Содержание

Список используемых сокращений

ОС – Операционная система

ОХД – Облачное хранилище данных

ПК – Персональный компьютер

ВМ – Виртуальная машина

ART – Android Runtime

SDK – Software Development Kit

БД – База данных

JAR – Java Archive

JVM – Java Virtual Machine

Введение

В современном мире, где информационные технологии стремительно развиваются, в основе любого процесса, связанного с информацией, лежит взаимодействие с персональными компьютерами.

Все наши данные хранятся на жестком диске ПК или на внешних носителях, таких как flash-накопителях, например. Однако всем хорошо известны недостатки такого подхода к хранению данных:

1. доступ к хранимой на ПК информации весьма ограничен (т.е. данные доступны только при работе с этим компьютером);

2. внешние носители могут быть легко потеряны, либо они могут выйти из строя и данные будут безвозвратно утеряны;

3. затрудненная синхронизации изменений данных при работе с одними и теми же файлами с разных рабочих мест;

4. данные на ПК могут быть уничтожены в результате попадания на устройство вируса.

Таким образом, современные люди все больше нуждаются в независимости своих данных от рабочих мест. Самым очевидным способом решения данной проблемы является использование облачных хранилищ данных (ОХД), позволяющее иметь доступ к файлам с разных устройств.

Итак, целью данной курсовой работы было решено выбрать тему «Разработка программного приложения для загрузки файлов в облачное хранилище», т.к. данная тематика крайне актуальна в условиях повсеместного и постоянного документооборота. Приложение может быть использовано как студентами, так и, например, офисными работниками для быстрого доступа к своим документам с разных устройств.

Важно отметить, что в рамках учебной программы разработка ведется для ОС Android. В ходе выполнения курсовой работы планируется освоить компетенции ПК-13 (готовность к использованию методов и инструментальных средств исследования объектов профессиональной деятельности).

1. Общие сведения

В ходе выполнения курсовой работы было разработано приложение для осуществления доступа к ОХД, реализованного с помощью сервера Debian 10.3, mysql Ver 15.1 Distrib 10.3.22-MariaD.

1.1. Обозначение и наименование программы

Приложение было названо «Calyx». Данное название полностью отражает суть и назначение данного программного приложения: «calyx» с английского языка переводится как «чашечка», т.е. приложение по сути является «сосудом» для файлов.

1.2. Программное обеспечение, необходимое для функционирования программы

Для функционирования приложения необходимо установить сам программный продукт на персональное устройство. Для корректной работы приложения необходимо, чтобы на устройство была установлена ОС Android 4.4 или другая, более новая версия (уровень API ≥ 19).

1.3. Языки программирования, на которых написана программа

Данное приложение написано на высокоуровневом, объектно-ориентированном языке программирования Java, который активно применяется для разработки мобильных приложений для ОС Android.

Также приложение использует собственное API для взаимодействия с сервером. При написании API использовался язык программирования PHP.

2. Функциональное назначение

Приложение «Calyx» предназначено для хранения файлов на удаленном сервере. Приложению требуется обеспечить для пользователя возможность регистрации, и авторизации в случае наличия учетной записи. Также необходимо обеспечить функции просмотра списка загруженных файлов, добавления файлов в хранилище, скачивания загруженных файлов и удаления их из хранилища. Требуется также обеспечить пользователю возможность выхода из текущей учетной записи.

3. Описание логической структуры

3.1. Серверная часть

Учет пользователей и их файлов ведется при использовании базы данных (СУБД MariaDB). Файлы хранятся в персональных папки, которые создаются для каждого пользователя при регистрации.

Взаимодействие с БД и файловой системой осуществляется с помощью собственного API, написанного на скриптовом языке программирования PHP.

3.2. Логическая структура программы

3.2.1. Декомпозиция

Все файлы программы условно можно разделить на пять групп, каждая из которых реализует определенную функциональную составляющую. Разбиение классов на группы приведено на рисунке 3.1.

Так, первая группа отвечает за авторизацию, вторая – за регистрацию, третья группа классов составляет функционал, который доступен авторизованному пользователю, четвертая группа обеспечивает функции проводника, и последнюю, пятую группу классов, составляют вспомогательные классы, использующиеся на разных этапах работы приложения.

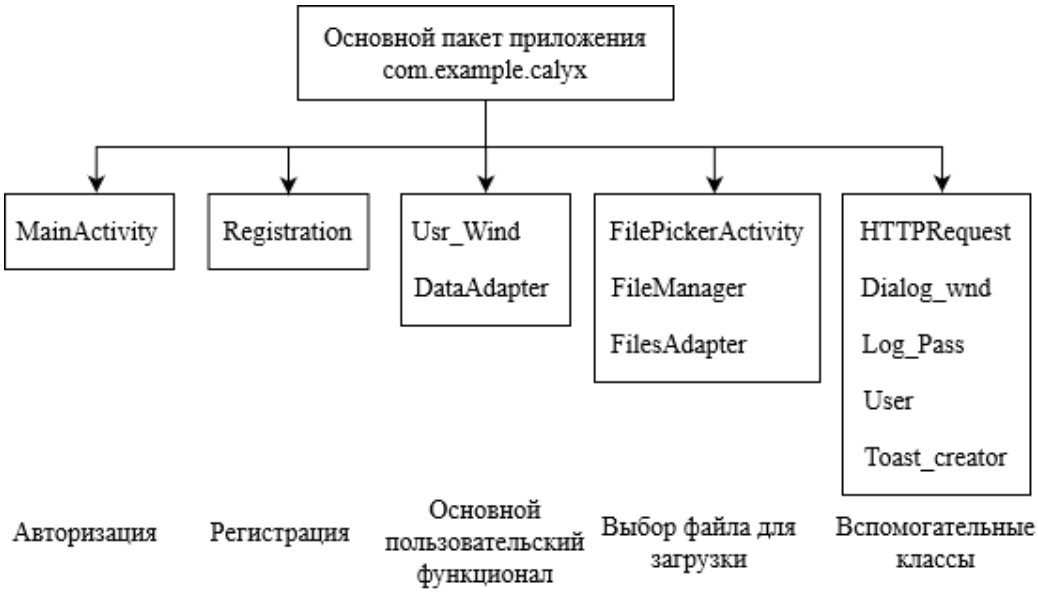

Рисунок 3.1 – Разбиение классов приложения на группы по функциям

Приложение содержит четыре активности (представлены классами, расширяющими класс AppCompatActivity):

1. MainActivity – класс, отвечающий за авторизацию пользователей. Эта активность содержит поля для ввода логина и пароля, кнопку регистрации.

2. Registration – класс, предназначенный для регистрации новых пользователей. Данная activity содержит поля ввода регистрационных данных (логин и пароль).

3. Usr_Wind – класс, обеспечивающий все функции авторизованного пользователя (добавление нового файла, скачивание файла из хранилища, удаление файлов, выход из учетной записи). Более подробное описание приведено в последующих разделах.

4. FilePickerActivity – класс, реализующий функции проводника, т.е. отображает список файлов текущей директории.

Кроме активностей в пакете содержатся и другие классы, которые обеспечивают весь необходимый функционал приложения. Подробное описание этих классов изложено в последующих разделах.

3.2.2. Описание зависимостей

В процессе разработки приложения использовалось несколько дополнительных библиотек, с помощью которых были реализованы некоторые функции приложения.

Так, для реализации связи с API сервера была использована библиотека OkHttp, которая представляет собой HTTP клиент. Классы библиотеки позволяют создавать запросы к серверу, получать данные, приходящие в ответ на запрос и многое другое.

Данная библиотека разработана компанией Square Capital, LLC. OkHttp была реализована для прямой работы с верхним уровнем сокетов Java, не используя при этом какие-либо дополнительные зависимости.

Библиотека поставляется в виде JAR-файла, поэтому она может быть использована на любых устройствах с JVM (в т.ч. и на Android-устройствах).

Для реализации проводника был использован класс RecyclerView. Функционал этого класса по умолчанию недоступен в AndroidStudio, поэтому необходимо было дополнительно включить поддержку данного класса в файле build.gradle (Module: app) в узле зависимостей (dependencies).

Класс RecyclerView предназначен для оптимизации работы со списками, также он позволяет повысить производительность по сравнению со стандартным элементом для работы со списками ListView [2].

3.2.3. Детальное описание классов

Для лучшего понимания структуры программы и работы ее составляющих далее будут приведены подробные описания классов, начиная со вспомогательных и заканчивая основными, где уже используются все вспомогательные классы.

3.2.3.1. Класс Toast_creator

Данный класс используется для создания всплывающих уведомлений, называемых «тостами».

Toast_creator имеет единственный метод – showToast. Метод является статическим, т.е. может вызываться без создания экземпляра класса.

Стуктура класса представленна наже, на рисунке 3.2.

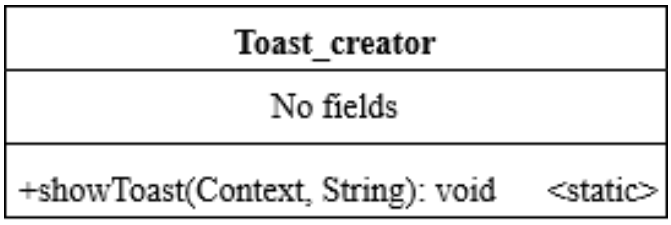

Рисунок 3.2 – UML-диаграмма класса Toast_creator

showToast принимает в качестве параметров объект класса Context, который указывает на активность, для которой нужно показать сообщение, и объект String (строка).

Метод showToast создает объект класса Toast с сообщением, содержащимся в параметре класса String, для активности, заданной с помощью параметра Context.

3.2.3.2. Класс Log_Pass

Данный вспомогательный класс используется для проверки вводимых в формы паролей и логинов на наличие запрещенных символов и правильность длинны. Структура класса изображена ниже, на рисунке 3.3.

В контексте данного приложения регистрационные данные не могут содержать следующие символы: пробел, /, ‘, “.

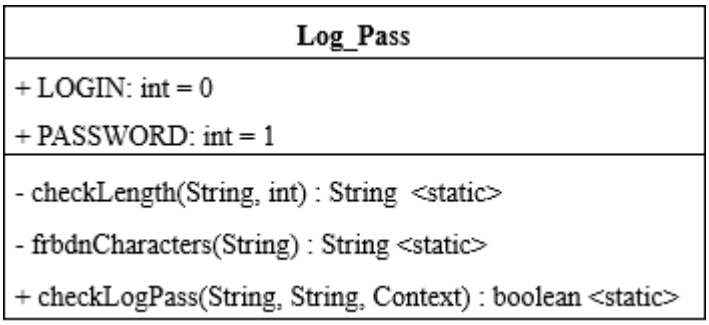

Рисунок 3.3 – UML-диаграмма класса Log_Pass

checkLength(String, int) – метод для проверки длинны паролей и логинов. Проверяемый логин или пароль передается в функцию как параметр класса String. Параметр типа int задает вид проверяемой строки (пароль или логин). Для того, чтобы задать тип проверяемой строки нужно передать в качестве параметра одну из двух статических констант (LOGIN или PASSWORD). Логин и пароль следует различать, потому что минимальные допустимые длины для них разные. checkLength возвращает строку, содержащую определенное сообщение в зависимости от замечаний, предъявленных к логину и паролю. Соответственно, если логин или пароль имеет длину, не удовлетворяющую требованиям приложения, метод вернет одну из двух следующих строк: «Логин должен содержать от 6 до 32 символов!» или «Пароль должен содержать от 8 до 32 символов!»; если же длинна удовлетворяет условиям, метод вернет пустую строку.

frbdnCharacters(String) – метод, применяемый для проверки паролей и логинов на содержание запрещенных символов. Наличие того или иного запрещенного символа проверяется с помощью метода String.contains (CharSequence sequence), где sequence – это символ или подстрока,

наличие которой требуется проверить. В случае, если подстрока найдена в проверяемой строке, функция возвращает true, в противном случае – false.

Таким образом, если при проверке полученной строки contains вернула true, метод frbdnCharacters возвращает строку с сообщением об ошибке: «Логин и пароль НЕ МОГУТ содержать следующие символы: ' " / и пробелы!», в противном случае метод возвращает пустую строку.

Методы checkLength и frbdnCharacters имеют модификатор доступа private, что делает их использование доступным только в пределах данного класса.

checkLogPass (String, String, Context) – метод, который предназначен для проверки логина и пароля, а также для выдачи полного сообщения о том, какие ошибки допущены пользователем во время запыления полей при регистрации или авторизации. Работа данного метода основана на использовании приватных методов этого же класса checkLength и frbdnCharacters. Сообщение «собирается» путем конкатенации строк, которые являются результатами работы вышеописанных методов, в локальную переменную warning. Затем, если в результате слияния строк получилась пустая строка, метод возвращает true; в ином случае с помощью класса Toast_creator создается всплывающее сообщение, содержащее текст, полученный в результате проверки логина и пароля, из локальной переменной warning, и возвращается значение false.

Все методы данного класса имеют модификатор static, что дает возможность использовать их, не создавая объект класса. В то же время, метод checkLogPass является публичным методом, что делает его очень удобным для использования, т.к. внутренняя реализация класса не обязывает разработчика создавать новый объект, чтобы проверить корректность логинов и паролей.

3.2.3.3. Класс Dialog_wnd

Класс Dialod_wnd предназначен для отображения диалогового окна с информацией о том, что чего нужны те или иные запрашиваемые у пользователя разрешения. Далее, на рисунке 3.4, изображена структура данного класса.

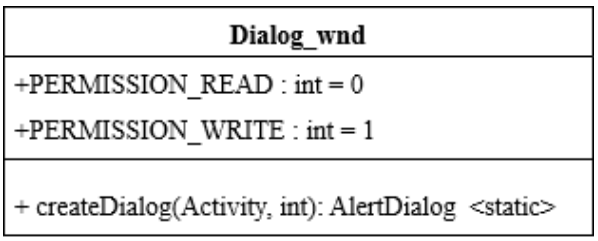

Рисунок 3.4 – UML-диаграмма класса Dialog_wnd

Единственный метод класса, createDialog, создает экземпляр класса androidx.appcompat.app.AlertDialog.Builder, с помощью методов которого далее задается заголовок, текст и кнопка, скрывающая диалоговое окно.

Метод принимает два параметра: ссылку на активити, для которого создается AlertDialog и идентификатор разрешения, в котором было отказано.

В зависимости от разрешения, выбирается текст для отображения. В случае, если отказано в разрешении на чтение внутреннего общего накопителя, диалоговое окно будет содержать следующий текст: «Для использования функционала данного приложения необходимо разрешить доступ к файлам устройства.», если же отказано в записи файлов в накопитель, текст будет таким: «Для использования функционала данного приложения необходимо разрешить запись файлов на ваше устройство.».

Важно отметить, что до Android 6 разрешения запрашивались перед установкой, а не в процессе работы приложения. Отсюда следует, что при установке приложения на устройство с версией ОС ниже версии Android 6, класс dialog_wnd не будет задействован.

3.2.3.4. Класс User

Данный класс используется для передачи данных пользователя между компонентами приложения. Так, класс User содержит три private поля типа String: login (логин), password (пароль), userKey (ключ пользователя).

Каждое из полей инициализируется либо в конструкторе, либо с помощью методов-сеттеров (англ. *setter* – установщик). Также значение каждого из полей может быть получено с помощью методов-геттеров (англ. *getter* – получатель).

Далее, на рисунке 3.5, приведена UML-диаграмма данного класса.

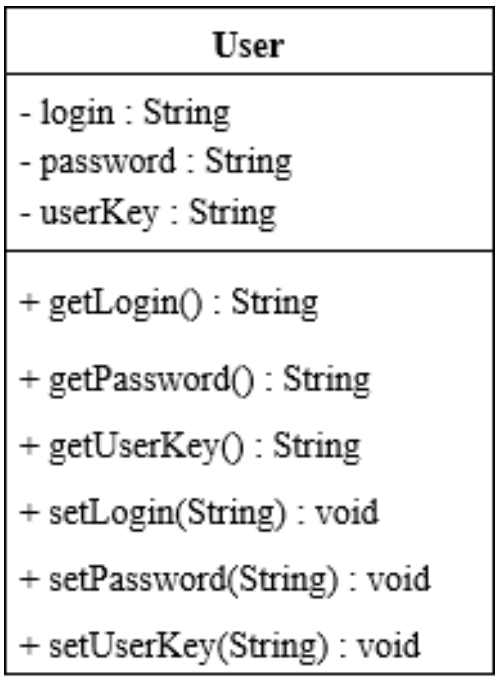

Рисунок 3.5 – UML-диаграмма класса User

Отдельно стоит пояснить, что логин и пароль используются только для регистрации новых пользователей и авторизации существующих. После авторизации пароль и логин больше не используются для связи с сервером, пользователь получает свой уникальный идентификатор – ключ пользователя (userKey).

Более детальное описание взаимодействия программы с сервером приведено в следующем разделе.

3.2.3.5. Класс HTTPRequest

Класс HTTPRequest отвечает за отправку запросов к серверу и обработку ответа, пришедшего с сервера.

Класс имеет большое количество полей, которые можно разделить на несколько групп: целочисленные константы – идентификаторы задач, строковые константы – адреса скриптов API сервера, поля-объекты для осуществления соединения с сервером и прочие поля, используемые при выполнении каждой из задач.

Далее, на рисунке 3.6, приведена структура данного класса.

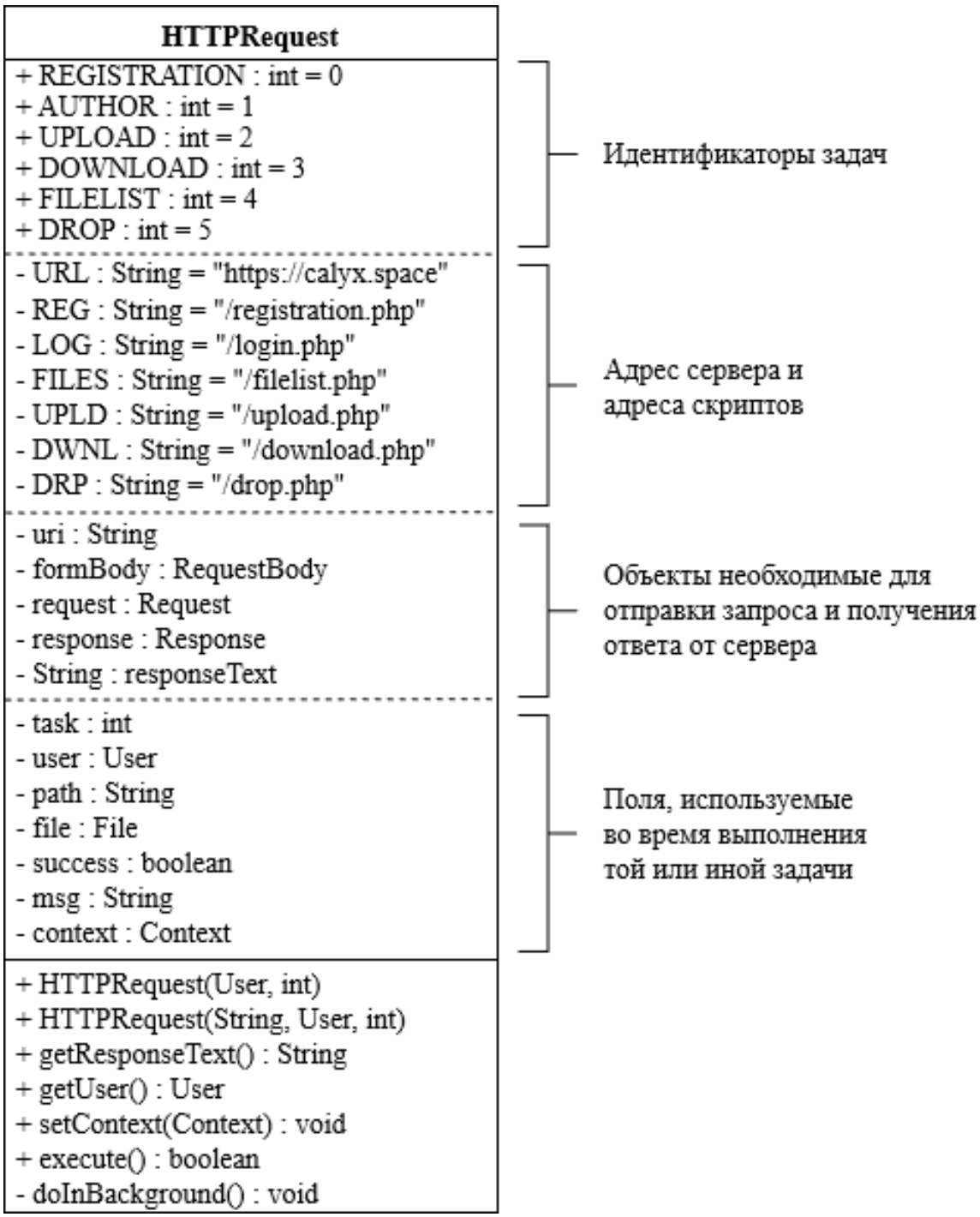

Рисунок 3.6 – UML-диаграмма класса HTTPRequest

В классе определены два конструктора:

HTTPRequest(User user, int task) – конструктор, применяемый тогда, когда необходимо выполнять задачи связанные только с пользовательскими данными (регистрация, авторизация, запрос списка файлов);

HTTPRequest(String path, User user, int task) – конструктор, используемый в случаях, если нужно выполнить задачу связанную с файлами (скачивание, загрузка и удаление файлов).

Оба конструктора принимают в качестве параметра объект класса User и целое число. Объект класса User (см. п. 2.2.3.4) содержит все необходимые данные для осуществления взаимодействия с сервером (логин и пароль для авторизации и регистрации, и ключ пользователя для всех остальных задач). Целое число, передаваемое конструктору, обозначает номер задачи. В программе для установки задачи используются целочисленные константы, т.к. они очень удобны в использовании, не нужно запоминать номер каждой задачи.

Метод doInBackground – самый важный метод класса HTTPRequest. Этот метод осуществляет всю работу по отправке запросов и получению ответов от сервера.

Рассмотрим алгоритм работы doInBackground.

После того как был вызван один из конструкторов, поля task и User проинициализированы, т.е. установлены данные пользователя и задача.

Данный метод открывает новый поток для работы с сервером.

В побочном потоке с помощью конструкции switch-case реализован выбор набора команд в соответствии с номером задачи (task). В рамках приложения предусмотрены следующие задачи: регистрация (0), авторизация (1), загрузка файла (2), скачивание файла (3), запрос списка уже загруженных файлов (4), удаление файла из хранилища (5).

В каждом case находятся команды для составления адреса конечного скрипта на сервере (путем конкатенации значений нужных строковых констант: URL + <нужная строковая константа>) и составления тела запроса, в который помещаются нужные для отправки данные.

После выполнения switch-case, когда тело запроса создано (объект класса RequestBody), создается сам запрос (объект класса Request) [4]. Далее в объект Response помещается результат отправки запроса, и в случае успеха переменная success принимает значение true (показатель, что запрос успешно отправлен), в противном случае – false.

Для каждой из задач API сервера предоставляет ответ, по которому можно судить об успешности выполнения задачи. Ответ сервера помещается в поле

responseText. Для обработки responseText используется еще один switch-case, в котором для каждой задачи устанавливается текст сообщения (поле msg) на случай успешного завершения выполнения и, также, на случай ошибки.

Ниже представлена таблица 3.1, в которой приведена информация о том, какие пользовательские данные отправляются в запросе к серверу, с какой целью, что программа получает в ответ и т.д.

Таблица 3.1 – Описание задач класса HTTPRequest

Задача		Регистр.	Авториз.	Загрузка	Скачивание	Запрос списка файлов	Удаление
Значение константы		0	1	2	3	4	5
Php-скрипт (*.php)		registration	login	upload	download	listfiles	drop
Данные, помещаемые в запрос		Логин, пароль	Логин, пароль	Ключ, объект типа File	Ключ, имя файла	Ключ	Ключ, имя файла
Ответ сервера	+	Пустая строка	Ключ	«Success»	Ссылка	JSON-файл	«Success»
	-	«Duplicate»	«Deny»	Пустая строка	Пустая строка	«No files»	Пустая строка
Активность		Registration	MainActivity	Usr_wind			

Процесс скачивания файла из хранилища немного отличается от всех остальных процессов. Скачивание проходит в два этапа: первый этап – отправка запроса с названием файла и ключом пользователя к серверу. В случае ошибки программа получит пустую строку, в случае успеха – ссылку для скачивания. В блоке обработки ответа сервера, с случае успеха, проводится второй этап: скачивание файла по ссылке. После скачивания на внутреннем общем накопителе создается новый файл (используется класс File) с именем, совпадающим с именем скачиваемого файла, далее с помощью потока вывода в файл (FileOutputStream) полученные данные записываются в новый файл. Таким образом, скачивание файла осуществляется в два запроса. Файл по умолчанию сохраняется в папку Download на внутреннем общем накопителе.

Важно отметить, что после запуска нового потока, для него вызывается метод join(). Это необходимо, чтобы основной поток ожидал завершения

побочного потока, т.к. основной поток приложения должен будет использовать данные пользователя, уже преобразованные в ходе работы побочного потока.

Также в классе HTTPRequest определен метод execute(). Этот метод осуществляет вызов doInBackground и реализует связь между вышеописанным методом и интерфейсом пользователя: с помощью класса Toast_creator создаются всплывающие сообщения о результате выполнения той или иной операции.

Перед вызовом метода execute(), обязательно нужно установить контекст с помощью метода setContext, для получения возможности отображения всплывающих уведомлений о результатах работы doInBackground. В качестве параметра следует передать методы ссылку на вызывающую активность.

Методы геттеры позволяют другим классам получить данные об успешности соединения с сервером или преобразованные данные пользователя (например, после авторизации необходимо получить ключ пользователя для осуществления всех пользовательских операций).

3.2.3.6. Класс FileManager

Класс FileManager – класс, ответственный за получение списка файлов. Ниже, на рисунке 3.7, приведена UML-диаграмма данного класса.

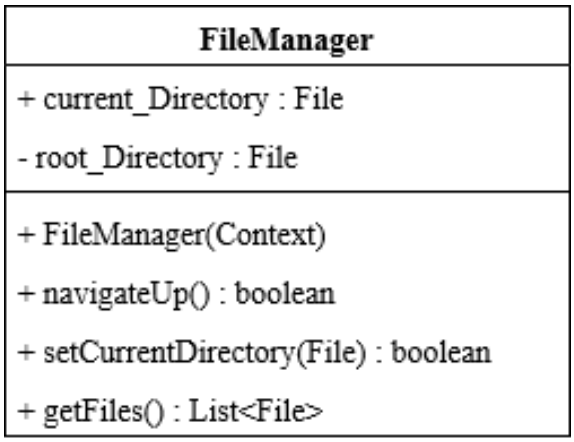

Рисунок 3.7 – UML-диаграмма класса FileManager

Класс хранит текущую директорию, и директорию, выше которой пользователь не сможет подняться. Данные поля представляют из себя объекты класса File. При создании объекта данного класса current_Directory совпадает с

root_Directory, т.к. при открытии проводника приложения необходимо показывать содержимое именно корневой директории.

В конструкторе инициализируются поле root_Directory. По умолчанию это корень внутреннего общего накопителя, если он доступен. Если доступа к накопителю нет, используется директория приложения, для доступа к которой понадобится контекст. Далее устанавливается текущая директория с помощью метода setCurrentDirectory(), который подробно описан ниже.

Метод setCurrentDirecory, в сущности, является методом установки новой текущей директории. С помощью этого метода и метода navigateUp (этот метод описан ниже) осуществляется переход по папкам устройства. Данный метод принимает в качестве параметра объект класса File, который представляет из себя директорию, содержимое которой пользователь хочет просмотреть. Класс File может содержать как директорию, так и файл, поэтому необходимо сделать проверку, что объект действительно содержит именно директорию. Также необходимо проверить, чтобы File не содержал rootDirectory. Если одно из условий не выполняется метод возвращает false, если же все требования к объекту класса File соблюдены, будет возвращено значение true.

Метод navigateUp осуществляет переход в прошлую директорию, т.е. в ту директорию, которая расположена выше по иерархии. Метод возвращает результат работы метода setCurrentDirectory для родительского файлового объекта текущей директории (родительский объект получается с помощью метода File getParentFile()).

За получение списка файлов текущей директории (currentDirectory) ответственен метод getFiles(). Список создается следующим образом: в локальную переменную типа ArrayList<File> добавляются все файлы и директории, содержащиеся в текущей, представленные в виде элементов списка. Метод основан на функционале классов ArrayList, Arrays и File.

3.2.3.7. Класс FilesAdapter

Адаптер необходим для отображения списка в активности и обеспечения реакции на нажатие на пункт списка.

Отметим, что для отображения списка файлов устройства используется класс RecyclerView, для которого нужен особенный адаптер, который расширяет класс RecyclerView.Adapter. Список отображает имена файлов, представленных объектами класса File.

Данный класс обязывает переопределить три метода: onCreateViewHolder (возвращает объект ViewHolder, который будет хранить данные по одному объекту File), onBindViewHolder (выполняет привязку объекта ViewHolder к объекту File по определенной позиции списка), getItemCount (возвращает количество объектов в списке) [1].

Также класс содержит метод для установки (обновления) списка – setFiles.

Для того, чтобы адаптер мог различать файлы и директории, предусмотрен метод getItemViewType(int). Метод принимает в качестве параметра номер пункта списка, и проверяет является ли объект Files, закрепленный за этим пунктом, директорией с помощью метода File isDirectory(). В зависимости от результата проверки, метод возвращает значение одной из целочисленных констант, определённых в классе (см. рис. 3.8).

Для обработки нажатия на элемент списка в адаптере имеется метод установки слушателя события нажатия setOnFileClickListener.

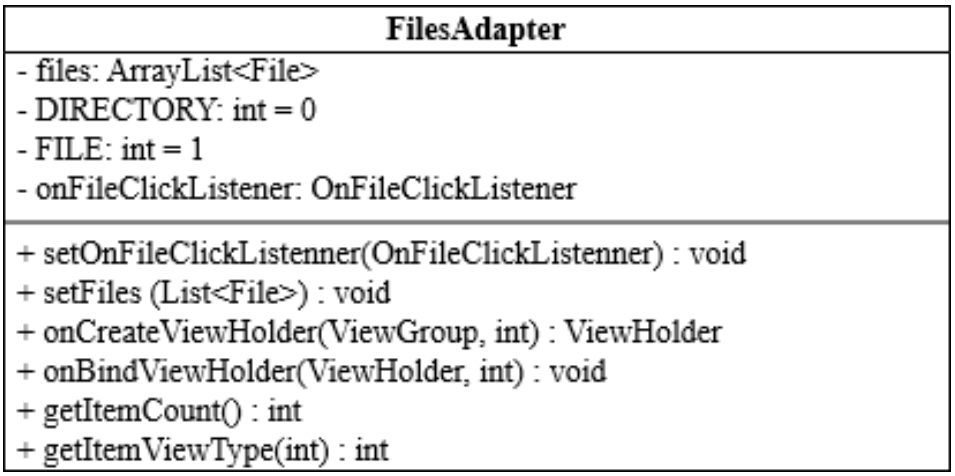

Рисунок 3.8 – UML-диаграмма класса FilesAdapter

3.2.3.8. Класс FilePickerActivity

Данный класс унаследован от класса AppCompatActivity и применяется как окно проводника при выборе файла для загрузки в облачное хранилище.

В классе активити определены четыре поля: менеджер файлов (объект класса FileManager), адаптер списка файлов (объект класса FilesAdapter), строковая константа, используемая как ключ для получения пути к выбранному файлу, и слушатель события нажатия на пункт из списка.

Для создания нового слушателя необходимо переопределить метод onFileClick(File). Поступивший в слушатель объект класса File обрабатывается следующим образом: если объект является директорией, то вызывается метод файлового менеджера для установки новой текущей директории, затем список обновляется с помощью метода updateList() (метод описан ниже). Если же пользователь нажал на пункт с файлом (т.е. . объект класса File - файл), то путь к этому файлу передается через Intent в основное окно приложения (Usr_wind, описан далее).

В классе переопределен метод onCreate. В нем создается объект класса RecyclerView, который и будет отображаться в данном активити, устанавливается разметка списка, заголовок активности, создается адаптер, который затем будет установлен в список. Далее запускается файловый менеджер с помощью метода intitFileManager() и на экране обновляется список файлов (метод updateFileList()).

В методе onStart в адаптер списка устанавливается слушатель нажатий на список, а в onStop необходимо отписаться от получений события нажатия (установка в адаптер null вместо слушателя).

Метод initFileManager() – метод создания файлового менеджера. Прежде чем создать новый менеджер, метод проверяет, есть ли у приложения разрешение на чтение внутреннего общего накопителя.

Для обновления списка при нажатии на папку предназначен метод updateFileList(). Данный метод создает новый List<Files>, получая список файлов

из файлового менеджера. Затем новый список передается адаптеру с помощью метода setFiles(List<File>). Далее необходимо вызвать метод notifySataSetChanded(), чтобы уведомить адаптер о том, что данные в списке изменились.

Чтобы перейти в прошлую директорию необходимо нажать кнопку «Назад». Клик на данную кнопку обрабатывается методом onBackPressed(), который необходимо переопределить. Данный метод проверяет, был ли создан файловый менеджер. Если менеджер создан, то вызывается метод обновления списка, в противном случае вызывается метод onBackPressed, определенный в родительском классе данной активности [5].

Далее, на рисунке 3.9, приведена структура описываемого класса.

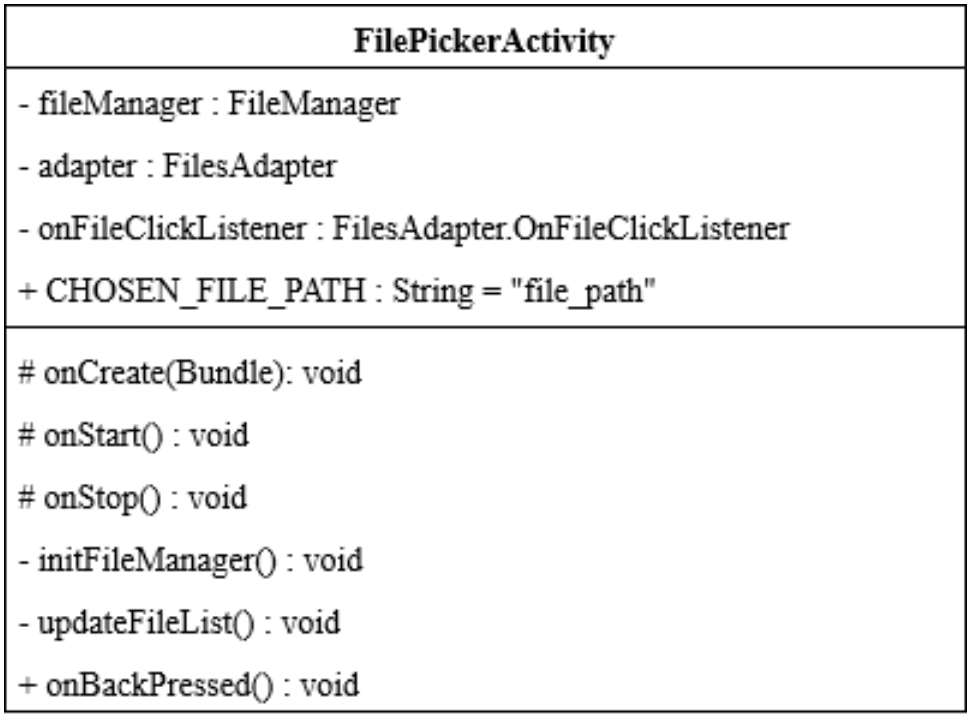

Рисунок 3.9 – UML-диаграмма класса FilePickerActivity

Таким образом, FilePickerActivity, FileManager и FilesAdapter в совокупности образую такой компонент приложения как проводник. Вместе они обеспечивают удобный и интуитивно понятный способ выбора файла для загрузки в хранилище.

3.2.3.9. Класс Usr_wind

Класс Usr_wind используется в качестве окна личного кабинета пользователя. Данный класс унаследован от класса AppCompatActivity. Ниже, на рисунке 3.10, представлена структура класса Usr_wind.

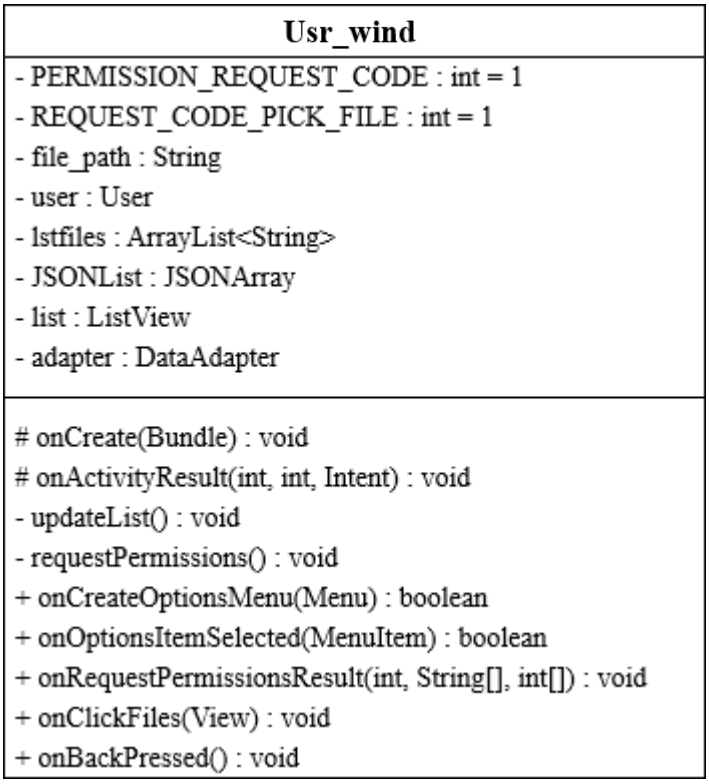

Рисунок 3.10 – UML-диаграмма класса Usr_wind

Usr_wind – ключевой класс для пользователя. Здесь собраны все пользовательские функции: загрузка нового файла в хранилище, скачивание уже загруженных файлов, удаление файлов, обновление списка файлов и выход из личного кабинета.

При создании активность получает объект класса User, который содержит имя пользователя (логин) и его ключ. В поле пароля перед отправкой помещается пустая строка в целях безопасности (т.е. пароль не передается между активностями). В методе onCreate инициализируется поле класса User и задается заголовок активити (имя пользователя), а также вызывается функция обновления списка загруженных файлов.

Обновление списка выполняет метод updateList. Данный метод создает объект класса HTTPRequest (см. п. 2.2.3.5) для отправления запроса на

получение списка загруженных файлов. Если произошла ошибка во время соединения с сервером, метод уведомит пользователя об этом путем создания всплывающего сообщения Toast. Затем, если на сервере есть файлы авторизованного пользователя, создается объект JSON массива куда помещается JSON массив, пришедший в ответ на запрос. Далее объект JSON массива конвертируется в стандартный ArrayList<String>. После того, как был получен строковый список файлов, создается сам объект списка в активности (используется класс ListView), а список строк передается адаптеру класса DataAdapter (описан в следующем разделе). Полученный адаптер необходимо установить в список активности.

Для обработки нажатия на список используются интерфейсы AdapterView.OnItemClickListener и AdapterView.OnItemLongClickListener. Первый слушатель обрабатывает обычное нажатие, второй – долгое нажатие.

При коротком нажатии происходит скачивание файла с помощью объекта HTTPRequest. В конструктор класса необходимо передать данные пользователя, название файла и идентификатор задачи (в данном случае нужен идентификатор RTTPRequest.DOWNLOAD), и затем вызвать метод execute().

При долгом нажатии на пункт списка происходит удаление файла из хранилища. Аналогично скачиванию, удаление файла обеспечивается механизмами класса HTTPRequest.

Окно личного кабинета также включает в себя кнопку выбора файла для загрузки. Нажатие этой кнопки обрабатывается методом on_Click_Files.

Так как класс Usr_wind запускает работу вспомогательных классов с внутренним общим накопителем, необходимо проверять наличие разрешения для доступа к памяти устройства.

Если пользователь еще не дал разрешение для чтения, on_Click_Files вызывает метод requestPermissions, в противном случае с помощью метода startActivityForResult вызывается FilePickerActivity (активность, используемая как проводник, см. п. 2.2.3.8).

Метод requestPermissions отвечает за запрос разрешения на чтение внутреннего накопителя.

За проверку наличия разрешения ответственен метод onRequestPermissionsResult. Данный метод проверяет, дал ли пользователь разрешение для чтения внутреннего общего накопителя. Если пользователь отказал в доступе к внутреннему общему накопителю, метод создает диалоговое окно, функция которого – уведомить пользователя о том, для чего нужно данное разрешение (используется класс Dialog_wnd).

Метод onActivityResult получает результат работы FilePickerActivity, которая возвращает объект Intent, содержащий путь к выбранному для загрузки файлу. Затем этот же метод создает объект класса HTTPRequest для загрузки файла, передав конструктору путь файла, данные пользователя и соответствующий идентификатор задачи.

Верхняя часть активности включает в себя ActionBar (меню), который содержит кнопки обновления списка файлов и выхода из аккаунта пользователя.

За создание меню отвечает метод onCreateOptionsMenu. Данный метод устанавливает разметку кнопок меню.

Метод onOptionsItemSelected обрабатывает нажатие на кнопку меню. Данный метод принимает объект View, который представляет из себя объект кнопки в интерфейсе.

Кнопка выхода из аккаунта очищает пользовательские данные, сохраненные на устройстве, вызывает главную активность (активность авторизации) и завершает работу текущей активности.

Кнопка обновления списка выполняет вызов метода updateList ().

3.2.3.10. Класс DataAdapter

Данный класс (рис. 3.11) является адаптером данных для списка загруженных в облачное хранилище файлов, который связывает элементы списка строк со списком ListView.

Класс содержит два поля: список для отображения в активности и контекст (объект класса Context).

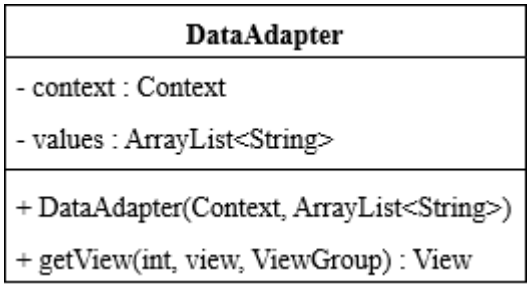

Рисунок 3.11 – UML-диаграмма класса DataAdapter

В адаптере определен конструктор, принимающий в качестве параметров объект класса Context и список строк, который необходимо отобразить в активности. Также данный метод устанавливает разметку для каждого пункта списка.

Метод getView «надувает» (от англ. *inflate* – раздувать, надувать) элементы списка разметкой и сопоставляет каждому текстовому полю из полученного View объект TextView. Затем устанавливается текст для каждого TextView из списка строк соответственно.

3.2.3.11. Класс MainActivity

Данный класс используется как окно для авторизации. Ниже, на рисунке 3.12, приведена UML-диаграмма класса MainActivity.

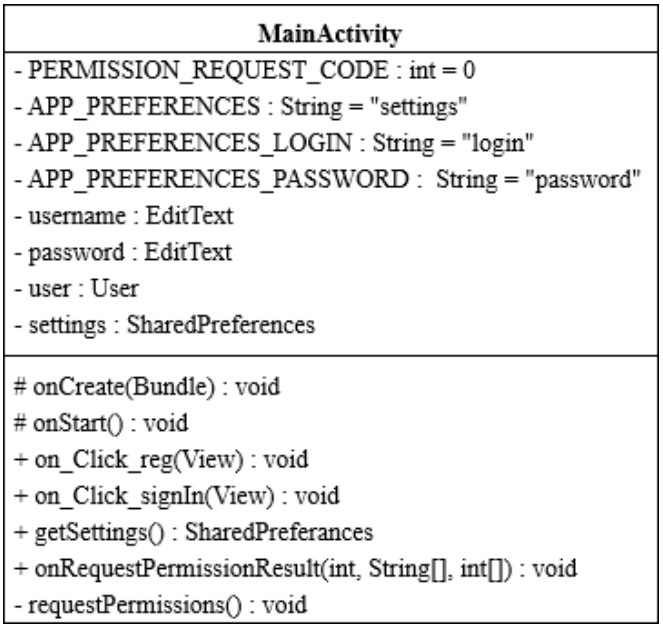

Рисунок 3.12 – UML-диаграмма класса MainActivity

Приложение поддерживает сохранение паролей и автоматическую авторизацию, если пользователь не выходил из своей учетной записи. Сохранение данных достигается путем использования механизмов класса SharedPreferences, который позволяет записывать передаваемые в объект данные в специальный файл, располагающийся в директории самого приложения, к которому пользователь устройства не имеет доступа [3].

При запуске приложения запускается метод onCreate. В начале своей работы он устанавливает разметку интерфейса, затем проверяет, есть ли данные в файле для сохранения данных авторизации. Если пароль и логин сохранены, выполняется запрос на авторизацию, и затем данные передаются в класс Usr_wind с помощью Intent. Если же пароль и логин учетной записи не были обнаружены, выполняется привязка элементов интерфейса, ответственных за ввод логина и пароля, к объектам класса TextEdit. Затем метод onStart () сбрасывает текст в полях ввода, если он там был.

За начало процесса авторизации отвечает кнопка «Войти». Нажатие на данную кнопку обрабатывается методом on_Click_SignIn. Перед непосредственно авторизацией метод проверяет наличие разрешения на запись файлов на устройство и при необходимости запрашивает его. Если пользователь отклонил запрос, авторизация выполнена не будет, также будет показано диалоговое окно с информацией о разрешении.

В случае, если у приложение получило требуемое разрешение, логин и пароль проходят проверку на корректность (задействуются механизмы класса Log_Pass). Если проверка не пройдена, авторизация не будет выполнена и на экране появится всплывающее уведомление с информацией о некорректности введенных данных. Если же проверка пройдена с помощью объекта класса HTTPRequest выполняется запрос к серверу на авторизацию. Если приложение в ответ на запрос получило ключ пользователя, данные объекта user обновляются (добавляется ключ), логин, пароль и ключ заносятся в файл для сохранения данных (SharedPreferences), затем логин и ключ с помощью Intent передаются в

класс Usr_wind, где в свою очередь, отправляется запрос на получение списка файлов, которые данный пользователь уже загрузил в хранилище.

Также из MainActivity можно перейти к процессу регистрации нового пользователя. За переход к окну регистрации отвечает кнопка «Зарегистрироваться», нажатие на которую обрабатывается методом on_Click_reg. Данный метод с помощью объекта класса Intent запускает активность класса Registration.

3.2.3.12. Класс Registration

Данный класс отвечает за регистрацию новых пользователей. Далее, на рисунке 3.13, приведена структура класса Registration.

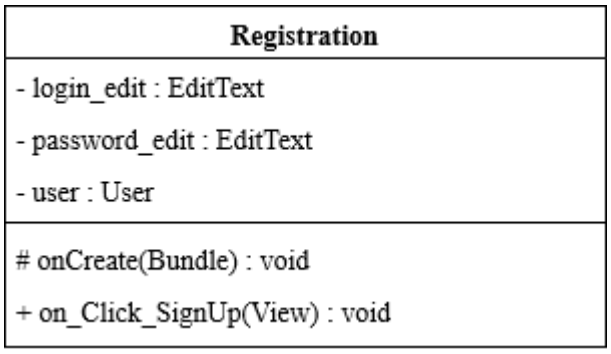

Рисунок 3.13 – UML-диаграмма класса Registration

Метод onCreate устанавливает разметку активности, заголовок окна, а также выполняет привязку элементов графического интерфейса к объектам класса EditText.

Для регистрации необходимо нажать на кнопку «Зарегистрироваться», нажатие на которою обрабатывает метод on_Click_SignUp. Данный метод проверяет корректность введенных регистрационных данных. Если данные не прошли проверку, метод сбрасывает текст в полях для ввода, в противном случае объект инициализируется данными пользователя, затем создается объект класса HTTPRequest для выполнения запроса на регистрацию. Если регистрация прошла успешно, приложение возвращает пользователя на страницу авторизации. Если же пользователь уже зарегистрирован, приложение уведомит его об этом и так же предложит авторизоваться.

3.3. Описание графического интерфейса

После установки приложения на рабочем столе появится значок приложения с подписью «Calyx» (рис. 3.14).

Рисунок 3.14 – Значок приложения Calyx

При первом запуске приложения пользователь попадет на страницу авторизации, представленную на рисунке 3.15.

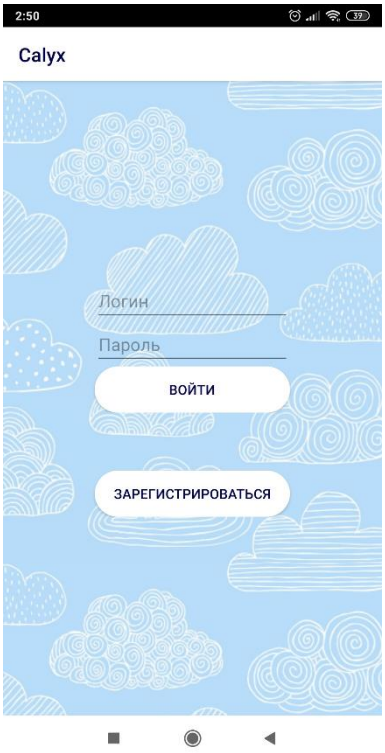

Рисунок 3.15 – Страница авторизации

Со страницы авторизации можно перейти к регистрации (рис. 3.16) нового пользователя путем нажатия на кнопку «Зарегистрироваться».

Рисунок 3.16 – Страница регистрации

Если регистрация прошла успешно, приложение уведомит об этом пользователя и перейдет к странице авторизации (рис. 3.17).

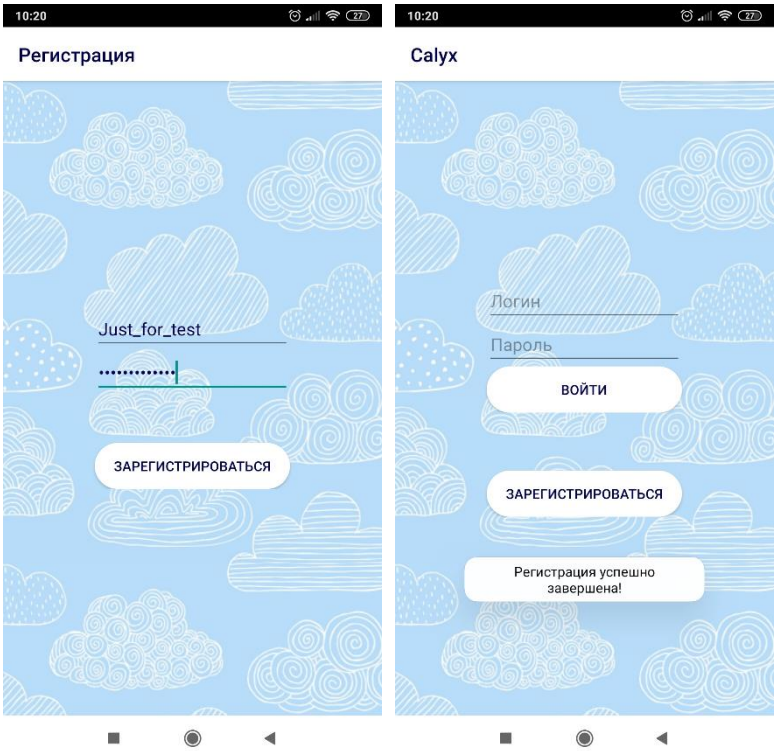

Рисунок 3.17 – Пример успешно завершенной регистрации

В случае же, если пользователь уже зарегистрирован, приложение так же предложит пользователю авторизоваться, показав уведомление, что пользователь уже зарегистрирован (рис. 3.18).

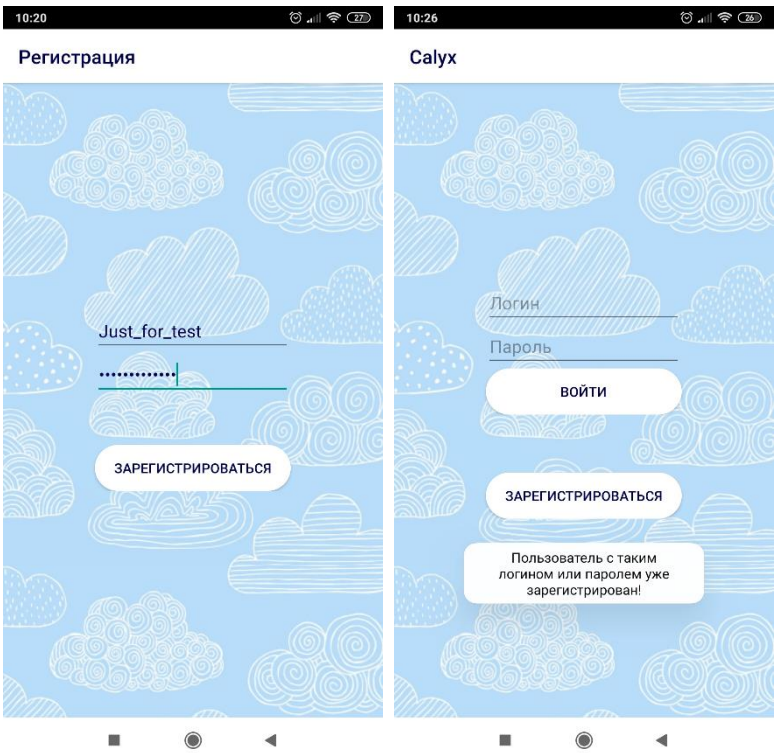

Рисунок 3.18 – Пример попытки повторной регистрации пользователя

Предусмотрена также проверка корректности логинов и паролей. При введение недопустимого логина или пароля, приложение сбросит текст в полях ввода и уведомит о допущенных ошибках (рис. 3.19).

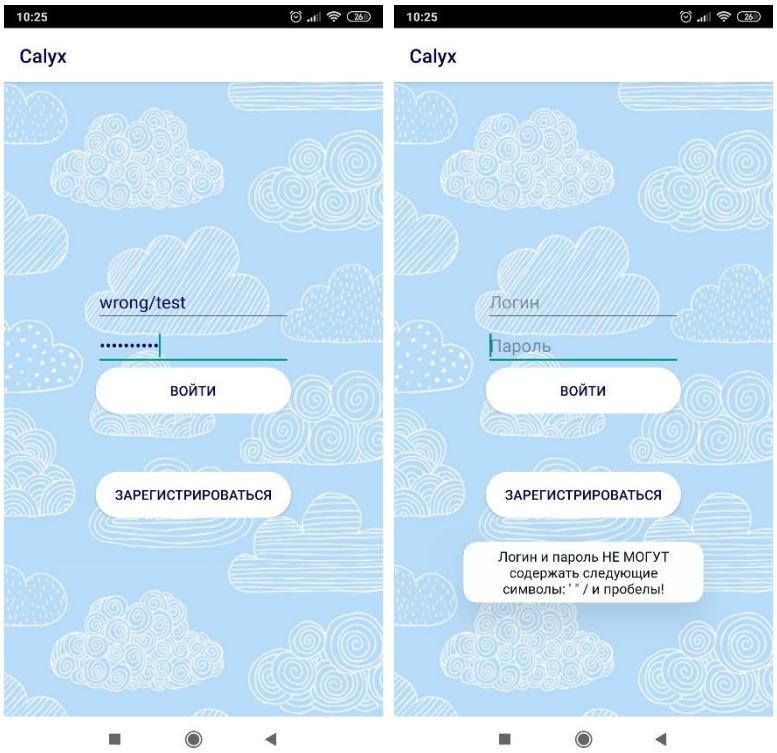

Рисунок 3.19 – Пример ввода некорректного логина и пароля

При регистрации логин и пароль проверяются аналогичным образом.

В первый запуск приложения, при попытке авторизоваться, пользователю будет предложен запрос на разрешение записи на внутренний общий накопитель. В случае отказа пользователю будет показано диалоговое окно с просьбой принять разрешение (рис. 3.20).

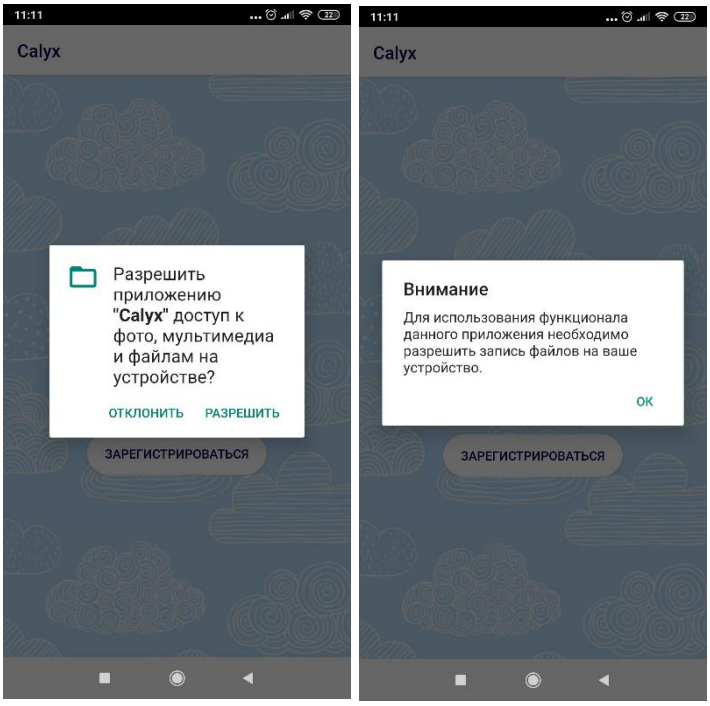

Рисунок 3.20 – Отказ в разрешении на запись файлов на устройство

При успешной авторизации приложение перейдет к странице личного кабинета и уведомит пользователя об успешном входе (рис. 3.21).

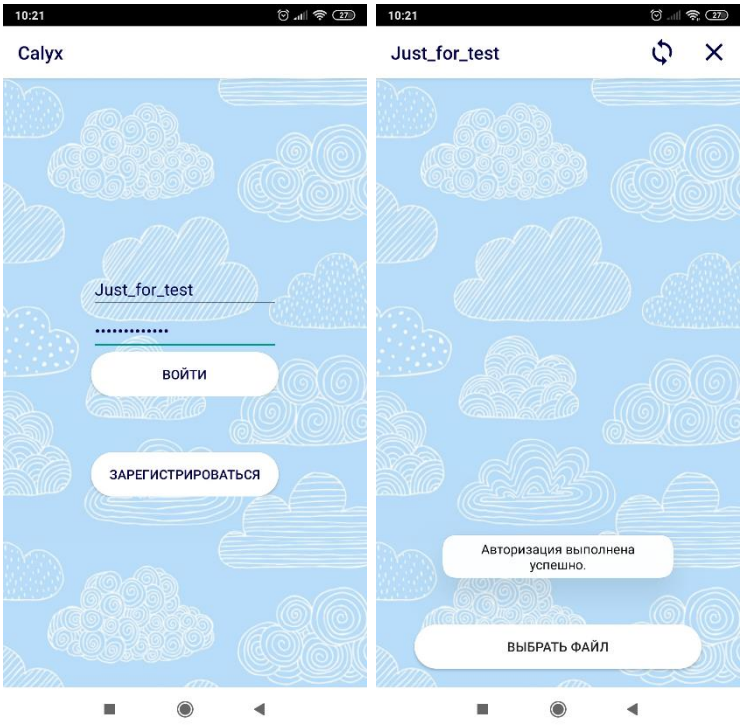

Рисунок 3.21 – Пример успешной авторизации

На рисунке выше, в окне авторизованного пользователя нет файлов, т.к. это новый пользователь. Приведем пример загрузки файла (рис. 3.22).

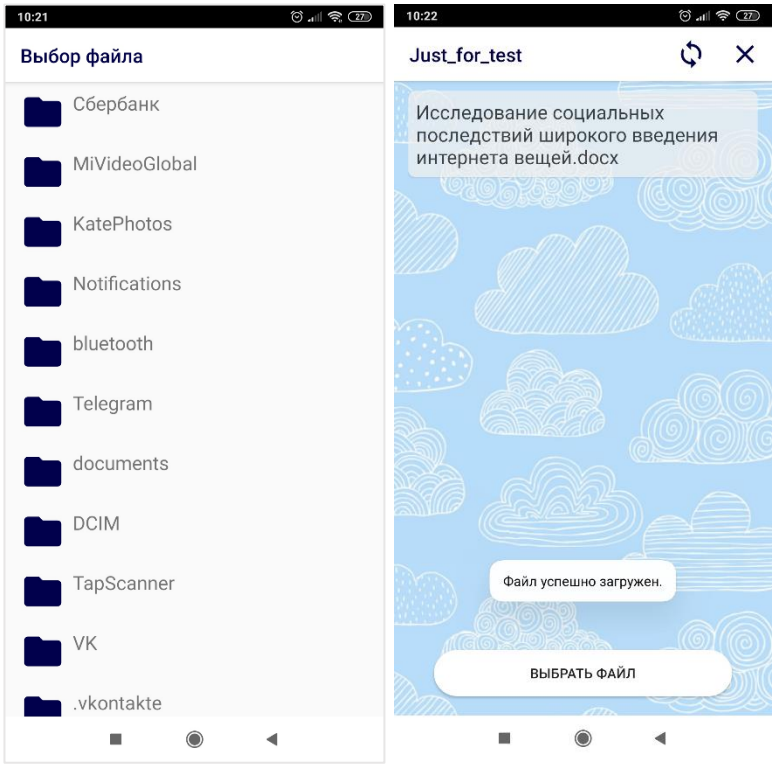

Рисунок 3.22 – Выбор файла и загрузка

Пользователь также может удалять файлы из хранилища (рис. 3.23), используя долгое нажатие на пункт списка.

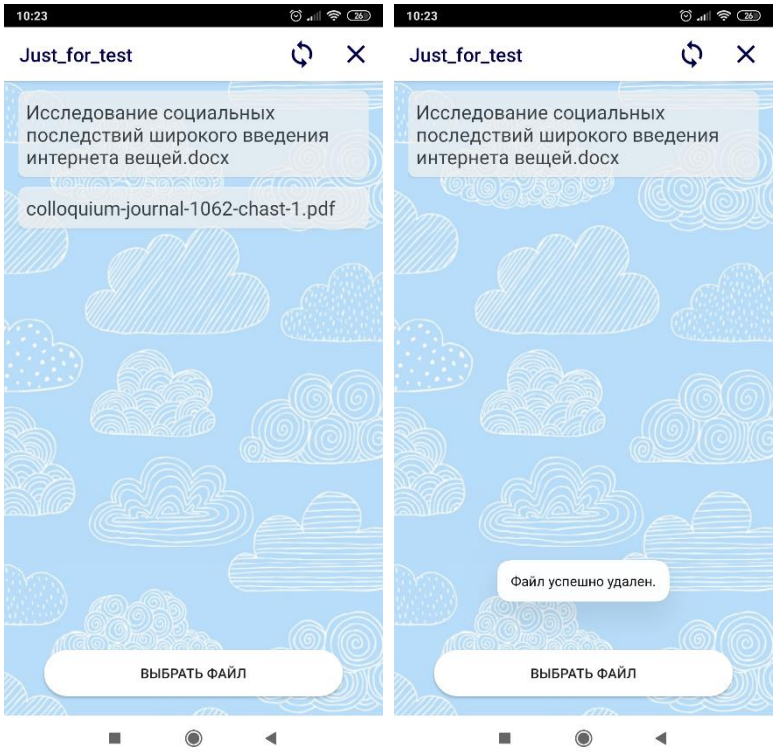

Рисунок 3.23 – Пример удаления файла (удален файл colloquium-journal)

Чтобы загрузить файл из облака на устройство нужно нажать на нужный пункт списка. Приложение уведомит пользователя о результате (рис. 3.24).

Рисунок 3.24 – Пример скачивания

Действительно, файл загрузился в указанную папку (рис. 3.25).

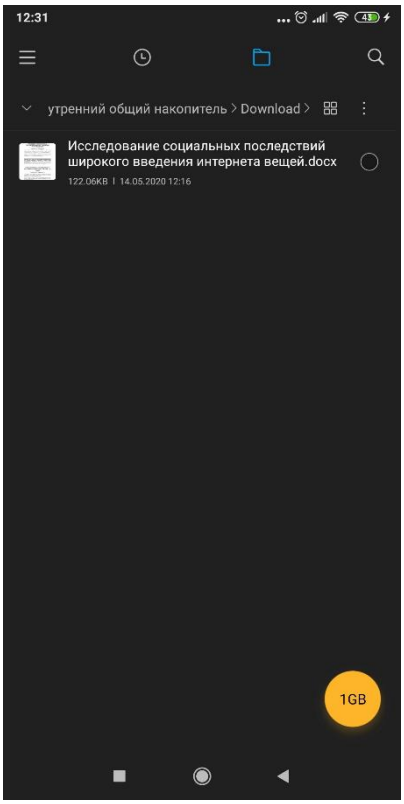

Рисунок 3.25 – Проверка результата работы функции загрузки

4. Используемые технические средства

Для корректного функционирования приложения необходимо иметь 2,23 МБ свободного места в внутреннем хранилище мобильного устройства. Также необходимо, чтобы на устройство была установлена версия ОС Android не старше версии 4.4 (KitKat, API level 19).

5. Вызов и загрузка

Для начала работы программы достаточно установить на мобильное устройство файл calyx.apk, содержащего архив с программой.

Заключение

По итогам разработки курсовой работы были выполнены следующие задачи:

— Был проведен анализ предметной области разработки программных приложений;

— Был определен перечень функциональных требований к программе;

— Были выбраны инструментальные средства для разработки программы;

— Были спроектированы модули программного приложения;

— Был разработан интерфейс программы;

— Был спроектирован и разработан функционал приложения;

Также были освоены работа с библиотекой OkHttp, методы отправки post и get запросов к серверу, изучены механизмы сохранения пользовательских данных на устройстве и автоматической авторизации.

Исходя из представленных тезисов, поставленную в начале данной работы цель можно считать выполненной.

Листинг приведен в приложении 2.

В результате выполнения курсовой работы была получена и отработана следующая компетенция: ПК-13 – готовность к использованию методов и инструментальных средств исследования объектов профессиональной деятельности.

Список использованных источников

1. RecyclerView [Электронный ресурс]: Metanit. Сайт о программировании. – Электрон. текстовые дан. – Режим доступа: https://metanit.com/java/android/5.11.php, свободный.

2. Documentation for app developers [Электронный ресурс]: Официальная документация для Android-разработчиков. RecyclerView. / Google Inc. – Электрон. текстовые дан. – Режим доступа: https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView?hl=he, свободный.

3. А. Климов. SharedPreferences [Электронный ресурс]: Сайт Александра Климова, 2020 – Режим доступа: developer.alexanderklimov.ru/android/theory/sharedpreferences.php, свободный.

4. OkHttp. Overview. [Электронный ресурс]: Официальная документация OkHttp. / Square, Inc. – Электрон. текстовые дан. – Режим доступа: https://square.github.io/okhttp/, свободный.

5. Реализуем выбор файла (File Picker). Система разрешений Android. Получение списка файлов. [Электронный ресурс]: Skillberg. – Электрон. текстовые дан. – Режим доступа: https://skillberg.com/courses/android/lessons/realizuem-vybor-fayla-sistema-razresheniy-android-poluchenie-spiska-faylov/, свободный

Приложение 1. Техническое задание

1.1. Введение

Составленное техническое задание по дисциплине “Разработка программных приложений” является документом к курсовой работе, который отображает этапы разработки, в том числе стадии проектирования и документирования программы.

1.1.1. Наименование программы

В качестве названия приложения было выбрано «Calyx».

1.1.2. Краткая характеристика области применения программы

Программа предназначена для загрузки файлов в облачное хранилище.

1.2. Основание для разработки

Основанием для разработки является потребность в эффективном и легковесном приложении для хранения файлов вне мобильного устройства с возможностью доступа к этим файлам с разных Android-устройств.

1.3. Назначение для разработки

Приложение предназначено для загрузки файлов пользователя в ОХД.

1.4. Требования, предъявляемые к программе

Приложение должно обеспечить возможность сохранения пользовательских файлов в облачном хранилище данных. Также требуется обеспечить пользователю удобный, интуитивно понятный интерфейс.

1.4.1. Взаимодействие продукта с другими продуктами и компонентами

Программный продукт должен взаимодействовать со сторонними API с помощью сети Интернет.

1.4.2. Функциональные требования

Приложению требуется обладать следующим функционалом:

1. предоставление пользователю возможность регистрации;
2. предоставление пользователю возможность авторизации;
3. загрузка файлов в ОХД;
4. удаление файлов из ОХД;
5. скачивание файлов из ОХД;

1.4.3. Требования к внешним средам

Приложению требуется доступ к сети Интернет и доступ к внутреннему общему накопителю устройства.

1.4.4. Требования к производительности

Требуется обеспечить стабильную работу приложения, таким образом, чтобы действия пользователя не вызвали аварийного завершения программы.

1.4.5. Нефункциональные требования

Предоставлять возможность открытия файлов в приложении не требуется.

1.5. Требования к программной документации

1. Пояснительная записка в соответствии со стандартами РТУ МИРЭА (СМКО МИРЭА 7.5.1/03.П.69-18);
2. Отчетная документация, составленная в соответствии с ГОСТ.

В процессе создания приложения вся проделанная работа должна быть задокументирована, должны быть отражены детали разработки. Всё вышеперечисленное должно быть отражено в пояснительной записке, которая прилагается к работе.

1.6. Технико-Экономические показатели

Технико-экономические требования к работе не предъявляются.

1.7. Стадии и этапы разработки

Этапы разработки:

1. Исследование концепции; (20.02.20 – 20.02.20)
2. Выработка требования и составление ТЗ; (21.02.20 – 21.02.20)
3. Проектирование; (13.03.20 – 15.03.20)
4. Реализация компонентов и написание логики программы; (15.03.20 – 30.04.20)
5. Интеграция компонентов: написание графического интерфейса; (30.04.20 – 05.05.20)
6. Отладка созданного приложения; (05.05.20 – 09.05.20)
7. Оформление отчетной документации; (10.03.19 – 13.05.20)
8. Представление работы к защите. (15.05.2020)

Дальнейшее сопровождение программы после сдачи и защиты курсового проекта не предусмотрено.

1.8. Порядок контроля и приема работы

Прием работы осуществляется путем защиты курсовой работы у преподавателя, назначенного руководителем курсовой работы. Защита осуществляется в соответствии с установленным графиком защиты курсовых работ.

Приложение 2. Код программы

Листинг 2.1 – Класс DataAdapter

```
package com.example.calyx;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.LinearLayout;
import android.widget.TextView;

import java.util.ArrayList;

public class DataAdapter extends ArrayAdapter<String> {
    private Context context;
    private ArrayList<String> values;

    public DataAdapter (Context context, ArrayList <String> values) {
        super(context, R.layout.server_file_itm, values);
        this.context = context;
        this.values = values;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

        LayoutInflater inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View view = inflater.inflate(R.layout.server_file_itm, parent, false);
        TextView textView = (TextView) view.findViewById(R.id.file_name);
        LinearLayout linearLayout = (LinearLayout)
view.findViewById(R.id.item);
        textView.setText(values.get(position));

        return view;
    }

}
```

Листинг 2.2 – Toast_creator

```
package com.example.calyx;

import android.content.Context;
import android.widget.Toast;

public class Toast_creator {
    public static void showToast (Context context, String text){
        Toast msg = Toast.makeText(context, text, Toast.LENGTH_LONG);
        msg.show();
    }
}
```

Листинг 2.3 – Класс Dialog_wnd

```
package com.example.calyx;

import android.app.Activity;
import androidx.appcompat.app.AlertDialog;
import android.content.DialogInterface;

public class Dialog_wnd {
    public static final int PERMISSION_READ = 0;
    public static final int PERMISSION_WRITE = 1;

    public static AlertDialog createDialog(Activity activity, int ID) {
        androidx.appcompat.app.AlertDialog.Builder builder = new
androidx.appcompat.app.AlertDialog.Builder(activity);

        builder.setTitle("Внимание");
        switch (ID) {
            case PERMISSION_READ:
                builder.setMessage("Для использования функционала данного
приложения необходимо разрешить доступ к файлам устройства.");
                break;

            case PERMISSION_WRITE:
                builder.setMessage("Для использования функционала данного
приложения необходимо разрешить запись файлов на ваше устройство.");
                break;
        }

        builder.setPositiveButton(android.R.string.ok, new
DialogInterface.OnClickListener() { // Кнопка ОК
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss(); // Отпускает диалоговое окно
            }
        });
        return builder.create();
    }
}
```

Листинг 2.4 – Класс FileManager

```
package com.example.calyx;
import android.content.Context;
import android.os.Environment;
import androidx.core.content.ContextCompat;
import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class FileManager {
    public File current_Directory;
    private final File root_Directory;

    public FileManager(Context context){
        File directory;

        if
(Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState())) {
            directory = Environment.getExternalStorageDirectory();
        } else {
            directory = ContextCompat.getDataDir(context);
        }
```

```
        root_Directory = directory;
        setCurrentDirectory(directory);
    }

    //ПЕРЕМЕЩЕНИЕ В ДИРЕКТОРИЮ (УСТАНОВКА ТЕКУЩЕЙ ДИРЕКТОРИИ)
    public boolean setCurrentDirectory(File directory) {

        if (!directory.isDirectory()) {// Если это не директория
            return false;
        }
        // Если влезли куда не надо
        if (!directory.equals(root_Directory) &&
root_Directory.getAbsolutePath().contains(directory.getAbsolutePath())) {
            return false;
        }

        current_Directory = directory;
        return true;
    }

    //BACK TO THE ПРОШЛАЯ ДИРЕКТОРИЯ
    public boolean navigateUp() {
        return setCurrentDirectory(current_Directory.getParentFile());
    }

    //СОЗДАНИЕ СПИСКА ФАЙЛОВ ДИРЕКТОРИИ
    public List<File> getFiles() {
        List<File> files = new ArrayList<>();
        files.addAll(Arrays.asList(current_Directory.listFiles()));// Добываем
список файлов и поддиректорий

        return files;
    }
}
```

Листинг 2.5 – Класс FilePickerActivity

```
package com.example.calyx;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;

import java.io.File;
import java.util.List;


public class FilePickerActivity extends AppCompatActivity {
   private FileManager fileManager;
   private FilesAdapter adapter;

   public static final String CHOSEN_FILE_PATH = "file_path";//КЛЮЧ ДЛЯ
ПЕРЕДАЧИ ПУТИ К ФАЙЛУ

   //СЛУШАТЕЛЬ СОБЫТИЯ "ПРИЛЕТЕЛ ФАЙЛИК"
   private final FilesAdapter.OnFileClickListener onFileClickListener = new
FilesAdapter.OnFileClickListener() {
```

```
        @Override
        public void onFileClick(File file) { //приняли событие

            //И ЧТО С ЭТИМ ДЕЛАТЬ ТЕПЕРЬ?
            if (file.isDirectory()) { //если кликнули на ДИРЕКТОРИЮ
                fileManager.setCurrentDirectory(file);//переходим в эту
директорию
                updateFileList();//и обновляем список на экране
            }
            else{ //если кликнули на ФАЙЛ
                Intent intent = new Intent(); //создаем интент
                intent.putExtra(CHOSEN_FILE_PATH, file.getAbsolutePath()); //
вместе с интентом передаем путь к файлу
                setResult(RESULT_OK, intent);//так же сообщаем, что все прошло
нормально
                finish();//возвращаемся к прошлой активити
            }
        }
    };


    //КОНСТРУКТОР
     @Override
     protected void onCreate(@Nullable Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_file_picker);

         RecyclerView recyclerView = findViewById(R.id.files_rv);
         recyclerView.setLayoutManager(new LinearLayoutManager(this));
         setTitle("Выбор файла");

         //установка адаптера, чтоббыло красиво и тык тык
         adapter = new FilesAdapter();
         recyclerView.setAdapter(adapter);

         //запуск создания списка файлов
         initFileManager();
         updateFileList();
     }


     @Override
     protected void onStart() {
         super.onStart();
         adapter.setOnFileClickListener(onFileClickListener);
     }


     // ПРИ ОСТАНОВКЕ АКТИВИТИ ОТПИСЫВАЕМСЯ ОТ ПОЛУЧЕНИЙ СОБЫТИЙ
     @Override
     protected void onStop() {
         adapter.setOnFileClickListener(null);
         super.onStop();
     }


     //СОЗДАЕМ ФАЙЛОВЫЙ МЕНЕДЖЕР
     private void initFileManager() {
         if (ContextCompat.checkSelfPermission(this,
Manifest.permission.READ_EXTERNAL_STORAGE)
                 == PackageManager.PERMISSION_GRANTED) { // Разрешение
предоставлено
```

```
            fileManager = new FileManager(this);
        }
    }

    //ОБНОВЛЕНИЕ СПИСКА, ЕСЛИ ЖМЯКНУЛИ ПО ПАПКЕ
    private void updateFileList() {
        List<File> files = fileManager.getFiles();

        adapter.setFiles(files);
        adapter.notifyDataSetChanged();
    }

    //ПРИ ЗАПУСКЕ АКТИВИТИ УСТАНАВЛИВАЕМ СЛУШАТЕЛЬ В АДАПТЕР


    @Override
    public void onBackPressed() {
        // если файловый менеджер создан и  можно подняться выше (т.е мы не в
rootDir)
        if (fileManager != null && fileManager.navigateUp()) {
            updateFileList();//обновить список
        }
        else {//если мы в rootDir, то вернуться в активити
            super.onBackPressed();
        }
    }

}
```

Листинг 2.6 – Класс FilesAdapter

```
package com.example.calyx;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.Nullable;
import androidx.recyclerview.widget.RecyclerView;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

public class FilesAdapter extends
RecyclerView.Adapter<FilesAdapter.ViewHolder> {

        private List<File> files = new ArrayList<>();
        private static final int DIRECTORY = 0;
        private static final int FILE = 1;

        @Nullable
        private OnFileClickListener onFileClickListener;


        //УСТАНОВКА СЛУШАТЕЛЯ ДЛЯ ПОЛУЧЕНИЯ КЛИКОВ ИЗ СПИСКА
        public void setOnFileClickListener(@Nullable OnFileClickListener
onFileClickListener) {
            this.onFileClickListener = onFileClickListener;
        }

        //ПЕРЕНОС ДАННЫХ В ОБЪЕКТ КЛАССА
        public void setFiles(List<File> files) {
```

```
            this.files = files;
        }


        //УСТАНОВКА РАЗМЕТКИ
        @Override
        public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
            LayoutInflater layoutInflater =
LayoutInflater.from(parent.getContext());
            View view;
            if (viewType == FILE) {
                view = layoutInflater.inflate(R.layout.file_mng_item, parent,
false);
            }
            else{
                view = layoutInflater.inflate(R.layout.dir_mng_item, parent,
false);
            }
            return new ViewHolder(view);
        }

        //СВЯЗЬ ФАЙЛОВ С ЭЛЕМЕНТАМИ СПИСКА
        @Override
        public void onBindViewHolder(ViewHolder holder, int position) {
            File file = files.get(position);
            holder.nameTv.setText(file.getName());
            holder.itemView.setTag(file);
        }

        //ПОЛУЧЕНИЕ КОЛИЧЕСТВА ФАЙЛОВ/ДИРЕКТОРИЙ
        @Override
        public int getItemCount() {
            return files.size();
        }


        //ДЛЯ ТОГО, ЧТОБ РАЗЛИЧАТЬ ФАЙЛЫ И ДИРЕКТОРИИ
        @Override
        public int getItemViewType(int position) {
            File file = files.get(position);
            if (file.isDirectory()) {
                return DIRECTORY;
            } else {
                return FILE;
            }
        }


        //ДЛЯ ОБРАБОТКИ НАЖАТИЯ
        class ViewHolder extends RecyclerView.ViewHolder {
        private final TextView nameTv;

        public ViewHolder(View itemView) {
            super(itemView);
            nameTv = itemView.findViewById(R.id.name_tv);

            itemView.setOnClickListener(new View.OnClickListener() {
//слушатель нажалий на item
                @Override
                public void onClick(View view) {
                    File file = (File) view.getTag(); // получение файла по
тегу
                    if (onFileClickListener != null) {
```

```
                        //список отправляет в активити файл
                        onFileClickListener.onFileClick(file);
                    }
                }
            });
        }
        }


        //ПЕРЕДАЧИ КЛИКА ИЗ СПИСКА В АКТИВИТИ / А-ЛЯ СИГНАЛ
        public interface OnFileClickListener {
            void onFileClick(File file);
        }
}
```

Листинг 2.7 – Класс HTTPRequest

```
package com.example.calyx;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.Nullable;
import androidx.recyclerview.widget.RecyclerView;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

public class FilesAdapter extends
RecyclerView.Adapter<FilesAdapter.ViewHolder> {

        private List<File> files = new ArrayList<>();
        private static final int DIRECTORY = 0;
        private static final int FILE = 1;

        @Nullable
        private OnFileClickListener onFileClickListener;


        //УСТАНОВКА СЛУШАТЕЛЯ ДЛЯ ПОЛУЧЕНИЯ КЛИКОВ ИЗ СПИСКА
        public void setOnFileClickListener(@Nullable OnFileClickListener
onFileClickListener) {
            this.onFileClickListener = onFileClickListener;
        }

        //ПЕРЕНОС ДАННЫХ В ОБЪЕКТ КЛАССА
        public void setFiles(List<File> files) {
            this.files = files;
        }


        //УСТАНОВКА РАЗМЕТКИ
        @Override
        public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
            LayoutInflater layoutInflater =
LayoutInflater.from(parent.getContext());
            View view;
            if (viewType == FILE) {
                view = layoutInflater.inflate(R.layout.file_mng_item, parent,
false);
            }
```

```
            else{
                view = layoutInflater.inflate(R.layout.dir_mng_item, parent,
false);
            }
            return new ViewHolder(view);
        }

        //СВЯЗЬ ФАЙЛОВ С ЭЛЕМЕНТАМИ СПИСКА
        @Override
        public void onBindViewHolder(ViewHolder holder, int position) {
            File file = files.get(position);
            holder.nameTv.setText(file.getName());
            holder.itemView.setTag(file);
        }

        //ПОЛУЧЕНИЕ КОЛИЧЕСТВА ФАЙЛОВ/ДИРЕКТОРИЙ
        @Override
        public int getItemCount() {
            return files.size();
        }


        //ДЛЯ ТОГО, ЧТОБ РАЗЛИЧАТЬ ФАЙЛЫ И ДИРЕКТОРИИ
        @Override
        public int getItemViewType(int position) {
            File file = files.get(position);
            if (file.isDirectory()) {
                return DIRECTORY;
            } else {
                return FILE;
            }
        }


        //ДЛЯ ОБРАБОТКИ НАЖАТИЯ
        class ViewHolder extends RecyclerView.ViewHolder {
        private final TextView nameTv;

        public ViewHolder(View itemView) {
            super(itemView);
            nameTv = itemView.findViewById(R.id.name_tv);

            itemView.setOnClickListener(new View.OnClickListener() {
//слушатель нажалий на item
                @Override
                public void onClick(View view) {
                    File file = (File) view.getTag(); // получение файла по
тегу
                    if (onFileClickListener != null) {
                        //список отправляет в активити файл
                        onFileClickListener.onFileClick(file);
                    }
                }
            });
        }
        }


        //ПЕРЕДАЧИ КЛИКА ИЗ СПИСКА В АКТИВИТИ / А-ЛЯ СИГНАЛ
        public interface OnFileClickListener {
            void onFileClick(File file);
        }
}
```

Листинг 2.8 – Класс Log_Pass

```
package com.example.calyx;

import android.content.Context;

public class Log_Pass {
    public static final int LOGIN = 0;
    public static final int PASSWORD = 1;

    private static String checkLength(String str, int type){

        String warning = "";
        switch (type) {
            case LOGIN:
                if ((str.length() < 6) || (str.length() > 32)) {
                    warning = "Логин должен содержать от 6 до 32 символов!\n";
                }
                break;
            case PASSWORD:
                if ((str.length() < 6) || (str.length() > 32)) {
                    warning = "Пароль должен содержать от 8 до 32
символов!\n";
                }
                break;
        }
        return warning;
    }

    private static String frbdnCharacters(String str){
        String warning ="";
        if (str.contains(" ") || str.contains("/") ||
                (str.contains("\'")) || str.contains("\"")){
            warning +="Логин и пароль НЕ МОГУТ содержать следующие символы: \'
\" / и пробелы!";
        }
        return warning;
    }

    public static boolean checkLogPass(String log, String pass, Context
context) {

        String warning = checkLength(log, Log_Pass.LOGIN)+checkLength(pass,
Log_Pass.PASSWORD);
        if (frbdnCharacters(log) != ""){
            warning += frbdnCharacters(log);
        }
        else{
            warning += frbdnCharacters(pass);
        }

        if (warning != "") { //Если есть какие то замечания
            Toast_creator.showToast(context,warning);
            return false;
        }
        else{
            return true;
        }
    }
}
```

Листинг 2.9 – Класс MainActivity

```
package com.example.calyx;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import android.Manifest;
import android.content.Context;
import android.content.Intent; // подключаем класс Intent
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {
    private static final int PERMISSION_REQUEST_CODE = 0;
    private EditText username;
    private EditText password;
    private User user;

    //для автоматической авторизации
    private static final String APP_PREFERENCES = "settings";
    private static final String APP_PREFERENCES_LOGIN = "login";
    private static final String APP_PREFERENCES_PASSWORD = "password";
    private static SharedPreferences settings;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        username = (EditText) (findViewById(R.id.login_Edit));
        password = (EditText) (findViewById(R.id.password_Edit));

        settings = getSharedPreferences(APP_PREFERENCES,
Context.MODE_PRIVATE);

        //если пароль был ранее сохранен
        if
((settings.contains(APP_PREFERENCES_LOGIN))&(settings.contains(APP_PREFERENCES
_LOGIN))){
            user = new User(settings.getString(APP_PREFERENCES_LOGIN, ""),
                            settings.getString(APP_PREFERENCES_PASSWORD,""),
                            "");

            HTTPRequest author = new HTTPRequest(user, HTTPRequest.AUTHOR);
            author.setContext(this);

            if (!author.execute()) Toast_creator.showToast(this, "Ошибка
соединения с сервером!");
            else{
                Intent signIn = new Intent(this, Usr_Wind.class);
                signIn.putExtra("login", user.getLogin());
                signIn.putExtra("password", "");
                signIn.putExtra("userKey", user.getUserKey());
                startActivity(signIn);
                finish();
            }
        }
        else {
```

```
            username = (EditText) (findViewById(R.id.login_Edit));
            password = (EditText) (findViewById(R.id.password_Edit));
        }
    }

    @Override
    protected void onStart() {
        super.onStart();
        username.setText(null);
        password.setText(null);
    }

    public static SharedPreferences getSettings() {
        return settings;
    }

    //КНОПКА РЕГИСТРАЦИИ
    public void on_Click_reg(View view) {
        Intent reg = new Intent(this, Registration.class);
        startActivity(reg);
    }

    //КНОПКА ВХОДА В АККАУНТ
    public void on_Click_sighIn(View view) {
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE)!=
PackageManager.PERMISSION_GRANTED) { // Разрешение НЕ предоставлено
            requestPermissions(); //то спросить разрешения
        }
        else {// если разрешение уже есть

            //проверяем пароль и логин
            if (!Log_Pass.checkLogPass(username.getText().toString(),
password.getText().toString(), getApplicationContext())) {
                username.setText(null);
                password.setText(null);
            }
            else { //делаем запрос на сервер
                user = new User(username.getText().toString(),
password.getText().toString(), "");
                HTTPRequest author = new HTTPRequest(user,
HTTPRequest.AUTHOR);
                author.setContext(this);

                if (!author.execute()) Toast_creator.showToast(this, "Ошибка
соединения с сервером!");
                else {
                    if (author.getUser().getUserKey().contains("Deny")) {
                        username.setText(null);
                        password.setText(null);
                    } else {
                        //записываем правильные логин и пароль в файл настроек
                        SharedPreferences.Editor editor = settings.edit();

editor.putString(APP_PREFERENCES_LOGIN,user.getLogin());

editor.putString(APP_PREFERENCES_PASSWORD,user.getPassword());
                        editor.apply();

                        //вызываем следующий активити
                        Intent signIn = new Intent(this, Usr_Wind.class);
                        signIn.putExtra("login", user.getLogin());
                        signIn.putExtra("password", "");
```

```
                        signIn.putExtra("userKey", user.getUserKey());
                        startActivity(signIn);
                    }
                }
            }
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);

        if (requestCode == PERMISSION_REQUEST_CODE) {

            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {//если
дали разрешение
                user = new User(username.getText().toString(),
password.getText().toString(), "");

                HTTPRequest author = new HTTPRequest(user,
HTTPRequest.AUTHOR);
                author.setContext(this);

                if (!author.execute()) Toast_creator.showToast(this, "Ошибка
соединения с сервером!");
                else {
                    if (author.getUser().getUserKey().contains("Deny")) {
                        username.setText(null);
                        password.setText(null);
                    } else {
                        settings = getSharedPreferences(APP_PREFERENCES,
Context.MODE_PRIVATE);
                        SharedPreferences.Editor editor = settings.edit();
                        editor.putString(APP_PREFERENCES_LOGIN,
user.getLogin());
                        editor.putString(APP_PREFERENCES_PASSWORD,
user.getPassword());
                        editor.apply();

                        Intent signIn = new Intent(this, Usr_Wind.class);
                        signIn.putExtra("login", user.getLogin());
                        signIn.putExtra("password", "");
                        signIn.putExtra("userKey", user.getUserKey());
                        startActivity(signIn);
                    }
                }

            } else { // если не дали разрешение
                AlertDialog dialog = Dialog_wnd.createDialog(this,
Dialog_wnd.PERMISSION_WRITE);
                dialog.show(); //покажем окошко с предупреждением
            }
        }
    }

    private void requestPermissions() { // запрос разрешения на запись файлов
        ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},
                PERMISSION_REQUEST_CODE);
    }
}
```

Листинг 2.10 – Класс Registration

```
package com.example.calyx;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class Registration extends AppCompatActivity {
    private EditText login_edit;
    private EditText password_edit;
    private User user;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registration);
        setTitle("Регистрация");
        login_edit = (EditText) (findViewById(R.id.name_ed));
        password_edit = (EditText) (findViewById(R.id.password_ed));
    }

    //ОБРАБОТЧИК КНОПКИ
    public void on_Click_SignUp(View view) {

        if (!Log_Pass.checkLogPass(login_edit.getText().toString(),
password_edit.getText().toString(), getApplicationContext())){
            login_edit.setText(null);
            password_edit.setText(null);
        }

        else {
            user = new User(login_edit.getText().toString(),
                            password_edit.getText().toString(),
                            null);

            HTTPRequest reg = new HTTPRequest(user, HTTPRequest.REGISTRATION);
            reg.setContext(this);

            if (!reg.execute()) Toast_creator.showToast(this, "Ошибка
соединения с сервером!");
            else {
                //Перезапуск акивити входа
                Intent restartMain = new Intent(this, MainActivity.class);
                startActivity(restartMain);
                finish();
            }
        }
    }

}
```

Листинг 2.11 – Класс User

```
package com.example.calyx;


public class User  {
    private String login;
    private String password;
    private String userKey;


    public User(String login, String password, String userKey){
        this.login = login;
        this.password = password;
        this.userKey = userKey;
    }

    public  String getLogin(){
        return login;
    }

    public String getPassword() {
        return password;
    }

    public String getUserKey() {
        return userKey;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public void setUserKey(String userKey) {
        this.userKey = userKey;
    }
}
```

Листинг 2.12 – Класс Usr_wind

```
package com.example.calyx;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import android.Manifest;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.util.Log;
import android.view.Menu;
import android.os.Bundle;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
```

```
import android.widget.TextView;
import org.json.JSONArray;
import org.json.JSONException;
import java.util.ArrayList;

public class Usr_Wind extends AppCompatActivity {
    private static final String TAG = "USER_WIND";
    private static final int PERMISSION_REQUEST_CODE = 1; //код запрашиваемого
разрешения
    private static final int REQUEST_CODE_PICK_FILE = 1;

    private String file_path;
    private User user;
    private JSONArray JSONlist;
    private ArrayList<String> listfiles;
    private ListView list;
    private DataAdapter adapter;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_usr_wind);

        Bundle arguments = getIntent().getExtras();

        user = new User(arguments.get("login").toString(),
                        "",
                        arguments.get("userKey").toString());

        setTitle(user.getLogin());

        updateList(); //обновление списка файлов

    }

    private void updateList() {
        //Запрос на получение списка файлов
        HTTPRequest files = new HTTPRequest(user, HTTPRequest.FILELIST);
        files.setContext(this);

        if (!files.execute()) Toast_creator.showToast(this, "Ошибка соединения
с сервером!");
        else {

            if(!files.getResponseText().contains("No files")) {
                try {
                    JSONlist = new JSONArray(files.getResponseText());
                } catch (JSONException e) {
                    System.out.println(e);
                }

                //Распарсим JSON
                listfiles = new ArrayList<>();
                for (int i = 0; i < JSONlist.length(); i++) {
                    try {
                        listfiles.add(JSONlist.getString(i).substring(2,
JSONlist.getString(i).length() - 2));
                        //System.out.println(listfiles.get(i));
                    } catch (JSONException e) {
                        Log.e(TAG, e.getMessage());
                    }
                }
```

```
                //Установим адаптер
                list = (ListView) findViewById(R.id.fileList);
                adapter = new DataAdapter(this, listfiles);
                list.setAdapter(adapter);
                list.setLongClickable(true);

                list.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
                    @Override
                    public void onItemClick(AdapterView<?> parent, View
itemClicked, int position, long id) {
                        TextView fileName = (TextView)
itemClicked.findViewById(R.id.file_name);

                        HTTPRequest download = new
HTTPRequest(fileName.getText().toString(), user, HTTPRequest.DOWNLOAD);
                        download.setContext(getApplicationContext());
                        if(!download.execute())
Toast_creator.showToast(getApplicationContext(),"Ошибка соединения с
сервером!");
                    }
                });

                list.setOnItemLongClickListener(new
AdapterView.OnItemLongClickListener() {
                    @Override
                    public boolean onItemLongClick(AdapterView<?> parent, View
itemClicked, int position, long id) {
                        TextView fileName = (TextView)
itemClicked.findViewById(R.id.file_name);

                        HTTPRequest drop = new
HTTPRequest(fileName.getText().toString(), user, HTTPRequest.DROP);
                        drop.setContext(getApplicationContext());
                        if (!drop.execute())
Toast_creator.showToast(getApplicationContext(), "Ошибка соединения с
сервером!");
                        else{
                            String selectedItem =
parent.getItemAtPosition(position).toString();

                            adapter.remove(selectedItem);
                            adapter.notifyDataSetChanged();
                        }

                        return true;
                    }
                });
            }
        }
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);

        MenuInflater menuInflater = getMenuInflater();
        menuInflater.inflate(R.menu.menu_usr_wind, menu);
        return  true;
    }
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()){
            case R.id.exit:
                //Очищение данных пользователя
                SharedPreferences.Editor editor =
MainActivity.getSettings().edit();
                editor.clear();
                editor.apply();

                Intent restartMain = new Intent(this, MainActivity.class);
                startActivity(restartMain);

                finish();
                break;

            case R.id.update:
                updateList();
                break;
        }
        return true;
    }


    private void requestPermissions() { // запрос разрешения
        ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.READ_EXTERNAL_STORAGE},
                PERMISSION_REQUEST_CODE);
    }


    //ДЛЯ ОБРАБОТКИ РАЗРЕШЕНИЯ
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);

        if (requestCode == PERMISSION_REQUEST_CODE) {

            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {//если
дали разрешение
                Intent startFileManager = new Intent(this,
FilePickerActivity.class); //запустить новое активити
                startActivity(startFileManager);

            } else { // если не дали разрешение
                AlertDialog dialog = Dialog_wnd.createDialog(this,
Dialog_wnd.PERMISSION_READ);
                dialog.show(); //покажем окошко с предупреждением
            }
        }
    }

    //КНОПА ДЛЯ ЗАГРУЗКИ НОВОГО ФАЙЛА
    public void on_Click_files(View view) {
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.READ_EXTERNAL_STORAGE)
                != PackageManager.PERMISSION_GRANTED) { // Разрешение НЕ
предоставлено
            requestPermissions(); //то спросить разрешения
            //ВЫЗОВ onRequestPermissionsResult
```

```
        } else {// если разрешение уже есть
            Intent startFileManager = new Intent(this,
FilePickerActivity.class);
            startActivityForResult(startFileManager, REQUEST_CODE_PICK_FILE);
// запуск нового активити
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        if (requestCode == REQUEST_CODE_PICK_FILE && resultCode == RESULT_OK)
{
            file_path =
data.getStringExtra(FilePickerActivity.CHOSEN_FILE_PATH);
            Log.i(TAG, file_path);
            HTTPRequest upload = new HTTPRequest(file_path, user,
HTTPRequest.UPLOAD);
            upload.setContext(this);
            if(!upload.execute())
Toast_creator.showToast(getApplicationContext(), "Ошибка соединения с
сервером!");
            else updateList();

        } else {
            super.onActivityResult(requestCode, resultCode, data);
        }
    }

    @Override
    public void onBackPressed() {
        SharedPreferences.Editor editor = MainActivity.getSettings().edit();
        editor.clear();
        editor.apply();

        Intent restartMain = new Intent(this, MainActivity.class);
        startActivity(restartMain);
        finish();
    }
}
```

Приложение 3. Файлы разметки и прочие файлы

Листинг 3.1 – Разметка activity_file_picker.xml

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

<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.recyclerview.widget.RecyclerView
        xmlns:app="http://schemas.android.com/apk/res-auto"
        app:layoutManager="androidx.recyclerview.widget.GridLayoutManager"

        android:id="@+id/files_rv"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />


</FrameLayout>
```

Листинг 3.2 – Разметка activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:background="@drawable/background"
    android:shrinkColumns="*"
    android:stretchColumns="wrap_content">

    <TableRow android:id="@+id/row_0"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center" >

        <EditText android:id="@+id/login_Edit"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:inputType="text"
            android:maxLength="32"
            android:hint="Логин"
            android:textColor="#00004d">
        </EditText>

    </TableRow>

    <TableRow android:id="@+id/row_1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center" >

        <EditText android:id="@+id/password_Edit"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textPassword"
            android:hint="Пароль"
            android:maxLength="32"
            android:textColor="#00004d">
        </EditText>
```

```
    </TableRow>
    <TableRow android:id="@+id/row_2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center" >

        <Button android:id="@+id/sign_in_Button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Войти"
            android:textColor="#00004d"
            android:background="@drawable/oval"
            android:onClick="on_Click_sighIn">
        </Button>

    </TableRow>

    <TableRow
        android:id="@+id/row_3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal">

        <Button
            android:id="@+id/reg_Button"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_marginTop="60dp"
            android:text="Зарегистрироваться"
            android:textColor="#00004d"
            android:background="@drawable/oval"
            android:onClick="on_Click_reg"
            android:paddingRight="20dp"
            android:paddingLeft="20dp">

        </Button>
    </TableRow>
</TableLayout>
```

Листинг 3.3 – Разметка activity_registration.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:background="@drawable/background"
    android:shrinkColumns="*"
    android:stretchColumns="wrap_content">

    <TableRow android:id="@+id/row_0"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center">

        <EditText android:id="@+id/name_ed"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:inputType="text"
            android:maxLength="32"
            android:hint="Ваше имя"
            android:textColor="#00004d">
```

```
        </EditText>
    </TableRow>


    <TableRow android:id="@+id/row_1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center">

        <EditText android:id="@+id/password_ed"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:inputType="textPassword"
            android:hint="Пароль"
            android:maxLength="32"
            android:textColor="#00004d">
        </EditText>
    </TableRow>

    <TableRow
        android:id="@+id/row_3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal">

        <Button
            android:id="@+id/reginstration"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="50dp"
            android:text="Зарегистрироваться"
            android:textColor="#00004d"
            android:background="@drawable/oval"
            android:paddingRight="20dp"
            android:paddingLeft="20dp"
            android:onClick="on_Click_SignUp">

        </Button>
    </TableRow>
</TableLayout>
```

Листинг 3.4 – Разметка activity_usr_wind.xml

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

      <RelativeLayout
          android:id="@+id/rel_layout"
          xmlns:android="http://schemas.android.com/apk/res/android"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:background="@drawable/background">

          <ListView
              android:id="@+id/fileList"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:layout_marginBottom="80dp"
              android:paddingHorizontal="15dp"
              android:dividerHeight="10dp"
              android:divider="@null"
              android:layout_marginTop="10dp">
          </ListView>
```

```
        <Button
            android:id="@+id/files_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_centerHorizontal="true"
            android:text="Выбрать файл"
            android:paddingLeft="100dp"
            android:paddingRight="100dp"
            android:layout_marginBottom="20dp"
            android:background="@drawable/oval"
            android:onClick="on_Click_files">
        </Button>

    </RelativeLayout>
```

Листинг 3.5 – Разметка dir_mng_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:paddingBottom="8dp"
    android:paddingEnd="8dp"
    android:paddingStart="16dp"
    android:paddingTop="8dp">

    <ImageView
        android:id="@+id/icon_iv"
        android:layout_width="46dp"
        android:layout_height="46dp"
        android:src="@drawable/ic_folder_blue_92dp" />

    <TextView
        android:id="@+id/name_tv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:textSize="20dp"/>
</LinearLayout>
```

Листинг 3.6 – Разметка file_mng_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:paddingBottom="8dp"
    android:paddingEnd="8dp"
    android:paddingStart="16dp"
    android:paddingTop="8dp">
    <ImageView
        android:id="@+id/icon_iv"
        android:layout_width="46dp"
        android:layout_height="46dp"
        android:src="@drawable/ic_file_blue_92dp"/>
    <TextView
        android:id="@+id/name_tv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:textSize="20dp"/>
</LinearLayout>
```

Листинг 3.7 – Разметка server_file_itm.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:paddingBottom="8dp"
    android:paddingEnd="8dp"
    android:paddingStart="16dp"
    android:paddingTop="8dp">

    <ImageView
        android:id="@+id/icon_iv"
        android:layout_width="46dp"
        android:layout_height="46dp"
        android:src="@drawable/ic_file_blue_92dp"/>

    <TextView
        android:id="@+id/name_tv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:textSize="20dp"/>

</LinearLayout>
```

Листинг 3.8 – Разметка menu_usr_wind.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    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"
    tools:context=".Usr_Wind">
    <item
        android:id="@+id/update"
        android:title="@string/update"
        app:showAsAction="always"
        android:icon="@drawable/ic_loop_blue_92dp">
    </item>

    <item
        android:id="@+id/exit"
        android:title="@string/exit"
        app:showAsAction="always"
        android:icon="@drawable/ic_exit_blue_92dp">
    </item>
</menu>
```

Листинг 3.9 – Файл настроек сетевой безопасности network_sequrity_config.xml

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">https://calyx.space</domain>
    </domain-config>
</network-security-config>
```

Листинг 3.10 – Файл строковых констант strings.xml

```
<resources>
    <string name="app_name">Calyx</string>
    <string name="exit">Выход</string>
    <string name="update">Обновить</string>
</resources>
```

Листинг 3.11 – Файл определения цветов colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#008577</color>
    <color name="colorPrimaryDark">#00574B</color>
    <color name="white">#ffffff</color>
    <color name="primary_text_default_material_dark">#00004d</color>
    <color name="black">#0a0909</color>>
</resources>
```

Листинг 3.12 – Манифест AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:targetSandboxVersion="1"
    package="com.example.calyx">

    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/orange_cup"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"

        android:networkSecurityConfig="@xml/network_security_config"
        android:usesCleartextTraffic="true">

        <activity
            android:name=".FilePickerActivity">
            <!--android:screenOrientation= "portrait">-->
        </activity>

        <activity
            android:name=".Usr_Wind">
        <!--android:screenOrientation= "portrait">-->
        </activity>

        <activity
            android:name=".Registration">
        <!--android:screenOrientation= "portrait">--></activity>
        <activity
            android:name=".MainActivity">
        <!--android:screenOrientation= "portrait">-->
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
</manifest>
```