**Введение**

На учебную практику по разработке и сопровождению программного обеспечения была поставлена задача разработать мобильное приложение “DIX”.

Цель учебной практики заключается в разработке мобильного мессенджера «DIX», которое позволит общаться с людьми на расстоянии через всемирную сеть Интеренет.

Создаваемая программа будет рассчитана на любого рода пользователей.

Далее приведем краткое описание разделов пояснительной записки.

Первый раздел носит название “Анализ задачи”. В нем вы сможете ознакомиться с постановкой задачи, которая включает в себя: исследование предметной области поставленной задачи, определение ее организационно-экономической сущности. Также в этом разделе вы сможете узнать о том, как данная задача решается в настоящее время. Все входные и выходные данные тоже будут описаны в первом разделе. В подразделе “Инструменты разработки” будет рассмотрена среда, в которой создается данный курсовой проект. Здесь также будут установлены минимальные и оптимальные требования к аппаратным характеристикам, обеспечивающим правильное функционирование поставленной задачей.

В разделе “Проектирование задачи” будут рассмотрены основные аспекты разработки программного продукта. Здесь можно будет узнать об организации данных в контексте среды разработки. В данном разделе будет четко описан пользовательский интерфейс, составлены алгоритмы процесса обработки информации, описана разработка системы справочной информации.

“Реализация задачи” – это третий раздел пояснительной записки, в котором описываются все элементы и объекты, которые будут использованы при реализации данного приложения. В этом разделе будут четко описаны функции пользователя и их структура. Здесь можно будет найти таблицу, в которой будет представлена полная аннотация файлов используемых в данном проекте.

Четвертый раздел – “Тестирование”. В нем будет описано полное и функциональное тестирование данной программы, т.е. будет оттестирован каждый пункт меню, каждая операция, которая выполняется приложением. Будут смоделированы все возможные действия пользователя при работе с программой, начиная от запуска до выхода.

В разделе “Применение” будет описано назначение, область применения, среда функционирования курсовой программы. Также в нем будет описано использование справочной системы.

“Заключение” будет содержать краткую формулировку задачи, результаты проделанной работы, описание использованных методов и средств, описание степени автоматизации процессов на различных этапах разработки.

В “Списке использованных источников” будет приведен список используемых при разработке источников.

В приложениях к пояснительной записке будет приведен листинг программы с необходимыми комментариями.

1. **Анализ задачи**

**1.1. Разработка постановки задачи**

**1.1.1. Организационно-экономическая сущность задачи:**

Темой данного курсового проекта является «Разработка мобильного приложения «DIX».

Первый мессенджер запустили в 1988 году. Он назывался Internet Relay Chat. Его разработал программист из Финляндии Яркко Ойкаринен. Сервис был популярным, в 2009 году им пользовалось более 500 тыс. человек. Но намного более известным в то время стал мессенджер ICQ. К началу 2010 года в нем были зарегистрированы почти 48 млн пользователей по всему миру. Со временем ICQ не выдержал конкуренции социальных сетей. Уже в 2012 году, с ростом популярности MySpace, Facebook\* и «Вконтакте», его аудитория снизилась на 31%.

Можно привести такие аналоги приложения как: Telegram,WhatsApp,Viber;

Периодичность использования данного программного продукта неограниченна.

Цель данной учебной практики – разработать программный продукт,

который позволит вести чаты с людьми в разных точках мира.

За основу программы следует взять из соответствующих источников

(мессенджеров, сайтов). Приложение должно располагать своим интерфейсом и функционалом к пользователю, чтобы ему было комфортно. Данный проект должен стать общедоступным для всех пользователей. В поставленной задаче необходимо реализовать максимально простой пользовательский интерфейс, позволяющий использовать проект пользователю, не обладающему дополнительными знаниями в интерфейсах мобильных приложений.

Программный продукт предоставляет функционал для следующего ряда пользователей: зарегистрированный пользователь.

**1.1.2. Функциональные требования:**

Разрабатываемый программный продукт должен позволять выполнять

следующие действия:

-регистрация и вход в аккаунт пользователя;

- обмен сообщениями текстового формата;

-поддержка отправки смайлов;

**1.1.3. Описание исходной (входной) информации:**

Перечень исходной информации для мессенджера включает в себя:

1. Текстовые сообщения
2. Смайлы

Формы представления документов по каждой позиции перечня могут быть следующими:

1. Текстовые сообщения - диалоговое окно чата

Примеры заполнения документов:

1. Текстовые сообщения - "Привет, как дела?"

Перечень пользователей исходной информации для мессенджера включает в себя всех зарегистрированных пользователей мессенджера, которые могут обмениваться сообщениями, файлами и проводить аудио и видео вызовы.

**1.1.4. Описание результатной (выходной) информации:**

Результатная (входная) информация для мессенджера может включать в себя следующее:

- Сообщения от других пользователей мессенджера

- Уведомления о новых сообщениях и других событиях

Формы представления результатной информации для мессенджера могут быть различными, включая текстовые сообщения, изображения, видео и звуковые уведомления.

Периодичность и сроки представления результатной информации зависят от активности других пользователей мессенджера и могут быть непредсказуемыми.

Пользователями результатной информации для мессенджера являются другие пользователи этого мессенджера, как индивидуальные лица, так и группы пользователей.

**1.1.5. Описание используемой условно-постоянной информации:**

- Классификаторы контактов: список контактов пользователей мессенджера

- Справочники групповых чатов: список доступных групповых чатов

- Таблица истории сообщений: показывает все отправленные и полученные сообщения

- Список уведомлений: список всех уведомлений о новых сообщениях, звонках и других событиях

Формы представления для мессенджера могут включать интерфейс чата, галерею изображений и видео, список контактов, список групповых чатов, историю сообщений и уведомления.

**1.1.6.** **Нефункциональные (эксплуатационные) требования:**

- Требования к применению включают создание интуитивно понятного пользовательского интерфейса, доступной документации и обучающих материалов.

- Требования к производительности могут включать ограничения на время загрузки приложения, скорость отправки и получения сообщений, использование памяти и другие ресурсы.

- Требования к реализации могут предписывать использование определенных технологий, языков программирования, архитектурных решений и стандартов безопасности.

- Требования к надежности могут определять допустимое время простоя системы, частоту сбоев и возможности восстановления данных.

- Требования к интерфейсу могут включать поддержку различных устройств (компьютеры, мобильные устройства), способы взаимодействия (клавиатура, сенсорный экран) и доступность для людей с ограниченными возможностями.

**1.1.7. Составление плана и графика работы над проектом (диаграмма Ганта)**

Диаграмма Ганта — это популярный тип столбчатых диаграмм, который используется для иллюстрации плана, графика работ по какому-либо проекту. Является одним из методов планирования проектов. Используется в приложениях по управлению проектами. Первый формат диаграммы был разработан Генри Л. Гантом в 1910 году.

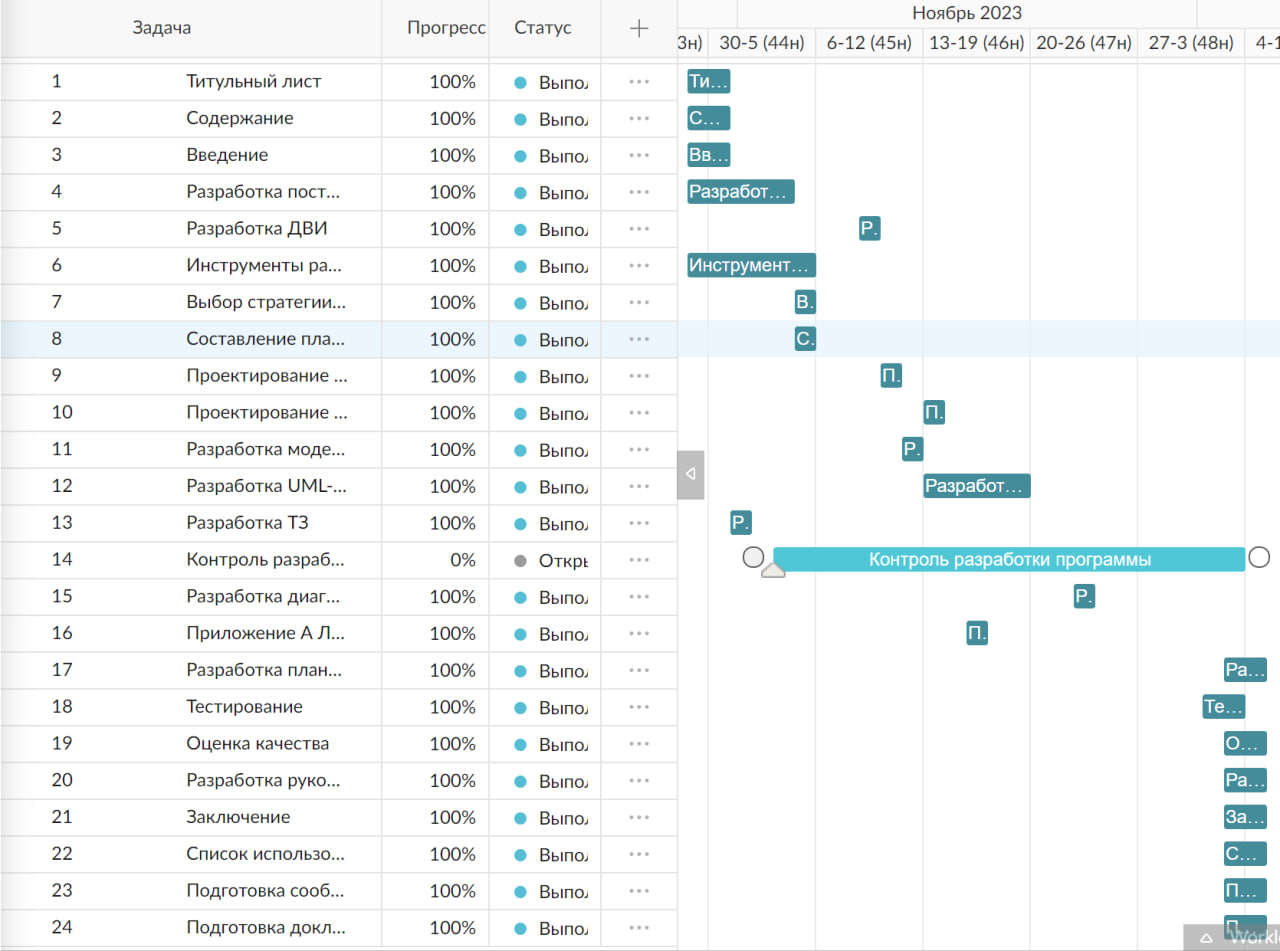


Рисунок 1- Диаграмма Ганта

**1.1.8. Эксплуатационные требования**

**Требования к применению:**Иметь доступ к интернету.

**Требования к реализации:** Для реализации статических страниц и шаблонов должен использоваться конструктор Android Studio.

**Требования к надежности:** Система может быть недоступна не более чем 24 часа в год. У администратора приложения должна быть возможность выгрузить и загрузить копию приложения, доступ ко всем данным зарегистрированных пользователей .

**Требования к интерфейсу:** При разработке приложения должны быть использованы преимущественно простые цвета. Основные разделы приложения должны быть доступны с первой страницы. Грамотный и удобный пользовательский интерфейс. Приложение должно адаптироваться под компьютер, телефон и планшет.

**1.2. Выбор стратегии разработки и модели жизненного цикла**

Для разработки чат-приложения «DIX» следует выбрать стратегию разработки и модель жизненного цикла. Осуществляем выбор посредством составления таблиц:

Таблица 1 **–** Выбор модели жизненного цикла на основе характеристик требований

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| №  крит ерия | Критерии категории требований | Каска дная | Vобра зная | RAD | Инкрем  ентная | Быстрого прототип ирования | Эволюци  онная |
| 1. | Являются ли требования к проекту легко определимыми и реализуемыми? | Да | Да | Да | Да | Нет | Нет |
| 2. | Могут ли требования быть сформулированы в начале ЖЦ? | Да | Да | Да | Да | Нет | Нет |
| 3. | Часто ли будут изменяться требования на протяжении ЖЦ? | Нет | Нет | Нет | Нет | Да | Да |

Продолжение (Таблица 1):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 4. | Нужно ли демонстрировать требования с целью их определения? | Нет | Нет | Да | Да | Да | Да |
| 5. | Требуется ли проверка концепции программного средства или системы? | Нет | Нет | Да | Да | Да | Да |
| 6. | Будут ли требования изменяться или уточняться с ростом сложности системы (программного средства) в ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 7. | Нужно ли реализовать основные | Нет | Нет | Да | Да | Да | Да |
|  | требования на ранних этапах разработки? |  |  |  |  |  |  |
|  | Итого | 2 | 2 | 3 | 6 | 5 | 5 |

Итог: На основе результатов заполнения табл. 1 подходящей является модель быстрого прототипирования и инкрементная.

Таблица 2**–** Выбор модели жизненного цикла на основе характеристик команды разработчиков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| №  критерия | Критерии категории команды разработчиков проекта | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |

Продолжение (Таблица 2):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1. | Являются ли проблемы предметной области проекта новыми для большинства разработчиков? | Нет | Нет | Нет | Нет | Да | Да |
| 2. | Являются ли инструментальны е средства, используемые в проекте, новыми для большинства разработчиков? | Да | Да | Нет | Да | Нет | Да |
| 3. | Изменяются ли роли участников проекта на протяжении ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 4. | Является ли структура процесса разработки более значимой для разработчиков, чем гибкость? | Да | Да | Нет | Да | Нет | Нет |
| 5. | Важна ли легкость распределения человеческих ресурсов проекта? | Да | Да | Да | Да | Нет | Нет |
| 6. | Приемлет ли команда разработчиков оценки, проверки, стадии разработки? | Да | Да | Нет | Да | Да | Да |
|  | Итого | 4 | 3 | 5 | 6 | 3 | 3 |

Итог: На основе результатов заполнения табл. 2 подходящими являются RAD и инкрементная модели.

Таблица 3 **–** Выбор модели жизненного цикла на основе характеристик коллектива пользователей

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| №  крите рия | Критерии категории коллектива пользователей | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |
| 1. | Будет ли присутствие пользователей  ограничено в  ЖЦ  разработки? | Да | Да | Нет | Да | Нет | Да |
| 2. | Будут ли пользователи оценивать текущее состояние программного продукта (системы) в процессе разработки? | Нет | Нет | Нет | Да | Нет | Да |
| 3. | Будут ли пользователи вовлечены во все фазы ЖЦ разработки? | Нет | Нет | Да | Нет | Нет | Нет |
| 4. | Будет ли заказчик отслеживать ход выполнения проекта? | Нет | Нет | Нет | Да | Нет | Да |
|  | Итого | 0 | 0 | 2 | 3 | 0 | 2 |

Итог: На основе результатов заполнения табл. 3 подходящей является инкрементная модель.

Таблица 4 **–** Выбор модели жизненного цикла на основе характеристик типа проектов и рисков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| №  критерия | Критерии категории типов проекта и рисков | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |
| 1. | Разрабатывается ли в проекте продукт нового для организации направления? | Нет | Нет | Нет | Да | Да | Да |
| 2. | Будет ли проект являться расширением существующей системы? | Да | Да | Да | Да | Нет | Нет |
| 3. | Будет ли проект крупно- или среднемасштабны м? | Нет | Нет | Нет | Да | Да | Да |
| 4. | Ожидается ли длительная эксплуатация продукта? | Да | Да | Нет | Да | Нет | Да |
| 5. | Необходим ли высокий уровень надежности продукта проекта? | Нет | Да | Нет | Да | Нет | Да |
| 6. | Предполагается ли эволюция продукта проекта в течение ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 7. | Велика ли вероятность изменения системы (продукта) на этапе сопровождения? | Нет | Нет | Нет | Да | Да | Да |

Продолжение (Таблица 4):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8. | Является ли график сжатым? | Нет | Нет | Да | Да | Да | Да |
| 9. | Предполагается ли повторное | Нет | Нет | Да | Да | Да | Да |
|  | использование компонентов? |  |  |  |  |  |  |
| 10. | Являются ли достаточными ресурсы (время, деньги, инструменты, персонал)? | Нет | Нет | Нет | Нет | Да | Да |
|  | Итого | 4 | 5 | 3 | 9 | 4 | 6 |

Итог: На основе результатов заполнения табл. 4 подходящей является Инкрементная и эволюционная модели.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Каска дная | Vобразная | RAD | Инкремент  ная | Быстрого прототипирова ния | Эволюционная |
| Таблица 1 | 2 | 2 | 3 | 6 | 5 | 5 |
| Таблица 2 | 4 | 3 | 5 | 6 | 3 | 3 |
| Таблица  3 | 0 | 0 | 2 | 3 | 0 | 2 |
| Таблица  4 | 4 | 5 | 3 | 9 | 4 | 6 |
| Общие итоги | 10 | 10 | 13 | 24 | 12 | 16 |

Общий итог: в итоге заполнения табл. 1 – 4 наиболее подходящей является инкрементная модель.

Инкрементная модель нам подходит, потому что задача достаточно сложная и требует изменения в процессе проектирования. Эта модель предполагает разбиение проекта на небольшие итерации, в ходе которых происходит анализ, дизайн, разработка, тестирование и интеграция продукта. Таким образом, мы можем получать работоспособные версии приложения на каждой итерации и постепенно улучшать его функциональность и качества.

* 1. **Инструменты разработки**

Для разработки данного проекта будет выбрана среда разработки Android Studio, которая является наиболее актуальной средой для создания приложений данного типа.

Разработка будет производится на таких языках программирования, как:

* Kotlin– статически типизированный, объектно-ориентированный

язык программирования, работающий поверх Java Virtual Machineсайта;

* Java­­­­– строго типизированный объектно-ориентированный язык

программирования общего назначения.

* WEB-ресурс DRAW.IO – будет использоваться для создания

графической части и разработки UML-диаграмм;

* Microsoft Office Word – для написания документации к

программному продукту;

* Firebase – будет использоваться в качестве

инструмента для администрирования Real Time Data Base;

Разработка проекта будет происходить на компьютерах со следующими параметрами:

Процессор: Intel® Core I5-11300h 4,3Gh, Intel® Core I5-11300h 4,3Gh, AMD Ryzen 5 5800U 4,3Ggh;

Объем оперативной памяти: 8-16 GB;

Объем места на жестком диске: 256-1024 GB;

Видеокарта: NVIDIA MX 450, NVIDIA RTX 3050, Intel Iris XE Graphics;

ОС: Windows 10-11.

**2. Проектирование задачи**

* 1. **Разработка структуры приложения, системы меню, навигации**



Рисунок 2- Система меню

* 1. **Разработка UML-диаграмм**

**2.2.1. Разработка ДВИ ( диаграммы вариантов использования)**

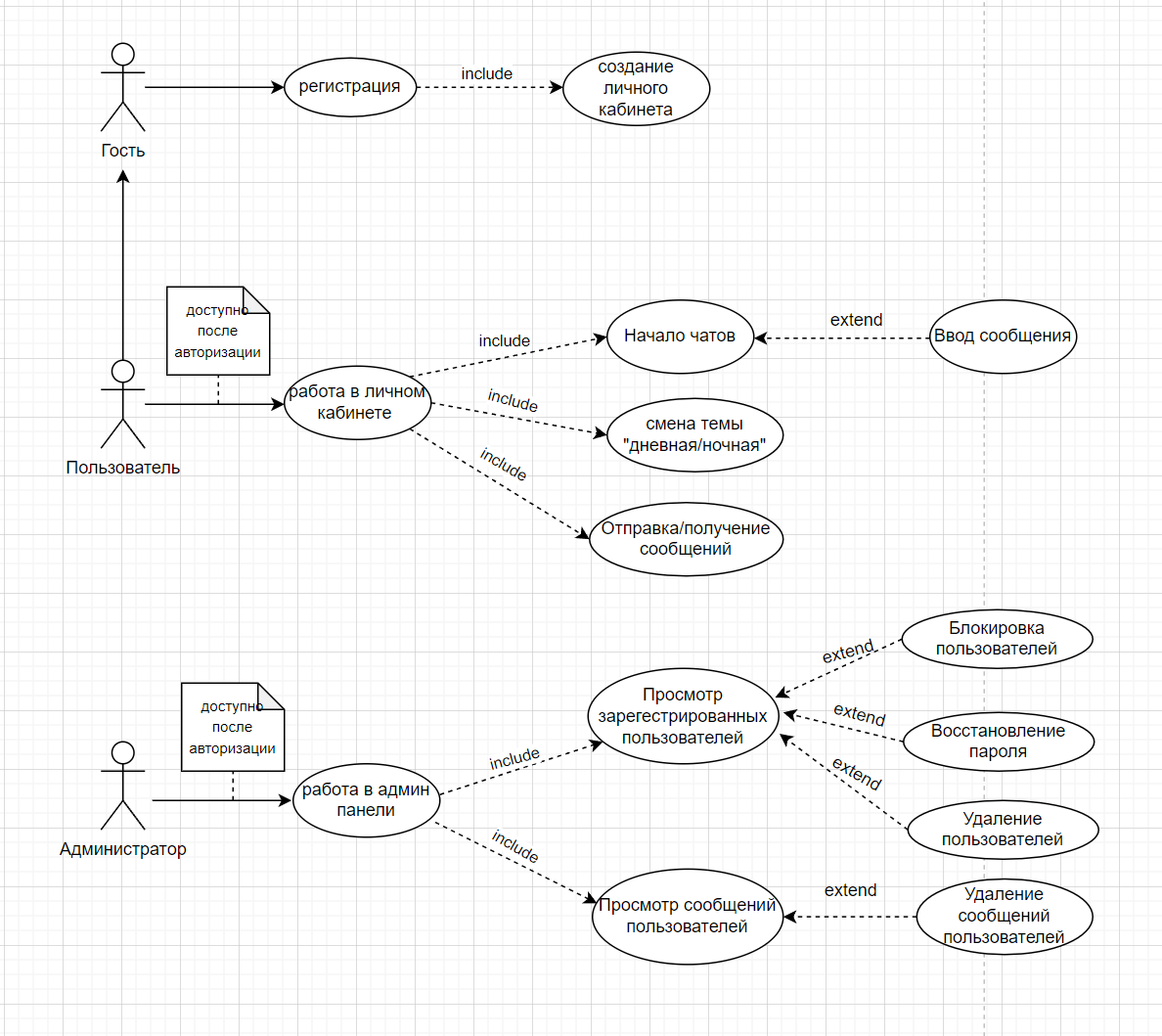


Рисунок 3- Диаграмма вариантов использования

* + 1. **Диаграмма деятельности**

Диаграмма деятельности — UML-диаграмма, на которой показаны действия, состояния которых описано на диаграмме состояний (рисунок 4).

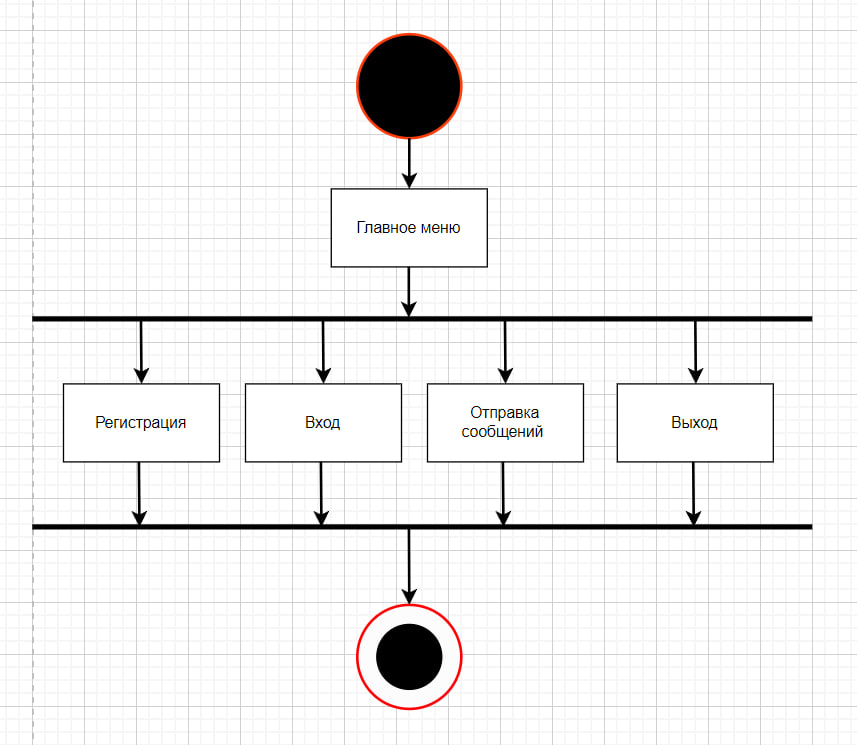


Рисунок 4- Диаграмма деятельности

* + 1. **Диаграмма классов**

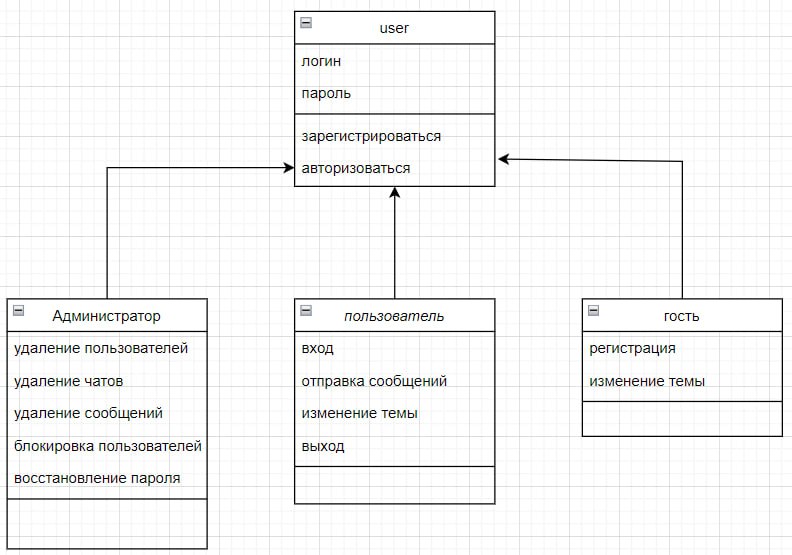


Рисунок 5- Диаграмма классов

* 1. **Разработка пользовательского интерфейса**

Важным элементом проектирования данного программного продукта является описание внешнего интерфейса разрабатываемого интернет-ресурса.

Для разработки визуального дизайн использовались сдержанные, мягкие цвета для удобства использования программного продукта.

В ходе разработки был спроектировано мобильное приложение “DIX”.

Для организации эффективной работы пользователя нужно создать целостный программный продукт данной предметной области, в котором все компоненты будут сгруппированы по функциональному назначению. При этом необходимо обеспечить удобный графический интерфейс пользователя. Интернет-ресурс должен позволить пользователю решать задачи, затрачивая значительно меньше усилий, чем при работе с разрозненными объектами. Все исходные данные будут разделены на несколько групп.

Прототип – это наглядная модель пользовательского интерфейса. В сущности, это «черновик» созданный на основе представления разработчика о потребностях пользователя. Итоговое отображение программы может отличаться от прототипа. C прототипами UX/UI можно ознакомиться в приложении А.

**3. Реализация**

Данный программный продукт был разработан с помощью конструктора мобильных приложений –Android Studio. Удобный и понятный в использовании, с большим функциональном и готовых шаблонов. В данной базе хранится вся информация приложения.

Функциональность Android Studio включает в себя:

1. Редактор кода: Android Studio предоставляет мощный редактор кода с поддержкой автодополнения, проверки ошибок, рефакторинга и других инструментов для удобной работы с кодом.

2. Дизайнер пользовательского интерфейса: Инструменты для создания пользовательского интерфейса приложения с помощью графического интерфейса и возможности предварительного просмотра интерфейса на различных устройствах.

3. Отладка: Встроенные инструменты для отладки приложений, включая возможность установки точек останова, отслеживание значений переменных, анализ стека вызовов и другие инструменты отладки.

4. Эмуляторы и устройства: Возможность запуска и тестирования приложений на встроенных эмуляторах или подключенных устройствах.

5. Интеграция с системами контроля версий: Возможность интеграции с системами контроля версий, такими как Git, для управления исходным кодом проекта.

6. Поддержка языков программирования: Android Studio поддерживает различные языки программирования, включая Java, Kotlin и C++.

Для создания примеров работ использовалась такие программы Adobe Illustrator 2023, Adobe XD, Adobe Photoshop 2024.

Также в программном продукте ”DIX” для регистрации/входа пользователя была использована база данных Firebase.

Преимущества Firebase включают в себя:

1. Расширенная функциональность: Firebase предоставляет широкий спектр инструментов и сервисов, таких как аутентификация, база данных в реальном времени, облачное хранилище, облачные функции, уведомления и многое другое, что позволяет разработчикам создавать полнофункциональные приложения.

2. Простота использования: Firebase предлагает простой и интуитивно понятный интерфейс, что делает его доступным для разработчиков всех уровней.

3. Масштабируемость: Firebase обеспечивает масштабируемость приложений, позволяя им расти по мере необходимости без необходимости переписывать код или изменять архитектуру.

4. Высокая доступность и надежность: Firebase предлагает высокую доступность и надежность своих сервисов благодаря облачной инфраструктуре Google.

5. Интеграция с другими сервисами Google: Firebase интегрируется с другими сервисами Google, такими как Google Analytics, AdMob, Google Cloud Platform и другими, что позволяет разработчикам получить доступ к широкому спектру инструментов для улучшения своих приложений.

6. Бесплатный уровень использования: Firebase предоставляет бесплатный уровень использования для многих своих сервисов, что позволяет начать использовать их без затрат на начальном этапе разработки.

**4.** [**Тестирование**](#_Достоинства_и_недостатки)

* 1. **Тесты на использование**

При разработке сайта “Логотип” многие возникающие ошибки и недоработки были исправлены на этапе реализации программного продукта. После завершения испытания реализации интернет-ресурса было проведено тщательное функциональное тестирование. Функциональное тестирование должно гарантировать работу всех элементов программного продукта в автономном режиме.

Таблица 5 – Тесты на использование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название теста** | **Действие** | **Ожидаемый результат** | **Физический результат** | **Результат тестирования** |
| Кнопка «Log in» | Переход на блок «Список чатов» | Переход на блок «Список чатов» | Переход на блок «Список чатов» | Выполнено |
| Кнопка «Sign Up» | Переход на блок «Регистрация» | Переход на блок «Регистрация» | Переход на блок «Регистрация» | Выполнено |
| Кнопка «Выход» | Выход из аккаунта | Выход из аккаунта | Выход из аккаунта | Выполнено |
| Кнопка «Отправить сообщение» | Отправка сообщения | Отправка сообщения | Отправка сообщения | Выполнено |

* 1. **Отчет о результатах тестирования**

В результате проведения тестирования выяснилось, что все ранее оговоренные функции и требования, были разработаны, а также протестированы. Тесты показали, что все функции работают правильно, следовательно, разработанный сайт можно выпускать.

В ходе тестирования программного обеспечения продукта на устройстве не было выявлено каких-либо ошибок, так как адаптивность интернет-ресурса была проведена на всех стадиях разработки.

1. **Руководство пользователя**

При входе в приложение нас встречает страница с регистрацией с простым, интуитивным для навигации дизайном (рисунок 6).

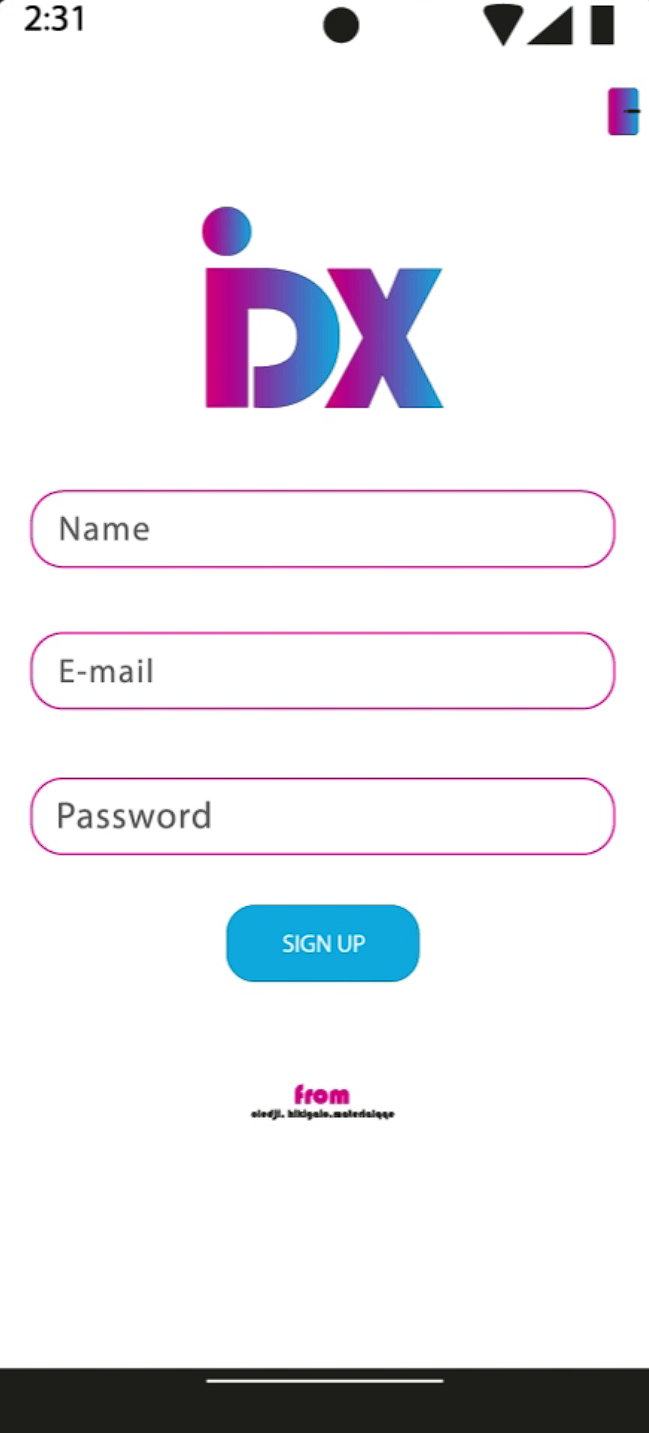


Рисунок 6- Страница регистрации

**5.Кнопка выхода из акканута**

**1. Поле для ввода имени**

**2. Поле для ввода эл.почты**

**3. Поле для ввода пароля**

**4. Кнопка регистрации**

1. Поле 1 предназначено для ввода имени будущего пользователя.
2. Поле 2 предназначено для ввода электронной почты.
3. Поле 3 предназначено для ввода пароля от аккаунта будущего пользователя.
4. Кнопка 4 отвечает за внесение введенных данных в базу данных с последующим входом в аккаунт.

После регистрации в приложении все данные занесенные пользователем помещаются в базу данных Firebase (рисунок 7).

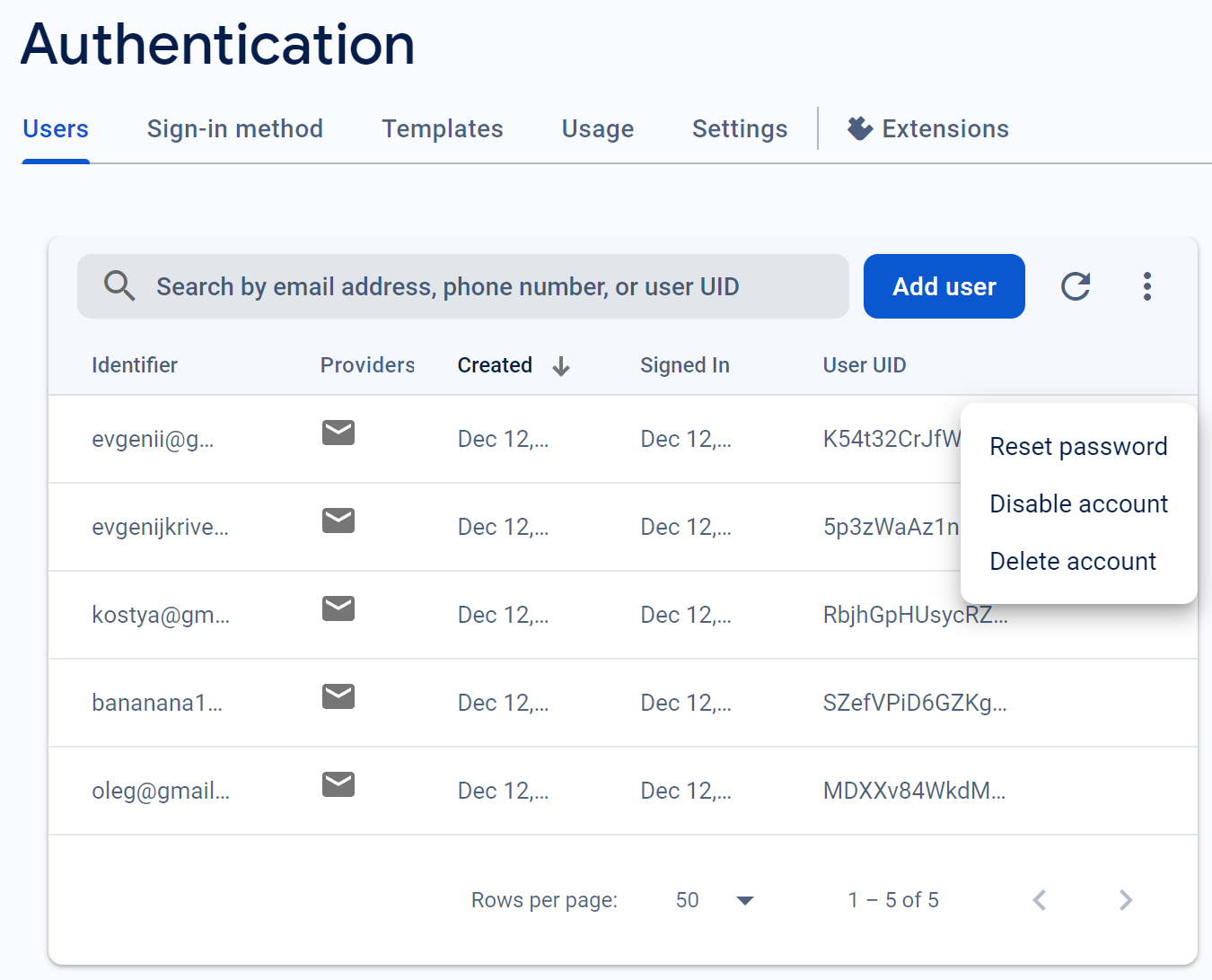


Рисунок 7- База данных зарегистрированных пользователей

Также администратор может как удалять, так и добавлять пользователей в базу данных, блокировать пользователей и восстанавливать пароли (рисунок 8).

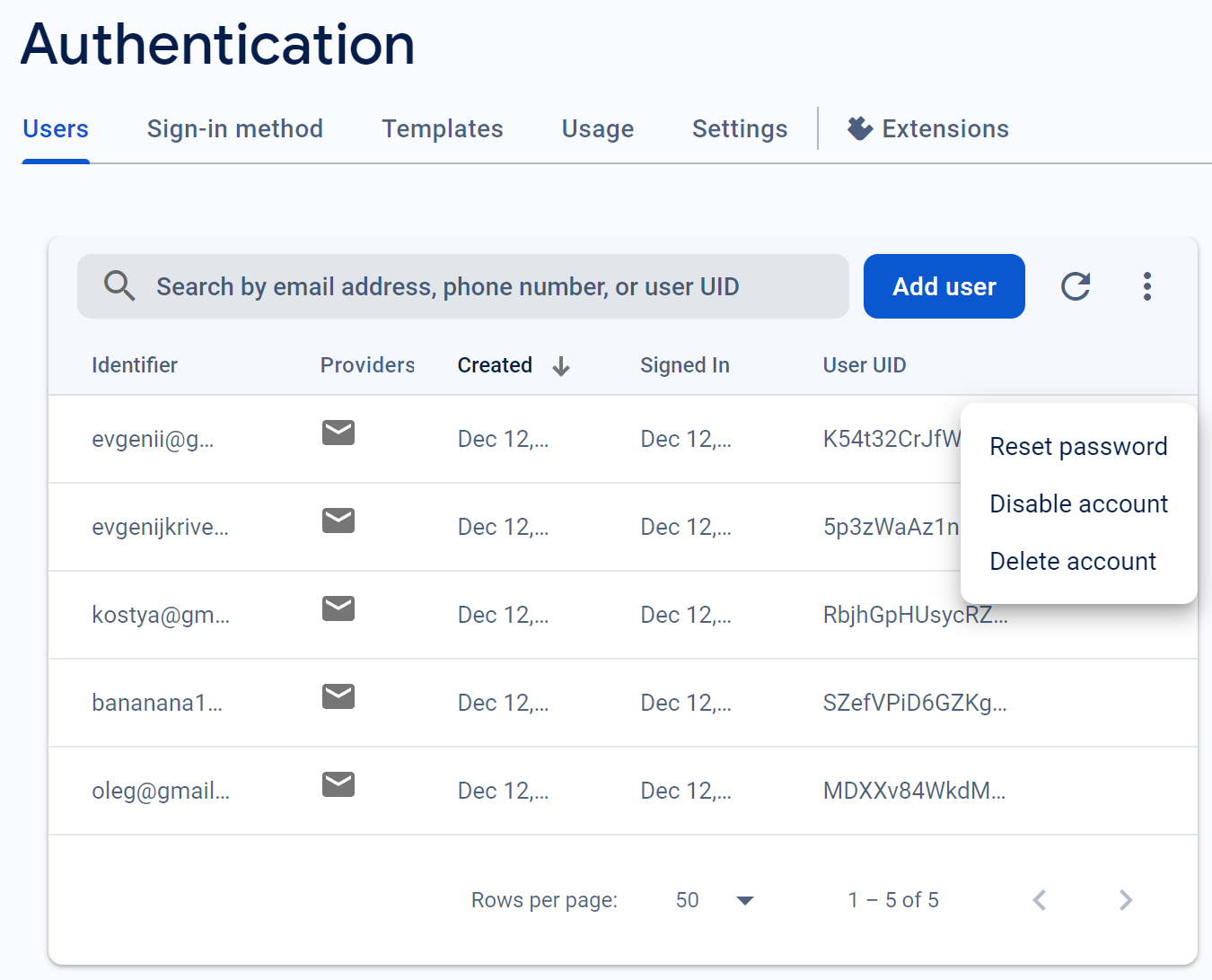


Рисунок 8- Возможности администратора

**1. Добавление пользователей**

**2.Восстановление пароля**

**3.Блокировка аккаунта**

**4.Удаление аккаунта**

1. Кнопка 1 отвечает за добавление пользователей в базу данных.

2. Кнопка 2 отвечает за восстановления пароля от аккаунта уже зарегистрированного пользователя.

3. Кнопка 3 отвечает за блокировку аккаунта пользователя.

4. Кнопка 4 отвечает за удаления аккаунта из базы данных.

**Заключение**

Разработка мессенджера «DIX», для общения была выполнена используя среду разработки Java.

для разработки программы использовались:

-Kotlin– статически типизированный, объектно-ориентированный язык программирования, работающий поверх Java Virtual Machineсайта;

-Java– строго типизированный объектно-ориентированный язык программирования общего назначения.

-WEB-ресурс DRAW.IO – будет использоваться для создания графической части и разработки UML-диаграмм;

-Microsoft Office Word – для написания документации к программному продукту;

-Firebase – будет использоваться в качестве инструмента для администрирования Real Time Data Base;

Использование данных методов и средств позволило создать полноценное приложение.

Были реализованы все основные функции приложения:

-регистрация и вход в аккаунт пользователя;

- обмен сообщениями текстового формата;

- отправка смайлов.

**Список использованных источников**

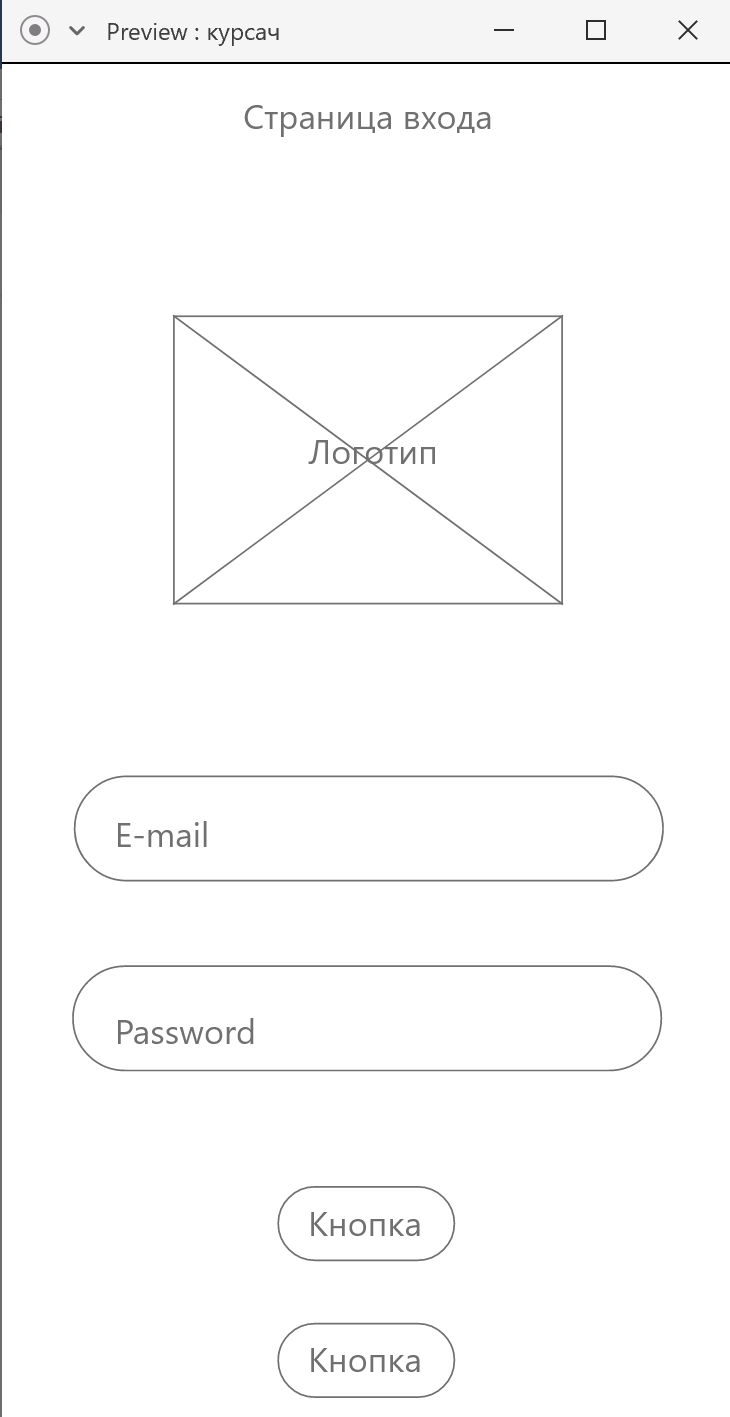
Конструктор для создания мобильных приложений «Android Studio» - <https://developer.android.com/studio>

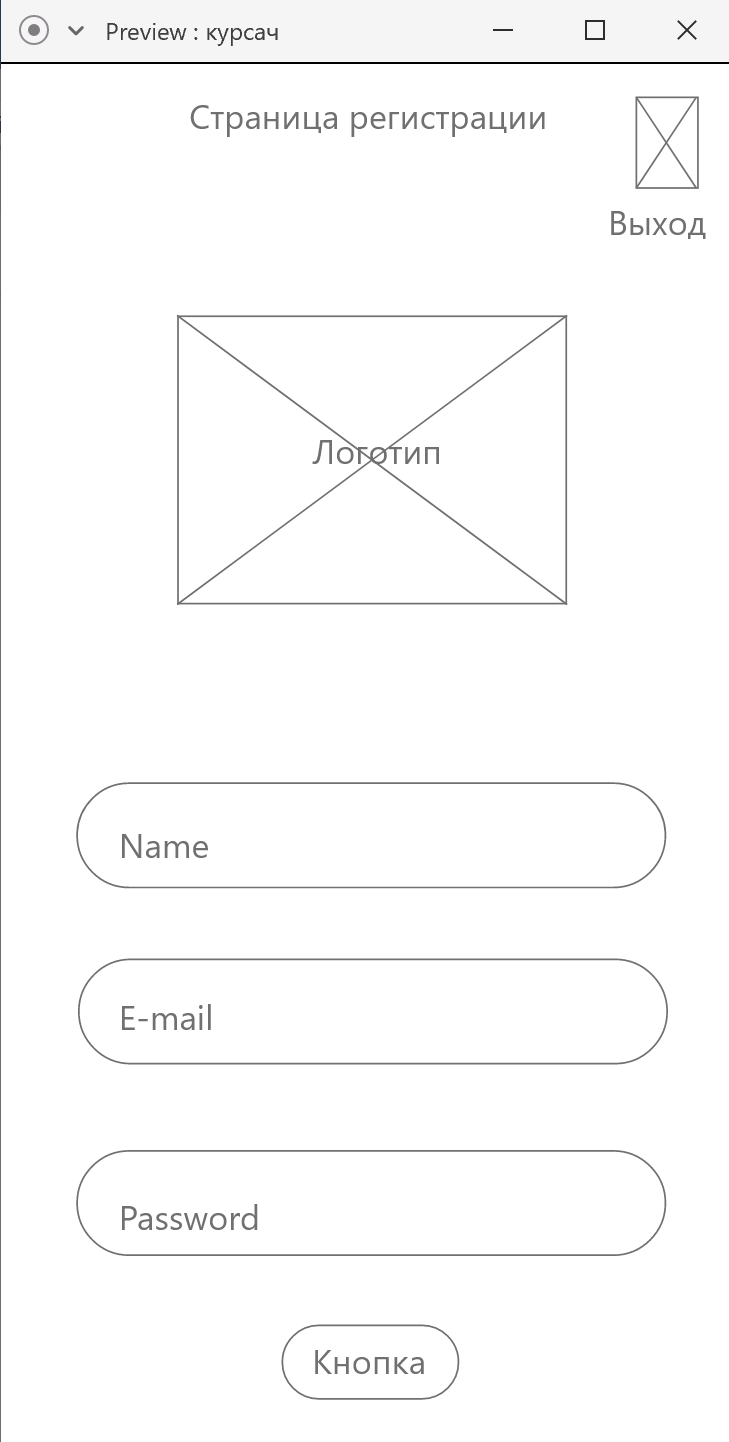
Draw io - <https://app.diagrams.net/>

Adobe Illustrator - <https://www.adobe.com/ru/products/illustrator.html>

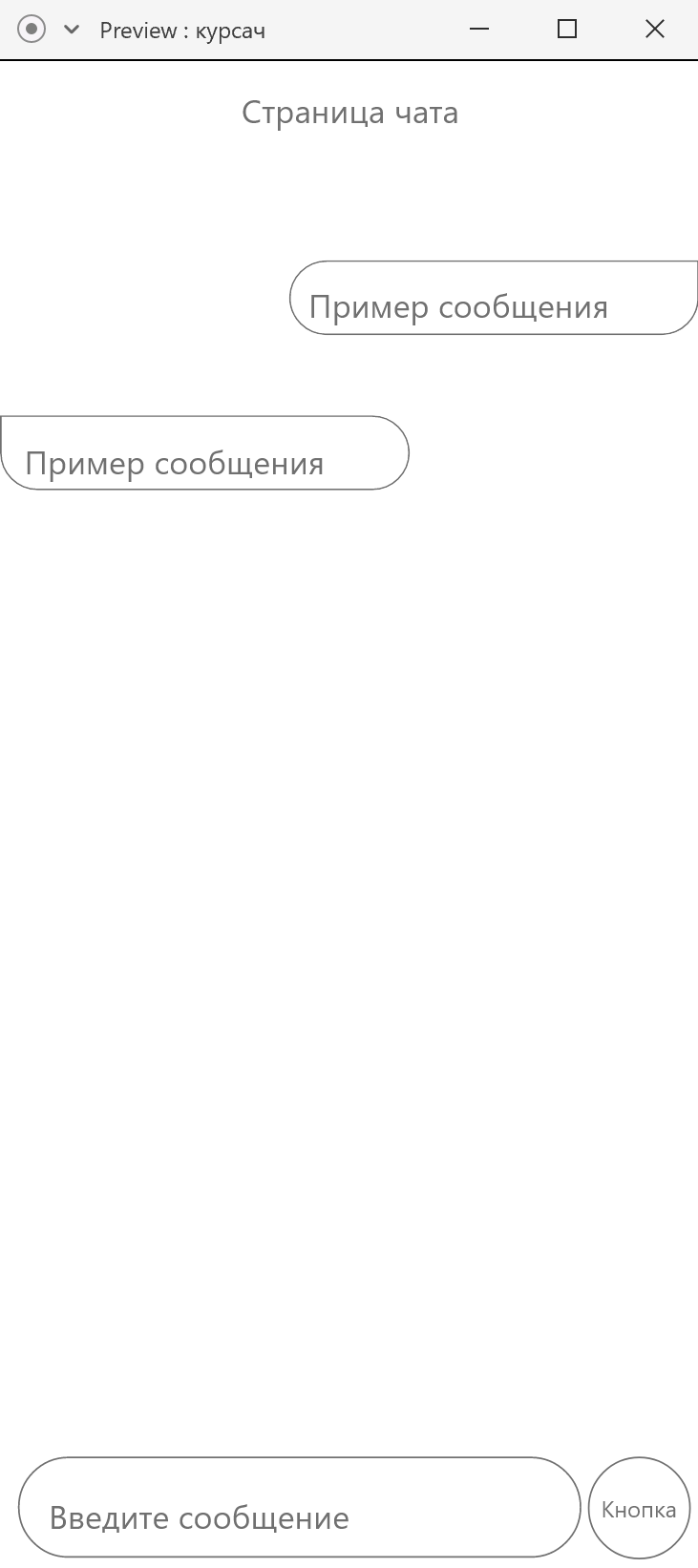
Обучающие видео по созданию мобильных приложений с использованием языка программирования Kotlin на YouTube - <https://www.youtube.com/>

Помощь по связыванию баз данных с приложением Firebase- https://firebase.google.com/docs?hl=ru









Xml  
btn\_backgrounf. Xml

<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android">

<corners android:radius="20dp"/>

<solid android:color="@color/GOLY"/>

</shape>

edt\_background.xml

<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android">

<corners android:radius="20dp"/>

<stroke android:color="@color/PINQ" android:width="3dp"/>

</shape>

ic\_launcher\_background.xml

<vector xmlns:android="http://schemas.android.com/apk/res/android"

android:width="108dp"

android:height="108dp"

android:viewportWidth="108"

android:viewportHeight="108">

<!-- Чисто белый фон -->

<path

android:fillColor="#FFFFFFFF"

android:pathData="M0,0h108v108h-108z" />

<!-- Убраны линии -->

</vector>

ic\_launcher\_foreground.xml

<vector xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:aapt="http://schemas.android.com/aapt"

android:width="108dp"

android:height="108dp"

android:viewportWidth="108"

android:viewportHeight="108">

<path android:pathData="M31,63.928c0,0 6.4,-11 12.1,-13.1c7.2,-2.6 26,-1.4 26,-1.4l38.1,38.1L107,108.928l-32,-1L31,63.928z">

<aapt:attr name="android:fillColor">

<gradient

android:endX="85.84757"

android:endY="92.4963"

android:startX="42.9492"

android:startY="49.59793"

android:type="linear">

<item

android:color="@color/GOLY"

android:offset="0.0" />

<item

android:color="#00000000"

android:offset="1.0" />

</gradient>

</aapt:attr>

</path>

<path

android:fillColor="@color/PINQ"

android:fillType="nonZero"

android:pathData="M65.3,45.828l3.8,-6.6c0.2,-0.4 0.1,-0.9 -0.3,-1.1c-0.4,-0.2 -0.9,-0.1 -1.1,0.3l-3.9,6.7c-6.3,-2.8 -13.4,-2.8 -19.7,0l-3.9,-6.7c-0.2,-0.4 -0.7,-0.5 -1.1,-0.3C38.8,38.328 38.7,38.828 38.9,39.228l3.8,6.6C36.2,49.428 31.7,56.028 31,63.928h46C76.3,56.028 71.8,49.428 65.3,45.828zM43.4,57.328c-0.8,0 -1.5,-0.5 -1.8,-1.2c-0.3,-0.7 -0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1 1.2,1.8C45.3,56.528 44.5,57.328 43.4,57.328L43.4,57.328zM64.6,57.328c-0.8,0 -1.5,-0.5 -1.8,-1.2s-0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1 1.2,1.8C66.5,56.528 65.6,57.328 64.6,57.328L64.6,57.328z"

android:strokeWidth="1"

android:strokeColor="#00000000" />

</vector>

message\_box\_background.xml

<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android">

<corners android:radius="20dp"/>

<stroke android:color="@color/PINQ" android:width="2dp"/>

</shape>

receive\_back.xml

<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android">

<corners android:radius="20dp"

android:topLeftRadius="0dp"/>

<solid android:color="@color/PINQ"/>

</shape>

sent\_back.xml

<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android">

<corners android:radius="20dp"

android:topRightRadius="0dp"/>

<solid android:color="@color/GOLY"/>

</shape>

activity\_chat.xml

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".ChatActivity">

<androidx.recyclerview.widget.RecyclerView

android:id="@+id/chatRecyclerview"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_above="@+id/linearLayout"

android:layout\_alignParentTop="true"

android:layout\_marginTop="1dp"

android:layout\_marginBottom="-1dp" />

<LinearLayout

android:id="@+id/linearLayout"

android:layout\_alignParentBottom="true"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginBottom="5dp"

android:weightSum="100">

<EditText

android:id="@+id/messageBox"

android:layout\_width="304dp"

android:layout\_height="match\_parent"

android:layout\_marginLeft="10dp"

android:layout\_weight="80"

android:background="@drawable/message\_box\_background"

android:hint="Сообщение"

android:paddingLeft="10dp"

/>

<ImageView

android:id="@+id/sentButton"

android:layout\_width="40dp"

android:layout\_height="40dp"

android:layout\_weight="20"

android:src="@drawable/send" />

</LinearLayout>

</RelativeLayout>

activity\_login.xml

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".Login">

<ImageView

android:id="@+id/app\_Logo"

android:layout\_width="150dp"

android:layout\_height="150dp"

android:src="@drawable/logo"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="70dp"/>

<ImageView

android:id="@+id/app\_from"

android:layout\_width="90dp"

android:layout\_height="90dp"

android:layout\_marginTop="600dp"

android:layout\_centerHorizontal="true"

android:src="@drawable/from" />

<EditText

android:id="@+id/edt\_email"

android:layout\_width="match\_parent"

android:layout\_height="50dp"

android:layout\_below="@id/app\_Logo"

android:layout\_marginTop="40dp"

android:hint="E-mail"

android:paddingLeft="10dp"

android:background="@drawable/edt\_background"

android:layout\_marginLeft="20dp"

android:layout\_marginRight="20dp"/>

<EditText

android:id="@+id/edt\_password"

android:layout\_width="match\_parent"

android:layout\_height="50dp"

android:layout\_below="@id/edt\_email"

android:layout\_marginTop="40dp"

android:hint="Password"

android:paddingLeft="10dp"

android:background="@drawable/edt\_background"

android:layout\_marginRight="20dp"

android:layout\_marginLeft="20dp"/>

<Button

android:id="@+id/btn\_Login"

android:layout\_width="120dp"

android:layout\_height="wrap\_content"

android:layout\_below="@id/edt\_password"

android:text="Log in"

android:background="@drawable/btn\_backgrounf"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="30dp"

android:textColor="@color/white"/>

<Button

android:id="@+id/btn\_Signup"

android:layout\_width="120dp"

android:layout\_height="wrap\_content"

android:layout\_below="@id/btn\_Login"

android:text="Sign up"

android:background="@drawable/btn\_backgrounf"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="30dp"

android:textColor="@color/white"/>

</RelativeLayout>

activity\_main.xml  
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<androidx.recyclerview.widget.RecyclerView

android:id="@+id/userRecyclerView"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="1.0"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_bias="1.0"

tools:listitem="@layout/user\_layout" />

</androidx.constraintlayout.widget.ConstraintLayout>

activity\_signup.xml

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".Login">

<ImageView

android:id="@+id/mini\_exit"

android:layout\_width="30dp"

android:layout\_height="30dp"

android:layout\_marginTop="5dp"

android:layout\_marginLeft="382dp"

android:src="@drawable/exit" />

<ImageView

android:id="@+id/app\_Logo"

android:layout\_width="150dp"

android:layout\_height="150dp"

android:src="@drawable/logo"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="70dp"/>

<ImageView

android:id="@+id/app\_from"

android:layout\_width="90dp"

android:layout\_height="90dp"

android:layout\_marginTop="600dp"

android:layout\_centerHorizontal="true"

android:src="@drawable/from" />

<EditText

android:id="@+id/edt\_name"

android:layout\_width="match\_parent"

android:layout\_height="50dp"

android:layout\_below="@id/app\_Logo"

android:layout\_marginTop="40dp"

android:hint="Name"

android:paddingLeft="10dp"

android:background="@drawable/edt\_background"

android:layout\_marginLeft="20dp"

android:layout\_marginRight="20dp"/>

<EditText

android:id="@+id/edt\_email"

android:layout\_width="match\_parent"

android:layout\_height="50dp"

android:layout\_below="@id/edt\_name"

android:layout\_marginTop="40dp"

android:hint="E-mail"

android:paddingLeft="10dp"

android:background="@drawable/edt\_background"

android:layout\_marginLeft="20dp"

android:layout\_marginRight="20dp"/>

<EditText

android:id="@+id/edt\_password"

android:layout\_width="match\_parent"

android:layout\_height="50dp"

android:layout\_below="@id/edt\_email"

android:layout\_marginLeft="20dp"

android:layout\_marginTop="40dp"

android:layout\_marginRight="20dp"

android:background="@drawable/edt\_background"

android:hint="Password"

android:paddingLeft="10dp" />

<Button

android:id="@+id/btnSignup"

android:layout\_width="120dp"

android:layout\_height="wrap\_content"

android:layout\_below="@id/edt\_password"

android:layout\_marginTop="30dp"

android:layout\_centerHorizontal="true"

android:background="@drawable/btn\_backgrounf"

android:text="Sign up"

android:textColor="@color/white" />

</RelativeLayout>

Receive.xml

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<RelativeLayout

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentLeft="true"

android:layout\_marginLeft="5dp"

android:layout\_marginTop="5dp"

android:background="@drawable/receive\_back">

<TextView

android:textColor="@color/white"

android:id="@+id/txt\_receive\_message"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Это полученное сообщение"

android:padding="10dp"

android:textSize="18sp"/>

</RelativeLayout>

</RelativeLayout>

Sent.xml

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<RelativeLayout

android:layout\_width="wrap\_content"

android:background="@drawable/sent\_back"

android:layout\_alignParentRight="true"

android:layout\_marginRight="5dp"

android:layout\_marginTop="5dp"

android:layout\_height="wrap\_content">

<TextView

android:id="@+id/txt\_sent\_message"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Это отправленое сообщение"

android:textColor="@color/white"

android:padding="10dp"

android:textSize="18sp"/>

</RelativeLayout>

</RelativeLayout>

user\_layout.xml

xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<TextView

android:layout\_marginTop="5dp"

android:id="@+id/txt\_name"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="addi"

android:textSize="30sp"

android:textStyle="bold"

android:layout\_marginLeft="10dp"/>

<View

android:layout\_below="@id/txt\_name"

android:layout\_marginTop="5dp"

android:layout\_width="wrap\_content"

android:layout\_height="1dp"

android:background="@color/black"/>

</RelativeLayout>

Menu.xml

<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android">

<item android:id="@+id/logout" android:title="Выйти"/>

</menu>

Kt

ChatActivity.kt  
package com.example.dix

import android.content.Intent

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

import android.widget.EditText

import android.widget.ImageView

import androidx.recyclerview.widget.LinearLayoutManager

import androidx.recyclerview.widget.RecyclerView

import com.google.firebase.auth.FirebaseAuth

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

class ChatActivity : AppCompatActivity() {

// RecyclerView для отображения сообщений чата

private lateinit var chatRecyclerView: RecyclerView

// EditText для ввода сообщений

private lateinit var messageBox: EditText

// ImageView для отправки сообщений

private lateinit var sendButton: ImageView

// Адаптер для управления и отображения сообщений в RecyclerView

private lateinit var messageAdapter: MessageAdapter

// Список для хранения сообщений

private lateinit var messageList: ArrayList<Message>

// Ссылка на Firebase Realtime Database

private lateinit var mDbRef: DatabaseReference

// Строки для хранения уникальных идентификаторов комнаты чата на основе UID пользователей

var receiveRoom: String? = null

var senderRoom: String? = null

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_chat)

// Получение имени и UID пользователя из intent

val name = intent.getStringExtra("name")

val receiveUid = intent.getStringExtra("uid")

// Получение текущего UID пользователя с использованием Firebase Authentication

val senderUid = FirebaseAuth.getInstance().currentUser?.uid

// Инициализация ссылки на Firebase Realtime Database

mDbRef = FirebaseDatabase.getInstance().getReference()

// Генерация уникальных идентификаторов комнаты чата на основе UID пользователей

senderRoom = receiveUid + senderUid

receiveRoom = senderUid + receiveUid

// Установка заголовка ActionBar в имя получателя чата

supportActionBar?.title = name

// Инициализация элементов пользовательского интерфейса

chatRecyclerView = findViewById(R.id.chatRecyclerview)

messageBox = findViewById(R.id.messageBox)

sendButton = findViewById(R.id.sentButton)

// Инициализация списка сообщений и адаптера для RecyclerView

messageList = ArrayList()

messageAdapter = MessageAdapter(this, messageList)

// Настройка RecyclerView с LinearLayoutManager и адаптером сообщений

chatRecyclerView.layoutManager = LinearLayoutManager(this)

chatRecyclerView.adapter = messageAdapter

// Настройка ValueEventListener для прослушивания изменений в узле messages комнаты отправителя

mDbRef.child("chats").child(senderRoom!!).child("messages")

.addValueEventListener(object : ValueEventListener {

override fun onDataChange(snapshot: DataSnapshot) {

// Очистка существующего списка сообщений

messageList.clear()

// Перебор сообщений в снимке данных и добавление их в список сообщений

for (postSnapshot in snapshot.children) {

val message = postSnapshot.getValue(Message::class.java)

messageList.add(message!!)

}

// Уведомление адаптера об изменениях в данных

messageAdapter.notifyDataSetChanged()

}

override fun onCancelled(error: DatabaseError) {

// Обработка отмены операции при чтении из базы данных

}

})

// Установка слушателя события для кнопки отправки сообщения

sendButton.setOnClickListener {

// Получение текста сообщения из EditText

val message = messageBox.text.toString()

// Создание объекта сообщения

val messageObject = Message(message, senderUid)

// Отправка сообщения в узел messages комнаты отправителя и получателя

mDbRef.child("chats").child(senderRoom!!).child("messages").push()

.setValue(messageObject).addOnSuccessListener {

mDbRef.child("chats").child(receiveRoom!!).child("messages").push()

.setValue(messageObject)

}

// Очистка EditText после отправки сообщения

messageBox.setText("")

}

}

}

Login.tk

package com.example.dix

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

import android.widget.Button

import android.widget.EditText

import android.content.Intent

import android.provider.ContactsContract.CommonDataKinds.Email

import android.widget.Toast

import com.google.firebase.auth.FirebaseAuth

class Login : AppCompatActivity() {

// Элементы пользовательского интерфейса

private lateinit var edtEmail: EditText

private lateinit var edtPasword: EditText

private lateinit var btnLogin: Button

private lateinit var btnSignup: Button

// Объект для аутентификации через Firebase

private lateinit var mAuth: FirebaseAuth

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_login)

// Инициализация объекта FirebaseAuth

mAuth = FirebaseAuth.getInstance()

// Скрытие ActionBar

supportActionBar?.hide()

// Инициализация элементов пользовательского интерфейса

edtEmail = findViewById(R.id.edt\_email)

edtPasword = findViewById(R.id.edt\_password)

btnLogin = findViewById(R.id.btn\_Login)

btnSignup = findViewById(R.id.btn\_Signup)

// Обработчик события для кнопки регистрации

btnSignup.setOnClickListener {

val intent = Intent(this, Signup::class.java)

startActivity(intent)

}

// Обработчик события для кнопки входа

btnLogin.setOnClickListener{

val email = edtEmail.text.toString()

val password=edtPasword.text.toString()

// Вызов метода для выполнения входа

login(email, password)

}

}

// Метод для выполнения входа пользователя

private fun login(email: String, password: String){

mAuth.signInWithEmailAndPassword(email, password)

.addOnCompleteListener(this) { task ->

if (task.isSuccessful) {

// Вход выполнен успешно, переход к главному экрану

val intent=Intent(this@Login,MainActivity::class.java)

finish()

startActivity(intent)

} else {

// В случае неудачного входа выводится уведомление

Toast.makeText(this@Login, "Пользователя не существует", Toast.LENGTH\_SHORT).show()

}

}

}

}

MainActivity.tk

package com.example.dix

import android.content.Intent

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

import android.view.Menu

import android.view.MenuItem

import androidx.recyclerview.widget.LinearLayoutManager

import androidx.recyclerview.widget.RecyclerView

import com.google.firebase.auth.FirebaseAuth

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

class MainActivity : AppCompatActivity() {

// RecyclerView для отображения списка пользователей

private lateinit var userRecyclerView: RecyclerView

// Список пользователей и адаптер для RecyclerView

private lateinit var userList: ArrayList<User>

private lateinit var adapter: UserAdapter

// Объекты для аутентификации и работы с базой данных Firebase

private lateinit var mAuth: FirebaseAuth

private lateinit var mDbRef: DatabaseReference

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// Инициализация объектов Firebase

mAuth = FirebaseAuth.getInstance()

mDbRef = FirebaseDatabase.getInstance().getReference()

// Инициализация списка пользователей и адаптера

userList = ArrayList()

adapter = UserAdapter(this, userList)

// Инициализация RecyclerView

userRecyclerView = findViewById(R.id.userRecyclerView)

// Настройка RecyclerView с LinearLayoutManager и адаптером

userRecyclerView.layoutManager = LinearLayoutManager(this)

userRecyclerView.adapter = adapter

// Добавление слушателя изменений в узел "user" базы данных

mDbRef.child("user").addValueEventListener(object : ValueEventListener {

override fun onDataChange(snapshot: DataSnapshot) {

// Очистка списка пользователей

userList.clear()

// Перебор данных из снимка для построения списка пользователей

for (postSnapshot in snapshot.children) {

val currentUser = postSnapshot.getValue(User::class.java)

// Исключение текущего пользователя из списка

if (mAuth.currentUser?.uid != currentUser?.uid) {

userList.add(currentUser!!)

}

}

// Уведомление адаптера об изменениях в данных

adapter.notifyDataSetChanged()

}

override fun onCancelled(error: DatabaseError) {

// Обработка отмены операции при чтении из базы данных

}

})

}

// Создание меню на верхней панели

override fun onCreateOptionsMenu(menu: Menu?): Boolean {

menuInflater.inflate(R.menu.menu, menu)

return super.onCreateOptionsMenu(menu)

}

// Обработка выбора пункта меню

override fun onOptionsItemSelected(item: MenuItem): Boolean {

if (item.itemId == R.id.logout) {

// Выход из аккаунта при выборе соответствующего пункта меню

mAuth.signOut()

val intent = Intent(this@MainActivity, Login::class.java)

startActivity(intent)

return true

}

return super.onOptionsItemSelected(item)

}

}

Message.tk

package com.example.dix

class Message {

var message: String?=null

var senderId: String?=null

constructor(){}

constructor(message: String?, senderId: String?){

this.message=message

this.senderId=senderId

}

}

MessageAdapter.tk

package com.example.dix

import android.content.Context

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.TextView

import androidx.recyclerview.widget.RecyclerView

import com.google.firebase.auth.FirebaseAuth

class MessageAdapter(val context: Context, val messageList: ArrayList<Message>):

RecyclerView.Adapter<RecyclerView.ViewHolder>() {

// Constants to define different types of message views

val ITEM\_RECEIVE = 1

val ITEM\_SENT = 2

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecyclerView.ViewHolder {

// Inflating the layout based on the view type

if(viewType == ITEM\_RECEIVE){

val view: View = LayoutInflater.from(context).inflate(R.layout.receive, parent, false)

return ReceiveViewHolder(view)

} else {

val view: View = LayoutInflater.from(context).inflate(R.layout.sent, parent, false)

return SentViewHolder(view)

}

}

override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {

val currentMessage = messageList[position]

// Binding data to the appropriate view holder based on its type

when (holder) {

is SentViewHolder -> {

holder.sentMessage.text = currentMessage.message

}

is ReceiveViewHolder -> {

holder.receiveMessage.text = currentMessage.message

}

}

}

override fun getItemViewType(position: Int): Int {

val currentMessage = messageList[position]

// Determining the type of view based on the sender's ID

return if (FirebaseAuth.getInstance().currentUser?.uid == currentMessage.senderId) {

ITEM\_SENT

} else {

ITEM\_RECEIVE

}

}

override fun getItemCount(): Int {

return messageList.size

}

// View holder for sent messages

class SentViewHolder(itemView: View): RecyclerView.ViewHolder(itemView) {

val sentMessage = itemView.findViewById<TextView>(R.id.txt\_sent\_message)

}

// View holder for received messages

class ReceiveViewHolder(itemView: View): RecyclerView.ViewHolder(itemView) {

val receiveMessage = itemView.findViewById<TextView>(R.id.txt\_receive\_message)

}

}

Signup.tk

package com.example.dix

import android.content.Intent

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

import android.widget.Button

import android.widget.EditText

import android.widget.ImageView

import android.widget.Toast

import com.google.firebase.auth.FirebaseAuth

import com.google.firebase.database.DatabaseReference

import com.google.firebase.database.FirebaseDatabase

class Signup : AppCompatActivity() {

// Элементы пользовательского интерфейса

private lateinit var edtName: EditText

private lateinit var edtEmail: EditText

private lateinit var edtPasword: EditText

private lateinit var btnSignup: Button

private lateinit var mAuth: FirebaseAuth

private lateinit var miniExit: ImageView

private lateinit var mDatabaseRef: DatabaseReference

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_signup)

// Инициализация объекта FirebaseAuth

mAuth = FirebaseAuth.getInstance()

// Скрытие ActionBar

supportActionBar?.hide()

// Инициализация элементов пользовательского интерфейса

edtName = findViewById(R.id.edt\_name)

edtEmail = findViewById(R.id.edt\_email)

edtPasword = findViewById(R.id.edt\_password)

btnSignup = findViewById(R.id.btnSignup)

miniExit = findViewById(R.id.mini\_exit)

// Обработчик события для кнопки выхода на экран входа

miniExit.setOnClickListener {

val intent = Intent(this, Login::class.java)

startActivity(intent)

}

// Обработчик события для кнопки регистрации

btnSignup.setOnClickListener {

val name = edtName.text.toString()

val email = edtEmail.text.toString()

val password = edtPasword.text.toString()

// Вызов метода для регистрации нового пользователя

signUp(name, email, password)

}

}

// Метод для регистрации нового пользователя

private fun signUp(name: String, email: String, password: String) {

mAuth.createUserWithEmailAndPassword(email, password)

.addOnCompleteListener(this) { task ->

if (task.isSuccessful) {

// Добавление пользователя в базу данных и переход к главному экрану

addUserToDatabase(name, email, mAuth.currentUser?.uid!!)

val intent = Intent(this@Signup, MainActivity::class.java)

finish()

startActivity(intent)

} else {

// В случае неудачной регистрации выводится уведомление

Toast.makeText(this@Signup, "Некорректные данные", Toast.LENGTH\_SHORT).show()

}

}

}

// Метод для добавления пользователя в базу данных Firebase

private fun addUserToDatabase(name: String, email: String, uid: String) {

mDatabaseRef = FirebaseDatabase.getInstance().getReference()

mDatabaseRef.child("user").child(uid).setValue(User(name, email, uid))

}

}

User.tk  
package com.example.dix;

import javax.xml.namespace.QName;

public class User {

var name: String?=null

var email: String?=null

var uid: String?=null

constructor(){}

constructor(name: String?, email: String?, uid: String?){

this.name= name

this.email= email

this.uid= uid

}

}

UserAdapter.tk

package com.example.dix

import android.content.Context

import android.content.Intent

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.TextView

import androidx.recyclerview.widget.RecyclerView

class UserAdapter(val context: Context, val userList: ArrayList<User>) :

RecyclerView.Adapter<UserAdapter.UserViewHolder>() {

// Создание нового представления пользователя (ViewHolder)

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): UserViewHolder {

val view: View = LayoutInflater.from(context).inflate(R.layout.user\_layout, parent, false)

return UserViewHolder(view)

}

// Возвращает общее количество элементов в списке пользователей

override fun getItemCount(): Int {

return userList.size

}

// Привязка данных пользователя к представлению (ViewHolder)

override fun onBindViewHolder(holder: UserViewHolder, position: Int) {

val currentUser = userList[position]

// Установка имени пользователя в соответствующее поле

holder.textName.text = currentUser.name

// Обработчик события при нажатии на элемент списка (пользователя)

holder.itemView.setOnClickListener{

// Создание и отправка интента для перехода к активности чата

val intent = Intent(context, ChatActivity::class.java)

// Передача данных (имени и UID) пользователя через интент

intent.putExtra("name", currentUser.name)

intent.putExtra("uid", currentUser.uid)

// Запуск активности чата

context.startActivity(intent)

}

}

// ViewHolder для представления пользователя в списке

class UserViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

val textName = itemView.findViewById<TextView>(R.id.txt\_name)

}

}

ExampleInstrumentedTest.tk

package com.example.dix

import androidx.test.platform.app.InstrumentationRegistry

import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test

import org.junit.runner.RunWith

import org.junit.Assert.\*

/\*\*

\* Instrumented test, which will execute on an Android device.

\*

\* See [testing documentation](http://d.android.com/tools/testing).

\*/

@RunWith(AndroidJUnit4::class)

class ExampleInstrumentedTest {

@Test

fun useAppContext() {

// Context of the app under test.

val appContext = InstrumentationRegistry.getInstrumentation().targetContext

assertEquals("com.example.dix", appContext.packageName)

}

}