**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Образовательная программа бакалавриата «Программная инженерия»

|  |  |  |  |
| --- | --- | --- | --- |
| СОГЛАСОВАНО  Научный руководитель,  доцент департамента  Программной инженерии  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Р.З. Ахметсафина  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | RU.17701729.04.03-01 81 01-1-ЛУ | | **МОБИЛЬНОЕ ПРИЛОЖЕНИЕ “StudyGo”**  **Пояснительная записка**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.04.03-01 81 01-1-ЛУ** | | | | | |
|  | |  | | | |
| Исполнитель  студент группы БПИ197  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Бакытбек уулу Н. /  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | | | |

**Москва 2020**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| УТВЕРЖДЕН  RU.17701729.04.03-01 81 01-1 |  | |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | RU.17701729.04.03-01 81 01-1-ЛУ | | **МОБИЛЬНОЕ ПРИЛОЖЕНИЕ “ VOBSHAGE”**  **Пояснительная записка**  **RU.17701729.04.03-01 81 01-1**  **Листов 39** | | | | |
|  | |  | | |
|  | | |
|  | | | | |
|  | | | |  |

**Москва 2020**

Оглавление

[1 Введение. 3](#_Toc40980599)

[1.1 Наименование программы: 3](#_Toc40980600)

[1.2 Документы, на основании которых ведется разработка: 3](#_Toc40980601)

[2 Назначение и область применения. 4](#_Toc40980602)

[2.1 Назначение программы 4](#_Toc40980603)

[2.1.1 Функциональное назначение 4](#_Toc40980604)

[2.1.2 Эксплуатационное назначение 4](#_Toc40980605)

[2.2 Краткая характеристика области применения 4](#_Toc40980606)

[3 Технические характеристики. 5](#_Toc40980607)

[3.1 Постановка задачи на разработку программы 5](#_Toc40980608)

[3.2 Описание функционирования программы 5](#_Toc40980609)

[3.2.1 Описание функционирования различных окон программы 5](#_Toc40980610)

[3.2.2 Описание структуры онлайн базы данных Firebase Realtime Database 14](#_Toc40980611)

[3.2.3 Диаграмма классов 17](#_Toc40980612)

[3.3 Описание и обоснование выбора метода организации входных и выходных данных 18](#_Toc40980613)

[3.3.1 Описание метода организации входных и выходных данных 18](#_Toc40980614)

[3.3.2 Обоснование выбора метода организации входных и выходных данных 18](#_Toc40980615)

[3.4 Описание и обоснование выбора состава технических и программных средств 19](#_Toc40980616)

[3.4.1 Состав технических и программных средств 19](#_Toc40980617)

[4 Ожидаемые технико-экономические показатели 20](#_Toc40980618)

[4.1 Предполагаемая потребность 20](#_Toc40980619)

[1.1 Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами 20](#_Toc40980620)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 21](#_Toc40980621)

[ПРИЛОЖЕНИЕ 1 КАРТА КОДА 23](#_Toc40980622)

[ПРИЛОЖЕНИЕ 2 ОПИСАНИЕ КЛАССОВ 24](#_Toc40980623)

[ПРИЛОЖЕНИЕ 3 ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ПОЛЕЙ МЕТОДОВ И СВОЙСТВ 25](#_Toc40980624)

[Лист регистрации изменений. 38](#_Toc40980625)

# Введение.

## Наименование программы:

“Веб-приложение “StudyGo” для поддержки образовательного процесса”

## Документы, на основании которых ведется разработка:

В качестве основания для разработки продукта выступает учебный план, посвященный подготовке бакалавров по направлению 09.03.04 "Программная инженерия", и тема курсового проекта, утвержденная академическим руководителем.

# Назначение и область применения.

## Назначение программы

### Функциональное назначение

{

Авторизация пользователей. Сохранение данных для обеспечения выполнения логики программы и предоставление пользователям целостной информации. Обеспечить обмен файлами внутри между пользователями.

}

Панель управления:

Задача панели управления:

- предоставление всех элементов страницы для добавления дальнейшего функционала и навигации по приложению:

* Кнопка для показа выпадающего окна выхода из аккаунта
* Кнопка для показа выпадающего окна управления текущей группой
* Кнопка для перехода на страницу просмотра оценок студентами за выполненные работы
* Кнопка для показа выпадающего окна со следующими элементами для просмотра содержания группы:
  + Пункт окна для перехода на страницу со всеми обязательными заданиями
  + Пункт окна для перехода на страницу со всеми необязательными заданиями
  + Пункт окна для просмотра содержания каналов

-Предоставить возможность комфортного взаимодействия с приложением.

-Переход между различными окнами приложения

Задача окон приложения:

* Предоставить удобный интерфейс для пользователя
* Просмотр главного окна с краткой информацией о команде и видео-инструкцией по пользованию приложением.
* Просмотр содержания групп
* Просмотр содержания каналов
* Просмотр обучающимися оценок и комментариев по своим выполненным работам
* эффективное хранение, представление и обновление данных

В программе можно создавать группы, ассоциированные с учебными группами.

В пределах каждой группы есть возможность создавать каналы, связанные с конкретной дисциплиной для группы. Строго только создатель канала обладает возможностью для создания трех видов сообщений: пост, задание, крайний срок сдачи определенной работы - с возможностью приложения файлов. Создатель канала имеет возможность отправить другим пользователям обратную связь в виде оценок и, если есть необходимость, комментариев по полученным работам.

Также другим участникам группы доступен просмотр всех сообщений всех видов со всех каналов. Участники группы могут отправлять свои выполненные работы в ответ на сообщения типа крайнего срока. Участникам также доступен просмотр результатов проверок на свои работы от создателей групп в виде оценок и комментариев.

Все пользователя могут добавляться в группы по уникальным ссылкам для каждой группы.

### Эксплуатационное назначение

{

Авторизация пользователей. Сохранение данных для обеспечения выполнения логики программы и предоставление пользователям целостной информации. Обеспечить обмен файлами внутри программы между пользователями.

}

Приложение нацелено для использования преподавателями, обучающимися для обмена учебными материалами и учебной информацией.

Сегодня многие преподаватели задают задания, ставят крайние сроки, публикуют объявления, пользуясь при этом разными средствами: почта - для отправки учебных материалов, мессенджеры - для публикации объявлений и так далее - что не совсем удобно, так как преподавателю приходится разрываться сразу на несколько площадок.

Текущее приложение помогает облегчить преподавателям и студентам обмен заданиями, решениями и обратной связью по выполненным работам.

Примечание: обмен заключается в том, что преподаватель может отправлять различные объявления, задания с приложенными к ним файлами заданий, а студент отправляет выполненные работы в ответ на обязательны задания от преподавателя и получает обратную связь по работе от преподавателя.

## Краткая характеристика области применения

Текущая программа может применяться для упрощения взаимодействия обучающихся и преподавателей в следующем виде: преподавателям требуется публиковать объявления и задания для студентов. Студенты загружают выполненные работы, а преподаватели внутри приложения дают обратную связь по принятым работам в виде оценок и комментариев.

# Технические характеристики.

## Постановка задачи на разработку программы

Для всех пользователей:

* + Авторизация пользователя.
  + Создание группы.
  + Создание каналов в каждой учебной группе, связанных с конкретными дисциплинами.
  + Приглашение и добавление пользователя в конкретную учебную группу по ее уникальной ссылке.
  + Просмотр ленты сообщений в конкретном канале.
  + Просмотр ленты всех необязательных заданий со всех каналов.
  + Просмотр ленты всех обязательных заданий со всех каналов.
  + Предоставление выполненных работ в ответ на сообщения с обязательным заданием.
  + Просмотр обратной связи в виде оценок и комментариев от преподавателей.
  + Редактирование названия и картинки группы.
  + Возможность покинуть группу.

Преподаватель:

* + Публикация сообщений:
    - Объявление.
    - Необязательное (без крайнего срока) задание.
    - Обязательное (с крайним сроком) задание.
  + Предоставление обратной связи по выполненным работам в виде оценок и комментариев.
  + Просмотр список студентов, которые отправили выполненные работы в ответ на обязательные задания, и их оценок, комментариев в случае их наличия.

## Описание функционирования программы

### Описание функционирования различных окон программы

Структуру каждого окна приложения можно редактировать в .html, .css, .js файлах соответствующего окна. ~~Для ознакомления с архитектурой проекта смотрите Приложение 1 (???)~~. В ходе разработки программы использовались ресурсы [4]-[12].

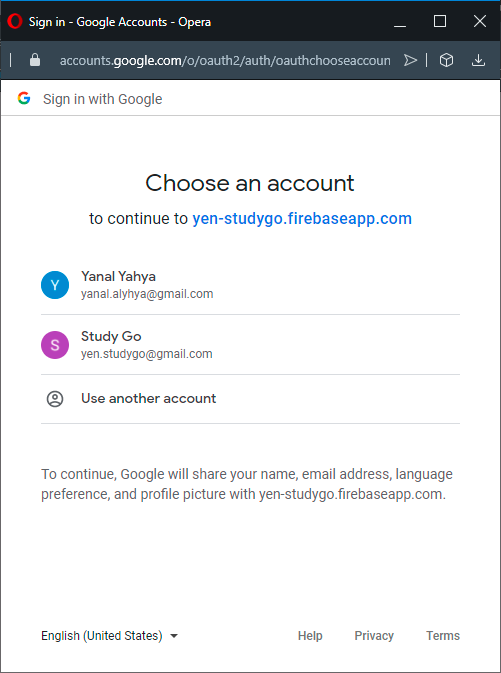
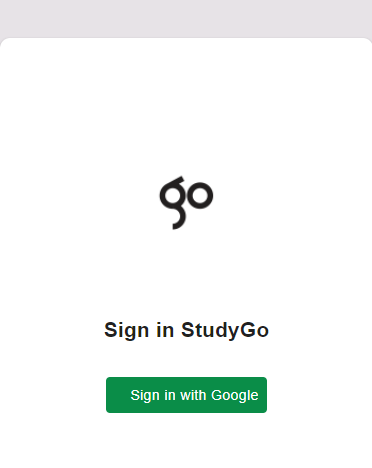


Рисунок 1 - Страница авторизации.

При входе в приложение пользователю предлагается авторизоваться (рисунок 1).

**Окно авторизации**



Для авторизации пользователей используется служба Firebase Authentication с помощью google аккаунта. Управление учетными записями происходит в личном кабинете Firebase Authentication (см. рисунок 2). Для работы с этой службой используется модуль react-firebase-hooks/auth при помощи которого можно получить доступ к данным Google-account пользователя.

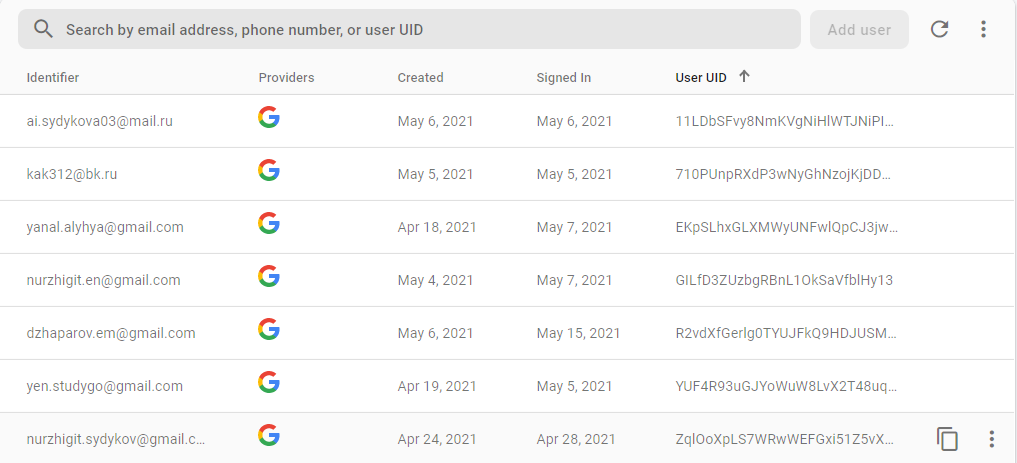


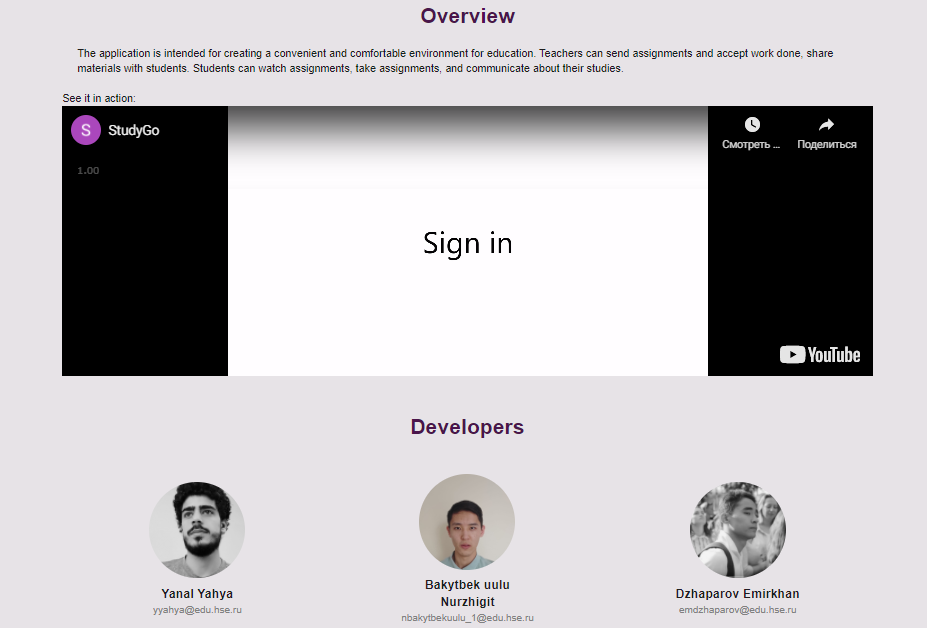
Рисунок 2. Управление учетными записями авторизованных.

В окне авторизации содержится:

1. Кнопка для авторизации через определенный учетную запись Google.
2. Всплывающее окно для авторизации в учетной записи Google.

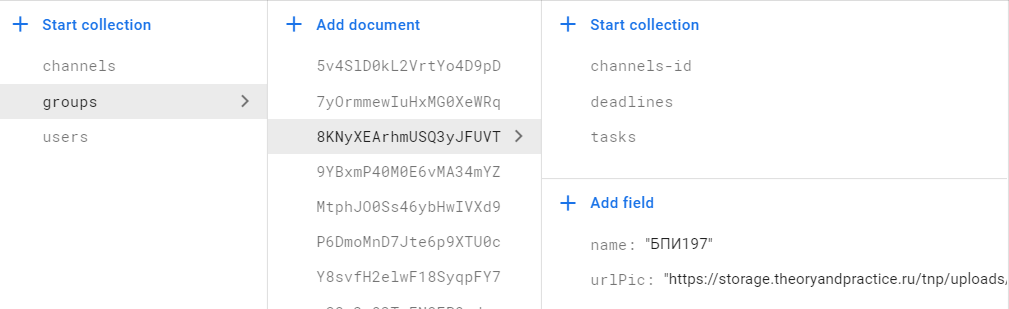
После авторизации пользователь попадет в главное **окно с кратким описанием приложения и видеоинструкцией по приложению** (рисунок 4)**.**

**Окно с кратким описанием приложения и видеоинструкция по приложению**



**Панель приложения**

Панель приложения получаем из файла App.js, извлекаем группу при помощи хука useDocument из модуля react-firebase-hooks/firestore, а пользователя при помощи хука useAuthState из модуля react-firebase-hooks/auth, где группа представлена в виде документа (рис ), важно отметить, что при извлечении данных происходит загрузка только полей, без коллекций. Пользователь представляет объект, который инкапсулирует информацию из учетной записи Google.



В панели приложения (рисунок №) содержится:

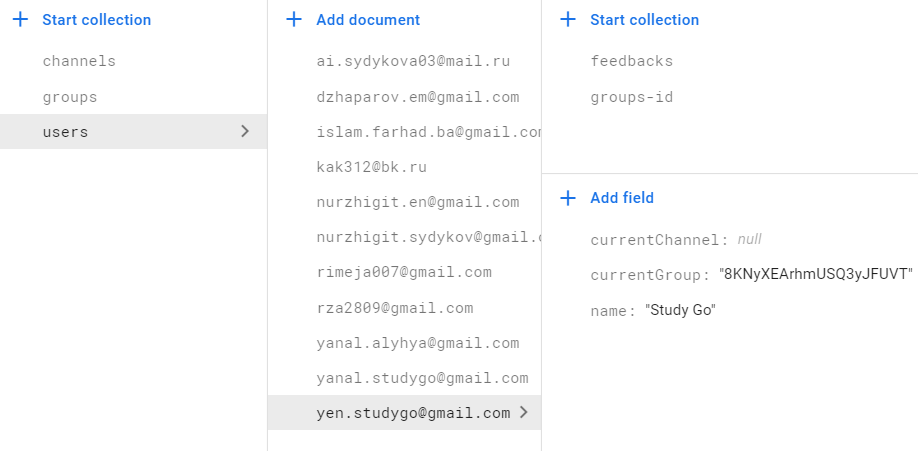


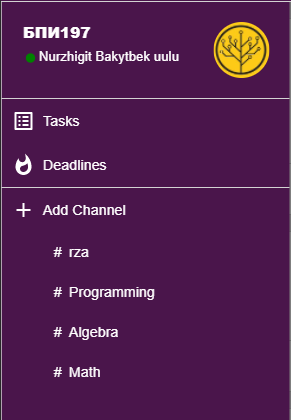
Рисунок №. Панель приложения

По нажатии на иконку меню (1) открывается **панель навигации**, при нажатии на аватар группы (2) появляется **выпадающее окно управления группой**. При нажатии на аватар пользователя (3) появляется **выпадающее окно для выхода из аккаунта**. При нажатии на иконку дневника (4) переходим в **окно с обратной связью от преподавателей**.

**Панель навигации**

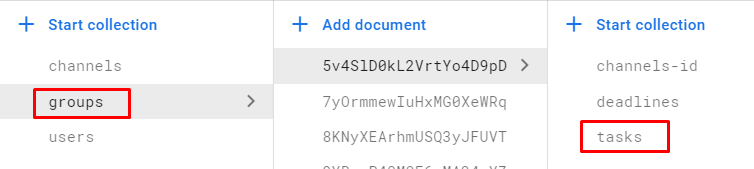
Компоненту панели навигации получаем из файла Sidebar.js. Извлекаем хуком useDocument группу (рис ) и пользователя (рис ). По нажатию на кнопку Tasks переходим в **окно необязательных задач**, а по нажатию на кнопку Deadlines в **окно обязательных заданий**. По нажатию на кнопку Add Channel, открывается **окно создания канала**. Извлекаем коллекцию идентификаторов каналов группы (рис) и по нажатию по названию канала переходим в **окно канала**, сохранив в Redux идентификатор канала.

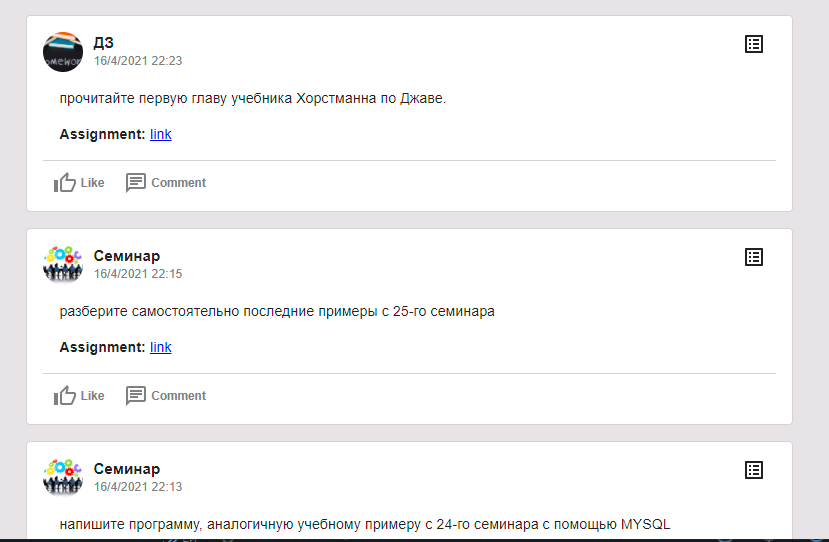
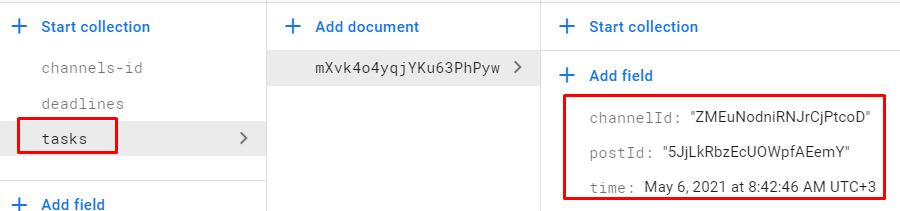




**Окно просмотра всех необязательных (без крайнего срока) заданий**

Компоненту необязательных заданий получаем из файла Tasks.js. Каждое задание представлено компонентой из файла TaskItem.js. В Tasks.js мы извлекаем данные при помощи модуля react-firebase-hooks/firestore мы извлекаем коллекцию tasks группы (рис ), которые содержат необходимые идентификаторы для представления задания, потом мы циклом отображаем эту коллекцию в виде компоненты из TaskItem.js, где извлекаем. Коллекцию tasks сортируем по неубыванию времени добавления. При создании необязательного задания внутри любого канала в коллекцию tasks группы добавляем новый элемент.





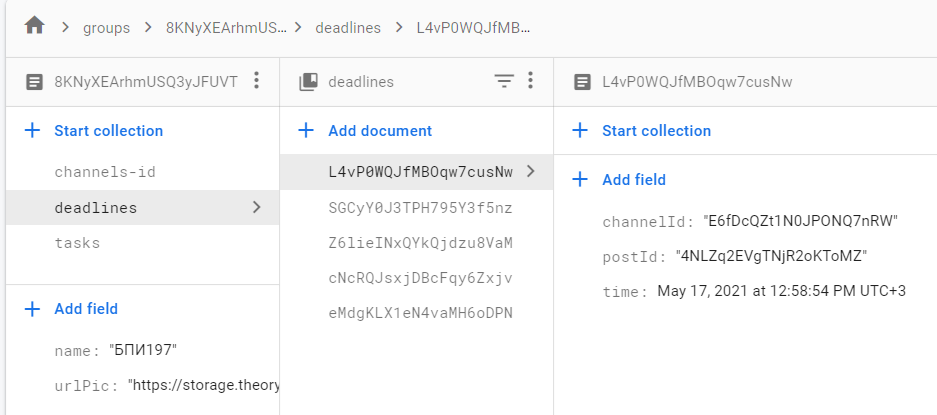
В окне содержится:

Лента блоков с описанием всех необязательных заданий по определенным предметам. Эти блоки содержат:

1. Картинку (аватар) канала, название канала, время создания необязательного задания.
2. Описание задания.
3. Ссылку на файл-описание задания.

**Окно просмотра всех обязательных (c крайним сроком) заданий**

Компоненту необязательных заданий получаем из файла Deadlines.js. Каждое задание представлено компонентой из файла TaskItem.js. В Tasks.js мы извлекаем данные при помощи модуля react-firebase-hooks/firestore мы извлекаем коллекцию tasks группы (рис ), которые содержат необходимые идентификаторы для представления задания, потом мы циклом отображаем эту коллекцию в виде компоненты из TaskItem.js, где извлекаем. Коллекцию tasks сортируем по неубыванию времени добавления. При создании необязательного задания внутри любого канала в коллекцию tasks группы добавляем новый элемент.

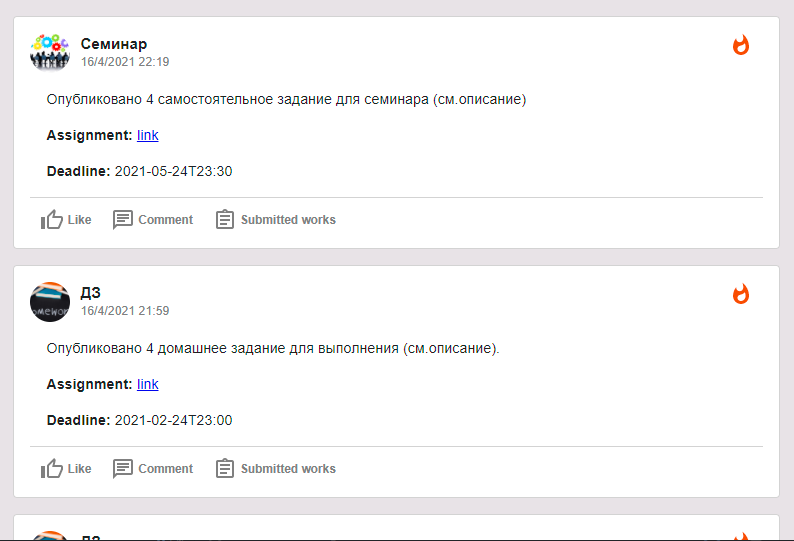


В окне содержится:

Лента блоков с описанием всех обязательных заданий по определенным предметам. Эти блоки содержат (рисунок №U):

1. Картинку (аватар) канала, название канала, время создания обязательного задания.
2. Описание задания.
3. Время крайнего срока.
4. Ссылку на просмотр и скачивание файл-описания задания.
5. Для создателя канала отображается кнопка (рисунок №, кнопка №) для перехода в окно, где хранится список студентов с их загруженными работами и оценками по этим работам.
6. Для обычного пользователя отображается кнопка (рисунок №, кнопка №) для открытия окна для загрузки работы.

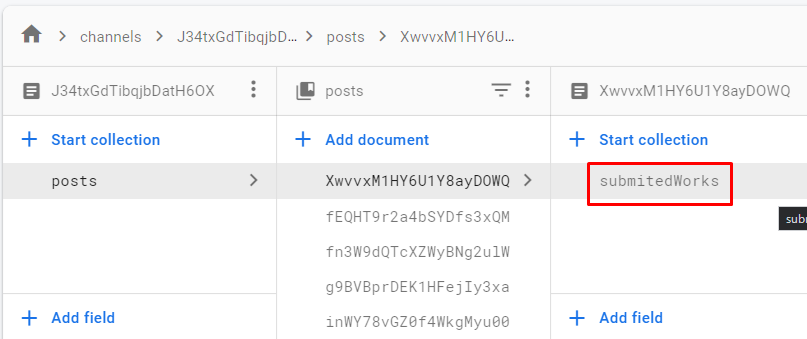
{са нужен еще один скриншот}



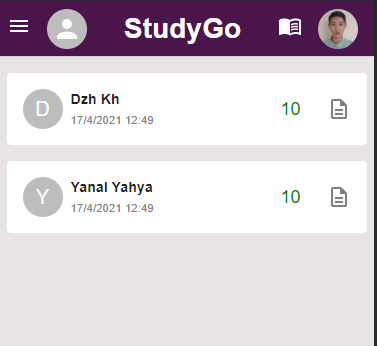
**Окно со списком отправленных выполненных работ от обучающихся**

{

Окно со списком отправленных выполненных работ от обучающихся получаем из файла Submits.js, где при помощи хука useCollection получаем submitedWorks (рисунок ). Элменты этой коллекции отображаем в виде компоненты из файла WorkItem.js.



}



Данное окно доступно для перехода из блока определенного обязательного задания, то есть преподаватель из блока с обязательным заданием просматривает список работ, отправленных ровно на это задание. Данное окно состоит из блоков, которые содержат:

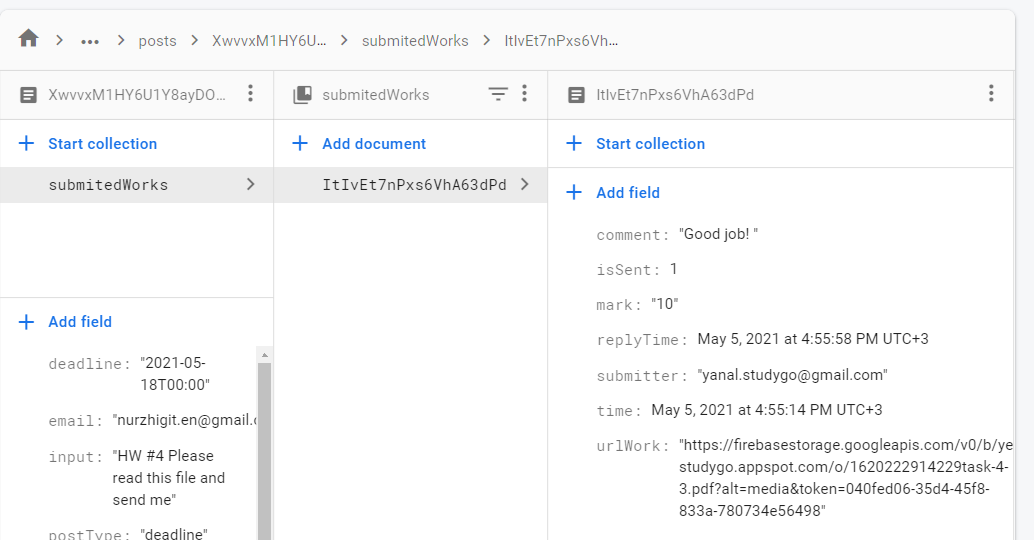
* Отправленный студентом файл с решением
* Инициалы студента, который отправил работу
* Дата и время отправки работы
* Оценка по работе
* Если оценки по работе нет, то присутствует кнопка для открытия окна выдачи обратной связи по работе

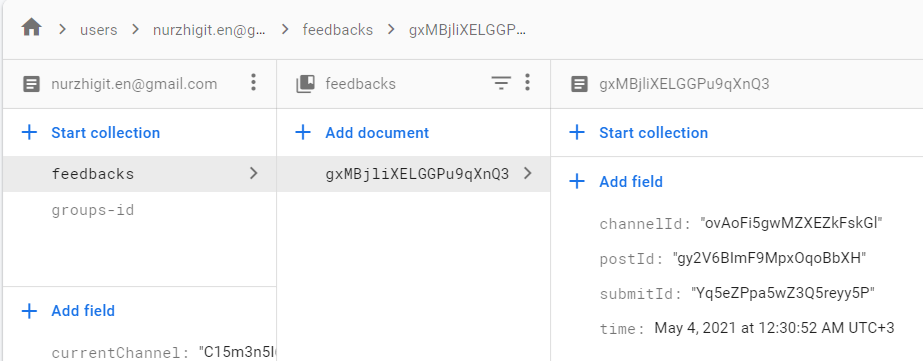
Преподаватель в этом окне имеет возможность оценивать и комментировать отправленные студентами работы. Для этой цели по нажатии на кнопку выдачи обратной связи откроется вспомогательное окно с полями ввода оценки, комментария по работе и кнопка для отправки обратной связи.

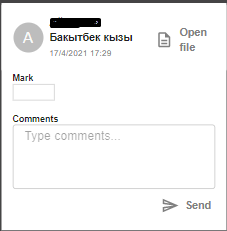
**Вспомогательное окно создания обратной связи**

{

Вспомогательное окно создания обратной связи получаем из файла WorkItemModal, получаем данные от пользователя и добавляем в коллекцию submitedWorks и добавляем к отправителю в коллекцию feedbacks необходимые данные.



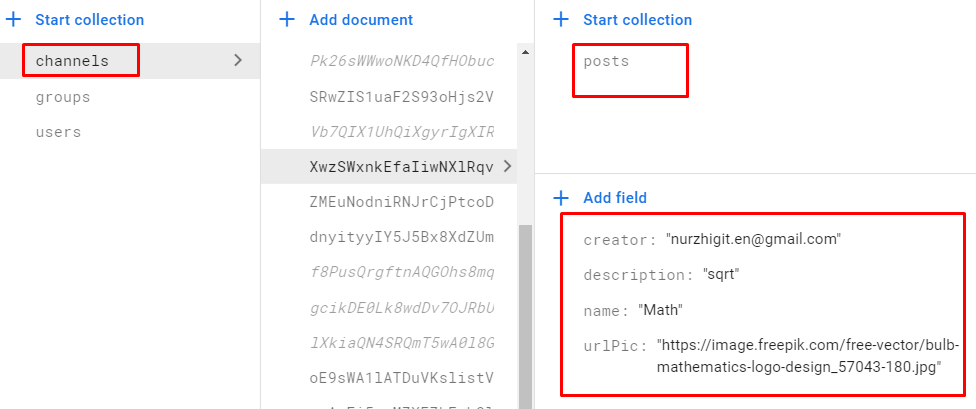
}

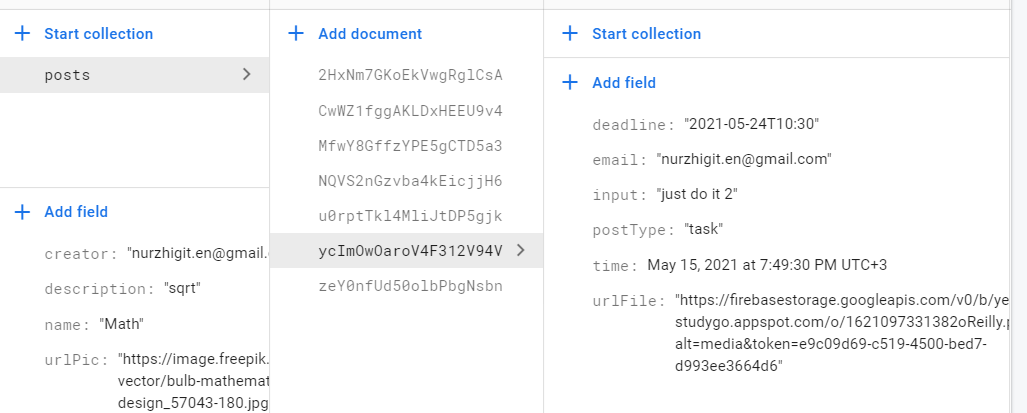


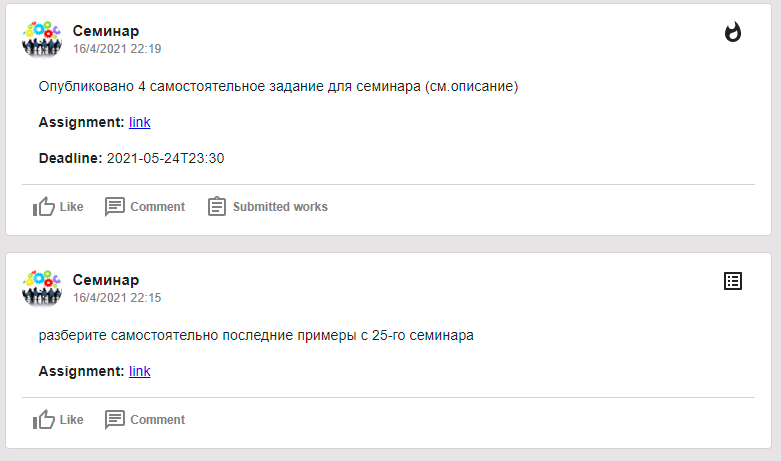
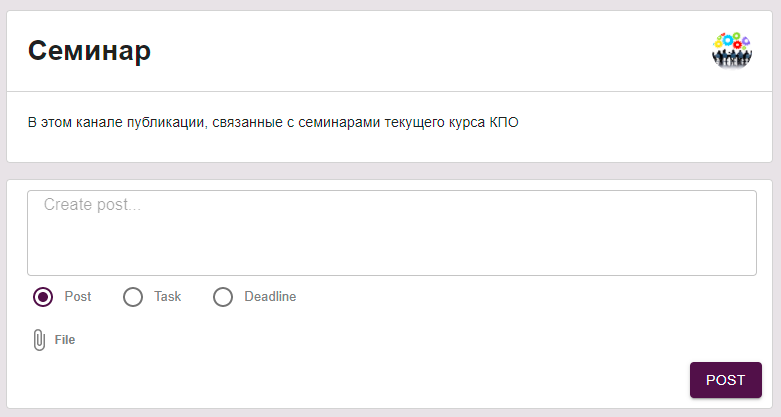
Для выдачи обратной связи имеется вспомогательное окно с полями ввода оценки, комментария по работе и с кнопкой для отправки обратной связи.

**Окно просмотра ленты конкретного канала**

Окно просмотра ленты получаем из файла ChannelPage.js, получаем пользователя хуком useAuthState, получаем идентификатор канала и группы из Redux при помощи хуков useSellector из модуля react-redux. Получаем по идентификатору канала элемент из коллекции channels. У этого элемента при помощи хука useCollection загружаем коллекцию posts. Циклом по элементам (рисунок ) коллекции posts отображаем элементы в виде компонент из файла PostItem.js.



{вставить картину структуры}



В окне содержится:

Лента блоков с учебными объявлениями и обязательными, необязательными заданиями. Также в ленте канала для создателя канала отображается блок с возможностью публикации конкретного сообщения.

Блок объявления содержит:

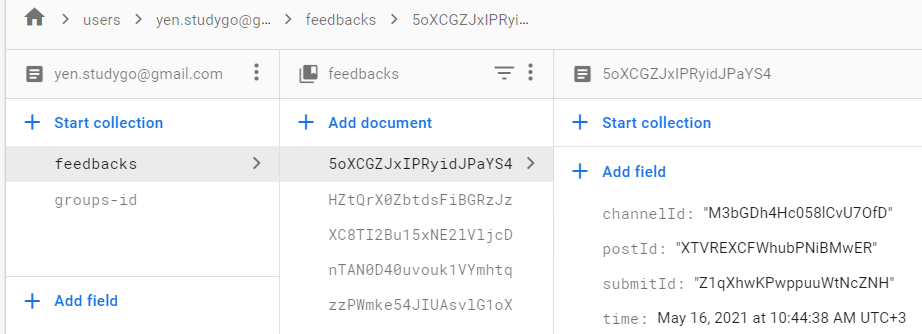
* + - 1. Картинку (аватар) канала, название канала, время создания необязательного задания.
      2. Описание задания.
      3. Ссылку на файл-описание задания.

Блок публикации сообщения:

* + - 1. Блок ввода текстовой информации;
      2. Кнопки выбора типа сообщения: объявление, необязательное и обязательное задание;
      3. Кнопка прикрепления файла для задания;
      4. Кнопка для того, чтобы опубликовать сообщение.

**Окно пользователя для просмотра списка оценок и комментариев от преподавателей:**

Окно оценок и комментариев от преподавателей получаем из файла Feedbacks.js при помози Redux получаем элемент из коллекции пользователей (рисунок ) и из него извлекаем коллекцию feedbacks. Элементы этой коллекции отображаем в виде компоненту из файла FeedbackItem.js, где извлекаем данные при помощи useDocument.

{}

В окне содержится:

Лента блоков с обратной связью от преподавателя по выполненным работам.

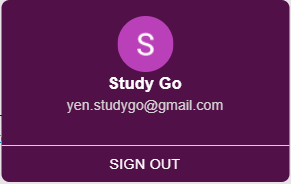
Блоки с обратной связью содержат:

* + - 1. Картинку (аватар) канала, название канала, время создания необязательного задания.
      2. Текстовое описание задания.
      3. Ссылку на файл-описание задания.
      4. Ссылку на файл-решение задания.
      5. Комментарий от преподавателя.
      6. Оценка от преподавателя.

**Выпадающее окно выхода из аккаунта**

По нажатии на кнопку личного аккаунта (рисунок №, кнопка №) из панели приложения показывается выпадающее окно (рисунок №), через которое можно выйти из аккаунта.

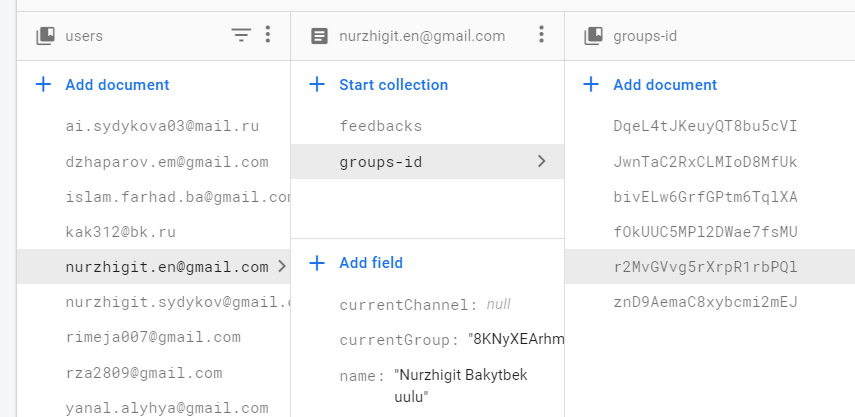
При нажатии на кнопку SIGN OUT, выходим из аккаунта при помощи объекта auth из файла firebase.js в файле UserSettingSidebar.js.



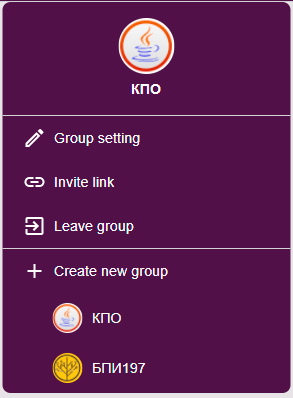
**Выпадающее окно управления группой**

[

При помощи Redux и хуков получаем текущую группу из коллекции groups. При нажатии на кнопку Leave group удаляем элемент из коллекции groups-id пользователя (рисунок )

При нажатии на иконку или название определенной группы мы меняем сосотояние в Redux и в поле currentGroup у пользователя.

]

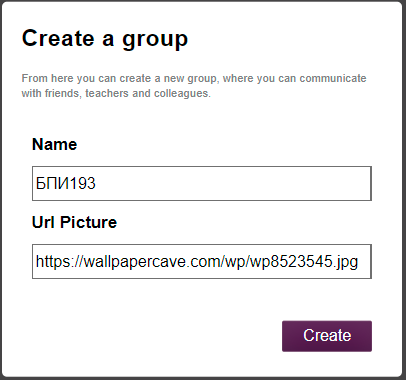


По нажатии на кнопку управления группой (Рисунок №, кнопка №) показывается выпадающее окно (рисунок №), в котором:

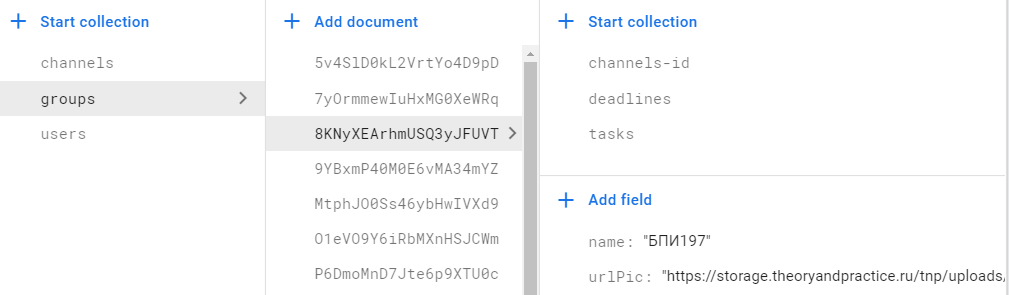
* Можно изменить базовые настройки группы (название и аватар) (рисунок №, пункт №)
* Можно скопировать ссылку - приглашение в группу (рисунок №, пункт №)
* Можно покинуть группу (рисунок №, пункт №)
* Можно создать новую группу (рисунок №, пункт №).
* Можно перейти к другой существующей группе (рисунок №, пункт №)

**Вспомогательное окно создания группы (рисунок №) содержит:**

**[**



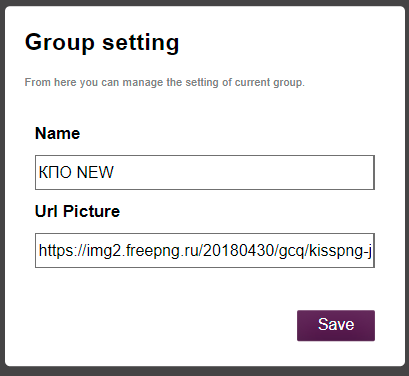
Вспомогательное окно создания группы получаем из файла CreateGroupModal.js. Получаем данные от пользователя и создаем группу в коллекции groups (рисунок ).



**]**

* Поле ввода названия создаваемой группы
* Поле ввода URL-ссылки на картинку (аватарку) создаваемой группы (опционально)
* Кнопка создания группы

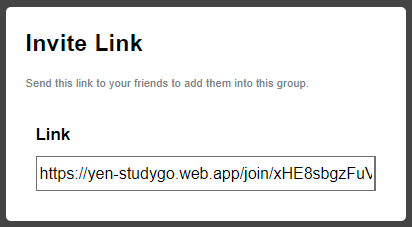
**Вспомогательное окно для редактирования настройки группы.**



Вспомогательное окно для редактирования настройки группы получаем из файла EditGroupSettingModal, где мы получаем новые данные и обновляем элемент в коллекции groups.

**Вспомогательное окно для получения ссылки для присоединения к группе.**

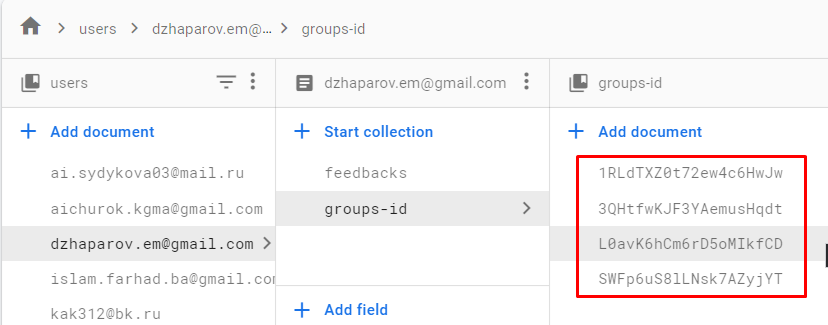
Берется идентификатор текущей группы и перед ним добавляется домен приложения и “/join/”.



**Окно присоединения к группе**

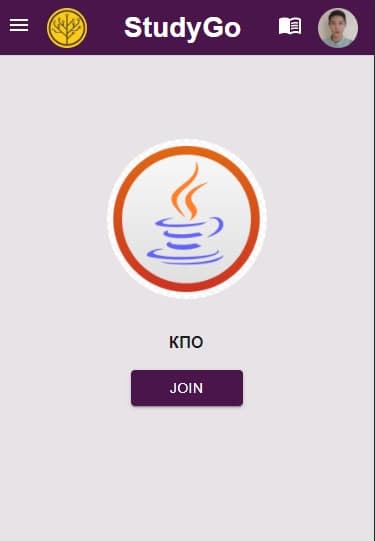
[

Окно присоединения к группе получаем из файла JoinPage.js. Получаем из адресной строки идентификатор группы и добавляем к коллекции groups-id (рисунок ).



]

При переходе по ссылке - приглашению авторизованному пользователю открывается окно с кнопкой, по нажатии на которую пользователь добавится в группу. Если пользователь не был ранее авторизован, то в самом начале ему откроется окно авторизации.

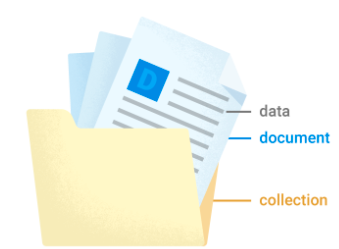


### Описание структуры онлайн базы данных Cloud Firestore

~~База данных в этом приложении представлена в виде нереляционной базы данных (NOSQL), где объекты хранятся не в таблице, а в коллекции JSON объекты, и каждый объект имеет уникальный идентификатор.~~

В текущей работе использовалась нереляционная база данных Cloud Firestore. Настоящая база данных выделяется документно-ориентированным подходом в хранении информации, то есть от реляционных баз данных она отличается тем, что в ней нет таблиц или строк. Данные хранятся в документах, которые, в свою очередь, распределены в коллекции.

Каждый документ содержит пары из ключа и значения. В каждом файле информация хранится в JSON-формате. Каждый документ содержится в коллекциях. Документ может содержать внутри себя вложенные коллекции и вложенные объекты, которые могут иметь в себе поля, строки или объекты, например списки.

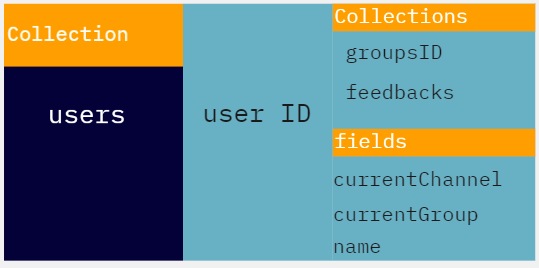


**КОЛЛЕКЦИЯ ПОЛЬЗОВАТЕЙ**

Коллекция пользователей — это набор пользователей, каждый из которых представлен в виде пары уникальный идентификатор пользователя и данные, ассоциированные с пользователем.

**ПОЛЬЗОВАТЕЛЬ**

Главный объект в базе данных — это пользователь, у которого имеются свои поля и коллекции.



Поля пользователя:

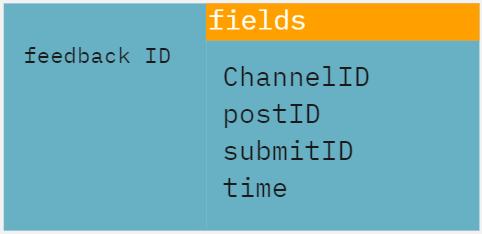
* Текущая группа, с которой работает пользователь в данный момент. Группа представлена в виде идентификатора — это сделано для того, чтобы запомнить последнюю группу, который посетил пользователь, то есть реализовано сохранение последнего сеанса.
* Текущий канал, с содержанием которого работает пользователь в данный момент. Канал представлен в виде идентификатора — это сделано для того, чтобы запомнить последний канал, который посетил пользователь.
* Имя пользователя.

Коллекции пользователя:

* Группы, в которых состоит пользователь. Представлены в виде уникальных идентификаторов, через которые можно получить доступ к соответствующим объектам - группам.

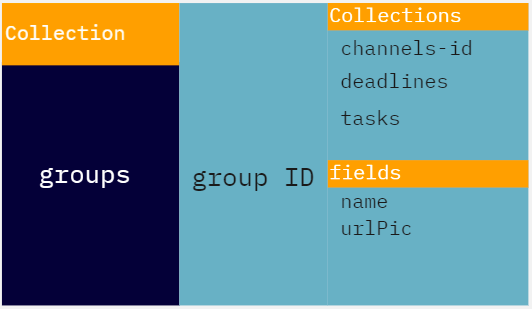


* Обратные связи (оценки, комментарии) от преподавателей представлены в виде идентификаторов. Внутри каждого объекта - обратной связи содержатся:
  + - Идентификатор канала, к которому относится эта обратная связь. С помощью идентификатора канала происходит обращение к коллекции каналов для получения самого объекта канала.
    - Идентификатор обязательного задания, на которое высылали выполненную работу, к которой была создана исходная обратная связь.
    - Идентификатор высланной выполненной работы. С помощью идентификаторов обязательного задания, канала и отправленной работы можно получить саму отправленную выполненную работу.
    - Время создания обратной связи. С помощью этого параметра происходит сортировка всех объектов - обратных связей.



**КОЛЛЕКЦИЯ ГРУПП**

Коллекция групп - то набор объектов - групп, каждый из которых представлен в виде пары уникальный идентификатор пользователя и данные, ассоциированные с группами.



**ОБЪЕКТ - ГРУППА**

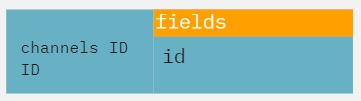
Объект-группа имеет следующие поля и коллекции:

Поля группы:

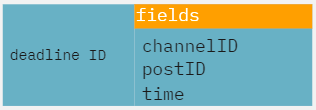
* Имя группы.
* URL-ссылка на картинку группы (аватар).

Коллекции группы:

* Каналы, которые находятся в этой группе. Представлены в виде уникальных идентификаторов, через которые можно получить доступ к соответствующим объектам - каналам.



* Обязательные задания от преподавателей представлены в виде идентификаторов. Внутри каждого объекта содержатся:
  + - Идентификатор канала, к которому относится это обязательное задание. С помощью идентификатора канала происходит обращение к коллекции каналов для получения самого объекта канала.
    - Идентификатор обязательного задания.
    - Время создания обязательного задания. С помощью этого параметра происходит сортировка всех объектов - обязательных заданий.



Имеется три главных узла:

1. Users -каждый подузел представлен в виде сериализованного Json объекта класса User (рисунок 16). Это узел для хранения данных о пользователях.

Таблица 1 – Описание структуры узла Users.

|  |  |
| --- | --- |
| Название | Назанчение |
| Id | Идентификатор |
| Name | Имя |
| TeamsId | Команды пользователя |
| Token | Токен для быстрых запросов |

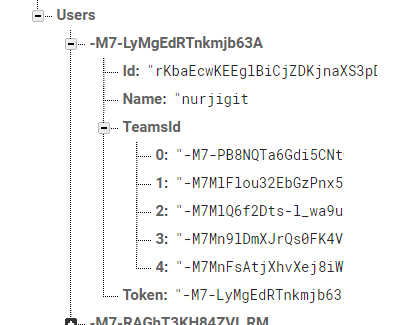


Рисунок 16 – Структура узла Users

1. CheckLists - каждый подузел представлен в виде сериализованного Json объекта класса CheckList (рисунок 17). Это узел для хранения чек-листов.

Таблица 2 – Описание узла CheckLists

|  |  |
| --- | --- |
| Название | Назначение |
| Date | Дата |
| Id | Идентификатор |
| Tasks | Текст задания и его результат |
| Time | Время для выполнения |
| Title | Заголовок |

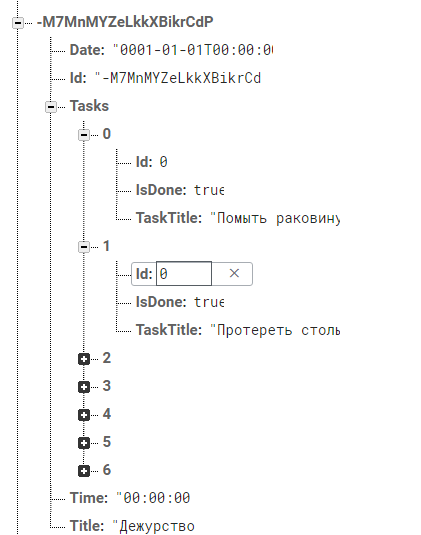


Рисунок 17 – Структура узла CheckLists

1. Teams - каждый подузел представлен в виде сериализованного Json объекта класса Team (рисунок 18). Это узел для хранения команд.

Таблица 3 – Описание узла Teams

|  |  |
| --- | --- |
| Название | Назначение |
| CheckListsId | Идентификаторы чек-листов команды |
| Id | Идентификтор |
| MembersTokens | Токены пользователей |
| Name | Название команды |

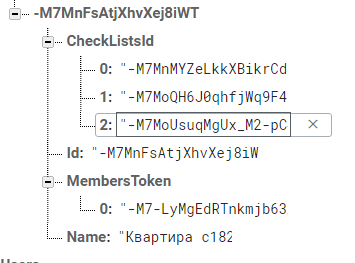


Рисунок 18 – Структура Teams

### Диаграмма классов

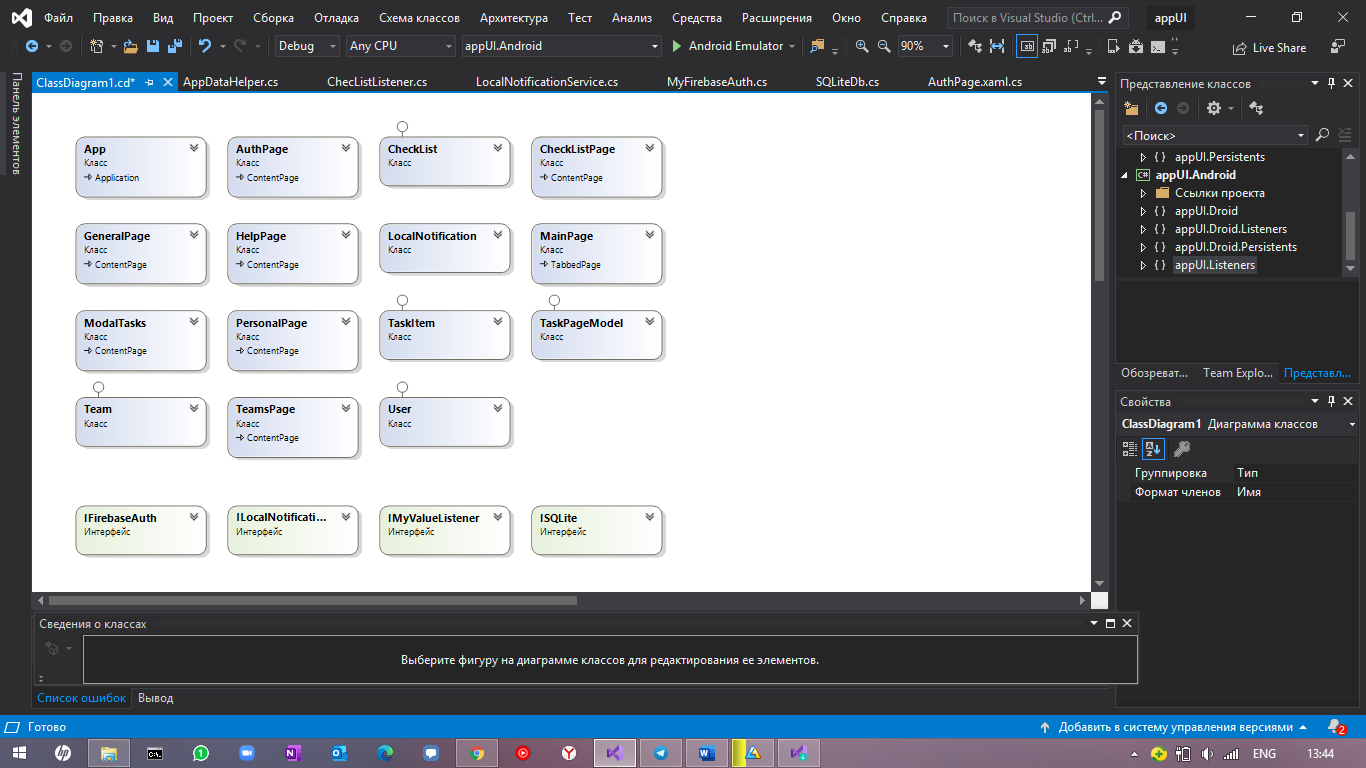


Рисунок 19 – Диаграмма классов

На данном рисунке приведена диаграмма классов Portable Class Library проекта Xamarin Forms. Описание классов приведены в Приложении 1,2,3.

## Описание и обоснование выбора метода организации входных и выходных данных

### Описание метода организации входных и выходных данных

Входными данными являются действия пользователей (нажатия на кнопки), тексты, получаемые из полей ввода и информации, получаемы от СУБД (Firebase Realtime Database) и локального файла базы данных SQLite. Общая схема на рисунке 20. Связи яввяются двухсторонними.

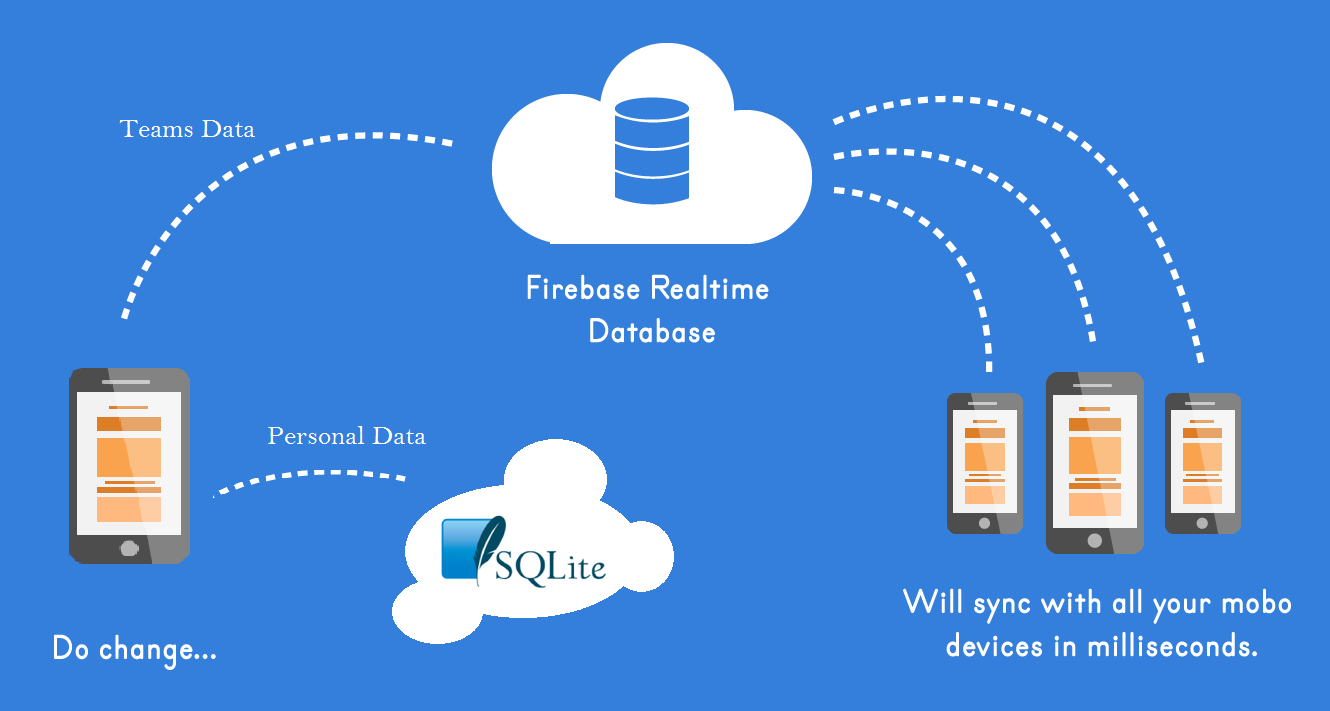


Рисунок 20 – Схема обмена данными

Выходными данными являются файл базы данных SQLite, хранящийся в локальном устройстве пользователя, а также информации, отправляемые в СУБД (Firebase Realtime Database).

### Обоснование выбора метода организации входных и выходных данных

База данных SQLite выбран из следующих соображений:

1. Файловая структура – все данные хранятся в единственном файле.
2. Кроссплатформенность – отлично подходит для проектов Xamarin Forms.
3. Простота и скорость использования базы данных.

Облачная СУБД Firebase Realtime Database выбран из следующих соображений:

1. Позволяет хранить и синхронизировать данные пользователей в режиме реального времени.
2. Данные имеют вид Json – позволяет быстро отправлять и получать информацию.
3. Нет необходимости создания сервера.
4. Надежная пользовательская безопасность.

## Описание и обоснование выбора состава технических и программных средств

### Состав технических и программных средств

Для функционирования программы требуется смартфон с интернетом со следующими характеристиками:

1. Количество ядер - 4
2. Частота работы процессора – 1.4 ГГц
3. Объем оперативной памяти - 2 ГБ
4. Объем встроенной памяти - 16 ГБ
5. ОС Android 10.0 и выше.

# Ожидаемые технико-экономические показатели

## Предполагаемая потребность

Приложение в первую очередь ориентировано студентам. Благодаря данной программе будут сохранены благоприятная обстановка и хорошие взаимоотношения между студентами, что необходимо для благополучного проживания в общежитии.

## Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами

На рынке Play Market существуют много различных аналогов, самые яркие из них:

1. “Trello: Organize anything with anyone, anywhere!”. Trello, Inc.
2. “Focus To-Do: Pomodoro Timer & To Do List”. Pomodoro Timer & To Do List – SuperElement Soft.

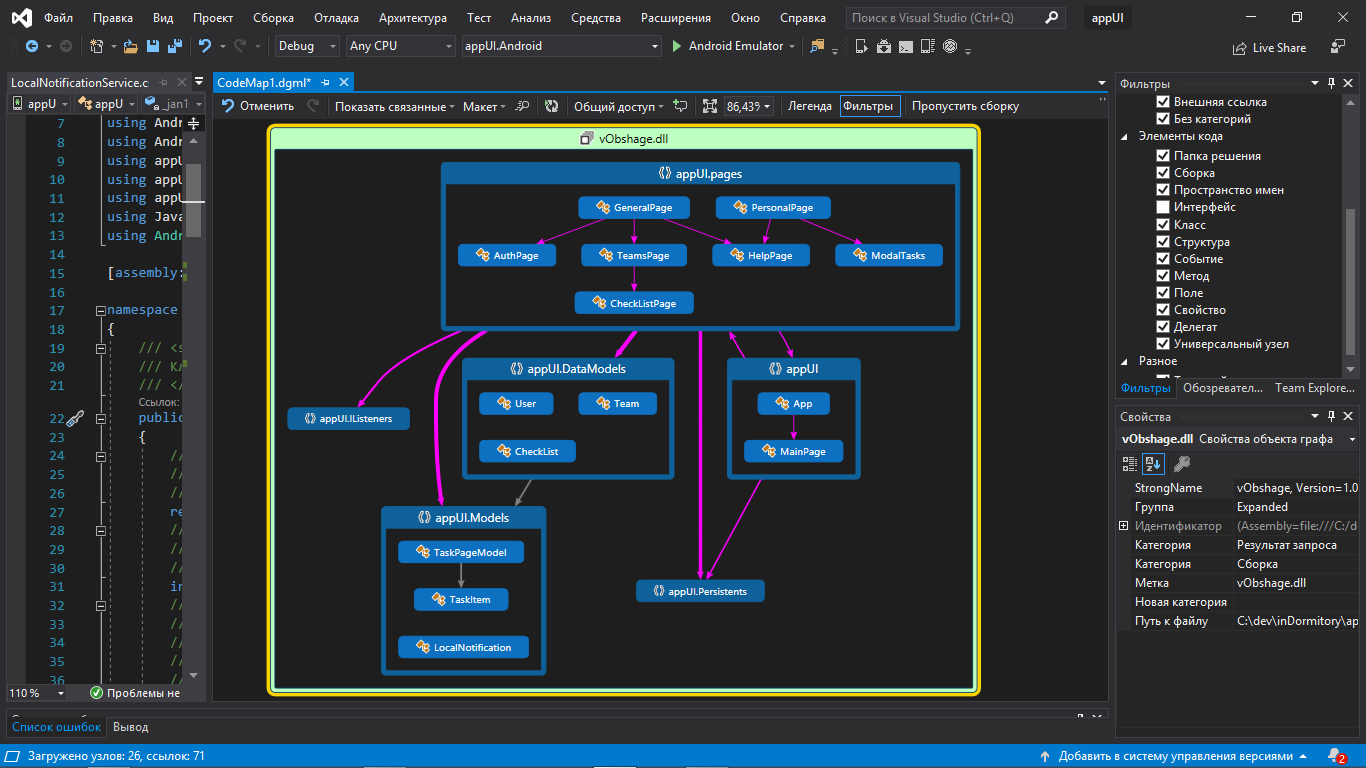
Преимуществами программы являются:

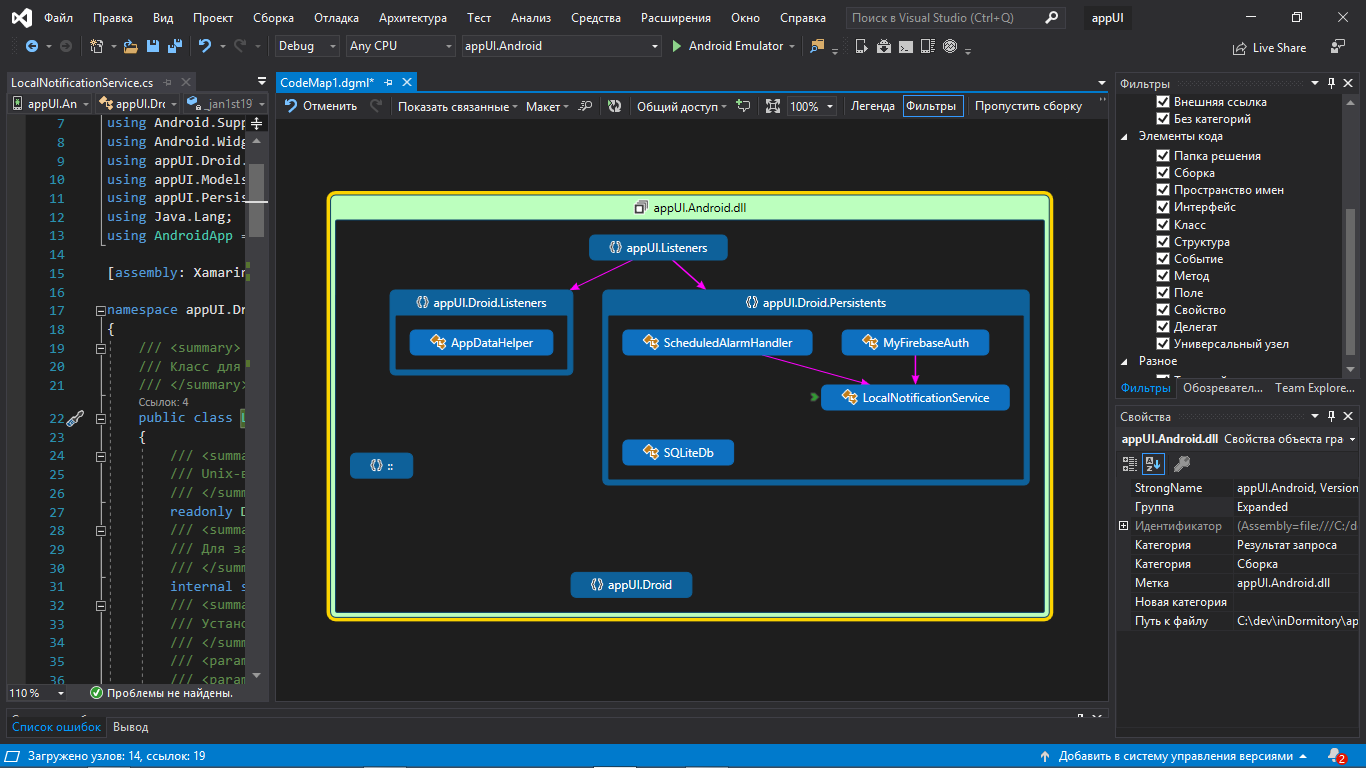
1. Является бесплатным.
2. Удобный интерфейс.
3. Возможность работы с другими пользователями.
4. Не требует денежных затрат.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19.101-77 Виды программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. Создание мобильных приложений с помощью Xamarin.Forms (книга) [Электронный ресурс]// URL: <https://docs.microsoft.com/ru-ru/xamarin/xamarin-forms/creating-mobile-apps-xamarin-forms/> (Дата обращения 8.4.2020, режим доступа: свободный).
5. How To Send Local Notification With A Repeat Interval (Day/Minute/Hour etc.) [Электронный ресурс]// URL: <https://www.c-sharpcorner.com/article/how-to-send-local-notification-with-repeat-interval-in-xamarin-forms/> (Дата обращения 28.4.2020, режим доступа: свободный).
6. How to make a Floating Action Button in Xamarin Forms [Электронный ресурс]// URL: <https://stackoverflow.com/questions/54517874/how-to-make-a-floating-action-button-in-xamarin-forms> (Дата обращения 15.4.2020, режим доступа: свободный).
7. How can I make my app send out notifications when it's loaded but not running in the foreground? [Электронный ресурс]// URL: <https://stackoverflow.com/questions/59622443/how-can-i-make-my-app-send-out-notifications-when-its-loaded-but-not-running-in> (Дата обращения 17.4.2020, режим доступа: свободный).
8. Vertically expand ListView inside an AbsoluteLayout [Электронный ресурс]// URL: <https://forums.xamarin.com/discussion/37897/vertically-expand-listview-inside-an-absolutelayout> (Дата обращения 22.4.2020, режим доступа: свободный).
9. Toast equivalent for Xamarin Forms [Электронный ресурс]// URL: <https://stackoverflow.com/questions/35279403/toast-equivalent-for-xamarin-forms> (Дата обращения 23.4.2020, режим доступа: свободный).
10. Firebase Cloud Messaging [Электронный ресурс]// URL: <https://docs.microsoft.com/ru-ru/xamarin/android/data-cloud/google-messaging/firebase-cloud-messaging> (Дата обращения 3.5.2020, режим доступа: свободный).
11. Firebase.Xamarin [Электронный ресурс]// URL: <https://github.com/rlamasb/Firebase.Xamarin/blob/master/README.md> (Дата обращения 6.5.2020, режим доступа: свободный).
12. Xamarin.Forms - Working with Firebase Realtime Database CRUD Operations [Электронный ресурс]// URL: <https://xamarinmonkeys.blogspot.com/2019/01/xamarinforms-working-with-firebase.html> (Дата обращения 9.5.2020, режим доступа: свободный).
13. Firebase Authentication Password Hashing [Электронный ресурс]// URL: <https://firebaseopensource.com/projects/firebase/scrypt/> (Дата обращения 11.5.2020, режим доступа: свободный).

# ПРИЛОЖЕНИЕ 1 КАРТА КОДА





# ПРИЛОЖЕНИЕ 2 ОПИСАНИЕ КЛАССОВ

Таблица П1 - Описание и функциональное назначение классов

|  |  |
| --- | --- |
| Класс | Назначение |
| CheckList | Модель чек-листа. |
| Team | Модель команды. |
| User | Модель пользователя. |
| LocalNotification | Класс для локальных уведомлений. |
| TaskItem | Модель задачи. |
| TaskPageModel | Модель страницы ModalTasks. |
| AuthPage | Страница для регистрации\авторизации пользователя. |
| CheckListPage | Страницы командного чек-листа. |
| GeneralPage | Страница для работы с командами. |
| HelpPage | Страница с подсказками. |
| ModalTasks | Страница персонального чек-листа. |
| PersonalPage | Страница с персональными чек-листами. |
| TeamsPage | Страница с чек-листами. |
| AppDataHelper | Класс для работы с базой данных Firebase Realtime Database. |
| CheckListListener | Класс для получения изменений на подписанный узел. |
| LocalNotificationService | Класс для оповещения пользователя. |
| MyFirebaseAuth | Класс для регистрации\авторизации пользователей. |
| SQLiteDb | Класс для получения доступа к локальной базе данных. |

# ПРИЛОЖЕНИЕ 3 ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ПОЛЕЙ МЕТОДОВ И СВОЙСТВ

Таблица П2 - Описание полей, методов и свойств класса CheckList.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| PropertyChangedEventHandler | public | event | Оповещает об изменении свойств. | |
| resourseName | public static readonly | string | Название узла в онлайн базе данных Firebase database. | |
| title | private | string | Заголовок чек-листа. | |
| id | private | string | Идентификатор чек-листа. | |
| date | private | DateTime | Дата чек-лиса. | |
| time | private | TimeSpan | Время чек-листа. | |
| Свойства | | | | |
| Имя | Модификатор | Тип | Доступ | Назначение |
| Title | public | string | get, set | Свойство доступа к title. Оповещает об изменениях. |
| Id | public | string | get, set | Свойство доступа к id. Оповещает об изменениях. |
| Date | public | DateTime | get, set | Свойство доступа к date. Оповещает об изменениях. |
| Time | public | TimeSpan | get, set | Свойство доступа к time. Оповещает об изменениях. |
| Tasks | public | ObservableCollection<TaskItem> | get, set | Задачи чек-лиcта. Оповещает об изменениях. |
| IsRemind | public | bool | get, set; | Свойство доступа к isRemind. Оповещает об изменениях. |
| RemindIcon | public | string | get | Возвращает картинку в зависимости от состояния уведомлений. |
| NotificfId | public | int? | get, set | Идентификатор уведомлений. |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| SetProperty | protected | void | string name | Оповещение об изменении определенного свойства. |

Таблица П3 - Описание полей, методов и свойств класса Team

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| resourseName | public static readonly | string | Название узла в онлайн базе данных Firebase database. | |
| id | private | string | Идентификатор команды | |
| name | private | string | Имя команды | |
| PropertyChanged | public PropertyChangedEventHandler | event | Оповещает об изменении свойств. Оповещает об изменениях. | |
| Свойства | | | | |
| Имя | Модификатор | Тип | Доступ | Назначение |
| MembersToken | public | List<string> | get, set | Идентификатор-токены членов команды. |
| CheckListsId | public | List<string> | get, set | Идентификатор чек-листов команды. |
| Name | public | string | get, set | Свойство для доступа name. Оповещает об изменениях. |
| Id | public | string | get, set | Свойство для доступа к id. Оповещает об изменениях. |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| SetProperty | protected | void | string name | Оповещение об изменении определенного свойства. |

Таблица П4 - Описание полей, методов и свойств класса User

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Модификатор доступа | Тип | Назначение |
| name | private | string | Имя пользователя |
| id | private | string | Идентификатор пользователя |

Продолжение таблицы П4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PropertyChanged | public PropertyChangedEventHandler | event | Оповещает об изменении свойств. Оповещает об изменениях. | |
| Свойства | | | | |
| Name | public | string | get, set | Свойство для доступа name. Оповещает об изменениях. |
| Id | public | string | get, set | Свойство для доступа к id. Оповещает об изменениях. |
| Token | public | string | get, set | Токен пользователя. |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| SetProperty | protected | void | string name | Оповещение об изменении определенного свойства. |

Таблица П5 - Описание полей, методов и свойств класса AuthPage

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Модификатор доступа | Тип | Назначение |
| auth | private readonly | IFirebaseAuth | Интерфейс для взаимодейтвия с Firebase Authentication SDK. |
| userPath | private readonly | string | Узел пользователей в базе данных Firebase Realtime Database. |
| notificationManager | private readonly | ILocalNotificationService | Интерфейс для оповещений. |
| fbUrl | private readonly | string | Ссылка для доступа к базе данных. |
| firebase | private readonly | FirebaseClient | Предоставляет доступ к онлайн базе данных. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| AuthPage | public | Конструктор | - | Установка начальных значений |
| LoginBtn\_Clicked | private async | void | object sender, EventArgs e | Вход и регистрация в системе Firebase Authentication SDK. |
| NewAccountBtn\_Clicked | private | void | object sender, EventArgs e | Меняет функции авторизации\входа. |

Таблица П6 - Описание полей, методов и свойств класса CheckListPage

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Модификатор доступа | Тип | Назначение |
| checListsPath | private readonly | string | Узел пользователей в базе данных Firebase Realtime Database. |
| notification | private readonly | ILocalNotificationService | Интерфейс для оповещений. |
| fbUrl | private readonly | string | Ссылка для доступа к базе данных. |
| firebase | private readonly | FirebaseClient | Предоставляет доступ к онлайн базе данных. |
| CurrentCheckList | private | CheckList | Модель страницы. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| CheckListPage | public | Конструктор | CheckList checkList | Установка начальных значений |
| OnAppearing | protected override | void | - | Подписка на события. |
| SaveCurrentCheckList | private async | void | - | Сохраняет привязанный чек-лист в базе данных. |
| PlusFab\_Clicked | private async | void | object sender, EventArgs e | Добавляет задачу в чек-лист. |
| Input | private async | Task<string> | string title, string message, string placeholder, string accept, string initialValue | Метод для ввода строки. |
| TasksList\_ItemSelected | private | void | object sender, SelectedItemChangedEventArgs e | Метод срабатывает, когда происходит нажатие на элемент списка. Нужно, чтобы цветом не выделялся выбранный элемент. |
| TasksList\_ItemTapped | private | void | object sender, ItemTappedEventArgs e | Метод срабатывает, когда происходит нажатие на элемент списка. Отмечает, что задание выполнено \ не выполнено. |

Продолжение таблицы П6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EditTask\_Clicked | private async | void | object sender, EventArgs e | Метод для редактирования текста задачи. |
| DeleteTask\_Clicked | private async | void | object sender, EventArgs e | Метод для удаления задачи из листа. |
| TasksList\_Refreshing | private async | void | object sender, EventArgs e | Обновление листа. |
| MyDatePicker\_DateSelected | private | void | object sender, DateChangedEventArgs e | Срабатывает после выбора даты. |
| MyTimePicker\_Unfocused | private | void | object sender, FocusEventArgs e | Срабатывает при закрытии окно выбора времени. |

Таблица П7 - Описание полей, методов и свойств класса GeneralPage

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Модификатор доступа | Тип | Назначение |
| auth | private readonly | IFirebaseAuth | Интерфейс для взаимодейтвия с Firebase Authentication SDK. |
| userPath | private readonly | string | Узел пользователей в базе данных Firebase Realtime Database. |
| notification | private readonly | ILocalNotificationService | Интерфейс для оповещений. |
| fbUrl | private readonly | string | Ссылка для доступа к базе данных. |
| firebase | private readonly | FirebaseClient | Предоставляет доступ к онлайн базе данных. |
| teamPath | private readonly | string | Название узла команд в онлайн базе данных. |
| CurrentUser | private | User | Текущий пользователь. |
| Teams | private | ObservableCollection<Team> | Команды пользователя. |

Продолжение таблицы П7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| Input | private async | Task<string> | string title, string message, string placeholder, string accept, string initialValue | Метод для ввода строки. |
| OnAppearing | protected async override | void | - | Задание "тяжелых" начальных значений. |
| GetTeamsAsync | private async | Task<Team[]> | (List<string> ids | Получение команд из онлайн базы данных. |
| PlusFab\_Clicked | private async | void | object sender, EventArgs e | Добавление\создание команд. |
| GeneralPage | public | Конструктор | - | Задание начальных значений. |
| LogOut\_Clicked | private | void | object sender, EventArgs e | Выход из системы авторизации\регистрации. |
| MainList\_ItemSelected | private | void | object sender, SelectedItemChangedEventArgs e | Метод срабатывает, когда происходит нажатие на элемент списка. Нужно, чтобы цветом не выделялся выбранный элемент. |
| MainList\_ItemTapped | private async | void | object sender, ItemTappedEventArgs e | Метод срабатывает, когда происходит нажатие на элемент списка. Отмечает, что задание выполнено \ не выполнено. |
| TeamEdit\_Clicked | private async | void | object sender, EventArgs e | Метод для редактирования названия команды. |
| TeamDelete\_Clicked | private async | void | object sender, EventArgs e | Метод для удаления команды. |
| CopyId\_Clicked | private async | void | object sender, EventArgs e | Копирование идентификатора команды. |
| HelpTb\_Clicked | private | void |  | Показ окна с подсказками. |

Таблица П8 Описание полей, методов и свойств класса HelpPage

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| HelpPage | public | Конструктор | - | Установка начальных значений |

Таблица П9 - Описание полей, методов и свойств класса TeamsPage

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Модификатор доступа | Тип | Назначение |
| auth | private readonly | IFirebaseAuth | Интерфейс для взаимодейтвия с Firebase Authentication SDK. |
| userPath | private readonly | string | Узел пользователей в базе данных Firebase Realtime Database. |
| notification | private readonly | ILocalNotificationService | Интерфейс для оповещений. |
| fbUrl | private readonly | string | Ссылка для доступа к базе данных. |
| firebase | private readonly | FirebaseClient | Предоставляет доступ к онлайн базе данных. |
| teamPath | private readonly | string | Название узла команд в онлайн базе данных. |
| CurrentUser | private | User | Текущий пользователь. |
| CheckLists | private | ObservableCollection<CheckList> | Чек-листы команды. |
| CurrentTeam | private | Team | Текущая команда. |
| checListsPath | private readonly | string | Название узла команд в онлайн базе данных. |
| checListsListener | private readonly | IMyValueListener | Интерфейс для определения изменения данных. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| Input | private async | Task<string> | string title, string message, string placeholder, string accept, string initialValue | Метод для ввода строки. |

Продолжение таблицы П9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OnAppearing | protected async override | void | - | Задание "тяжелых" начальных значений. |
| GetListsAsync | private async | Task<CheckList[]> | - | Получение чек-листов из онлайн базы данных по их идентификаторам. |
| PlusFab\_Clicked | private async | void | object sender, EventArgs e | Добавление нового чек-листа. |
| MainList\_ItemSelected | private | void | object sender, SelectedItemChangedEventArgs e | Метод срабатывает, когда происходит нажатие на элемент списка. Нужно, чтобы цветом не выделялся выбранный элемент. |
| MainList\_ItemTapped | private async | void | object sender, ItemTappedEventArgs e | Показ страницы с чек-листом. |
| EditCheckList\_Clicked | private async | void | object sender, EventArgs e | Редактирование названия чек-листа. |
| DeleteCheckList\_Clicked | private async | void | object sender, EventArgs e | Удаление чек-листа. |
| MainList\_Refreshing | private async | void | object sender, EventArgs e | Обновление чек-листов. |
| CopyId\_Clicked | private async | void | object sender, EventArgs e | Копирование идентификтора команды. |

Таблица П10 - Описание полей, методов и свойств класса ModalTasks

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Модификатор доступа | Тип | Назначение |
| connection | private readonly | SQLiteAsyncConnection | Предостовляет асинхронное соединение с локальной базой данных. |
| notification | private readonly | ILocalNotificationService | Интерфейс для оповещений. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Свойства | | | | |
| Model | public | TaskPageModel | get, private set | Модель страницы |
| Id | public | string | get, set | Свойство для доступа к id. Оповещает об изменениях. |

Продолжение таблицы П10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| ModalTasks | public | Конструктор | TaskPageModel pageModel, SQLiteAsyncConnection connection | Задание начальных значений. |
| OnAppearing | protected override | void | - | Задание "тяжелых" начальных значений. |
| TasksList\_ItemTapped | private async | void | object sender, ItemTappedEventArgs e | Отметка\снятие галочки о выполнении. |
| Edit\_Clicked | async private | void | object sender, EventArgs e | Метод для редактирования текста задачи. |
| Delete\_Clicked | private async | void | object sender, EventArgs e | Метод для удаления задачи из листа. |
| Fabplus\_Clicked | private async | void | object sender, EventArgs e | Добавляет задачу в чек-лист. |
| RemaindTb\_Clicked | private async | void | object sender, EventArgs e | Включение\отключение уведомлений. |
| StarTb\_Clicked | private async | void | object sender, EventArgs e | Устанавливает приоритетность чек-листа. |
| TasksList\_ItemSelected | private | void | object sender, SelectedItemChangedEventArgs e | Метод срабатывает, когда происходит нажатие на элемент списка. Нужно, чтобы цветом не выделялся выбранный элемент. |
| CopyId\_Clicked | private async | void | object sender, EventArgs e | Копирует текст задания. |

Таблица П11 - Описание полей, методов и свойств класса PersonalPage

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Модификатор доступа | Тип | Назначение |
| sqConnection | private readonly | SQLiteAsyncConnection | Предостовляет асинхронное соединение с локальной базой данных. |
| notification | private readonly | ILocalNotificationService | Интерфейс для оповещений. |
| auth | private readonly | IFirebaseAuth | Интерфейс для взаимодейтвия с Firebase Authentication SDK. |

Продолжение таблицы П11

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Свойства | | | | |
| Имя | Модификатор | Тип | Доступ | Назначение |
| Models | public | ObservableCollection<TaskPageModel> | get, private set | Модели чек-листов. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| PersonalPage | public | Конструктор | - | Задание начальных значений. |
| OnAppearing | protected override | void | - | Задание "тяжелых" начальных значений. |
| PlusFab\_Clicked | private async | void | object sender, EventArgs e | Добавление персонального чек-листа. |
| MainList\_ItemTapped | private async | void | object sender, ItemTappedEventArgs e | Показ страницы с чек-листом. |
| Important\_Clicked | private | void | object sender, EventArgs e | Устанавливает приоритетность чек-листу. |
| Delete\_Clicked | private async | void | object sender, EventArgs e | Удаление чек-листа. |
| Edit\_Clicked | private async | void | object sender, EventArgs e | Редактирование названия чек-листа. |
| MyDay\_Clicked | private async | void | object sender, EventArgs e | Показывает сегодняшние чек-листы. |
| ImportantLists\_Clicked | private async | void | object sender, EventArgs e | Показывает приоритетные чек-листы. |
| All\_Clicked | private | void | object sender, EventArgs e | Показывает все чек-листы. |
| MainList\_ItemSelected | private | void | object sender, SelectedItemChangedEventArgs e | Метод срабатывает, когда происходит нажатие на элемент списка. Нужно, чтобы цветом не выделялся выбранный элемент. |
| HelpTb\_Clicked | private | void | object sender, EventArgs e | Показывает страницу с подсказками. |

Таблица П12 - Описание полей, методов и свойств класса AppDataHelper

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| GetDatabase | public static | FirebaseDatabase | - | Возвращает экземпляр для работы с базой данных Firebase Realtime Database. |

Таблица П12 - Описание полей, методов и свойств класса PersonalPage

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Модификатор доступа | Тип | Назначение |
| OnChange | public | event EventHandler | Событие для получения уведомлений об измении в определенном узле базы данных. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| OnDataChange | public | void | DataSnapshot snapshot | Получение оповещения об изменении данных. |
| Subscribe | public | void | string path | Устанавливает данный объект к узлу в базе данных, чтобы он оповещал о изменениях. |

Таблица П13 - Описание полей, методов и свойств класса SQLiteDb

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Модификатор доступа | Тип | Назначение |
| path | private readonly | string | Путь для сохранения файла базы данных. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| GetConnectionPersonal | public | SQLiteAsyncConnection | string id | Возвращает соединение для работы с базой данных. |

Таблица П14- Описание полей, методов и свойств класса MyFirebaseAuth

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| LoginWithEmailPassword | public async | Task<string> | string email, string password | Авторизация пользователя. |
| CreateWithEmailPassword | public async | Task<string> | string email, string password | Регистрация нового пользователя. |
| IsCurrentUser | public | bool | - | Определяет выполнил ли пользователь вход. |
| GetUserID | public | string | - | Возвращает уникальный идентификатор пользователя. |
| SignOut | public | void | - | Выполняет выход учетной записи из системы. |

Таблица П15 - Описание полей, методов и свойств класса LocalNotificationService

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Модификатор доступа | Тип | Назначение |
| \_jan1st1970 | private readonly | DateTime | Unix-время |
| \_randomNumber | internal | string | Для задания уникального идентификатора. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| LocalNotification | public | void | string title, string body, int id, DateTime notifyTime, long repeateMs | Установка локальных уведомлений на устройстве. |
| Cancel | public | void | int id | Отмена уведомления |
| GetLauncherActivity | public static | Intent | - | Возвращает "хороший" механизм для выполнения задач в переднем плане. |
| CreateIntent | private | Intent | int id | Возвращает подзадачу, используется для показа уведомлений. |
| GetAlarmManager | private | AlarmManager | - | Возвращает системный сервис для управления сигналами. |

Продолжение таблицы П15

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SerializeNotification | private | AlarmManager | LocalNotification notification | Сериализует уведомление. |
| LongAlert | public | void | string message | Показ длительного Toast уведомления. |
| ShortAlert | public | void | string message | Показ кратковременного Toast уведомления. |

Таблица П16 - Описание полей, методов и свойств класса ScheduledAlarmHandler

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Модификатор доступа | Тип | Назначение |
| manager | private | NotificationManager | Уведомление пользователя о событии. |
| LocalNotificationKey | public const | string | Ключ-константа. |
| channelId | private const | string | Идентификатор канала. |
| channelName | private const | string | Имя канала. |
| channelDescription | private const | string | Описание канала. |
| channelInitialized | private | bool | Флаг инициализации канала. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Название | Модификатор | Тип | Аргументы | Назначение |
| OnReceive | public override | void | Context context, Intent intent | Этот метод вызывается, когда ScheduledAlarmHandler получает механизм для вещания. |
| CreateNotificationChannel | private | void | - | Создание канала для оповещения. |
| DeserializeNotification | private | LocalNotification | string notificationString | Десериализация оповещения. |

# Лист регистрации изменений.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | Новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |