* Создать календарь, который позволяет добавлять мероприятия и привязывать к дате необходимые траты.

# Исследование рынка

Для начала работы необходимо решить вопрос о выборе платформы, которая должна отвечать минимуму требований:

1. Большое количество материалов и примеров для изучения работы выбранной платформы
2. Доступность IDE и компиляторов под языки программирования для данной системы

С выбором платформы нам помогло изучение крупных компаний, которые проводят аналитику мобильных устройств населения разных стран и в открытом доступе делятся своими исследованиями. Так, например, с помощью сайта StatCounter – Ирландского инструмента для обработки информации - можно понять, как в целом в мире распределяются мобильные операционные системы.

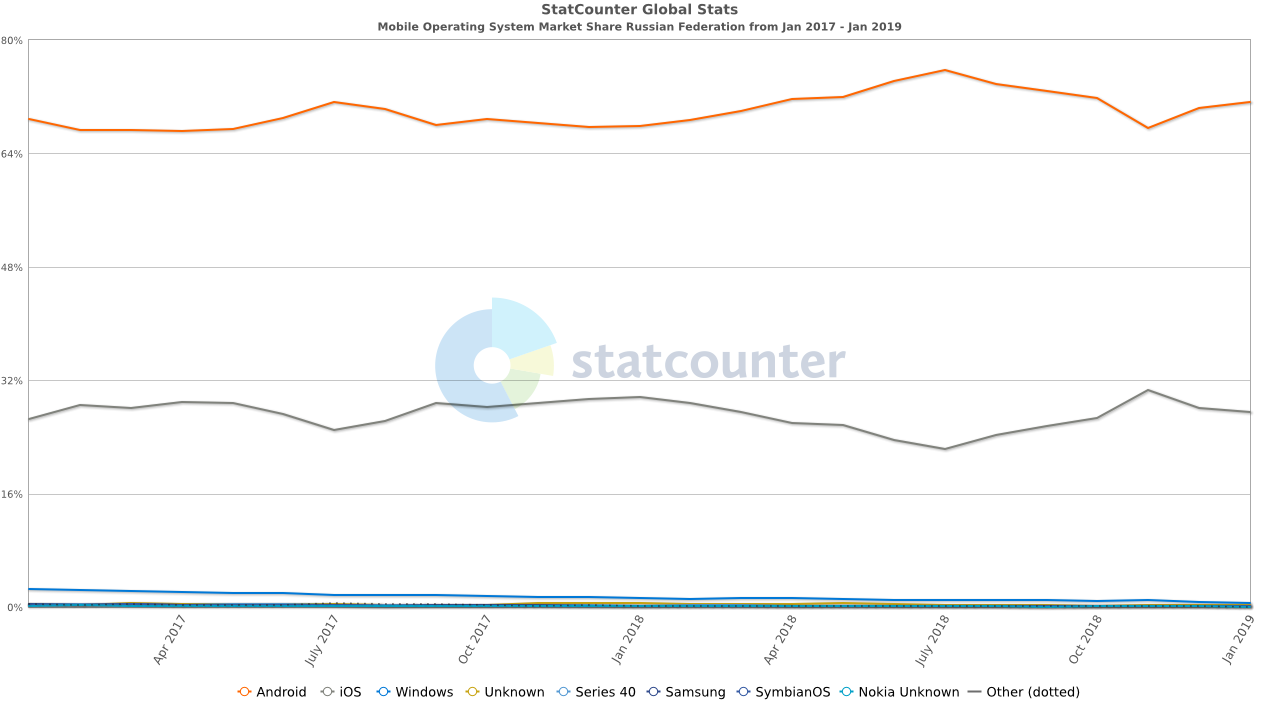


Рисунок 1. Исследование актуальности платформ

Самые популярные платформы в мире - ios и android. Выбор пал на android, так как количество смартфонов на этой платформе все же больше. Анализ доступности и удобства различных IDE помог убедиться в правильности выбора.

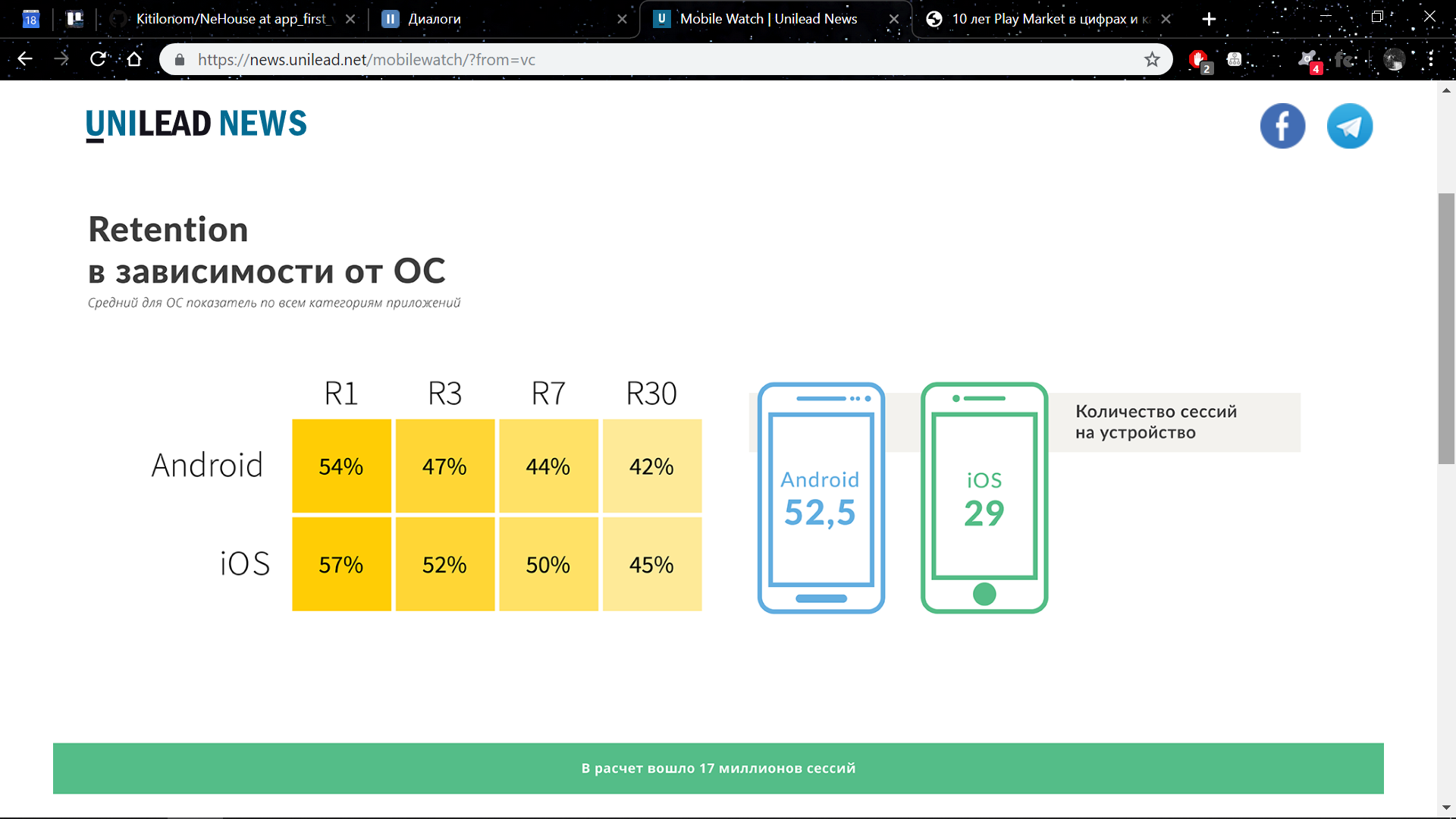


Рисунок 2. Сравнение популярности android и ios

Следующее, что предстояло изучить – приложения похожего типа, которые уже пользуются успехом на рынке. Мы рассмотрели аналоги на play market.

|  |  |  |
| --- | --- | --- |
| Название приложения | Поддерживаемые платформы | Описание |
| Дребеденьги  https://lh3.ggpht.com/tqjlIJ7Z7qogVCYpbXFeukved2dmRfs8Zfr3brpmG8pqidXPyijDEfa1RCWVL5UBe1I=s180 | * Android * Ios * Winphone * Web | * 549 рублей в год * Поддерживает разные счета * Поддерживает валюты * Интеграция с банком * Индивидуальные траты * Напоминания |
| Дзен-мани  https://is4-ssl.mzstatic.com/image/thumb/Purple124/v4/8a/78/10/8a781017-21bf-ea8e-3f60-a7d48d19c5cc/AppIcon-0-1x_U007emarketing-0-0-85-220-0-10.png/246x0w.jpg | * Android * Ios * Web | * 1249 рублей в год * Поддерживает разные счета * Поддерживает валюты * Интеграция с банком * Индивидуальные траты * Напоминания |
| CoinKeeper: учет расходовhttps://is4-ssl.mzstatic.com/image/thumb/Purple124/v4/ea/a7/a5/eaa7a5e1-525e-ae2d-902d-5bf613ec180d/AppIcons-0-1x_U007emarketing-0-85-220-0-6.png/246x0w.jpg | * Android * Ios * Web | * 250 рублей в месяц * Поддерживает разные счета * Поддерживает валюты * Интеграция с банком * Индивидуальные траты * Напоминания |
| Toshl Финансы - учет денегhttps://is5-ssl.mzstatic.com/image/thumb/Purple114/v4/3d/7d/06/3d7d0649-1bff-7ea1-4f65-4fe6d30ec89c/AppIcon-0-1x_U007emarketing-0-85-220-0-4.png/246x0w.jpg | * Android * Ios * Winphone * Web | * 19,9$ рублей в год * Поддерживает разные счета * Поддерживает валюты * Интеграция с банком * Индивидуальные траты * Напоминания |
| EasyFinance  https://is2-ssl.mzstatic.com/image/thumb/Purple118/v4/84/82/76/84827609-ad32-57f0-1ab4-841082df3427/mzl.lpditqpc.jpg/246x0w.jpg | * Android * Ios * Web | * 149 рублей в месяц * Поддерживает разные счета * Поддерживает валюты * Интеграция с банком * Индивидуальные траты * Напоминания |
| Alzex Finance  https://is4-ssl.mzstatic.com/image/thumb/Purple114/v4/b1/29/2b/b1292bce-09c2-8644-a81c-1f9ad2a6fe18/AppIcon-0-1x_U007emarketing-0-0-GLES2_U002c0-512MB-sRGB-0-0-0-85-220-0-0-0-10.png/246x0w.jpg | * Android * Ios * Web | * бесплатно * Поддерживает разные счета * Поддерживает валюты * Интеграция с банком * Индивидуальные траты * Напоминания |

*Таблица 1.Сравниение приложений*

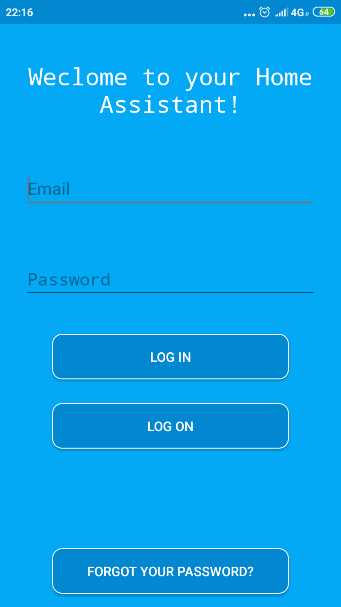
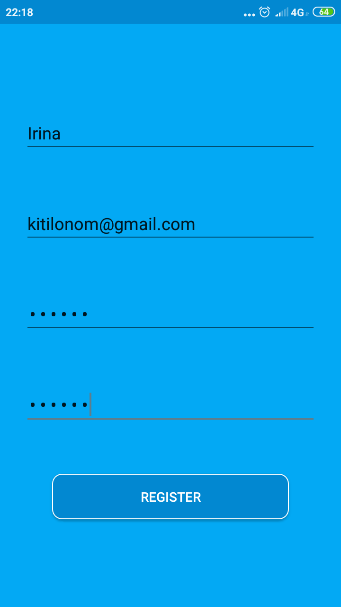
Таким образом мы отметили, что преимуществом многих программ является поддержка различных банковских систем, что на данном этапе сложно осуществимо и поддержка различных счетов, чем мы и воспользуемся для реализации своего приложения.

Так же стоит отметить главное отличие нашего приложения от тех, что уже есть на рынке – это сообщения с оповещением всех членов одной группы.

# Руководство пользователя

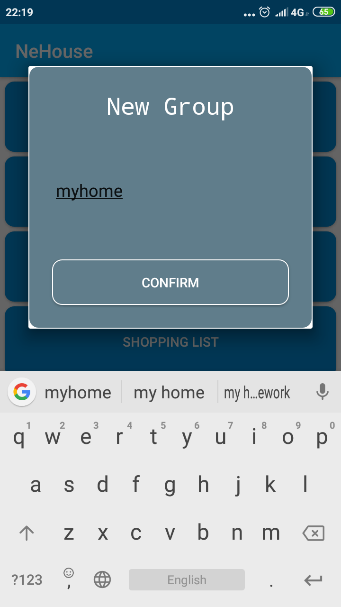
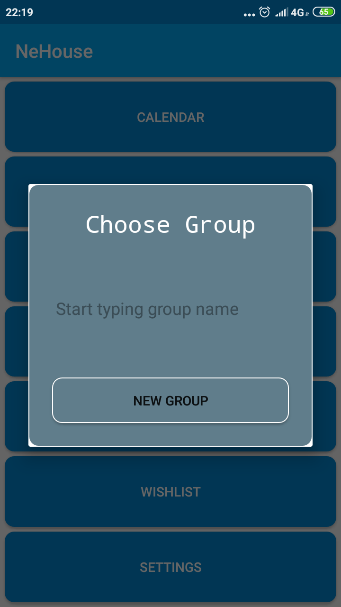
Для начала пользователю необходимо установить на телефон приложение. Для этого необходимо скопировать с компьютера HomeAssistant.apk, посредством кабеля usb. Установка производится без дополнительных действий, приложение сразу готово к работе.

При открытии приложение будет предлагать войти в личный аккаунт: ввести свои данные (почта и пароль) и нажать «log in». Конечно, если аккаунта у пользователя еще нет, он должен выбрать log on и зарегистрироваться (рис. 4).



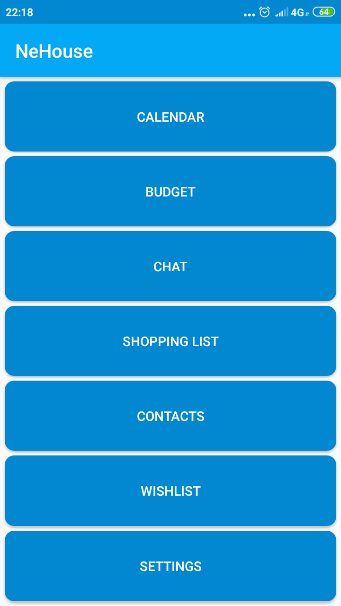
*Рисунок 3.Страница входа в аккаунт и страница регистрации.*

Если пользователь только зарегистрировался, то во время входа ему предлагается так же установить группу – выбрать из существующих или создать новую.

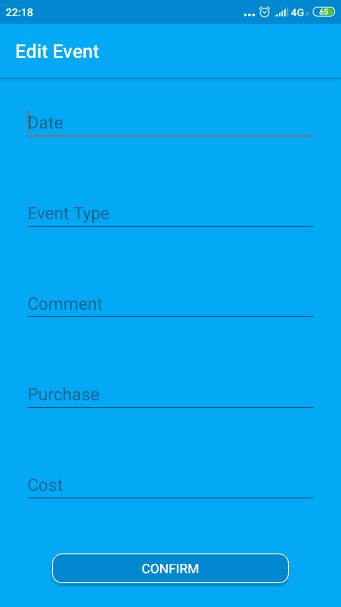
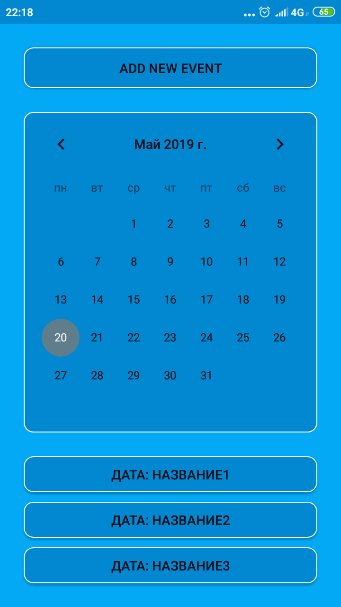
*Рисунок 5. Выбор группы.*

После выбора группы идентификация пользователя заканчивается, и он может использовать все функции приложения. Перед ним открывается главный экран, который содержит следующие разделы: календарь, бюджет, чат, список покупок, контакт, лист пожеланий, настройки.



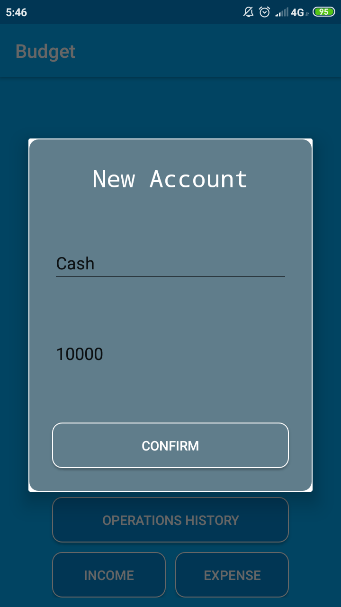
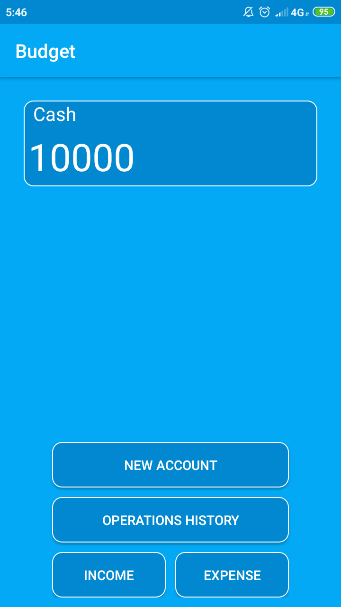
*Рисунок 6. Главное меню.*

При переходе на страницу календаря появляется небольшое табло с окном самого календаря, перечнем ближайших дат и мероприятий, кнопка добавления нового события. Для каждого события есть информация о его названии, дате, типе события, комментарии. Также предусмотрены запланированные траты: поле наименования покупок (purchase) и цены(cost).

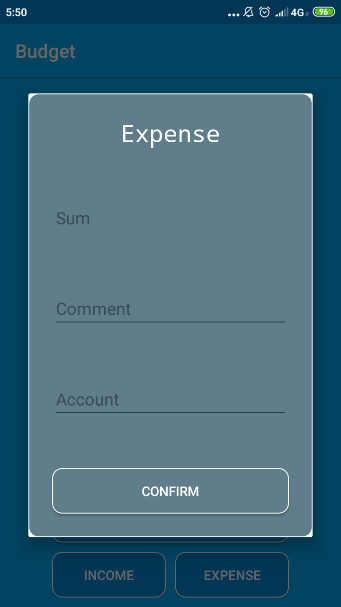
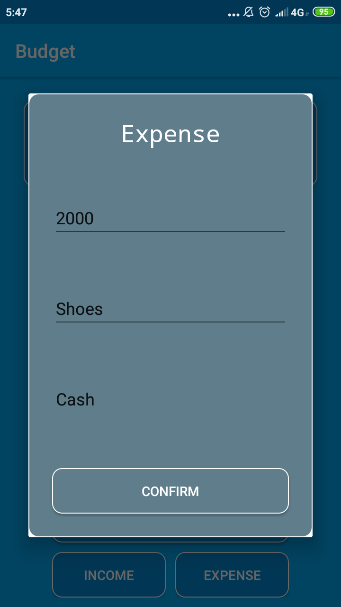
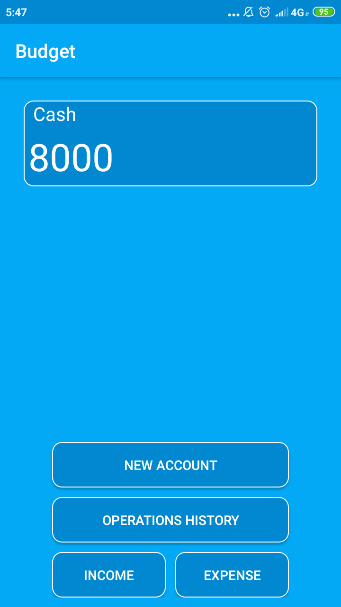
 

*Рисунок 7. Календарь.*

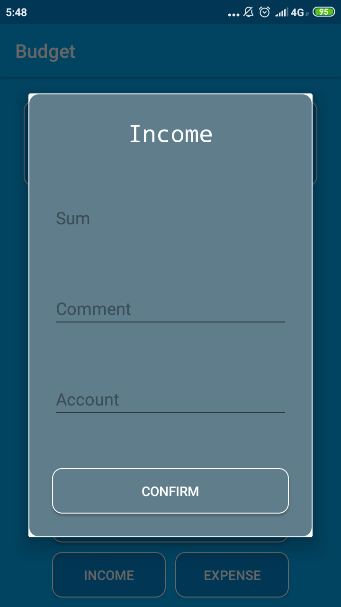
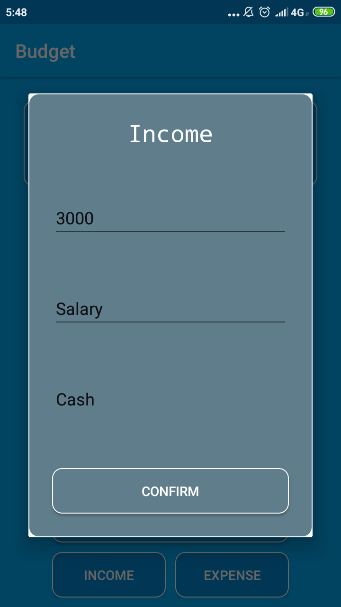
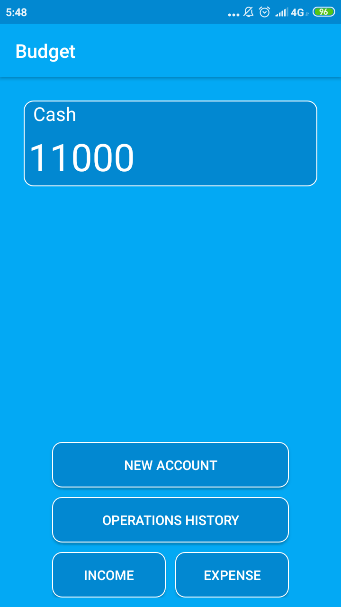
Следующий раздел бюджета подробнее показывает работу с денежными операциями. Экран бюджета представляет собой перечень доступных счетов, каждый из которых содержит в себе информацию о названии счета, сумме, плане расходов и доходов. Можно добавлять (рис. 8) новые счета, редактировать их данные: записать доход (рис. 10) и расход (рис. 9) с комментарием и информацией о владельце.

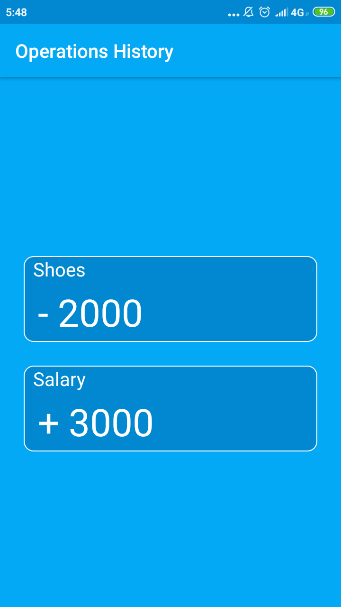
*Рисунок 8. Добавление нового счета.*

*Рисунок 9. Изменение счета. Указание расходов.*

*Рисунок 10. Изменение счета. Указание доходов*



*Рисунок 11. История операций с данным счетом.*

Основным преимуществом нашего приложения является чат. Чат представляет собой экран переписки между людьми, находящимися в одной группе.

# Руководство программиста

Код приложения можно найти в открытом доступе по адресу: <https://github.com/Kitilonom/NeHouse>

## Описание структуры программы

В корневой папке проекта находится файл *AndroidManifest.xml*. В нем объявлены все компоненты приложения c расширением .java.

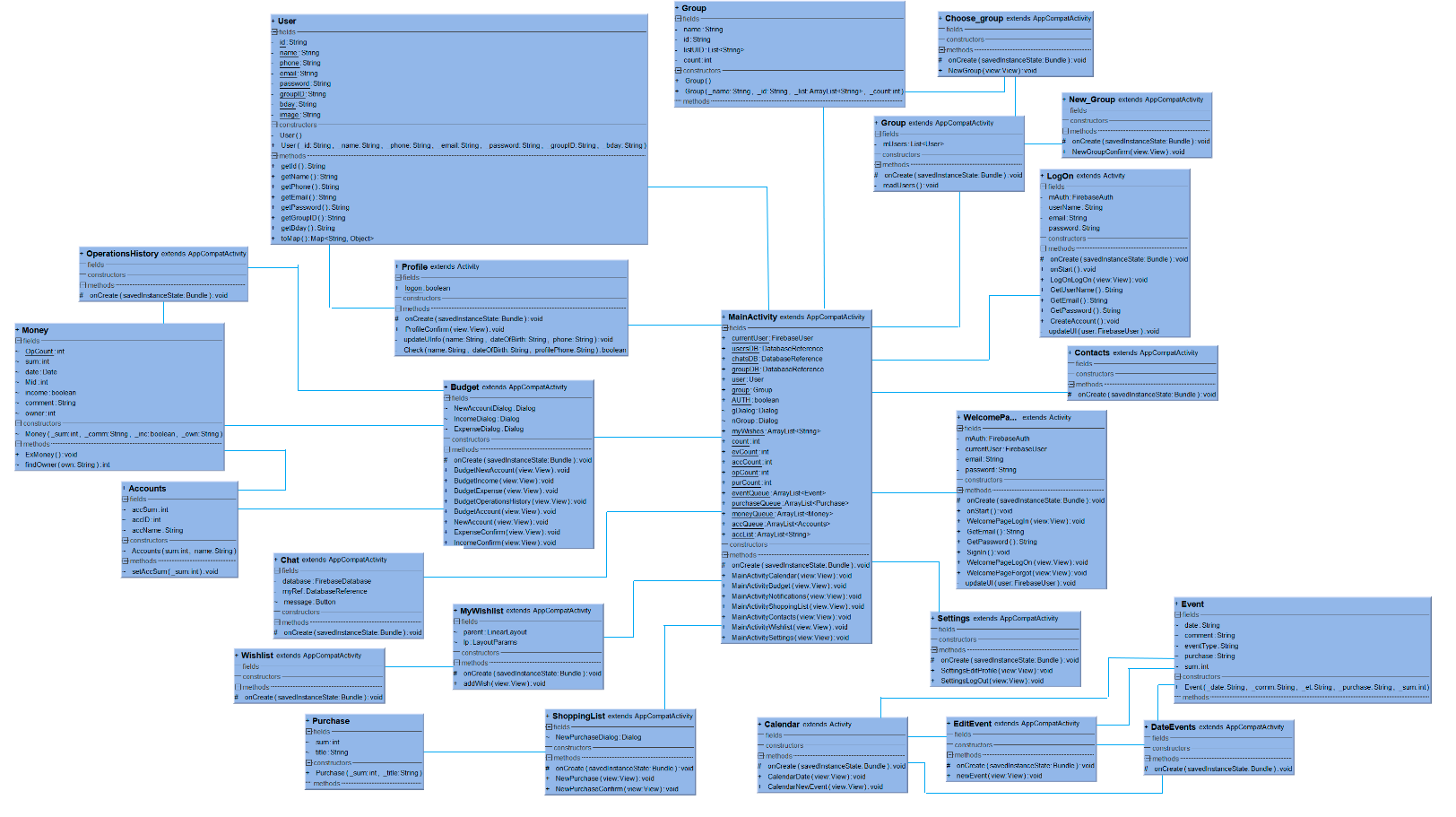
В каталоге *res* содержатся все ресурcы, используемые в коде приложения:

* xml-файлы(*res/layout*)
* значок, который обозначает приложение(*res/mipmap*)
* цветовая схема(*res/values/colors*)
* «константные» строки(*res/values/strings*)
* стили(*res/values/styles*)

В каталоге java находятся все файлы с расширением .*java*. Каждый файл описывает некий класс. Все классы можно разделить на два типа: описание структуры данных и описание пользовательского интерфейса. Классы, описывающие структуру данных, находятся в каталоге *java/Model*.

## Описание структур данных

Иерархию классов можно отследить на диаграмме ниже:



*Рисунок 13. Иерархия классов.*

Все классы можно разделить на два типа: классы, описывающие структуры данных и классы, описывающие пользовательский интерфейс.

Облачные услуги платформы Firebase

В проекте используются 2 сервиса платформы Firebase. Каждую из которых рассмотрим подробно.

**Firebase Realtime Database**

Firebase Realtime Database(FRD) - это облачная база данных. Данные хранятся в формате JSON и синхронизируются в режиме реального времени с каждым подключенным клиентом. Когда вы создаете кроссплатформенные приложения с помощью Firebase SDK все клиенты совместно используют один экземпляр базы данных в реальном времени и автоматически получают обновления с самыми новыми данными.

В случае приложения HomeAssistant использование этой базы данных обусловлено необходимостью хранить общие данные, такие как список участников группы или сообщения, а также данных, необходимых для входа в личные аккаунты пользователей.

Бесплатный тариф пользования услугами Firebase в рамках пакета Spark:

|  |  |
| --- | --- |
| Одновременные соединения | 100 |
| Объем хранения данных | 1 ГБ |
| Максимальное количество данных, доступное для скачивания | 10 ГБ/мес. |

*Таблица 2. Тарифный план «Spark» для БД*

В настоящий момент в рамках проекта в FRD хранятся структуры:

1. User

Данные о каждом пользователе хранятся в базе данных firebase под названием Users. Минимальные требования к пользовательским данным это наличие электронной почты (на одну почту может быть зарегистрирован только один пользователь), имя пользователя и id, сгенерированного при прохождении регистрации, что исключает повторяющиеся ключи и обеспечивает корректное создание ключа для каждого, чьи данные есть в базе. Реализована возможность менять данные пользователя с пользовательского экрана.

1. Messages

База данных Group содержит название группы – как ключ, который уникален и не может повторяться, и список id пользователей, состоящих в данной группе. Каждый пользователь может состоять лишь в одной группе.

1. Groups

Весе чаты сообществ реализованы в базе данных Messages. Объекты Messages – группы. Каждая группа хранит очередь сообщений, каждое из которых имеет уникальный номер, и несет в себе информацию о отправителе и тексте сообщения.

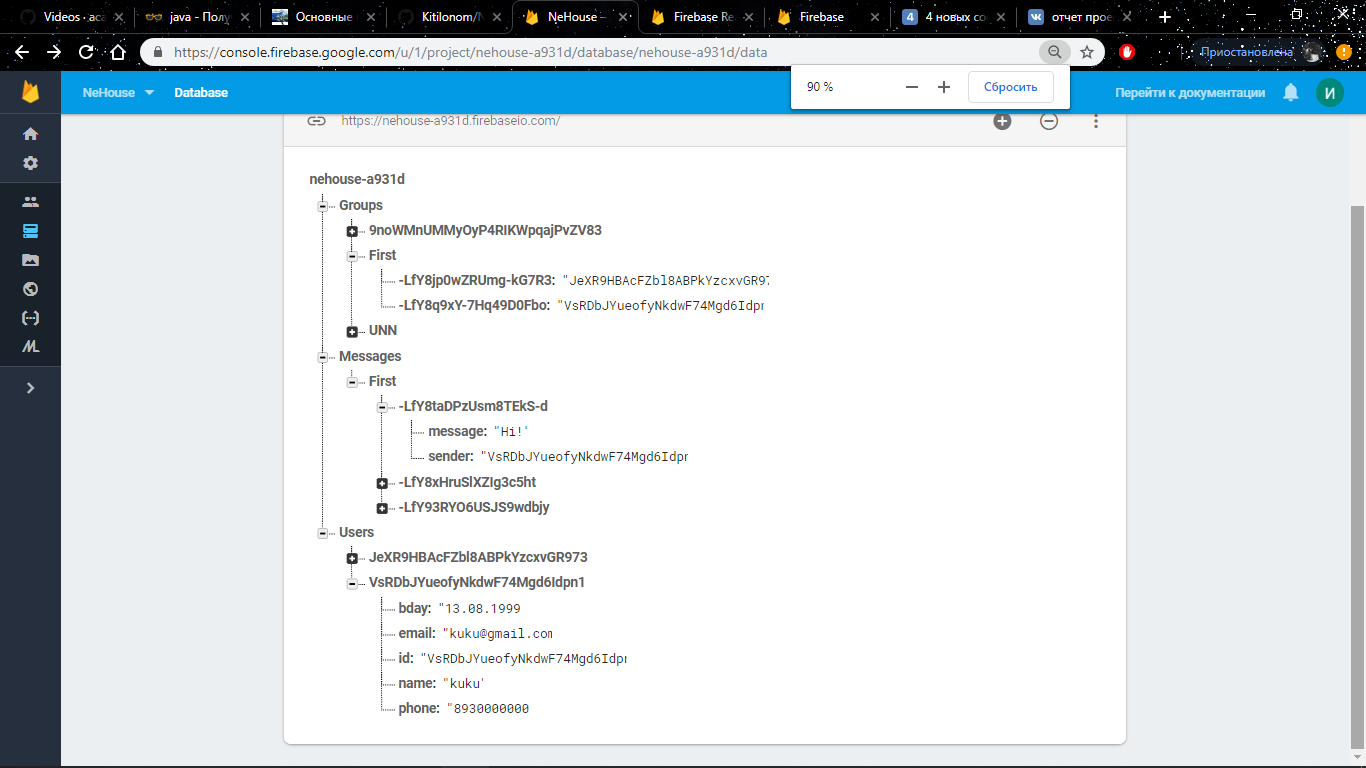


Рисунок 14. Данные в FRD.

**Firebase Authentication**

Поскольку наше приложение сдержит много слишком личных данных перед нами встала необходимость наиболее безопасно сохранять пользовательские данные в облаке. Для этого в работе была использована Firebase Authentication.

Аутентификация Firebase предоставляет бэкэнд-сервисы, простые в использовании SDK и готовые библиотеки пользовательского интерфейса для аутентификации пользователей. Она поддерживает аутентификацию с использованием пароля и электронной почты. Аутентификация Firebase тесно интегрируется с другими сервисами Firebase, что во многом облегчило работу с FRD, и использует отраслевые стандарты, такие как OAuth 2.0 и OpenID Connect.

Бесплатный тариф пользования услугами Firebase в рамках пакета Spark:

|  |  |
| --- | --- |
| Количество авторизованных пользователей | 10 тыс./мес. |

*Таблица 3. Тарифный план «Spark» для авторизации*

Список всех зарегистрированных пользователей и содержимое Баз данных доступно в личном аккаунте на сайте Firebase.

***Структуры данных***

***User***

Публичный класс пользователя. Хранит информацию о самом пользователе. Класс предназначен для интеграции с облачной базой данных.

Поля объявлены со спецификатором доступа private и модификатором static, и имеют тип String: id, name, phone, email, password, groupID, bday.

Класс содержит конструктор по умолчанию и конструктор копирования. Из публичных методов в классе реализованы только методы доступа к данным Get и метод toMap:

*public Map<String, Object> toMap()* – своего рода конструктор преобразования типов. Из текущего класса, создается объект класса Map.

***MyChat***

Публичный класс сообщения. Хранит информацию о отправителе и тексте сообщения. Класс предназначен для интеграции с облачной базой данных.

Поля объявлены со спецификатором доступа private и имеют тип String: sender, message.

Класс содержит конструктор по умолчанию. Из публичных методов в классе реализованы только методы доступа к данным Get и Set.

***Message***

Публичный класс чата. Хранит текст всех сообщений в чате. Класс предназначен для интеграции с облачной базой данных.

Поля:

* private ArrayList<String> mesList – содержит текст всех сообщений в чате.
* private String chat – название чата.

Класс содержит конструктор по умолчанию и конструктор копирования. Из публичных методов в классе реализованы только методы доступа к данным Get и Set.

***Group***

Публичный класс группы. Хранит id всех пользователей, состоящих в данной группе. Класс предназначен для интеграции с облачной базой данных.

Поля:

* private ArrayList<String> listUID – содержит id всех пользователей, состоящих в данной группе.
* private String name – название группы.

Класс содержит конструктор по умолчанию и конструктор копирования. Из публичных методов в классе реализованы только методы доступа к данным Get и Set, и метод toMap() (аналогично методу в классе User).

***Event***

Класс события. Предназначен для хранения и работы с событиями.

Содержит следующие поля: String date - дата, String eventType – тип события, String comm - комментарий, String purchase – необходимая покупка, int sum – предполагаемая цена.

***Purchase***

Класс покупок. Хранит название покупки (String name) и предполагаемую стоимость (int sum).

***Money***

Класс транзакций. Предназначен для сохранения истории операций пользователя.

Содержит поля int sum(сумма), String comment(комментарий), boolean inc (флажок доход/расход), int own (порядковый номер счета-влаельца).

***Account***

Класс счета. Хранит информацию о его состоянии.

Хранит имя счета (String name) и его состояние (int sum).

***Пользовательский интерфейс***

**Календарь**

Обеспечивает работу с событиями. Для реализации написаны такие классы как:

* Calendar, DateEvents – связь с xml-файлами(разметка экрана), создание событий
* Events – хранение данных

**Доска бюджета**

Обеспечивает работу со счетами. История операций доступна пользователю списком. Весь бюджет хранится локально и доступен лично каждому, без доступа третьим лицам. Классы:

* Budget, OperationsHistory – связь с xml-файлами, создание счета и денежных операций, просмотр истории операций
* Accounts, Money – хранение данных

**Чат**

Позволяет пользователям обмениваться сообщениями. Классы:

* Сhat – связь с xml

**Список покупок**

Создание списка необходимых приобретений с примерной стоимостью. Классы:

* ShoppingList – связь с xml, создание покупки, просмотр списка
* Purchase – хранение данных

**Контакты**

Просмотр списка участников группы, взаимодействие с ними. Классы:

* Contacts – связь с xml, просмотр списка
* Group – хранение данных

**Желания**

Создание списка желаний, доступного другим пользователям. Классы:

* MyWishes – связь с xml, создание пожеланий

**Настройки**

Просмотр id, изменение личных данных, выход из аккаунта

* Settings – связь с xml, работа с данными

**Авторизация**

Создание личного профиля пользователя, связь с firebase. Классы:

* WelcomePage, LogOn, LogIn – связь с xml, авторизация с использованием баз данных

**MainActivity** – связь с xml, хранение глобальных данных.

## Описание алгоритмов.

**NewEventConfirm** — создание нового события:

* + - Собрать введенную пользователем информацию из EditText (дата, тип события, комментарий, покупка и предлагаемая сумма)
    - Создать объект класса Event с заданными параметрами и добавить его в список событий
    - Создать объект класса Purchase с параметрами покупка и предполагаемая сумма и добавить его в список покупок

**NewAccountConfirm** — создание нового счета:

* Собрать введенную информацию из EditText (название, стартовая сумма)
* Создать объект типа Accounts с заданными параметрами
* Добавить объект в список счетов
* Добавить название элемента в список названий счетов

**ExpenseConfirm** — создание статьи расхода:

* Собрать введенную информацию из EditText (сумма, комментарий, счет-владелец)
* Создать объект типа Money с заданными параметрами
* Добавить объект в список операций
* Изменить состояние счета-владельца в соответствии с расходом

**IncomeConfirm** — создание статьи дохода:

* Собрать введенную информацию из EditText (сумма, комментарий, счет-владелец)
* Создать объект типа Money с заданными параметрами
* Добавить объект в список операций
* Изменить состояние счета-владельца в соответствии с доходом

**GetGroup** – подключение пользователя к группе:

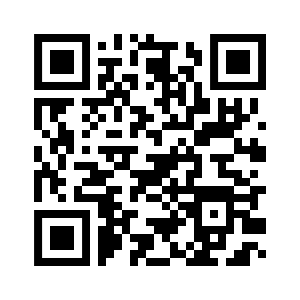
* Пользователь уже состоит в группе?
* Да – предлагаем выбор «Создать новую группу» или «Подключиться к уже существующей»
* Нет – значит пользователь уже состоит в какой –либо группе, и мы предоставляем ему доступ к соответствующему чату и списку «одногруппников».

# Заключение.

Мобильные приложения пользуются успехом во многих сферах, и оптимизация домашнего быта не исключение. В своей работе мы рассмотрели различные android-приложения на рынке, спроектировали свое приложение и создали его, используя язык программирования Java, подключив внешний сервис Firebase от google.

В итоге получилось рабочее приложение. Его удобно применять в повседневной жизни. Кроме того оно содержит множество возможностей для дальнейшего развития: развивать идею push уведомлений о покупках, обратная связь от членов группы о совершении некоторой покупки. Это может пригодиться не только для удобства пользователей, но и для монетизации ресурса, потому что информация о том, какие потребности возникают у пользователей и какие потребности они уже удовлетворили в зависимости от их местоположения и времени, представляет интерес для различных магазинов.

Ход и результаты нашей работы находятся в открытом доступе на ресурсе GitHub. С ними можно ознакомиться, отсканировав QR-код ниже.



# Список литературы

1. Голощапов, Алексей Google Android. Программирование для мобильных устройств / Алексей Голощапов. - М.: БХВ-Петербург, 2012. - 448 c.
2. Давыдов, Станислав IntelliJ IDEA. Профессиональное программирование на Java. Наиболее полное руководство (+ CD-ROM) / Станислав Давыдов , Алексей Ефимов. - М.: БХВ-Петербург, **2011**. - 800 c.
3. Дэрси, Лорен Android за 24 часа. Программирование приложений под операционную систему Google / Лорен Дэрси , Шейн Кондер. - М.: Рид Групп, 2011. - 464 c.
4. Камаев А.М., Сиднев А.А., Сысоев А.В. Об одном подходе к анализу эффективности приложений // Труды 50-й научной конференции МФТИ «Современные проблемы фундаментальных и прикладных наук»: Часть I. Радиотехника и кибернетика. - М.: МФТИ, 2007.
5. Касперски К. Техника оптимизации программ. Эффективное использование памяти. — СПб.: БХВ-Петербург, 2003. — 464 с.
6. Мозговой М. В. С++ Мастер-класс. 85 нетривиальных проектов, решений и задач. – Спбю: Наука и техника, 2007. – 272 с.
7. Нотон Java. Справочное руководство. Все, что необходимо для программирования на Java / Нотон, Патрик. - М.: Бином, **2015**. - 448 c.
8. Осипов, Дмитрий Delphi. Программирование для Windows, OS X, iOS и Android / Дмитрий Осипов. - М.: "БХВ-Петербург", 2014. - 464 c.
9. Роджерс, Рик Android. Разработка приложений / Рик Роджерс и др. - М.: ЭКОМ Паблишерз, 2010. - 400 c.
10. Седжевик, Роберт, Уэйн, Кевин. Алгоритмы на Java, 4-е издание.: Пер. с англ. – М.: ООО «И. Д. Вильямс», 2013. – 848 с.
11. Цехнер Марио Программирование игр под Android; Питер - Москва, 2012. - 688 c.
12. Statcounter: [сайт] <http://gs.statcounter.com/>
13. <https://github.com/Kitilonom/NeHouse/>

# Приложения

## ****Event.java****

package com.nehouse.nehouse.Model;

import com.nehouse.nehouse.MainActivity;

import java.util.Date;

public class Event {

public String date;

String comment;

public String eventType;

String purchase;

int sum;

public Event(String \_date, String \_comm, String \_et, String \_purchase, int \_sum) {

date = \_date;

comment = \_comm;

eventType = \_et;

purchase = \_purchase;

sum = \_sum;

MainActivity.evCount++;

MainActivity.eventQueue.add(this);

}

}

## ****Group.java****

package com.nehouse.nehouse.Model;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class Group {

private String name;

private ArrayList<String> listUID;

public Group() { }

public Group(String \_name, ArrayList<String> \_list, int \_count) {

name = name;

listUID = \_list;

}

public void setName(String \_name) {

name = \_name;

}

public String getName() {

return name;

}

public void setFriends(String uid) {

listUID.add(uid);

}

public Map<String, Object> toMap() {

HashMap<String, Object> res = new HashMap<>();

res.put("name", name);

res.put("list", listUID);

return res;

}

}

## ****Message.java****

package com.nehouse.nehouse.Model;

import java.util.ArrayList;

public class Message {

private ArrayList<String> mesList;

private String chat;

public Message() {

}

public String getChat() {

return chat;

}

public ArrayList<String> getMeslist() {

return mesList;

}

public Message(ArrayList<String> list, String \_chat){

mesList = list;

chat = \_chat;

}

public void setMessage(String mes){

mesList.add(mes);

}

}

## ****MyChat.java****

package com.nehouse.nehouse.Model;

public class MyChat {

private String sender;

private String message;

public MyChat(String sender, String message) {

this.sender = sender;

this.message = message;

}

public MyChat() {}

public String getSender() {

return sender;

}

public void setSender(String sender) {

this.sender = sender;

}

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

}

## ****Purchase.java****

package com.nehouse.nehouse.Model;

import com.nehouse.nehouse.MainActivity;

public class Purchase {

int sum;

String title;

public Purchase (int \_sum, String \_title) {

sum = \_sum;

title = \_title;

MainActivity.purchaseQueue.add(this);

MainActivity.purCount++;

}

}

## ****User.java****

package com.nehouse.nehouse.Model;

import com.google.firebase.database.Exclude;

import com.google.firebase.database.IgnoreExtraProperties;

import java.util.HashMap;

import java.util.Map;

@IgnoreExtraProperties

public class User {

private static String id;

private static String name;

private static String phone;

private static String email;

private static String groupID;

private static String bday;

public User() {

}

public User(String \_id, String \_name, String \_phone, String \_email, String \_groupID, String \_bday) {

id = \_id;

name = \_name;

phone = \_phone;

email = \_email;

groupID = \_groupID;

bday = \_bday;

}

public String getId() {

return id;

}

public String getName() {

return name;

}

public String getPhone() {

return phone;

}

public String getEmail() {

return email;

}

public String getGroupID() {

return groupID;

}

public String getBday() {

return bday;

}

@Exclude

public Map<String, Object> toMap() {

HashMap<String, Object> result = new HashMap<>();

result.put("id", id);

result.put("name", name);

result.put("email", email);

result.put("phone", phone);

result.put("groupID", groupID);

result.put("bday", bday);

return result;

}

public User(String \_email, String \_id) {

email= \_email;

id = \_id;

}

public void setGroupID(String key) {

groupID = key;

}

public void setId(String key) {

id = key;

}

public void setPhone(String key) {

phone = key;

}

public void setEmail(String key) {

email = key;

}

public void setName(String key) {

name = key;

}

public void setBday(String key) {

bday = key;

}

}

## ****Accounts.java****

package com.nehouse.nehouse;

import java.util.ArrayList;

public class Accounts {

int accSum;

int accID;

String accName;

Accounts(int sum, String name) {

accSum = sum;

accName = name;

accID = ++MainActivity.accCount + 20000;

MainActivity.accList.add(name);

MainActivity.accQueue.add(this);

}

void setAccSum (int \_sum) {

this.accSum = \_sum;

}

}

## ****Budget.java****

package com.nehouse.nehouse;

import android.app.Dialog;

import android.content.Context;

import android.content.Intent;

import android.os.Bundle;

import android.os.Vibrator;

import android.support.v7.app.AppCompatActivity;

import android.view.Gravity;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ArrayAdapter;

import android.widget.AutoCompleteTextView;

import android.widget.EditText;

import android.widget.LinearLayout;

import android.widget.TextView;

import java.util.ArrayList;

public class Budget extends AppCompatActivity{

Dialog NewAccountDialog, IncomeDialog, ExpenseDialog;

LinearLayout parent;

View promptsView;

EditText T, S;

LinearLayout.LayoutParams lp1 = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT, 100, Gravity.CENTER);

@Override

protected void onCreate(Bundle savedInstanceState) {

LayoutInflater li = LayoutInflater.from(this);

promptsView = li.inflate(R.layout.new\_account, null);

super.onCreate(savedInstanceState);

setContentView(R.layout.budget);

parent = findViewById(R.id.AccountsQueue);

LinearLayout Acc = new LinearLayout(Budget.this);

TextView name = new TextView(Budget.this);

TextView sum = new TextView(Budget.this);

if (MainActivity.accCount != 0) {

for (int i = 0; i < MainActivity.accCount; i++) {

Acc.setLayoutParams(lp1);

Vibrator v = (Vibrator)getSystemService(Context.VIBRATOR\_SERVICE);

v.vibrate(1000);

Acc.setBackgroundResource(R.drawable.button\_desidn);

name.setText(MainActivity.accQueue.get(i).accName);

sum.setText(MainActivity.accQueue.get(i).accSum);

sum.setTextColor(getResources().getColor(R.color.colorWhite));

name.setTextColor(getResources().getColor(R.color.colorWhite));

name.setTextSize(30);

sum.setTextSize(40);

parent.addView(Acc);

Acc.addView(name);

Acc.addView(sum);

}}

NewAccountDialog = new Dialog(Budget.this);

NewAccountDialog.setContentView(R.layout.new\_account);

IncomeDialog = new Dialog(Budget.this);

IncomeDialog.setContentView(R.layout.income);

ExpenseDialog = new Dialog(Budget.this);

ExpenseDialog.setContentView(R.layout.expense);

// AutoCompleteTextView chooseAcc1 = (AutoCompleteTextView)findViewById(R.id.ExpenseOwner); //установка выпадающего списка

// if (MainActivity.accCount != 0) {chooseAcc1.setAdapter(new ArrayAdapter<>(this, android.R.layout.simple\_dropdown\_item\_1line, MainActivity.accList));}

//

// AutoCompleteTextView chooseAcc2 = (AutoCompleteTextView)findViewById(R.id.IncomeOwner);

// if (MainActivity.accCount != 0) {chooseAcc2.setAdapter(new ArrayAdapter<>(this, android.R.layout.simple\_dropdown\_item\_1line, MainActivity.accList));}

}

public void BudgetNewAccount (View view) {

NewAccountDialog.show();

}

public void BudgetIncome (View view) {

IncomeDialog.show();

}

public void BudgetExpense (View view) {

ExpenseDialog.show();

}

public void BudgetOperationsHistory (View view) {

Intent intent = new Intent(Budget.this, OperationsHistory.class);

startActivity(intent);

}

public void BudgetAccount (View view) {

//нажатие на счет

}

public void NewAccount (View view) {

T = promptsView.findViewById(R.id.NewAccountName);

S = promptsView.findViewById(R.id.NewAccountSum);

int sum = Integer.parseInt(S.getText().toString());

String name = T.getText().toString();

new Accounts(sum, name);

}

public void ExpenseConfirm (View view) {

EditText ETsum = (EditText)findViewById(R.id.ExpenseSum);

EditText ETcomm = (EditText)findViewById(R.id.ExpenseComment);

EditText ETown = (EditText)findViewById(R.id.ExpenseOwner);

String Sum = ETsum.getText().toString();

int sum = Integer.parseInt(Sum);

String comm = ETcomm.getText().toString();

String own = ETown.getText().toString();

Money m = new Money(sum, comm, false, own);

m.ExMoney();

finish();

}

public void IncomeConfirm (View view) {

EditText ETsum = (EditText)findViewById(R.id.ExpenseSum);

EditText ETcomm = (EditText)findViewById(R.id.ExpenseComment);

EditText ETown = (EditText)findViewById(R.id.ExpenseOwner);

String Sum = ETsum.getText().toString();

int sum = Integer.parseInt(Sum);

String comm = ETcomm.getText().toString();

String own = ETown.getText().toString();

Money m = new Money(sum, comm, true, own);

m.ExMoney();

finish();

}

}

## ****Calendar.java****

package com.nehouse.nehouse;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.view.Gravity;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.LinearLayout;

import static java.lang.Math.min;

public class Calendar extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.calendar);

LinearLayout parent = findViewById(R.id.evQ);

for (int i = 0; i < min(MainActivity.evCount, 3); i++) {

Button Event1 = new Button(Calendar.this);

Event1.setBackgroundResource(R.drawable.button\_desidn);

Event1.setTextColor(getResources().getColor(R.color.colorWhite));

LinearLayout.LayoutParams lp = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT, ViewGroup.LayoutParams.WRAP\_CONTENT, Gravity.CENTER);

lp.setMargins(50, 30, 50, 0);

Event1.setLayoutParams(lp);

Event1.setId(500000 + i);

Event1.setText(MainActivity.eventQueue.get(i).date + System.getProperty("line.separator") + MainActivity.eventQueue.get(i).eventType);

parent.addView(Event1);

}

}

public void CalendarDate(View view) {

Intent intent = new Intent(Calendar.this, DateEvents.class);

startActivity(intent);

}

public void CalendarNewEvent(View view) {

Intent intent = new Intent(Calendar.this, com.nehouse.nehouse.EditEvent.class);

startActivity(intent);

}

}

## ****Chat.java****

package com.nehouse.nehouse;

import android.content.Intent;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.EditText;

import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.database.ChildEventListener;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.ValueEventListener;

import com.nehouse.nehouse.Model.MyChat;

import java.util.ArrayList;

public class Chat extends AppCompatActivity {

ArrayList<Chat> mchat; // список всех сообщений.

// LinearLayout parent;

// LinearLayout.LayoutParams lp = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT, ViewGroup.LayoutParams.WRAP\_CONTENT,Gravity.CENTER);

@Override

protected void onCreate(Bundle savedInstanceState) {

if (MainActivity.user.getGroupID() == null) {

Intent intent = new Intent(Chat.this, ChooseGroup.class);

startActivity(intent);

}

super.onCreate(savedInstanceState);

setContentView(R.layout.chat);

String group = MainActivity.user.getGroupID();

MainActivity.messagesDB.child(group).addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

readMesagges();

//Тут реализовать вывод на экран всех сообщений из списка mchat

//Перед выводом сообщений нужно удалить все сообщения что есть на экране и потом заново создать все полученные

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {}

});

/\*\* lp.setMargins(25, 5,25,5);

parent = (LinearLayout)findViewById(R.id.MessageQueue);

MainActivity.messagesDB.addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(@NonNull0 DataSnapshot dataSnapshot) {

MainActivity.message = dataSnapshot.getValue(Messages.class);

TextView mes = new TextView(Chat.this);

int size = MainActivity.message.getMeslist().size();

mes.setText(MainActivity.message.getMeslist().get(size - 1));

mes.setBackgroundResource(R.drawable.button\_desidn);

mes.setLayoutParams(lp);

mes.setGravity(Gravity.CENTER);

parent.addView(mes);

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {

}

});\*/

}

public void ChatSend(View view) {

String sender = MainActivity.currentUser.getUid();

EditText ETmessage = findViewById(R.id.editText2);

final String message = ETmessage.getText().toString();

String groupID = MainActivity.user.getGroupID();

/\*\*TextView mes = new TextView(Chat.this);

mes.setText(message);

mes.setBackgroundResource(R.drawable.button\_desidn);

mes.setLayoutParams(lp);

mes.setGravity(Gravity.CENTER);

parent.addView(mes);

HashMap<String, Object> hm = new HashMap<>();

hm.put("sender", sender);

hm.put("receiver", receiver);

hm.put("message", message );\*/

MyChat ch = new MyChat(sender, message);

String key = MainActivity.messagesDB.child(groupID).push().getKey();

MainActivity.messagesDB.child(groupID).child(key).setValue(ch).addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

Toast.makeText(Chat.this, "Send",

Toast.LENGTH\_SHORT).show();

}

});

}

public void readMesagges() {

String groupID = MainActivity.user.getGroupID();

mchat = new ArrayList<>();

MainActivity.messagesDB.child(groupID).addChildEventListener(new ChildEventListener() {

@Override

public void onChildAdded(@NonNull DataSnapshot dataSnapshot, @Nullable String s) {

mchat.clear();

Chat chat = dataSnapshot.getValue(Chat.class);

mchat.add(chat);

}

@Override

public void onChildChanged(@NonNull DataSnapshot dataSnapshot, @Nullable String s) {}

@Override

public void onChildRemoved(@NonNull DataSnapshot dataSnapshot) {}

@Override

public void onChildMoved(@NonNull DataSnapshot dataSnapshot, @Nullable String s) {}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {}

});

}

}

## ****ChooseGroup.java****

package com.nehouse.nehouse;

import android.content.Intent;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.EditText;

import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.ValueEventListener;

import com.nehouse.nehouse.Model.Group;

import com.nehouse.nehouse.Model.User;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class ChooseGroup extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.choose\_group);

}

@Override

public void onStart() {

super.onStart();

if (MainActivity.user.getGroupID() != null) {

Intent intent = new Intent(ChooseGroup.this, Chat.class);

startActivity(intent);

}

}

public void NewGroup(View view) {

Intent intent = new Intent(ChooseGroup.this, NewGroup.class);

startActivity(intent);

}

public void ChoosenGroup(View view) {

EditText ETname = (EditText) findViewById(R.id.ChooseGroup);

final String name = ETname.getText().toString();

if (name.equals(null)) {

Toast.makeText(ChooseGroup.this, "Filed is required", Toast.LENGTH\_SHORT).show();

} else {

MainActivity.groupDB.addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

if(dataSnapshot.child(name).exists()) {

MainActivity.user.setGroupID(name);

String userId = MainActivity.user.getId();

HashMap<String, String> hm = new HashMap<>();

hm.put("groupID", name);

MainActivity.groupDB.child(userId).setValue(hm);

final DataSnapshot ds = dataSnapshot;

final String key = MainActivity.groupDB.child(name).push().getKey();

Map<String, Object> info = new HashMap<>();

info.put(key, userId);

MainActivity.groupDB.child(name).updateChildren(info).addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

MainActivity.group = ds.getValue(Group.class);

Intent intent = new Intent(ChooseGroup.this, Chat.class);

Toast.makeText(ChooseGroup.this, "Succesfull.",

Toast.LENGTH\_SHORT).show();

startActivity(intent);

}

});

} else {

Toast.makeText(ChooseGroup.this, "This group isn't exists.",

Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {

}

});

}

}

}

## ****Contacts.java****

package com.nehouse.nehouse;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

public class Contacts extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.contacts);

}

}

## ****DateEvents.java****

package com.nehouse.nehouse;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

public class DateEvents extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.date\_events);

}

}

## ****EditEvent.java****

package com.nehouse.nehouse;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.EditText;

import android.widget.TextView;

import com.nehouse.nehouse.Model.Event;

import com.nehouse.nehouse.Model.Purchase;

import java.text.SimpleDateFormat;

import java.util.Date;

public class EditEvent extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.edit\_event);

}

public void newEvent(View view) {

EditText ETdate = findViewById(R.id.EventDate);

String date = ETdate.getText().toString();

EditText ETtype = findViewById(R.id.EventType);

EditText ETcomm = findViewById(R.id.EventComment);

EditText ETpur = findViewById(R.id.EventPresent);

EditText ETcost = findViewById(R.id.EventCost);

String type = ETtype.getText().toString();

String comm = ETcomm.getText().toString();

String pur = ETpur.getText().toString();

int cost = Integer.parseInt(ETcost.getText().toString());

MainActivity.eventQueue.add(new Event(date, comm, type, pur, cost));

MainActivity.purchaseQueue.add(new Purchase(cost, pur));

MainActivity.purCount++;

finish();

}

}

## ****Groups.java****

package com.nehouse.nehouse;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.v7.app.AppCompatActivity;

import android.support.v7.widget.LinearLayoutManager;

import android.support.v7.widget.RecyclerView;

import android.view.View;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import com.nehouse.nehouse.Model.User;

import java.util.ArrayList;

import java.util.List;

public class Groups extends AppCompatActivity {

private List<User> mUsers;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.group);

mUsers = new ArrayList<>();

readUsers();

}

private void readUsers() {

final FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();

DatabaseReference reference = FirebaseDatabase.getInstance().getReference("Users");

reference.addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

mUsers.clear();

for(DataSnapshot snapshot : dataSnapshot.getChildren()) {

User user = snapshot.getValue(User.class);

if(user.getId().equals(firebaseUser.getUid())) {

mUsers.add(user);

}

}

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {

}

});

}

}

## ****LogOn.java****

package com.nehouse.nehouse;

import android.accounts.Account;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.text.TextUtils;

import android.util.Log;

import android.view.View;

import android.widget.EditText;

import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import com.nehouse.nehouse.Model.User;

import java.util.HashMap;

import static android.content.ContentValues.TAG;

import static com.nehouse.nehouse.MainActivity.currentUser;

public class LogOn extends Activity {

private FirebaseAuth mAuth;

private String userName, email, password;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.log\_on);

mAuth = FirebaseAuth.getInstance();

}

@Override

public void onStart() {

super.onStart();

currentUser = mAuth.getCurrentUser();

updateUI(currentUser);

}

public void LogOnLogOn (View view) {

String \_password = GetPassword();

String \_email = GetEmail();

String \_userName = GetUserName();

if (\_userName.isEmpty() || \_password.isEmpty() || \_email.isEmpty()) {

Toast.makeText(LogOn.this, "All fileds are required", Toast.LENGTH\_SHORT).show();

} else if (\_password.length() < 6) {

Toast.makeText(LogOn.this, "Password must be at least 6 characters", Toast.LENGTH\_SHORT).show();

} else {

userName = GetUserName();

email = GetEmail();

password = GetPassword();

CreateAccount();

}

}

public String GetUserName(){

EditText ETusername = (EditText) findViewById(R.id.UserName); //Read info from user interface

return ETusername.getText().toString();

}

public String GetEmail(){

EditText ETemail = (EditText) findViewById(R.id.Email); //Read info from user interface

return ETemail.getText().toString();

}

public String GetPassword(){

EditText ETpassword = (EditText) findViewById(R.id.Password1); //Read info from user interface

return ETpassword.getText().toString();

}

public void CreateAccount() {

mAuth.createUserWithEmailAndPassword(email, password)

.addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if (task.isSuccessful()) {

Log.d(TAG, "createUserWithEmail:success");

FirebaseUser user = mAuth.getCurrentUser();

Toast.makeText(LogOn.this, "Authentication succesfull.",

Toast.LENGTH\_SHORT).show();

String userID = user.getUid();

MainActivity.user = new User();

MainActivity.user.setEmail(email);

MainActivity.user.setId(userID);

HashMap<String, String> user\_info = new HashMap<>();

user\_info.put("id", userID);

user\_info.put("name", userName);

user\_info.put("password", password);

user\_info.put("email", email);

//MainActivity.user = new User(MainActivity.currentUser.getEmail(), MainActivity.currentUser.getUid());

MainActivity.usersDB.child(userID).setValue(user\_info).addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if (task.isSuccessful()) {

Toast.makeText(LogOn.this, "DB update is succesfull.",

Toast.LENGTH\_SHORT).show();

}

}

});

updateUI(user);

} else {

Log.w(TAG, "createUserWithEmail:failure", task.getException());

Toast.makeText(LogOn.this, "Authentication failed.",

Toast.LENGTH\_SHORT).show();

updateUI(null);

}

}

});

}

private void updateUI(FirebaseUser user) {

currentUser = user;

if (user != null) {

Intent intent = new Intent(LogOn.this, MainActivity.class);

startActivity(intent);

}

}

}

## ****MainActivity.java****

package com.nehouse.nehouse;

import android.app.Dialog;

import android.content.Intent;

import android.support.annotation.NonNull;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import com.nehouse.nehouse.Model.Event;

import com.nehouse.nehouse.Model.Group;

import com.nehouse.nehouse.Model.Message;

import com.nehouse.nehouse.Model.MyChat;

import com.nehouse.nehouse.Model.Purchase;

import com.nehouse.nehouse.Model.User;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

private FirebaseAuth mAuth;

public static FirebaseUser currentUser;

public static DatabaseReference usersDB = FirebaseDatabase.getInstance().getReference("Users");

public static DatabaseReference messagesDB = FirebaseDatabase.getInstance().getReference("Messages");

public static DatabaseReference groupDB = FirebaseDatabase.getInstance().getReference("Groups");

public static User user;

public static Group group;

public static ArrayList<String> myWishes;

public static int count, evCount, accCount, opCount, purCount;

public static ArrayList<Event> eventQueue;

public static ArrayList<Purchase> purchaseQueue;

public static ArrayList<Money> moneyQueue;

public static ArrayList<Accounts> accQueue;

public static ArrayList<String> accList;

@Override

protected void onCreate(Bundle savedInstanceState) {

currentUser = FirebaseAuth.getInstance().getCurrentUser();

if (currentUser == null) {

Intent intent = new Intent(MainActivity.this, com.nehouse.nehouse.WelcomePage.class);

startActivity(intent);

}

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

count = 0;

evCount = 0;

accCount = 0;

opCount = 0;

purCount = 0;

accList = new ArrayList<>();

eventQueue = new ArrayList<>();

purchaseQueue = new ArrayList<>();

moneyQueue = new ArrayList<>();

accQueue = new ArrayList<>();

myWishes = new ArrayList<>();

/\*\* usersDB.child(currentUser.getUid()).addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

user = dataSnapshot.getValue(User.class);

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {

}

});\*/

}

public void MainActivityCalendar (View view) {

Intent intent = new Intent(MainActivity.this, com.nehouse.nehouse.Calendar.class);

startActivity(intent);

}

public void MainActivityBudget (View view) {

Intent intent = new Intent(MainActivity.this, com.nehouse.nehouse.Budget.class);

startActivity(intent);

}

public void MainActivityChat (View view) {

Intent intent = new Intent(MainActivity.this, Chat.class);

startActivity(intent);

}

public void MainActivityShoppingList (View view) {

Intent intent = new Intent(MainActivity.this, ShoppingList.class);

startActivity(intent);

}

public void MainActivityGroup (View view) {

Intent intent = new Intent(MainActivity.this, Chat.class);

startActivity(intent);

}

public void MainActivityWishlist (View view) {

Intent intent = new Intent(MainActivity.this, com.nehouse.nehouse.MyWishlist.class);

startActivity(intent);

}

public void MainActivitySettings (View view) {

Intent intent = new Intent(MainActivity.this, Settings.class);

startActivity(intent);

}

}

## ****Money.java****

package com.nehouse.nehouse;

import java.util.Date;

public class Money {

static int OpCount = 0;

int sum;

Date date;

int Mid;

boolean income;

String comment;

int owner;

Money(int \_sum, String \_comm, boolean \_inc, String \_own) {

sum = \_sum;

comment = \_comm;

income = \_inc;

owner = findOwner(\_own);

date = new Date();

Mid = ++MainActivity.opCount + 10000;

}

public void ExMoney () {

MainActivity.moneyQueue.add(this);

int sum1;

if (income) {

sum1 = MainActivity.accQueue.get(owner).accSum + sum;

}

else {

sum1 = MainActivity.accQueue.get(owner).accSum - sum;

}

MainActivity.accQueue.get(owner).setAccSum(sum1);

}

int findOwner(String own) {

for(int j = 0; j < MainActivity.accCount; j++) {

if (MainActivity.accQueue.get(j).accName == own) return j;

}

return -1;

}

}

## ****MyWishlist.java****

package com.nehouse.nehouse;

import android.media.AudioManager;

import android.media.ToneGenerator;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.view.Gravity;

import android.view.View;

import android.view.ViewGroup;

import android.widget.EditText;

import android.widget.LinearLayout;

import android.widget.TextView;

import java.util.ArrayList;

public class MyWishlist extends AppCompatActivity {

LinearLayout parent;

static boolean first;

LinearLayout.LayoutParams lp = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT, 70, Gravity.CENTER);

public final View.OnLongClickListener lis = new View.OnLongClickListener() {

@Override

public boolean onLongClick(View view) {

if (first) {MainActivity.myWishes.remove(view.getId() - (int)1);}

else {MainActivity.myWishes.remove(view.getId());}

parent.removeAllViews();

MainActivity.count--;

for(int i = 0; i < MainActivity.count; i++) {

TextView wish = new TextView(MyWishlist.this);

wish.setText(MainActivity.myWishes.get(i));

wish.setBackgroundResource(R.drawable.button\_desidn);

wish.setLayoutParams(lp);

wish.setId(i);

wish.setTextColor(getResources().getColor(R.color.colorWhite));

wish.setGravity(Gravity.CENTER);

wish.setOnLongClickListener(lis);

parent.addView(wish);

}

first = false;

return true;

}

};

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.my\_wishlist);

first = true;

lp.setMargins(25, 10,25,0);

parent = (LinearLayout)findViewById(R.id.MyWishesQueue);

if (MainActivity.count != 0) {

for(int i = 0; i < MainActivity.count; i++) {

TextView wish = new TextView(MyWishlist.this);

wish.setText(MainActivity.myWishes.get(i));

wish.setBackgroundResource(R.drawable.button\_desidn);

wish.setLayoutParams(lp);

wish.setId(i);

wish.setTextColor(getResources().getColor(R.color.colorWhite));

wish.setGravity(Gravity.CENTER);

wish.setOnLongClickListener(lis);

parent.addView(wish);

}

}

}

public void addWish(View view) {

EditText txt = (EditText)findViewById(R.id.newWish);

String text = txt.getText().toString();

TextView wish = new TextView(MyWishlist.this);

wish.setText(text);

wish.setBackgroundResource(R.drawable.button\_desidn);

wish.setLayoutParams(lp);

wish.setGravity(Gravity.CENTER);

wish.setId(++MainActivity.count);

wish.setTextColor(getResources().getColor(R.color.colorWhite));

wish.setOnLongClickListener(lis);

parent.addView(wish);

MainActivity.myWishes.add(text);

txt.setText(null);

}

}

## ****NewGroup.java****

package com.nehouse.nehouse;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import com.nehouse.nehouse.Model.Group;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

import static com.nehouse.nehouse.MainActivity.user;

public class NewGroup extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.new\_group);

}

@Override

public void onStart() {

super.onStart();

if (MainActivity.user.getGroupID() != null) {

Intent intent = new Intent(NewGroup.this, Chat.class);

startActivity(intent);

}

}

public void NewGroupONConfirm(View view) {

EditText ETname = findViewById(R.id.TVNewGroupName);

final String name = ETname.getText().toString();

if (name.equals(null)) {

Toast.makeText(NewGroup.this, "Filed is required", Toast.LENGTH\_SHORT).show();

} else {

MainActivity.groupDB.addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

if(dataSnapshot.child(name).exists()) {

Toast.makeText(NewGroup.this, "This group is already exists.",

Toast.LENGTH\_SHORT).show();

} else {

MainActivity.user.setGroupID(name);

String userId = MainActivity.user.getId();

HashMap<String, String> hm = new HashMap<>();

hm.put("groupID", name);

MainActivity.groupDB.child(userId).setValue(hm);

final DataSnapshot ds = dataSnapshot;

final String key = MainActivity.groupDB.child("friends").push().getKey();

Map<String, Object> info = new HashMap<>();

info.put(key, userId);

MainActivity.groupDB.child(name).updateChildren(info).addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

MainActivity.group = ds.getValue(Group.class);

Intent intent = new Intent(NewGroup.this, Chat.class);

Toast.makeText(NewGroup.this, "Succesfull.",

Toast.LENGTH\_SHORT).show();

startActivity(intent);

}

});

}

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) { }

});

}

}

}

## ****OperationsHistory.java****

package com.nehouse.nehouse;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.widget.LinearLayout;

import android.widget.TextView;

public class OperationsHistory extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.operations\_history);

LinearLayout OpQueue = (LinearLayout)findViewById(R.id.OperationsQueue);

TextView comm = (TextView)findViewById(R.id.OpComm);

TextView own = (TextView)findViewById(R.id.OpOwn);

TextView sum = (TextView)findViewById(R.id.OpSum);

for (int i = 0; i < Money.OpCount; i++) {

LinearLayout Mon = (LinearLayout)findViewById(R.id.Mon);

comm.setText(MainActivity.moneyQueue.get(i).comment);

own.setText(MainActivity.moneyQueue.get(i).owner);

if(MainActivity.moneyQueue.get(i).income) {

sum.setText(MainActivity.moneyQueue.get(i).sum);

}

else {

sum.setText("- " + MainActivity.moneyQueue.get(i).sum);

}

OpQueue.addView(Mon);

}

}

}

## ****Profile.java****

package com.nehouse.nehouse;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.util.Log;

import android.view.View;

import android.widget.EditText;

import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.FirebaseUser;

import com.nehouse.nehouse.Model.User;

import java.util.HashMap;

import java.util.Map;

import static android.content.ContentValues.TAG;

public class Profile extends Activity {

public static boolean logon = true;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.profile);

}

public void ProfileConfirm (View view) {

if (!logon) { //если вызвана из settings

logon = true; //возвращаем флаг в предыдущее состояние

finish(); //завершаем активность

}

EditText ETname = findViewById(R.id.Name);

EditText ETDateOfBirth = findViewById(R.id.DateOfBirth);

EditText ETProfilePhone = findViewById(R.id.ProfilePhone);

String name = ETname.getText().toString();

String DateOfBirth = ETDateOfBirth.getText().toString();

String ProfilePhone = ETProfilePhone.getText().toString();

if (Check(name, DateOfBirth, ProfilePhone)) {

MainActivity.user.setName(name);

MainActivity.user.setBday(DateOfBirth);

MainActivity.user.setPhone(ProfilePhone);

updateUInfo(name, DateOfBirth, ProfilePhone);

Intent intent = new Intent(Profile.this, MainActivity.class);

startActivity(intent);

}

}

private void updateUInfo(String name, String dateOfBirth, String phone) {

String userId = MainActivity.user.getId();

String id = MainActivity.user.getId();

String email = MainActivity.user.getEmail();

String group = MainActivity.user.getGroupID();

User user = new User(id, name, phone, email, group, dateOfBirth);

Map<String, Object> userMap = user.toMap();

Map<String, Object> childUpdates = new HashMap<>();

childUpdates.put(userId, userMap);

MainActivity.usersDB.updateChildren(childUpdates).addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if (task.isSuccessful()) {

Toast.makeText(Profile.this, "Update succesfull.",

Toast.LENGTH\_SHORT).show();

}

}

});

}

private boolean Check(String name, String dateOfBirth, String profilePhone) {

if (name.isEmpty() || dateOfBirth.isEmpty() || profilePhone.isEmpty()) {

Toast.makeText(Profile.this, "All fileds are required", Toast.LENGTH\_SHORT).show();

return false;

}

return true;

}

}

## ****Settings.java****

package com.nehouse.nehouse;

import android.content.Intent;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.TextView;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import com.nehouse.nehouse.Model.User;

public class Settings extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.settings);

TextView userID = (TextView)this.findViewById(R.id.UserID); //переменная userID хранит TextView с id == "UserID"

TextView name = (TextView)this.findViewById(R.id.AccName);

TextView bday = (TextView)this.findViewById(R.id.AccDateOfBirth);

TextView email = (TextView)this.findViewById(R.id.AccProfileEmail);

TextView phone = (TextView)this.findViewById(R.id.AccProfilePhone);

TextView group = (TextView)this.findViewById(R.id.GroupName);

userID.setTextColor(getResources().getColor(R.color.colorWhite));

name.setTextColor(getResources().getColor(R.color.colorWhite));

bday.setTextColor(getResources().getColor(R.color.colorWhite));

email.setTextColor(getResources().getColor(R.color.colorWhite));

phone.setTextColor(getResources().getColor(R.color.colorWhite));

group.setTextColor(getResources().getColor(R.color.colorWhite));

userID.setText("UserID: " + System.getProperty("line.separator") + MainActivity.currentUser.getUid());

name.setText("Name: " + MainActivity.user.getName());

bday.setText("DateOfBirth: " + MainActivity.user.getBday());

email.setText("Email: " + MainActivity.user.getEmail());

phone.setText("Phone: " + MainActivity.user.getPhone());

group.setText("Group: " + MainActivity.user.getGroupID());

}

public void SettingsEditProfile (View view) {

Intent intent = new Intent(Settings.this, Profile.class);

Profile.logon = false;

startActivity(intent);

}

public void SettingsLogOut (View view) {

FirebaseAuth.getInstance().signOut();

Intent intent = new Intent(Settings.this, WelcomePage.class);

startActivity(intent);

}

}

## ****WelcomePage.java****

package com.nehouse.nehouse;

import android.app.Activity;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.text.TextUtils;

import android.util.Log;

import android.view.View;

import android.widget.EditText;

import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.nehouse.nehouse.Model.User;

import static android.content.ContentValues.TAG;

public class WelcomePage extends Activity {

private FirebaseAuth mAuth;

private FirebaseUser currentUser;

private String email, password;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.welcome\_page);

mAuth = FirebaseAuth.getInstance();

}

@Override

public void onStart() {

super.onStart();

currentUser = mAuth.getCurrentUser();

updateUI(MainActivity.currentUser);

}

public void WelcomePageLogIn (View view) {

String \_password = GetPassword();

String \_email = GetEmail();

if (\_password.isEmpty() || \_email.isEmpty()) {

Toast.makeText(WelcomePage.this, "All fileds are required", Toast.LENGTH\_SHORT).show();

} else {

email = \_email;

password = \_password;

SignIn();

}

}

public String GetEmail(){

EditText ETemail = (EditText) findViewById(R.id.Email);

return ETemail.getText().toString();

}

public String GetPassword(){

EditText ETpassword = (EditText) findViewById(R.id.Password);

return ETpassword.getText().toString();

}

public void SignIn()

{

mAuth.signInWithEmailAndPassword(email, password)

.addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if (task.isSuccessful()) {

Log.d(TAG, "signInWithEmail:success");

FirebaseUser user = mAuth.getCurrentUser();

currentUser = user;

updateUI(user);

} else {

Log.w(TAG, "signInWithEmail:failure", task.getException());

Toast.makeText(WelcomePage.this, "Authentication failed.",

Toast.LENGTH\_SHORT).show();

updateUI(null);

}

}

});

}

public void WelcomePageLogOn (View view) {

Intent intent = new Intent(WelcomePage.this, LogOn.class);

startActivity(intent);

}

public void WelcomePageForgot (View view) {

Intent intent = new Intent(WelcomePage.this, ForgotPassword.class);

startActivity(intent);

}

private void updateUI(FirebaseUser user) {

currentUser = user;

//MainActivity.currentUser = user;

if (user != null) {

Intent intent = new Intent(WelcomePage.this, MainActivity.class);

startActivity(intent);

}

}

}