**Введение**

На проект по технологической практике была поставлена задача разработка мобильного приложения для автоматизации некоторых процессов школы «MySchool».

Цель задачи заключается в реализации приложения в среде разработки Android Studio с помощью языка программирования Kotlin.

Первый раздел носит название «Структура и общая технологическая схема производства, характеристики основных видов продукции». Этот раздел позволяет понять, как организовано производство, какие продукты выпускаются и какие требования предъявляются к качеству и безопасности продукции. Это помогает обеспечить прозрачность и открытость информации о предприятии, что важно для потенциальных инвесторов, партнеров и потребителей. Второй раздел носит название

Второй раздел «Описание структуры ВЦ, АСУП и производственного отдела, где проходит практика». Этот раздел технической документации описывает структуру ВЦ, АСУП и производственного отдела. В нем указывается общая структура организации, функции и обязанности каждого отдела, их взаимодействие друг с другом, штатное расписание, процессы, оборудование и программное обеспечение, требования к квалификации персонала и действующие процедуры и регламенты.

В третьем разделе «Должностные обязанности оператора ЭВМ, техника-программиста, инженера-программиста» описаны должностные обязанности специалистов в области информационных технологий: оператора ЭВМ, техника-программиста и инженера-программиста. Для каждого из указанных специалистов перечислены основные задачи, которые могут включать ввод и обработку данных, обеспечение сохранности и резервного копирования, поддержку пользователей, разработку и тестирование программного обеспечения, руководство командой разработчиков и другие.

В четвёртом разделе «Анализ задачи», описан процесс изучения исходных данных и требований к проекту, а также выявления возможных проблем и рисков, так же в данном разделе описаны инструменты разработки, диаграмма вариантов использования, модель данных и постановка задачи.

Пятый раздел «Проектирование задачи» в данном разделе описывается и анализируется проблема или задача, которую необходимо решить в рамках проекта. Здесь определяются цели, требования и ограничения проекта, а также выбираются оптимальные методы и подходы для его выполнения.

Шестой раздел «Реализация» технической документации описывает процесс создания продукта, указывает используемые технологии и методы, а также отслеживает изменения и проблемы в ходе разработки.

Седьмой раздел «Тестирование», описывающий полное и функциональное тестирование программы. В нем тестируется каждый пункт меню и операции, выполняемые приложением. Все возможные действия пользователя при работе с программой моделируются, начиная от запуска и до выхода.

Восьмой раздел «Применение» технической документации, который описывает, как и где использовать продукт или услугу, а также какие меры предосторожности следует соблюдать при их использовании.

В «Списке используемых источников» приведен список используемых источников при разработке программного продукта.

В приложении А к пояснительной записке представлен листинг мобильного приложения.

**1 Структура и общая технологическая схема производства, характеристики основных видов продукции**

Технологическая практика проходила в школе ГУО «Квасовская Средняя Школа».

Структура и общая технологическая схема в ГУО «Квасовская Средняя Школа»

первый уровень – директор, главное административное лицо, воплощающее единоначалие и несущее персональную ответственность за все, что делается в образовательном учреждении всеми субъектами управления. На этом же уровне модели находятся высшие органы коллегиального и общественного управления, имеющие тот или иной правовой статус: Управляющий Совет школы, педагогический Совет, органы самоуправления учащихся. Субъекты управления этого уровня обеспечивает единство управляющей системы в целом, определяют стратегическое направление развития образовательного учреждения, всех его подразделений;

второй уровень – первый заместитель директора по УВР образовательного учреждения, заместитель директора по информатизации , заместитель директора по воспитательной работе ,  заместитель директора главный бухгалтер, заместитель директора по хозяйственной части, педагог-организатор, органы, входящие в сферу влияния каждого из членов администрации. Этот уровень выступает звеном опосредованного руководства директора образовательной системой. Его главная функция согласование деятельности всех участников процесса в соответствии с заданными целями, программой и ожидаемыми результатами, то есть добиваться тактического воплощения стратегических задач и прогнозов;

третий уровень - методические объединения. К управленцам этого уровня относятся руководители методических объединений. Взаимодействие субъектов управления этого уровня осуществляется через специализацию функций при их одновременной интеграции. Руководство на этом уровне основано преимущественно на личных контактах, осуществляется с учетом индивидуальных особенностей и не формализовано;

четвертый уровень - учащиеся, родители и учителя. Развитие самоуправления на этом уровне обеспечивает реализацию принципа демократизации. Участие детей в управляющей системе формирует их организаторские способности и деловые качества.

**2 Описание структуры отдела программного обеспечения, где проходит практика**

Технологическая практика проходила в отделе информатизации предприятия ГУО «Квасовская Средняя Школа».

Сектор системного администрирования осуществляет администрирование серверного оборудования, систем хранения данных и сетевого оборудования.

Отдел находится в подчинении заместителя директора. Отделом руководит начальник отдела информатизации. В его подчинении находятся системные администраторы.

**3 Должностные обязанности системного администратора**

Должностные обязанности системного администратора определяются его квалификацией и уровнем ответственности, а также спецификой деятельности предприятия. В целом, к основным обязанностям системного администратора относятся:

Администрирование серверов, систем хранения данных и сетевого оборудования

Поддержка работы программного обеспечения.

Обеспечение информационной безопасности.

Обслуживание пользователей.

Разработка и внедрение новых информационных технологий.

Администрирование серверов, систем хранения данных и сетевого оборудования.

Системный администратор отвечает за установку, настройку и обслуживание серверного оборудования, систем хранения данных и сетевого оборудования. Он должен быть знаком с принципами работы этих систем, уметь их конфигурировать и поддерживать в работоспособном состоянии.

Поддержка работы программного обеспечения.

Системный администратор отвечает за установку, настройку и поддержку работы программного обеспечения, используемого на предприятии. Он должен быть знаком с различными операционными системами, офисными пакетами, приложениями для обработки данных и другими видами программного обеспечения.

Обеспечение информационной безопасности.

Системный администратор отвечает за обеспечение информационной безопасности на предприятии. Он должен знать основные принципы информационной безопасности, уметь выявлять и устранять уязвимости в системах, а также защищать данные от несанкционированного доступа.

Обслуживание пользователей.

Системный администратор отвечает за обслуживание пользователей, использующих компьютеры и программное обеспечение на предприятии. Он должен быть готов оказать помощь пользователям в решении их проблем, связанных с использованием информационных технологий.

Разработка и внедрение новых информационных технологий.

Системный администратор может участвовать в разработке и внедрении новых информационных технологий на предприятии. Он должен быть в курсе последних тенденций в области информационных технологий и уметь применять их на практике.

Кроме того, системный администратор может выполнять и другие обязанности, связанные с его квалификацией и уровнем ответственности. Например, он может отвечать за управление ИТ-проектами, обучение пользователей или проведение аудита информационных систем.

**4 Анализ задачи**

**4.1 Анализ предметной области**

В понятие разработки мобильных приложений для смартфонов, планшетов и прочих мобильных устройств входит написание программного кода с целью создания программ, которые будут работать на определенных мобильных платформах (на сегодняшний день существует 2 основные платформы мобильных операционных систем — Android и iOS, и менее популярные Windows Phone и Symbian). Эти программы и приложения могут предварительно устанавливаться на мобильные телефоны, персональные цифровые помощники, корпоративные цифровые ассистенты, смартфоны и прочие мобильные устройства до того, как устройства попадут в руки пользователю, либо загружаться пользователями в устройство непосредственно в процессе использования.

«MySchool» — это мобильное приложение для Android, которое автоматизирует некоторые процессы школы. В данном приложении пользователи могут смотреть и размещать новости школы, смотреть электронный журнал школы, выставлять оценки, смотреть домашнее задание и размещать домашнее задание, смотреть статистику ученика.

Пользоваться приложением представляется возможным только после регистрации пользователя и одобрении администратором внутри приложения. Для регистрации пользователю потребуется внести личные данные в форму и отправить данную форму на проверку администратору.

**4.2 Инструменты разработки**

Для реализации проекта будет выбран фреймворк Android Studio. Так же будут использованы другие инструменты разработки:

* Firebase - это облачная база данных, которая позволяет пользователям хранить и получать сохраненную информацию, а также имеет удобные средства и методы взаимодействия с ней;
* Microsoft Word – программа, используемая для создания документации к программному продукту.

Для оптимального функционирования разрабатываемого приложния необходимо наличие у телефонов и планшетов следующих параметров:

* 1 GB RAM;
* android 6.0 и выше;
* стабильный интернет.

**4.3 Диаграмма вариантов использования**

Диаграмма вариантов использования – это инструмент UML для описания функциональных требований системы и ее взаимодействия с внешними сущностями (актёрами) в возможных сценариях использования.

Диаграмма вариантов использования используется для определения функциональных требований к системе, которые должны быть реализованы. Диаграмма описывает варианты использования системы, которые могут возникнуть при взаимодействии пользователей с ней. В центре диаграммы находится система, а вокруг нее располагаются актеры – пользователи, которые взаимодействуют с системой. Для каждого актера определяются варианты использования системы, в соответствии с его потребностями.

Актёр — это любой человек, группа людей или внешняя система, которые участвуют в определенном процессе или взаимодействии с системой (Рисунок 1).

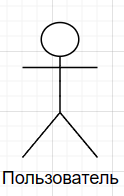


Рисунок 1 – Актёр

Вариант использования – это список конкретных шагов, необходимых для достижения определенной цели, такой как настройка устройства или использование программного обеспечения изображено на рисунке 2. На диаграмме вариантов использования отображаются все возможные варианты использования системы, что позволяет определить и проанализировать функциональные требования.

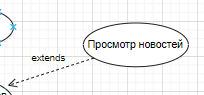


Рисунок 2 - Вариант использования

В данном веб-ресурсе выделены следующие варианты использования для пользователя:

* Авторизация;
* Регистрация;
* Просмотр фрагментов приложения;
* Просмотр новостей;
* Просмотр электронного журнала;
* Просмотр домашнего задания;
* Просмотр своей статистики или статистики своего ребенка;
* Просмотр своего профиля;

Для администратора:

* Авторизация;
* Создание новости;
* Удаление новости;
* Выставить оценку;
* Разместить домашнее задание;
* Просмотр нерассмотренных заявок на регистрацию аккаунта;
* Одобрение или отклонение заявки на регистрацию аккаунта.

Отношение — это описание взаимодействия между актерами и системой, представленное в графическом виде с помощью стрелок. Отношения в диаграмме помогают разработчикам понимать, какие действия могут выполнять пользователи или другие системы в системе, и как система должна реагировать на эти действия.

Отношения ассоциации в диаграмме вариантов использования (Use Case) представляют связи между элементами диаграммы, например, между акторами и вариантами использования или между различными вариантами использования изображено на рисунке 3. Они могут быть двусторонними или односторонними и указывать на связь, обмен информацией или взаимодействие между различными частями системы. Отношения ассоциации могут быть представлены стрелками, которые указывают направление связи или без стрелок, если связь двусторонняя.

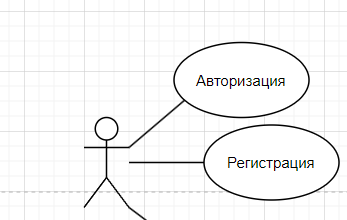


Рисунок 3 - Отношение ассоциации

Отношение дополнения в диаграмме вариантов использования используется для связывания двух взаимоисключающих вариантов использования изображено на рисунке 4. Это означает, что если один из вариантов использования выполняется, то другой не может быть выполнен. Отношение дополнения обозначается открытой стрелкой, указывающей на вариант использования, который является дополнением.

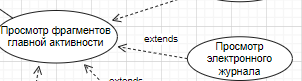


Рисунок 4 – Отношение дополнения

Отношение включения в диаграмме вариантов использования указывает на то, что один вариант использования может быть включен в другой вариант использования изображено на рисунке 5. Это означает, что выполнение первого варианта использования является необходимым для выполнения второго варианта использования.

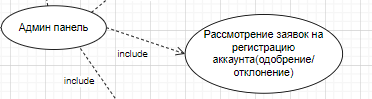


Рисунок 5 – Отношение включения

Через анализ вариантов использования и связей между ними для конкретного актера, мы составим полную диаграмму вариантов использования изображено на рисунке 6.

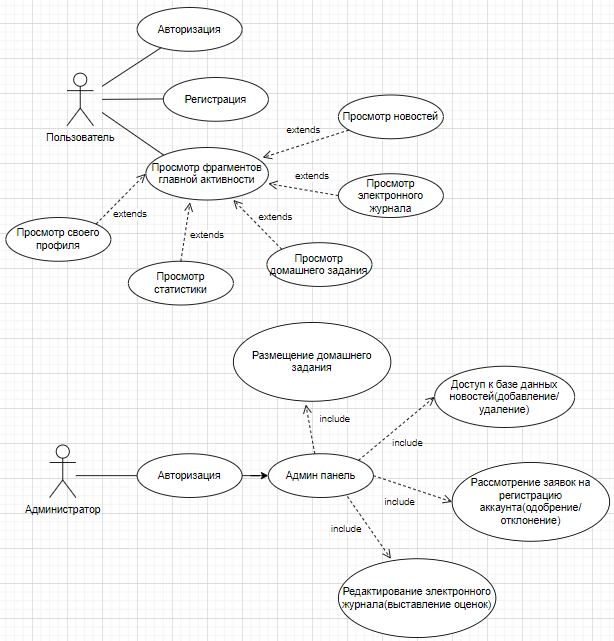


Рисунок 6 – Диаграмма вариантов использования

**4.4 Модель данных**

Модель данных — это абстрактное представление структуры, связей и характеристик данных, используемых в информационной системе или базе данных. Она определяет, как данные будут организованы, храниться и обрабатываться. Модель данных описывает сущности, их атрибуты, связи между ними и ограничения на данные. В зависимости от способа представления структуры данных, существуют различные модели данных, такие как иерархическая, сетевая, реляционная, объектно-ориентированная и другие. Каждая модель данных имеет свои специфические характеристики и применяется в определенных ситуациях в зависимости от требований проекта.

Сущности — это основные объекты данных в модели. Они представляют реальные объекты или концепции, которые хранятся и обрабатываются в системе. Например, в модели данных для интернет-магазина могут быть сущности "пользователь", "товар", "заказ".

Связи — это связи или отношения между сущностями. Они определяют, как сущности взаимодействуют друг с другом. Например, в модели данных для интернет-магазина может быть связь "заказ" между сущностями "пользователь" и "товар", чтобы отразить тот факт, что пользователь делает заказ на определенный товар.

Модель данных изображена на рисунке 7.

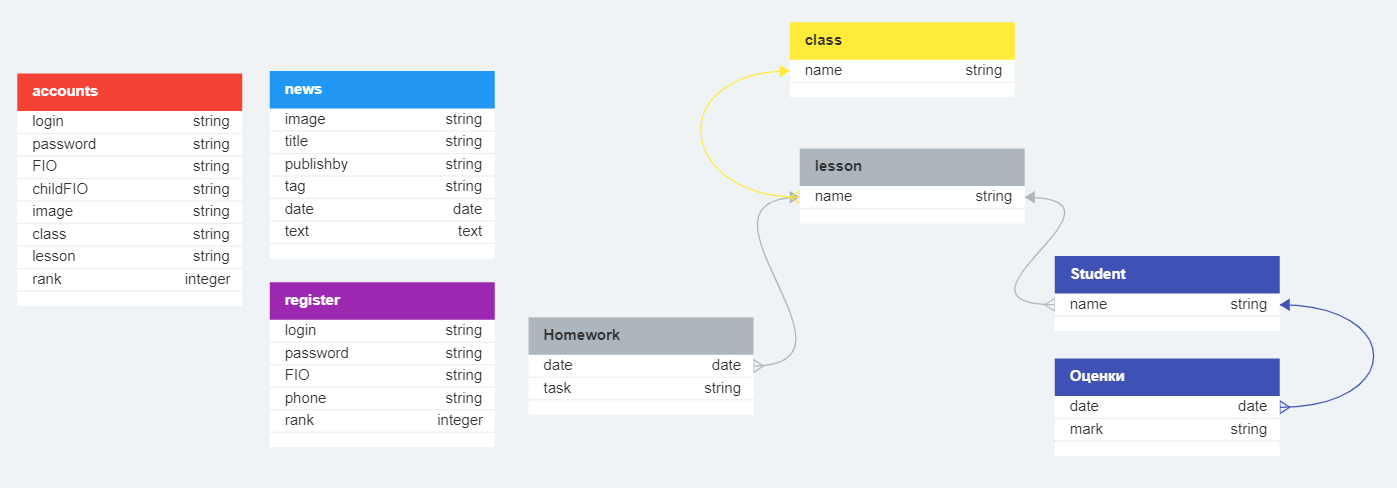


Рисунок 7 - Модель данных

**4.5 Постановка задачи**

Наименование задачи: Мобильное приложение по автоматизации некоторых процессов школы «MySchool».

Цель задачи: создание приложения, в котором пользователь может смотреть новости, электронный журнал, домашнее задание, статистику и свой профиль.

Назначение: мобильное приложение предназначено для автоматизации и в последствии, упрощении получения информации, так или иначе связанной со школой.

Периодичность использования: в любое время;

Источники и способы получения данных: интернет-ресурсы;

Обзор существующих аналогичных программных продуктов: рассмотрим мобильное приложение под название Schools.by. В данном приложении пользователь может смотреть дневник, новости и другое.

Описание перечня функций и задач, которые должно выполнять приложение:

Пользователь:

* регистрация;
* авторизация;
* выход;
* просмотр новостей;
* просмотр электронного журнала;
* просмотр домашнего задания;
* просмотр личной статистики.

Администратор:

* авторизация;
* создание и удаление новости;
* выставлять оценки;
* размещать Д/З;
* рассматривать заявки на регистрацию, в том числе одобрять или отклонять их;
* просмотр информации о пользователях.

Описание процессов с входной, выходной и условно-постоянной информацией:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Таблица 1 – Описание входной и выходной информации | | | | | |
| Категория пользователей | Наименование процесса | Краткое описание алгоритма выполнения процесса | Входная информация | Выходная информация | Условно - постоянная информация |
| Пользователь | Авторизация | Вход в личный кабинет/ввод логина и пароля | Отсутствует | Отсутствует | Авторизация |
| Пользователь | Регистрация | Создание учётной записи | Отсутствует | Отсутствует | Регистрация |
| Пользователь | Просмотр товаров | Просмотр новости | Отсутствует | Получение информации о новости | Информация в приложении |
| Пользователь | Просмотр фрагментов приложения | Просмотр приложение | Фрагмент приложения | Отсутствует | Получение информации о фрагментах приложения |
| Администратор | Авторизация | Вход в админ панель | Отсутствует | Отсутствует | Вход в админ панель |
| Администратор | Размещения домашнего задания | Добавление Д/З в базу данных | Отсутствует | Д/З | База данных |
| Администратор | Добавление пользователей | Добавление нового пользователя | Отсутствует | Отсутствует | База данных |
| Администратор | Добавление новости | Добавление новости | Отсутствует | Отсутствует | База данных |
| Администратор | Доступ к базе данных электронного журнала и редактирование | Доступ к базе данных приложения и редактирование данных | Отсутствует | Отсутствует | Просмотр базы данных |
| Администратор | Доступ к базе данных пользователей и редактирование | Доступ к базе данных приложения и редактирование данных | Отсутствует | Отсутствует | Просмотр базы данных |

Требования к применению: приложение должно показывать наличие различных новостей, электронный журнал, домашнее задание и статистику.

Требования к реализации: для реализации статических страниц и шаблонов должен использоваться язык XML. Для реализации интерактивных элементов клиентской части должен использоваться язык Kotlin. Для реализации базы данных должен быть использован язык программирования SQL.

Описание пользовательского интерфейса: при разработке веб-ресурса должны быть использовано несколько цветовых оттенков для грамотной палитры цветов. Основные разделы сайта должны быть доступны с первой страницы. Грамотный пользовательский интерфейс.

Требования к хостингу:

* Поддерживает любые CMS;
* Круглосуточный мониторинг серверов 24/7: сбои устраняются раньше, чем их можно заметить;
* Объем дискового пространство 200Мб;
* Эффективная защита от спама и взлома;
* Базы данных хранятся на SSD-дисках;
* Максимальный размер базы данных - 2 Гб;
* Все данные проходят через процедуру ежедневного резервного копирования;
* Копии хранятся в течение 20 дней;

**5 Проектирование задачи**

Для проектирования задачи выбрана среда разработки Android Studio. Данная среда разработки позволит организовать логическую структуру приложения. Пользовательский интерфейс будет реализован просто и понятно для пользователя любого возраста. Админ панель так же имеет просто и понятный интерфейс.

Создание самого приложения осуществляется с помощью файлов с расширением .kt и .xml для реализации интерфейса. При связывании интерфейса и функциональной части на выходе выходит готовое приложение.

При первом входе в приложение пользователю откроется активность для авторизации, после авторизации последующие входы будут переадресовывать нас сразу в основную активность. Активность для авторизации изображена на рисунке 8.

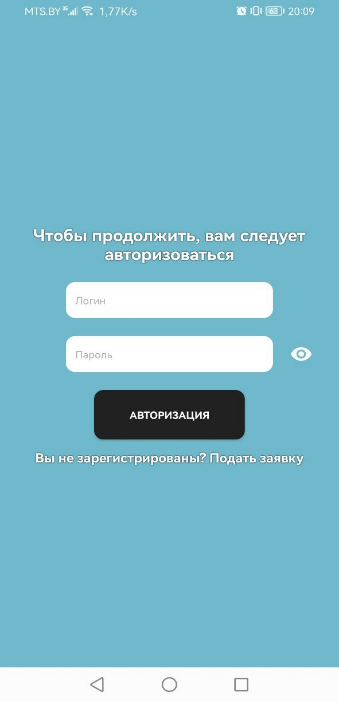


Рисунок 8 – Активность авторизации

После авторизации мы становимся полноценным пользователем приложения и нас переадресовывает в основную активность. Основная активность изображена на рисунке 9.

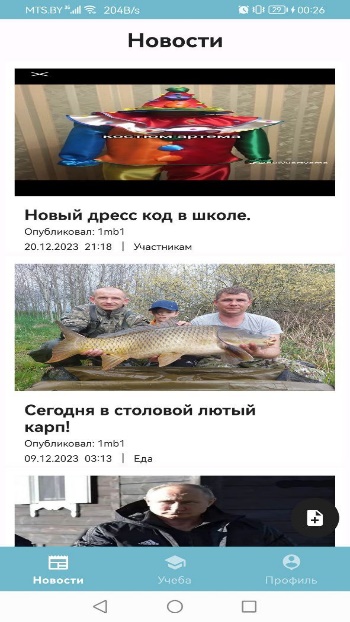


Рисунок 9 – Основная активность

Интерфейс полной новости состоит из: картинка, заголовок, автор, дата, тэг, текст. Интерфейс полной новости изображён на рисунке 10.

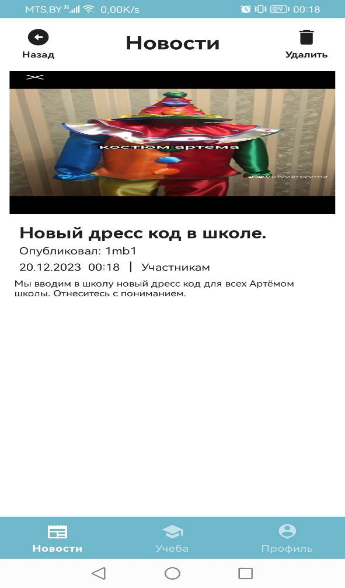


Рисунок 10 – Интерфейс новости

**6 Реализация**

Для реализации проекта выбрана среда разработки Android Studio. С помощью данного инструмента реализованы пользовательский интерфейс и функциональная часть приложения. Данная среда разработки позволяет удобно редактировать функциональную и визуальную часть приложения.

Модули веб-ресурса:

* activity\_login.xml – файл разметки активности с авторизацией;
* activity\_main.xml – файл разметки основной активности;
* fragment\_news.xml – файл разметки фрагмента с новостями;
* fragment\_study.php – файл разметки фрагмента с электронным журналом, домашним заданием и статистикой;
* fragment\_news.xml – файл разметки фрагмента с профилем и админ панелью;
* LoginActivity.kt – файл для реализации функциональной составляющей активности с авторизацией;
* MainAcitivty.kt – файл для реализации функциональной составляющей основной активности;
* NewsFragment.kt – файл для реализации функциональной составляющей фрагмента с новостями;
* StudyFragment.kt – файл для реализации функциональной составляющей фрагмента с электронным журналом, домашним заданием и статистикой;
* ProfileFragment.kt – файл для реализации функциональной составляющей фрагмента с профилем и админ панелью;
* SchoolDbHelper.kt – класс, который создает, пересоздает или удаляет локальную базу данных, хранящую в себе « куки » файлы, которые определяют, вошли вы в аккаунт или нет;
* SchoolDbNames.kt – класс, который хранит в себе команды создания и удаления таблиц;
* SchoolDbManager.kt – класс в котором реализованы пользовательские функции для упрощения работы с локальной базы данных;
* ClassAdapter.kt – класс в котором реализованы пользовательские настройки адаптера для RecyclerView;
* HomeworkAdapter.kt – класс в котором реализованы пользовательские настройки адаптера для RecyclerView;
* NewsAdapter.kt – класс в котором реализованы пользовательские настройки адаптера для RecyclerView;
* RegAdapter.kt – класс в котором реализованы пользовательские настройки адаптера для RecyclerView;
* StudentAdapter.kt – класс в котором реализованы пользовательские настройки адаптера для RecyclerView;
* CustomToast.kt – класс в котором реализованы кастомные настройки всплывающих сообщений.

Спецификация файлов с расширением .xml является тем, что в данном файле разметка активностей и фрагментов.

Спецификация файлов с расширением .kt является тем, что в данном файле находится код для реализации функциональной части приложения. Данный файл позволяет связать все фрагменты приложения и позволяет корректно работать приложению.

**7 Тестирование**

Таблица 2 – отчёт о дефектах и проделанных тест-кейсах.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Иде-нтификатор | Приоритет | Модуль ресурса | Название | Шаги | Ожидаемый результат | Результ-ат |
| TС1 | Высокий | Активность авторизации | Авторизация | 1 Ввести правильный логин  2 Ввести правильный пароль  3 Войти | Отображение главной активности | Выполнено |
| TС2 | Высокий | Основная активность,  фрагмент новостей | Просмотр новостей | Открыть новости | Отображение новостей | Выполнено |
| ТС3 | Высокий | Просмотр полной новости | Просмотр фрагмента новости | Нажать на новость | Отображение полной новости | Выполнено |
| TС4 | Высокий | Активность авторизации | Проверка работы фрагмента регистрации | 1 Отрыть форму регистрации  2 Ввести правильные данные  3 Отправить заявку | Всплывающее окно, говорящее нам о том, что заявка отправлена успешно | Выполнено |
| TС5 | Высокий | Основная активность,  фрагмент учеба | Проверка фрагмента учеба | 1 Открыть электронный журнал  2 Закрыть электронный журнал  3 Открыть домашнее задание  4 Закрыть домашнее задание  5 Открыть статистику  6 Закрыть статистику | Успешное открытие и закрытие отдельных фрагментов | Выполнено |
| TC6 | Высокий | Основная активность,  фрагмент новостей | Проверка создания и публикации новости | 1 Открыть форму создания новости  2 Заполнить поля правильными данными  3 Опубликовать | Осуществляется публикация новости | Выполнено |
|  |  |  |  |  |  |  |
| Продолжение - отчёт о дефектах и проделанных тест-кейсах. | | | | | | |
| ТС7 | Высокий | Основная активность,  фрагмент профиль | Проверка одобрения заявки на регистрацию | 1 открыть админ панель  2 Выбрать пункт « заявки »  3 Нажать продолжить на нужной заявке  4 Заполнить недостающую информацию  5 Одобрить | В базе данных создается аккаунт и в последствии на него можно войти | Выполнено |
| ТС8 | Высокий | Основная активность,  фрагмент учеба | Проверка взаимодействия с базой данных классов | 1 Открыть электронный журнал  2 Выбрать класс  3 Выбрать ученика  4 Нажать « Оценка »  5 Заполнить форму  6 Выставить | Осуществляется взаимодействие с базой данных классов | Выполнено |

**8 Применение**

Руководство пользователя включает в себя следующие характеристики.

Данное приложение разработано под любое Android устройство.

Наименование приложения: « MySchool ». Приложение разработано в среде разработки Android Studio с помощью языков программирования Kotlin и расширяемого языка программирования XML.

Назначение выполняемого приложения:

* + Слежка за новостями школы или публикация новых;
  + Возможность вести и следить за журналом классов;
  + Возможность смотреть домашние задания;
  + Возможность смотреть статистику;

На этапе тестирования программного продукта были выявлены минимальные системные требования, которые позволят работать приложению без задержек.

Быстродействие данного приложения во многом зависит от скорости

подключенного к выбранному для эксплуатации мобильному устройству интернета. Несмотря на все реализованные задачи, сайт легко запускается и функционирует на любых устройствах.

Минимальные системные требования для других гаджетов:

* + Разрешение экрана от 320 пикселей;
  + 100MHz процессор;
  + 512 RAM.

Руководство программиста включается в себя следующие характеристики.

Входными данными приложения являются файлы составляющие структуру ресурса, база данных, хранящая данные пользователей, информации о товарах.

Выходными данными являются визуальная составляющая приложения.

XML разметка фрагментов и активностей располагается в файлах acitivity\_login.xml, activity\_main.xml, fragment\_news.xml, fragment\_study.xml, fragment\_profile.xml .

Kotlin код реализующий функционал располагается в файлах LoginActivity.kt, MainActivity.kt, NewsFragment.kt, StudyFragment.kt, ProfileFragment.kt, SchoolDbNames.kt, SchoolDbHelper.kt, SchoolDbManager.kt, ClassAdapter.kt, StudentAdapter.kt, RegAdapter.kt, NewsAdapter.kt, HomeworkAdapter.kt, CustomToast.kt.

Спецификация файлов приведена в таблице 3.

|  |  |
| --- | --- |
| Таблица 3 – Спецификация файлов | |
| Имя файла | Назначение |
| acitivity\_login.xml | В данном файле находится разметка активности авторизации |
| activity\_main.xml | В данном файле находится разметка основной активности |
| fragment\_news.xml | В данном файле находится разметка фрагмента новостей |
| fragment\_study.xml | В данном файле находится разметка фрагмента учебы |
| fragment\_profile.xml | В данном файле находится разметка фрагмента профиля |
| LoginActivity.kt | В данном файле реализована функциональная составляющая активности с авторизацией |
| MainActivity.kt | В данном файле реализована функциональная составляющая основной активности |
| NewsFragment.kt | В данном файле реализована функциональная составляющая фрагмента с новостями |
| StudyFragment.kt | В данном файле реализована функциональная составляющая фрагмента с учебой |
| ProfileFragment.kt | В данном файле реализована функциональная составляющая фрагмента с профилем |
| SchoolDbNames.kt | В данном файле хранится класс, который хранит в себе команды создания и удаления таблиц |
| SchoolDbHelper.kt | В данном файле хранится класс, который создает, пересоздает или удаляет локальную базу данных, хранящую в себе « куки » файлы, которые определяют, вошли вы в аккаунт или нет |
| SchoolDbManager.kt | В данном файле хранится класс, который хранит в себе команды создания и удаления таблиц; |
| ClassAdapter.kt, StudentAdapter.kt, RegAdapter.kt, NewsAdapter.kt, HomeworkAdapter.kt | В данном файле хранится класс в котором реализованы пользовательские настройки адаптера для RecyclerView; |
| CustomToast.kt | В данном файле хранится класс в котором реализованы кастомные настройки всплывающих сообщений. |

**Заключение**

Целью проекта по технологической практике была проектирование мобильного приложения для автоматизации некоторых процессов школы «MySchool».

Особенность данного задания заключалась в том, чтобы, реализовать мобильное приложение с помощью Android Studio.

В поставленной задаче был реализован простой и понятный пользовательский интерфейс.

В ходе реализации поставленной задачи был укреплён пройденный курс программирования в среде разработки Android Studio, а также получено много дополнительной информации о Kotlin, XML, SqlLite.

После тестирования мобильного приложения были выявлены недоработки, которые были по большей части исправлены на стадии проектирования, и полностью исключены на стадии тестирования приложения. При реализации приложения, были выполнены все условия, перечисленные в предыдущих разделах пояснительной записки.

# Список использованных источников

* Android Studio [Электронный ресурс]. – Электронные данные. Режим доступа: https://developer.android.com/studio;
* Firebase [Электронный ресурс]: – Электронные данные. Режим доступа: https://firebase.google.com/docs/database?hl=ru.

**Приложение А**

**Листинг программы**

package com.holysource.myschool.Activities  
  
import android.annotation.SuppressLint  
import android.app.Activity  
import android.content.Context  
import android.content.Intent  
import android.os.Bundle  
import android.text.InputType  
import android.util.Log  
import android.view.View  
import android.view.animation.Animation  
import android.view.animation.AnimationUtils  
import android.view.inputmethod.InputMethodManager  
import android.widget.Button  
import android.widget.EditText  
import android.widget.ImageButton  
import android.widget.LinearLayout  
import android.widget.Spinner  
import android.widget.TextView  
import android.widget.Toast  
import androidx.appcompat.app.AppCompatActivity  
import com.google.firebase.firestore.ktx.firestore  
import com.google.firebase.ktx.Firebase  
import com.holysource.myschool.Adapters.CustomToast  
import com.holysource.myschool.R  
import com.skydoves.powerspinner.PowerSpinnerView  
import com.squareup.picasso.Picasso  
import db.SchoolDbManager  
import kotlin.math.log  
  
class LoginActivity : AppCompatActivity() {  
  
 var count = 0  
 val fdb = Firebase.*firestore* val db = SchoolDbManager(this)  
   
 private lateinit var mLoginMenu: LinearLayout  
 private lateinit var mLoginField: EditText  
 private lateinit var mPasswordField: EditText  
 private lateinit var mPasswordVisiblityButton: ImageButton  
 private lateinit var mTryToLoginButton: Button  
 private lateinit var mToRegisterMenuButton: TextView  
 private lateinit var mRegisterMenu: LinearLayout  
 private lateinit var mRegisterLoginField: EditText  
 private lateinit var mRegisterPasswordField: EditText  
 private lateinit var mRegisterPasswordVisiblityButton: ImageButton  
 private lateinit var mRegisterAgainPasswordField: EditText  
 private lateinit var mRegisterFIOField: EditText  
 private lateinit var mRegisterRankSpinner: PowerSpinnerView  
 private lateinit var mRegisterPhoneField: EditText  
 private lateinit var mTryToRegisterButton: Button  
 private lateinit var mToLoginMenuButton: TextView  
  
 @SuppressLint("MissingInflatedId")  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 db.openSchoolDb()  
 val cookies = db.Cookies()  
 db.closeSchoolDb()  
 if(cookies.get("IsAuth")=="true") {  
 checkAuth(login = cookies.get("Login")!!, password = cookies.get("Password")!!)  
 } else {  
 setContentView(R.layout.*activity\_login*)  
  
val animAlpha: Animation = AnimationUtils.loadAnimation(*applicationContext*, R.anim.*alpha*)  
 val animScale: Animation = AnimationUtils.loadAnimation(*applicationContext*, R.anim.*scale*)  
  
 mLoginMenu = findViewById(R.id.*login\_menu*)  
 mRegisterMenu = findViewById(R.id.*register\_menu*)  
 mLoginField = findViewById(R.id.*login\_field*)  
 mRegisterLoginField = findViewById(R.id.*register\_login\_field*)  
 mPasswordField = findViewById(R.id.*password\_field*)  
 mPasswordVisiblityButton = findViewById(R.id.*password\_visiblity\_button*)  
 mPasswordVisiblityButton.setOnClickListener **{  
 it**.startAnimation(animScale)  
 if(mPasswordField.*inputType*!= InputType.*TYPE\_CLASS\_TEXT* or InputType.*TYPE\_TEXT\_VARIATION\_PASSWORD*) {  
 mPasswordField.*inputType* = InputType.*TYPE\_CLASS\_TEXT* or InputType.*TYPE\_TEXT\_VARIATION\_PASSWORD* } else {  
 mPasswordField.*inputType* = InputType.*TYPE\_TEXT\_VARIATION\_VISIBLE\_PASSWORD* }  
 mPasswordField.clearFocus()  
 **}** mRegisterPasswordField = findViewById(R.id.*register\_password\_field*)  
 mRegisterAgainPasswordField = findViewById(R.id.*register\_password\_again\_field*)  
 mRegisterPasswordVisiblityButton = findViewById(R.id.*register\_password\_visiblity\_button*)  
 mRegisterPasswordVisiblityButton.setOnClickListener **{  
 it**.startAnimation(animScale)  
 if(mRegisterPasswordField.*inputType*!= InputType.*TYPE\_CLASS\_TEXT* or InputType.*TYPE\_TEXT\_VARIATION\_PASSWORD*) {  
 mRegisterPasswordField.*inputType* = InputType.*TYPE\_CLASS\_TEXT* or InputType.*TYPE\_TEXT\_VARIATION\_PASSWORD* mRegisterAgainPasswordField.*inputType* = InputType.*TYPE\_CLASS\_TEXT* or InputType.*TYPE\_TEXT\_VARIATION\_PASSWORD* } else {  
 mRegisterPasswordField.*inputType* = InputType.*TYPE\_TEXT\_VARIATION\_VISIBLE\_PASSWORD* mRegisterAgainPasswordField.*inputType* =  
 InputType.*TYPE\_TEXT\_VARIATION\_VISIBLE\_PASSWORD* }  
 mRegisterPasswordField.clearFocus()  
 mRegisterAgainPasswordField.clearFocus()  
 **}** mRegisterFIOField = findViewById(R.id.*register\_fio\_field*)  
 mRegisterRankSpinner = findViewById(R.id.*register\_rank\_spinner*)  
 mRegisterPhoneField = findViewById(R.id.*register\_return\_call\_field*)  
 mTryToLoginButton = findViewById(R.id.*try\_to\_login\_button*)  
 mTryToLoginButton.setOnClickListener **{  
 it**.startAnimation(animAlpha)  
 if(mLoginField.*text*.*isNotEmpty*() && mPasswordField.*text*.*isNotEmpty*()) {  
 if(mLoginField.*text*.length>=4 && mPasswordField.*text*.length>=6) {  
 login(mLoginField.*text*.toString(),mPasswordField.*text*.toString())  
 mPasswordField.clearFocus()  
 mPasswordField.*text*.clear()  
 mLoginField.clearFocus()  
 mLoginField.*text*.clear()  
 } else {  
 if(mLoginField.*text*.length<4) {  
 toast("Упс... Логин состоит минимум из 4 символов!",null)  
 mLoginField.startAnimation(animScale)  
 } else {  
 toast("Упс... Пароль состоит минимум из 6 символов!",null)  
 mPasswordField.startAnimation(animScale)  
 }  
 }  
 } else {  
 if(mLoginField.*text*.*isEmpty*()) {  
 mLoginField.startAnimation(animScale)  
 } else {  
 mPasswordField.startAnimation(animScale)  
 }  
 toast("Упс... Кажется не все поля заполены!",null)  
 }  
 hideKeyboardFrom(*applicationContext*,mTryToLoginButton)  
 **}** mTryToRegisterButton = findViewById(R.id.*try\_to\_register\_button*)  
 mTryToRegisterButton.setOnClickListener **{  
 it**.startAnimation(animAlpha)  
 if(mRegisterLoginField.*text*.*isNotEmpty*() && mRegisterPasswordField.*text*.*isNotEmpty*() && mRegisterAgainPasswordField.*text*.*isNotEmpty*() && mRegisterFIOField.*text*.*isNotEmpty*() && mRegisterRankSpinner.selectedIndex>=0 && mRegisterPhoneField.*text*.*isNotEmpty*()) {  
 if(mRegisterPasswordField.*text*.toString()==mRegisterAgainPasswordField.*text*.toString()) {  
 if(mRegisterLoginField.*text*.length>=4) {  
 if(mRegisterPasswordField.*text*.length>=6) {  
 register(mRegisterLoginField.*text*.toString(),mRegisterPasswordField.*text*.toString(),mRegisterFIOField.*text*.toString(),(mRegisterRankSpinner.selectedIndex+1).toString(),mRegisterPhoneField.*text*.toString())  
 } else {  
 toast("Пароль должен состоять минимум из 6 символов!",null)  
 }  
  
 } else {  
 toast("Логин должен состоять минимум из 4 символов",null)  
 }  
 } else {  
 toast("Пароли не совпадают. Попробуйте еще раз.",null)  
 }  
 } else {  
 toast("Упс... Кажется не все поля заполены!",null)  
 }  
 hideKeyboardFrom(*applicationContext*,mTryToRegisterButton)  
 **}** mToRegisterMenuButton = findViewById(R.id.*to\_register\_menu\_button*)  
 mToRegisterMenuButton.setOnClickListener **{  
 it**.startAnimation(animScale)  
 mLoginMenu.*visibility* = View.*GONE* mRegisterMenu.*visibility* = View.*VISIBLE* **}** mToLoginMenuButton = findViewById(R.id.*to\_login\_menu\_button*)  
 mToLoginMenuButton.setOnClickListener **{  
 it**.startAnimation(animScale)  
 mLoginMenu.*visibility* = View.*VISIBLE* mRegisterMenu.*visibility* = View.*GONE* **}** }  
 }  
  
 fun toast(text: String, isDone: Boolean?) {  
 CustomToast.makeText(*applicationContext*, text, isDone).show()  
 }  
  
 fun checkAuth(login: String,password: String) {  
 var ErrorCode = 1  
 var FIO = ""  
 var childFIO = ""  
 var rank = ""  
 var clas = ""  
 var lesson = ""  
 var image = "null"  
 fdb.collection("accounts").document(login).get().addOnSuccessListener **{** result **->** if(password==result.*data*?.get("password")) {  
 ErrorCode = 0  
 FIO = result.*data*!!.get("FIO").*toString*()  
 rank = result.*data*!!.get("rank").*toString*()  
 clas = result.*data*!!.get("class").*toString*()  
 lesson = result.*data*!!.get("lesson").*toString*()  
 image = result.*data*!!.get("image").*toString*()  
 childFIO = result.*data*!!.get("childFIO").*toString*()  
 }  
 db.openSchoolDb()  
 if(ErrorCode==0) {  
 db.updateCookies(login,password,FIO,childFIO,rank,clas,lesson,image,"true")  
 db.closeSchoolDb()  
 val intent = Intent(this@LoginActivity, MainActivity::class.*java*)  
 startActivity(intent)  
 } else {  
 db.updateCookies("","","","","","","","null","false")  
 db.closeSchoolDb()  
 super.onCreate(null)  
 }  
 **}**.addOnFailureListener **{** Log.e("TAG", "Ошибка: ${**it**}")  
 **}** }  
  
 fun register(login: String,password: String,fio: String,rank: String,phone: String) {  
  
 val reg = *hashMapOf*(  
 "login" *to* login,  
 "password" *to* password,  
 "FIO" *to* fio,  
 "rank" *to* rank,  
 "phone" *to* phone,  
 )  
 fdb.collection("accounts").whereEqualTo("login",login).get().addOnSuccessListener **{** result **->** if(result.size()==0) {  
 fdb.collection("register").whereEqualTo("login",login).get().addOnSuccessListener **{** res **->** if(res.size()==0) {  
  
 fdb.collection("register").document(login).set(reg)  
 .addOnSuccessListener **{** toast("Вы оставили заявку. Ждите, пока администратор рассмотрит ее.",true)  
 **}** .addOnFailureListener **{** toast("Произошла неизвестная ошибка. Попробуйте еще раз.",false)  
 **}** } else {  
 toast("Заявка на аккаунт с таким логином уже существует",false)  
 }  
  
 **}**.addOnFailureListener **{** toast("Произошла неизвестная ошибка. Попробуйте еще раз.",false)  
 **}** }  
  
 **}**.addOnFailureListener **{** toast("Аккаунт с таким логином уже существует.",false)  
 **}** }  
  
 fun login(login: String,password: String) {  
 val animScale: Animation = AnimationUtils.loadAnimation(*applicationContext*, R.anim.*scale*)  
 var ErrorCode = 2  
 var FIO = ""  
 var childFIO = ""  
 var rank = 0  
 var clas = ""  
 var lesson = ""  
 var image = "null"  
 fdb.collection("accounts").document(login).get().addOnSuccessListener **{** result **->** if(login==result.*data*?.get("login")) {  
 if(password==result.*data*?.get("password")) {  
 ErrorCode = 0  
 FIO = result.*data*!!.get("FIO").*toString*()  
 rank = result.*data*!!.get("rank").*toString*().*toInt*()  
 clas = result.*data*!!.get("class").*toString*()  
 lesson = result.*data*!!.get("lesson").*toString*()  
 image = result.*data*!!.get("image").*toString*()  
 childFIO = result.*data*!!.get("childFIO").*toString*()  
 } else {  
 ErrorCode = 1  
 }  
 }  
 when(ErrorCode) {  
 0 -> {  
 db.openSchoolDb()  
 db.updateCookies(login,password,FIO,childFIO,rank.toString(),clas,lesson,image,"true")  
 db.closeSchoolDb()  
 when(rank) {  
 1 -> {  
 toast("Вы авторизовались как Ученик",true)  
 }  
 2 -> {  
 toast("Вы авторизовались как Родитель",true)  
 }  
 3 -> {  
 toast("Вы авторизовались как Преподаватель",true)  
 }  
 4 -> {  
 toast("Вы авторизовались как Администратор",true)  
 }  
 }  
 val intent = Intent(this@LoginActivity, MainActivity::class.*java*)  
 startActivity(intent)  
 }  
 1 -> {  
 toast("Упс... Вы ввели неправильный пароль! Попробуйте еще раз.",false)  
 mPasswordField.startAnimation(animScale)  
 }  
 2 -> {  
 toast("Такого аккаунта не существует. Подайте заявку, чтобы продолжить.",false)  
 mToRegisterMenuButton.startAnimation(animScale)  
 }  
 }  
 **}**.addOnFailureListener **{** Log.e("TAG", "Ошибка: ${**it**}")  
 **}** }  
  
 override fun onResume() {  
 count = 0  
 super.onResume()  
 }  
  
 @Deprecated("Deprecated in Java")  
 @SuppressLint("MissingSuperCall")  
 override fun onBackPressed() {  
 if(count==0) {  
 toast("Нажмите еще раз, чтобы свернуть",null)  
 count = 1  
 } else {  
 val startMain = Intent(Intent.*ACTION\_MAIN*)  
 startMain.addCategory(Intent.*CATEGORY\_HOME*)  
 startMain.*flags* = Intent.*FLAG\_ACTIVITY\_NEW\_TASK* startActivity(startMain)  
 }  
 }  
  
 fun hideKeyboardFrom(context: Context, view: View?) {  
 val imm = context.getSystemService(Activity.*INPUT\_METHOD\_SERVICE*) as InputMethodManager  
 imm.hideSoftInputFromWindow(view?.*windowToken*, 0)  
 }  
}

package com.holysource.myschool.Activities  
  
import android.annotation.SuppressLint  
import android.content.Intent  
import android.os.Bundle  
import androidx.appcompat.app.AppCompatActivity  
import androidx.navigation.findNavController  
import androidx.navigation.ui.AppBarConfiguration  
import androidx.navigation.ui.setupWithNavController  
import com.google.android.material.bottomnavigation.BottomNavigationView  
import com.holysource.myschool.Adapters.CustomToast  
import com.holysource.myschool.R  
import com.holysource.myschool.databinding.ActivityMainBinding  
  
  
class MainActivity : AppCompatActivity() {  
  
 var count = 0  
 private lateinit var binding: ActivityMainBinding  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
  
 binding = ActivityMainBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
  
 val navView: BottomNavigationView = binding.navView  
  
 val navController = *findNavController*(R.id.*nav\_host\_fragment\_activity\_main*)  
 *// Passing each menu ID as a set of Ids because each  
 // menu should be considered as top level destinations.* val appBarConfiguration = *AppBarConfiguration*(  
 *setOf*(  
 R.id.*navigation\_news*, R.id.*navigation\_study*, R.id.*navigation\_profile* )  
 )  
 navView.*setupWithNavController*(navController)  
 }  
  
 fun toast(text: String, isDone: Boolean?) {  
 CustomToast.makeText(*applicationContext*, text, isDone).show()  
 }  
  
 override fun onResume() {  
 count = 0  
 super.onResume()  
 }  
  
 @SuppressLint("MissingSuperCall")  
 override fun onBackPressed() {  
 if(count==0) {  
 toast("Нажмите еще раз, чтобы свернуть",null)  
 count = 1  
 } else {  
 val startMain = Intent(Intent.*ACTION\_MAIN*)  
 startMain.addCategory(Intent.*CATEGORY\_HOME*)  
 startMain.*flags* = Intent.*FLAG\_ACTIVITY\_NEW\_TASK* startActivity(startMain)  
 }  
 }  
}

**Приложение Б**

**Диаграммы UML**

