Содержание

Введение

На сегодняшний день облачные хранилища снискали большую популярность среди пользователей настольных ПК и мобильных устройств. Такие сервисы предоставляют доступ к данным с любого устройства, имеющего выход в Интернет, позволяют относительно легко организовать совместный доступ к файлам. Имеют повышенную отказоустойчивость, потеря данных за счет аппаратных или программных сбоев практически исключена, т. к. все процедуры по резервному копированию и сохранению целостности данных производятся провайдером «облачного» центра, не вовлекая в этот процесс клиента.

Однако, при работе с данными на некоторых устройствах могут возникнуть сложности из-за неудобного интерфейса или каких-то других программных или аппаратных ограничений. В связи с этим возникла необходимость в разработке приложения, упрощающего работу с данными в облачном хранилище.

1 Анализ предметной области

На данном этапе работы необходимо провести обзор наиболее популярных существующих решений в сфере облачных хранилищ, выделить их основной функционал, обосновать актуальность разработки и сформировать требования к программе в виде технического задания.

1.1 Обзор существующих решений

Среди всех решений наиболее популярными являются Google Диск, Яндекс.Диск, Microsoft OneDrive и Dropbox. Каждый из них имеет веб-интерфейс для работы с хранилищем, настольные клиенты для ОС Windows и macOS, а также предоставляет API для разработчиков сторонних приложений. Ниже представлены особенности рассматриваемых решений.

1.1.1 Google Диск

Google Диск – это файловый хостинг, созданный и поддерживаемый компанией Google. Его функции включают хранение файлов в Интернете, общий доступ к ним и совместное редактирование. В состав Google Диска входят Google Документы, Таблицы и Презентации – набор офисных приложений для совместной работы над текстовыми документами, электронными таблицами, презентациями, чертежами, веб-формами и другими файлами.

Сервис предоставляет 15 гигабайт для бесплатного хранения данных, если выделенного объёма недостаточно, можно приобрести дополнительно от 100 гигабайт до 30 терабайт.

Официального настольного клиента для Linux нет. Вместо него можно использовать сторонние сервисы для доступа к хранилищу по протоколу WebDAV. При этом имеется ограничение на максимальный объем загружаемого файла в 250 мегабайт.

1.1.2 Яндекс.Диск

Яндекс.Диск – облачный сервис, принадлежащий компании Яндекс, позволяющий пользователям хранить свои данные на серверах в облаке и передавать их другим пользователям в Интернете. В отличие от Google Диска не имеет собственных приложений для работы с офисными документами, а использует Microsoft Office Online. Файлы, загруженные через веб-интерфейс, ограничены в размере 10 гигабайтами.

Сервис предлагает бесплатно 10 гигабайт для хранения данных, которые можно расширить до одного терабайта. Также Яндекс.Диск периодически проводит акции, где участвующие пользователи могут бесплатно получить дополнительное пространство.

Существуют официальные клиенты Яндекс.Диска для таких платформ, как Windows, macOS и Linux. Также имеется экспериментальная версия настольно клиента для Windows и macOS, которая позволяет загружать файлы на Диск, просматривать, скачивать и делиться ими. При этом файлы не занимают места на ПК пользователя. Официально поддерживается доступ к Яндекс.Диску через протокол WebDAV.

1.1.3 Microsoft OneDrive

OneDrive – облачное хранилище, управляемое компанией Microsoft.

Сервис предоставляет 5 гигабайт для бесплатного хранения данных, если выделенного объёма недостаточно, можно приобрести дополнительно от 50 гигабайт до одного терабайта. При покупке одного терабайта предоставляется набор офисных приложений Office 365.

Существует поддержка Office Online в OneDrive. Это позволяет пользователям загружать, создавать, редактировать и обмениваться документами Microsoft Office непосредственно в веб-браузере. Пользователи могут создавать, просматривать и редактировать документы Word, Excel, PowerPoint и OneNote прямо в браузере.

OneDrive не имеет официального клиента для Linux. При этом есть недокументированный доступ по протоколу WebDAV.

1.1.4 Dropbox

Dropbox является одним из старейших сервисов в своем роде.

Сервис предлагает бесплатно 2 гигабайта для хранения данных, которые можно увеличить бесплатно до 16 гигабайт, приглашая новых пользователей или же получить несколько гигабайт после выполнения заданий (установка приложения Dropbox на мобильный телефон и т. д.). Также можно купить один терабайт.

Существуют официальные клиенты Dropbox для таких платформ, как Windows, macOS и Linux.

Файлы, загруженные через клиент, не имеют ограничения на размер, но файлы, загруженные через веб-интерфейс, ограничены 20 гигабайтами. Есть также возможность выкладывать файлы и папки для общего доступа через ссылки. Доступна автоматическая синхронизация файлов и папок и хранение версий с возможностью отката. Также имеется возможность работы с офисными документами, используя Microsoft Office Online.

1.2 Актуальность разработки

При рассмотрении функционала каждого из существующих решений было замечено, что для Dropbox не имеется бесплатного сервиса, обеспечивающего доступ к хранилищу без синхронизации файлов на ПК пользователя. В качестве замены возможно использование веб-версии клиента, но работа с ней может быть затруднена на ПК, не обладающих достаточной производительностью. Поэтому возникла необходимость в создании облегченного варианта настольного клиента для Dropbox.

Основными преимуществами данной программы являются:

– Небольшое потребление памяти относительно веб-версии.

– Повышенное быстродействие: программа не синхронизирует файлы на ПК пользователя.
– Отсутствие посредников: программа работает с хранилищем напрямую, используя Dropbox API, и не использует сторонние сервисы.
– Программа не взымает плату за доступ к хранилищу.

1.3 Техническое задание

Наименование программы – "QDDClient".

1.3.1 Краткая характеристика области применения

Программа предназначена к применению на домашних настольных ПК.

1.3.2 Требования к составу выполняемых функций

Программа должны обеспечивать возможность выполнения перечисленных ниже функций:

1) Функции аутентификации пользователя.
2) Функции создания пустой папки.
3) Функции удаления файла (папки).
4) Функции просмотра содержимого папки.
5) Функции загрузки файла в облачное хранилище.
6) Функции скачивания файла на локальный ПК.

1.3.3 Требования к организации входных данных

Размер файлов, предназначенных к загрузке в облачное хранилище, не должен превышать объем свободного пространства в облачном хранилище.

1.3.4 Требования к организации выходных данных

Требования к организации выходных данных не предъявляются.

1.3.5 Требования к обеспечению надежного функционирования программы

Надежное (устойчивое) выполнение программы должно быть обеспечено выполнением пользователем совокупности организационно-технических мероприятий, перечень которых приведен ниже:

1) Организацией бесперебойного питания технических средств.
2) Организацией бесперебойного доступа к сети Интернет.
3) Использованием лицензионного программного обеспечения.

1.3.6 Отказы из-за некорректных действий пользователя

Отказы программы возможны вследствие некорректных действий пользователя при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

1.3.7 Требования к составу и параметрам технических средств

В состав технических средств должен входить IBM-совместимый персональный компьютер, включающий в себя:

1) x86-совместимый процессор.
2) Не менее 50 мегабайт оперативной памяти.
3) Не менее 100 мегабайт свободного дискового пространства.
4) Доступ к облачному хранилищу Dropbox.

1.3.8 Требования к программным средствам, используемым программой

Системные программные средства, используемые программой, должны быть представлены лицензионной версией операционной системы Windows 7 или выше.

1.3.9 Специальные требования

Программа должна обеспечивать взаимодействие с пользователем посредством графического пользовательского интерфейса.

1.3.10 Требования к программной документации

Состав программной документации должен включать в себя:

1) Техническое задание.
2) Руководство пользователя.
3) Техническую документацию.

На основе проведенного анализа предметной области были выявлены преимущества и недостатки существующих решений в области облачных хранилищ. Сформирована проблема – отсутствие для пользователей Dropbox настольного клиента без синхронизации файлов, не требовательного к ресурсам ПК. Исходя из поставленной проблемы было составлено техническое задание.

2 Разработка структуры приложения

На данном этапе работы необходимо в соответствии с требованиями, поставленными в техническом задании, разработать функциональную структуру приложения, в соответствии с ней определить и разработать алгоритмы функционирования; разработать блочную и модульную структуры приложения.

2.1 Разработка функциональной структуры приложения

При разработке функциональной структуры приложения с точки зрения потоков данных были выделены две внешние сущности, взаимодействующие с приложением: пользователь и сервер, принимающий запросы Dropbox API.

Пользователь является источником следующих данных:

- Запрос на изменение содержимого данных.
- Запрос на загрузку (скачивание) файлов.
- Учетные данные пользователя Dropbox
- Код подтверждения учетных данных

Сервер является источником следующих данных:

- Блок содержимого файла для скачивания
- Результат запроса
- Токен пользователя

Обрабатывая информацию от внешних сущностей приложение порождает следующие данные, поступающие на сервер:

- Блок содержимого файла для загрузки
- Запрос на получение токена пользователя
- Запрос к Dropbox API

Также приложение передает данные пользователю в виде результата

операции, запущенной пользователем, и информационных сообщений. Контекстная диаграмма потоков данных представлена в приложении Г.

2.2 Разработка блочной структуры приложения

При декомпозиции контекстной диаграммы основной элемент (приложение) был разбит на следующие функциональные блоки:

- Обозреватель содержимого папки
- Менеджер загрузок
- Журнал событий
- Хранилище токенов
- Блок аутентификация

Обозреватель содержимого папки принимает запросы на изменение содержимого папки, запросы на загрузку (скачивание) файла, результат запроса к Dropbox API, токен пользователя. В ходе обработки данных функциональный блок генерирует запросы к Dropbox API и результаты операций. Обозреватель содержимого папки соответствует функциям создания пустой папки, удаления файла (папки), просмотра содержимого папки. Для обеспечения функционирования данного блока необходимо разработать следующие алгоритмы:

- Алгоритм создания папки
- Алгоритм копирования/вырезки файла/папки
- Алгоритм вставки файла/папки
- Алгоритм удаления файла/папки
- Алгоритм загрузки содержимого папки

Менеджер загрузок принимает блоки содержимого файлов для скачивания и выдает блоки содержимого файлов для загрузки и статусы задач. Данный блок соответствует функциям загрузки и скачивания файлов. Для обеспечения функционирования менеджера загрузок необходимо

разработать алгоритмы загрузки файла в хранилище и скачивания файла на ПК.

Журнал событий обрабатывает входные данные в виде статусов выполнения задач и результатов запросов к Dropbox API и выдает служебные информационные сообщения.

Хранилище токенов отвечает за хранение и выдачу токенов пользователя и приложения. В отличие от токена пользователя, токены приложения не изменяются.

Блок аутентификации получает в качестве входных данных токены приложения, учетные данные пользователя Dropbox и код подтверждения учетных данных; соответствует функции аутентификации. Результатом обработки поступивших данных является токен пользователя. Для функционирования данного блока необходимо разработать алгоритм аутентификации пользователя.

Детализирующая диаграмма потоков данных представлена в приложении Д.

2.3 Разработка модульной структуры приложения

Приложение было разбито на несколько модулей, каждый из которых решает одну или несколько схожих задач.

Обозреватели содержимого папки и дерева папок решают задачи, связанные с отображением и изменением содержимого папок.

Панель инструментов отвечает за запуск некоторых процессов, реализованных в других модулях приложения.

Обозреватели и панель инструментов соответствуют функциональному блоку обозревателя содержимого папки.

Модуль аутентификации пользователя реализует единственную функцию – аутентификация пользователя. Выделена в отдельный модуль, так как предполагается, что ей будут пользоваться редко. Соответствует

функциональному блоку аутентификации.

Модуль загрузки и скачивания файлов управляет процессами передачи файлов между облачным хранилищем и локальным ПК. Загрузка и скачивание файлов объединены в один модуль в виду схожести задач. Соответствует менеджеру загрузок.

Модуль основного окна решает задачи, связанные с инициализацией приложения и настройкой взаимодействия между модулями.

2.3.1 Модуль основного окна

Добавлено примечание ([РАЭо1]): Слишком сухо написано

Модуль основного окна должен решать следующие задачи:

- Отображение информационных сообщений в строке уведомлений
- Отображение пути к текущей папке в заголовке окна
- Загрузка и проверка токена пользователя
- Запуск процесса аутентификации
- Выполнение предварительных действий по скачиванию файлов (выбор места сохранения на ПК пользователя)
- Выполнение подготовительных действий по загрузке файлов (выбор загружаемого файла)
- Отправка запроса на загрузку (скачивание) файла
- Создание, компоновка и начальная настройка элементов пользовательского интерфейса

2.3.2 Обозреватель дерева папок

Обозреватель дерева папок должен решать следующие задачи:

- Отображение иерархии папок в хранилище Dropbox
- Отправка запроса на загрузку содержимого папки
- Отправка информационных сообщений в строку уведомлений

2.3.3 Обозреватель содержимого папки

Обозреватель содержимого папки должен решать следующие задачи:

- Отображение содержимого текущей папки
- Навигация по содержимому облачного хранилища
- Изменение содержимого путем копирования, вставки, перемещения или удаления файлов и папок
- Запуск процедуры скачивания файла
- Создание новых папок
- Отправка информационных сообщений в строку уведомлений

2.3.4 Модуль панели инструментов

Модуль панели инструментов должен решать следующие задачи:

- Отправка запроса на обновление дерева папок и содержимого текущей папки
- Выполнение подготовительных действий по созданию папки (ввод и проверка названия папки)
- Отправка запроса на создание новой папки
- Запуск процедуры загрузки файла

2.3.5 Модуль загрузки и скачивания файлов

Модуль загрузки и скачивания файлов должен решать следующие задачи:

- Запуск и остановка задач
- Отправка сведений о текущем состоянии выполнения задач

2.3.6 Модуль аутентификации пользователя

Модуль аутентификации пользователя должен решать следующие задачи:

– Ввод и проверка кода для подтверждения доступа к хранилищу Dropbox
– Получение токена пользователя, используя Dropbox API

2.4 Разработка алгоритмов функционирования

Алгоритм создания папки:

1) Запросить у пользователя название для новой папки
2) Если название является пустой строкой, состоит исключительно из пробелов или содержит один из недопустимых символов, то выдать пользователю сообщение о недопустимом названии и перейти к п. 1, в противном случае перейти к п. 3
3) Создать папку, используя запрос /files/create_folder
4) Проверить результат запроса: если он прошел успешно, обновить содержимое текущей папки, в противном случае сообщить пользователю об ошибке

Схема алгоритма создания папки представлена на рисунке 1.

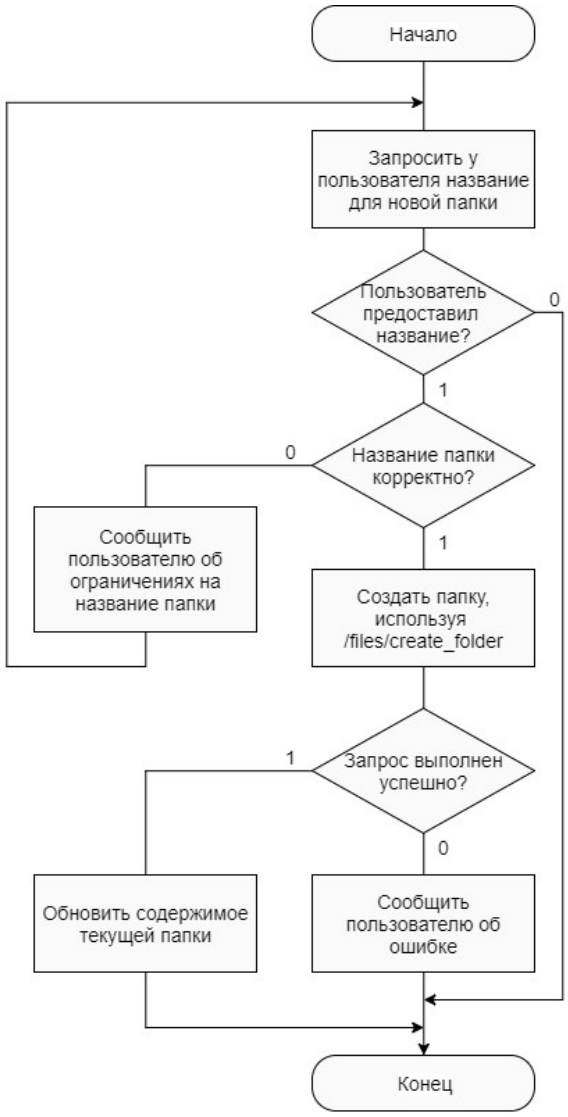


Рисунок 1 – Схема алгоритма создания папки

Алгоритм копирования/вырезки файла/папки:

1) Определить какое действие выполнено над файлом/папкой: «Вырезать» или «Копировать»

2) Занести файл/папку в буфер обмена с меткой соответствующего действия

В силу тривиальности, алгоритм описывается только в текстовой форме.

Алгоритм вставки файла/папки:

1) На основе информации из буфера обмена определить какой действие необходимо выполнить над файлом/папкой: «Вырезать» или «Копировать»
2) Если нужно скопировать, то выполнить запрос /files/copy_v2, перейти к п. 4, иначе перейти к п. 3
3) Переместить файл/папку, используя запрос /files/move_v2, перейти к п. 5
4) Проверить результат запроса: если он прошел успешно, обновить содержимое текущей папки, в противном случае сообщить пользователю об ошибке
5) После успешного выполнения перемещения удалить информацию из буфера обмена

Схема алгоритма вставки файла/папки представлена на рисунке 2.

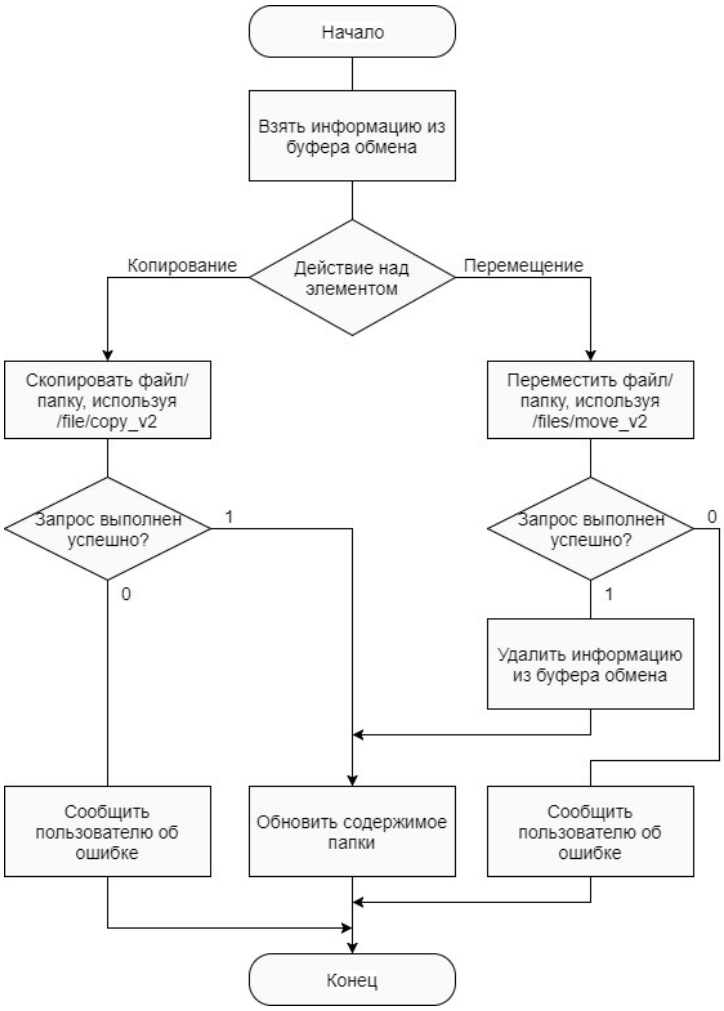


Рисунок 2 – Схема алгоритма вставки файла/папки

Алгоритм удаления файла/папки:

1) Запросить у пользователя подтверждение на удаление
2) Если пользователь отказался, завершить работу алгоритма
3) Удалить файл/папку, используя запрос /files/delete_v2
4) Проверить результат запроса: если он прошел успешно, обновить

содержимое текущей папки, в противном случае сообщить пользователю об ошибке

В силу тривиальности, алгоритм описывается только в текстовой форме.

Алгоритм загрузки содержимого папки:

1) Загрузить по блокам список содержимого папки, используя запросы /files/list_folder (для первого блока) и /files/list_folder_continue (для следующих блоков)
2) Если при загрузке блоков произошла ошибка, то сообщить о ней пользователю и завершить работу алгоритма, иначе перейти к следующему пункту
3) Очистить область просмотра содержимого
4) Отобразить список файлов и папок в области просмотра папки

Схема алгоритма изображена на рисунке 3.

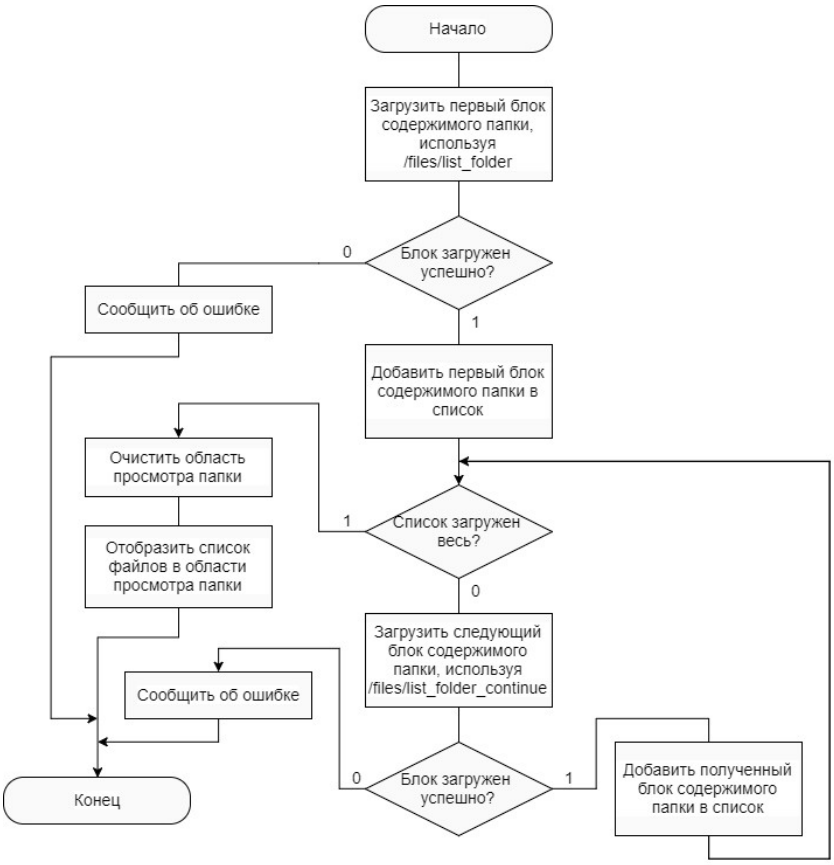


Рисунок 3 – Схема алгоритма загрузки содержимого папки

Алгоритм загрузки файла в хранилище:

1) Запросить у пользователя путь к загружаемому файлу
2) Если пользователь отказался ввести путь, то завершить работу алгоритма, иначе перейти к следующему пункту
3) Получить дескриптор временного файла (ID сесиии) в Dropbox, используя /files/upload_session_start.
4) Получить дескриптор файла на локальном ПК

5) Если при получении дескрипторов произошла ошибка, то сообщить о ней пользователю, завершить работу алгоритма, иначе перейти к следующему пункту
6) Выполнить загрузку файла по блокам, используя /files/ upload_session_append_v2
7) Если при загрузке произошла ошибка, то сообщить о ней пользователю, закрыть дескрипторы файлов и завершить работу алгоритма, иначе перейти к следующему пункту
8) Присвоить временному файлу в Dropbox имя загружаемого файла, используя /files/upload_session_finish
9) Закрыть дескрипторы файлов

Схема алгоритма изображена на рисунке 4.

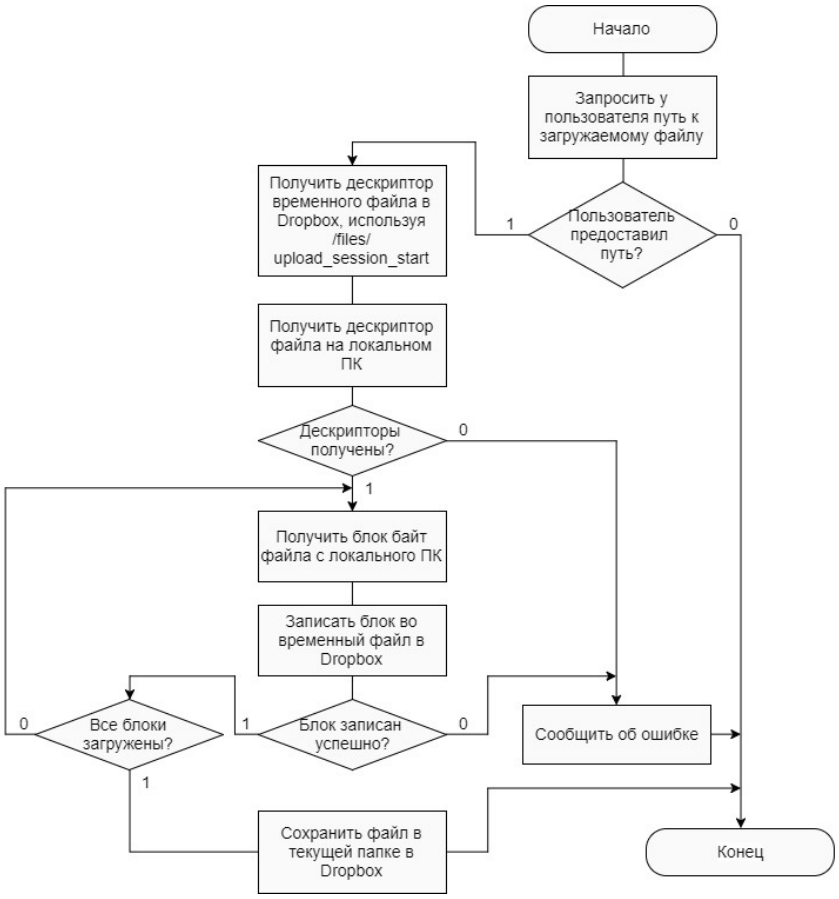


Рисунок 4 – Схема алгоритма загрузки файла в хранилище

Алгоритм скачивания файла на ПК:

1) Запросить у пользователя путь для сохранения файла
2) Если пользователь отказался ввести путь, то завершить работу алгоритма, иначе перейти к следующему пункту
3) Получить дескриптор файла (ответ на HTTP-запрос) из Dropbox, используя /file/download.
4) Получить дескриптор файла на локальном ПК

5) Если при получении дескрипторов произошла ошибка, то сообщить о ней пользователю, завершить работу алгоритма, иначе перейти к следующему пункту
6) Выполнить скачивание файла по блокам
7) Если при скачивании произошла ошибка, то сообщить о ней пользователю
8) Закрыть дескрипторы файлов

Схема алгоритма изображена на рисунке 5.

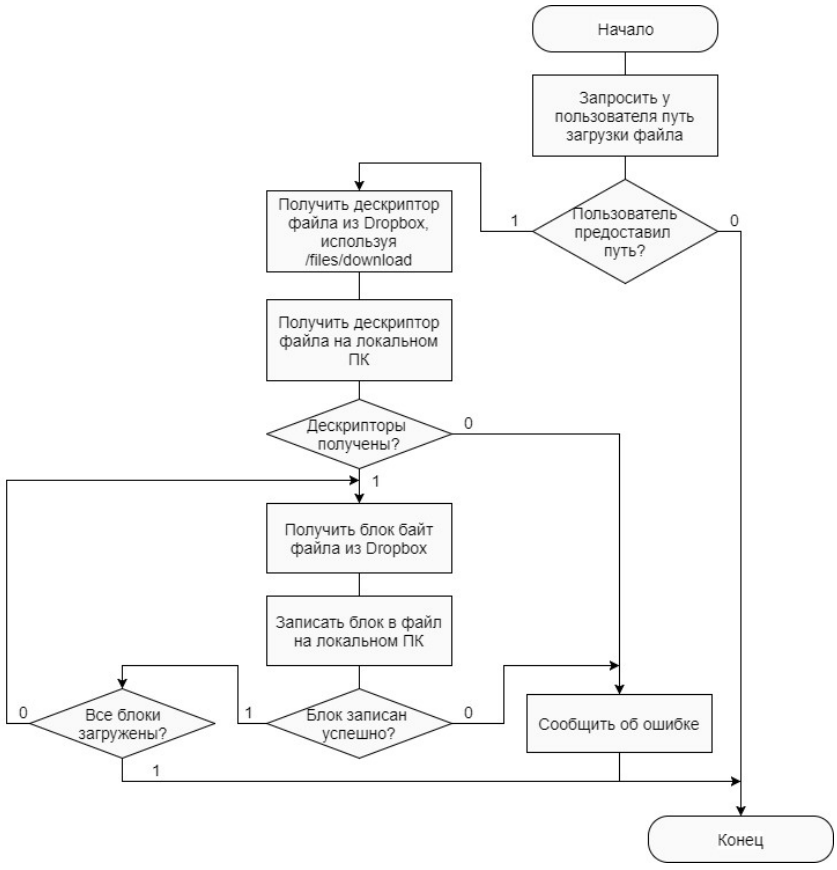


Рисунок 5 – Схема алгоритма скачивания файла на ПК

Алгоритм аутентификации пользователя:

1) Создать ссылку, по который пользователь должен перейти для ввода учетных данных от аккаунта Dropbox.
2) Получить от пользователя код подтверждения
3) Если пользователь ввел код, то перейти к следующему пункту, иначе завершить работу алгоритма
4) Проверить код. Если проверки прошла удачно, то получить токен пользователя и завершить работу алгоритма, иначе сообщить пользователю об ошибке и перейти к п. 2

Схема алгоритма аутентификации пользователя представлена на рисунке 6

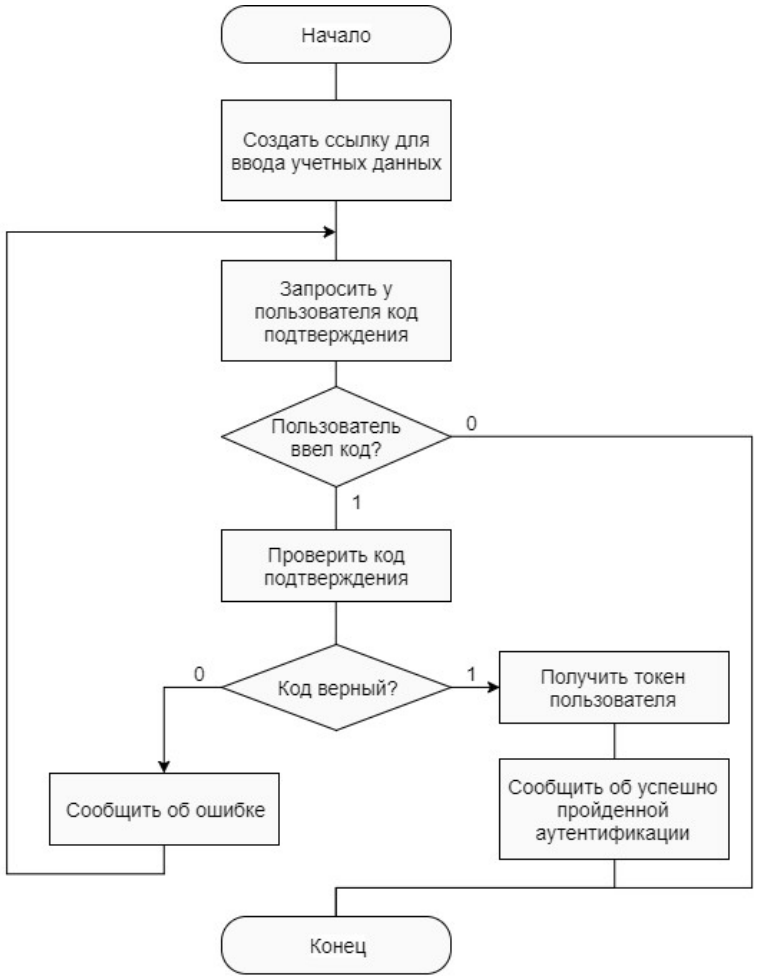


Рисунок 6 – Схема алгоритма аутентификации пользователя

Исходя из поставленных в техническом задании требований были разработаны функциональная, блочная и модульная структуры приложения (приведена в приложении В), алгоритмы функционирования, а также диаграмма переходов состояний, которая приведена в приложении Б.

3 Программная реализация

На данном этапе работы необходимо выбрать инструменты для кодирования приложения (язык программирования, библиотеки), выполнить реализацию разработанных ранее модулей и алгоритмов, а также разработать графический интерфейс пользователя.

3.1 Выбор языка программирования

Исходные коды программы должны быть реализованы на языке Python, имеющем следующие преимущества:

- Поддержка большинства операционных систем.
- Высокая популярность языка.
- Наличие фреймворков для построения GUI.
- Наличие встроенных средств модульного тестирования.
- Богатая стандартная библиотека.

В качестве инструмента для работы с многопоточностью и обеспечением взаимодействия модулей приложения друг с другом, а также для построения графического пользовательского интерфейса задействован фреймворк PyQt.

3.2 Утилиты

Из-за особенностей работы фреймворка PyQt, Dropbox API, а также для упрощения кодирования приложения было разработано несколько вспомогательных классов и функций, не входящих в состав модулей приложения.

3.2.1 Защита пользовательских данных

В качестве базовой защиты от несанкционированного доступа к

токенам пользователя и приложения используется кодирование данных с использованием алгоритма base64. Токены хранятся в закодированных файлах в формате JSON. Для кодирования и декодирования информации были реализованы функции load_json и dump_json.

Функция load_json получает в качестве параметра путь к закодированному файлу, информацию из которого необходимо считать. С помощью встроенной функции open файл открывается для чтения, считывается его содержимое, которое декодируется функцией decodebytes модуля base64 стандартной библиотеки Python. Полученный в ходе преобразования текст переводится в структуры данных языка Python с помощью функции loads модуля json. Полученный объект возвращается как результат функции.

Функция dump_json получает в качестве параметров путь к файлу, в который необходимо записать закодированную информацию, и объект с данными. Функцией dumps модуля json информация в объекте переводится в формат JSON, а полученный текст кодируется с помощью функции encodebytes модуля base64. Встроенной функцией open файл открывается для записи. В него записывается закодированный текст.

3.2.2 Представление путей к папкам и файлам

Для удобства при работе с путями в хранилище Dropbox был реализован класс Entry, предоставляющий следующий функционал:

- Хранение полного пути к папке (файлу).
- Получение родительского каталога папки (файла).
- Хранение названия папки (файла).

Одной из особенностей Dropbox API является представление путей в хранилище. Среди них можно отметить:

- Путь к корневой папке является пустой строкой.

– Все остальные пути должны начинаться с символа ‘/’.

– Названия папок и файлов являются регистронезависимыми.

При перемещении по иерархии папок возможна ситуация, при которой представление пути может оказаться недопустимым. Для разрешения таких ситуаций была реализована функция normalize_path, которая приводит данный ей путь к допустимому виду.

3.2.3 Логирование

Для упрощения отладки приложения используется логирование с помощью модуля стандартной библиотеки Python logging. По умолчанию ведется логирование только исключительных ситуаций. Переменной окружения пользователя QDDC_ENV управляется уровень ведения логов: если переменная QDDC_ENV равна dev, то включается более подробное логирование.

3.2.4 Выполнение длительных операций в фоновом режиме

Для сохранения отзывчивости пользовательского интерфейса при выполнении длительных операций был реализован класс AsyncTask, который предоставляет следующий функционал:

– Выполнение определенной задачи в отдельном потоке.

– Обработка результата задачи в отдельной функции.

– Обработка исключительных ситуаций в отдельной функции.

Задача должна быть реализована в виде отдельной функции без аргументов, которая передается как параметр в конструктор класса. Также в виде отдельных функций с одним аргументом должны быть реализованы обработчики результата вычислений и исключительных ситуаций и переданы как параметры в конструктор класса. В случае если они не нужны, то вместо них будет использован обработчик по умолчанию – пустая функция с одним

аргументом.

В связи с тем, что сигналы в PyQt распространяются на всех подписчиков, необходимо определять для какого именно подписчика был создан сигнал. Для этого был реализован вспомогательных класс _Signals, в котором имеется один сигнал result. На данный сигнал подписан единственный обработчик _callback, принимающий в качестве аргументов функцию и аргумент для нее и вызывающий ее с этим аргументом. Вызов _callback происходит не мгновенно, а помещается в очередь сообщений PyQt.

Выполнение задачи происходит в защищенном блоке в методе run, который выполняется в глобальном пуле потоков, предоставляемым QThreadPool. При успешном завершении задачи создается сигнал result, которому в качестве параметров передается обработчик результата и сам результат. В случае возникновения ошибок сигналу передается обработчик ошибок и объект исключения.

3.2.5 Буфер обмена

Для хранения информации при копировании (перемещении) папок или файлов был реализован класс Clipboard, предоставляющий следующий функционал буфера обмена:

- Хранение папки (файла), скопированной (вырезанной).
- Хранение действия, выполняемого над элементом.

Доступ к содержимому буфера обмена предоставляется через поле content, хранящее кортеж из переменной типа Entry (файл или папка) и код действия в виде целого числа.

3.3 Реализация модулей приложения

Каждый модуль приложения должен быть реализован в виде отдельного класса. Для организации взаимодействия между модулями

используется механизм сигналов и слотов PyQt.

3.3.1 Модуль основного окна

Среди всех функций данного модуля можно выделить две основные: загрузка токена пользователя и создание графического интерфейса.

Сам запуск приложения выполняется методом start, в котором через сигнал app_started вызывается метод _load_access_token. В этом методе токен пользователя загружается из файла user_credentials с помощью функции load_json. Если файл был поврежден и токен загрузить невозможно, то вызывается метод _open_login_dialog, который запускает процедуру аутентификации. После того как токен был успешно загружен, выполняется его проверка в методе _accept_access_token. В случае если токен оказался недействительным, запускается процедура аутентификации.

В методе _accept_access_token токен проверяется с помощью запроса Dropbox API /users/get_current_account. После успешной проверки токен сохраняется в файле user_credentials с помощью функции dump_json и вызывается метод _build_ui, в котором выполняется построение интерфейса пользователя.

В методе _open_login_dialog создается диалоговое окно DropboxLoginDialog и вызывается метод exec, результатом которого будет код завершения процедуры, успешное или неуспешное. При успешном исходе в поле access_token диалогового окна хранится токен пользователя, который передается в качестве параметра методу _accept_access_token. В случае неудачи приложение завершает свою работу.

Токен сохраняется в специальной переменной _dropbox, которая помимо этого, предоставляет доступ ко всему Dropbox API. Ссылка на эту переменную передается в качестве параметра конструктора остальным модулям.

Список сигналов, на которые реагирует модуль:

- upload_file – запускается процедура загрузки файла (вызывается метод _upload_file).
- message_sent – отображается информационное сообщение в строке уведомлений (реализовано в методе _show_message).
- current_path_changed – обновляется путь к папке в заголовке окна (вызывается метод _update_window_title).
- download_file – запускается процедура скачивания файла (вызывается метод _download_file).

В методе _upload_file пользователь выбирает с помощью диалогового окна файл, который необходимо загрузить в облачное хранилище. После этого создается экземпляр класса DropboxLoadingTask, с информацией о том, какой файл загрузить, куда загрузить, а также код задачи (UPLOAD). Запуск задачи выполняется в методе _run_new_task.

В методе _download_file пользователь выбирает с помощью диалогового окна папку, в которую нужно сохранить файл. После этого также создается экземпляр класса DropboxLoadingTask, но уже с кодом DOWNLOAD. Задача запускается в методе _run_new_task.

Для того, чтобы при закрытии приложения выполняющиеся в фоновом режиме задачи были корректно завершены, в методе _run_new_task выполняется привязка каждой задачи к сигналу _app_closed. При его создании у каждой задачи вызывается метод close.

Метод _update_window_title перед тем, как изменить заголовок окна, проверяет путь к папке. Если путь является пустой строкой (корневая папки хранилища), то он заменяет ее на строку ‘/’. После этого заголовок окна меняется на новый.

Данный модуль реализован в виде класса DropboxMainWindow.

3.3.2 Обозреватель дерева папок

Одной из особенностей Dropbox API является то, что для получения списка всех файлов и папок хранилища необходимо выполнить несколько запросов. Однако API не предоставляет возможности выдачи списка, упорядоченного по полному пути элементов. Поэтому было принято решение выполнять загрузку содержимого папок только по требованию, а именно только для тех папок, которые пользователь выбрал. В данном случае под содержимым папки понимается список папок, непосредственно находящихся в данной папке.

Для отображения дерева папок был задействован класс QTreeWidget, в качестве узлов дерева используется класс _TreeItem, производный от класса QTreeWidgetItem. Класс _TreeItem содержит два дополнительных поля: entry, хранящий информацию о папке, представляемой данным узлом; loaded, булевый флаг, показывающий нужно ли создавать дочерние узлы.

При выборе пользователем какой-либо папки создается сигнал item_clicked, уведомляющий о том, что необходимо загрузить содержимое данной папки. Одновременно с созданием сигнала запускается метод _load_item_content, отвечающий за загрузку и отображение содержимого папки в дереве. В качестве параметров метод получает переменную типа _TreeItem, представляющую папку, которую выбрал пользователь. В данном методе выполняются следующие действия:

1) Проверяется поле loaded, если оно истинно, то метод завершает работу.
2) Выполняется загрузка содержимого папки с использованием AsyncTask. Результатом работы является список переменных типа Entry, в которых хранится информация о папках.
3) В обработчике результата выполняется создание узлов дерева _TreeItem с соответствующими переменными Entry, добавление их в качестве дочерних к узлу выбранной пользователем папки и

запись в ее поле loaded значения True.

Также данный модуль реагирует на сигнал update_content от модуля панели инструментов, вызывая при этом метод reset. В данном методе загружаются папки корневой директории с использованием AsyncTask аналогично _load_item_content и заново строится дерево папок. При этом в дерево добавляется только узел корневой папки и папок, непосредственно содержащихся в ней.

Привязка к сигналам панели инструментов (update_content) выполняется в методе attach_toolbar.

Данный модуль реализован в виде класса DropboxTreeView.

3.3.3 Обозреватель содержимого папок

Данный модуль может создавать следующие сигналы:

- download_file, сообщающий о том, что необходимо запустить процесс скачивания файла.
- message_sent, отправляющий информационное сообщение в строку уведомлений.
- current_path_changed, сообщающий о том, что путь к текущей папке изменился.

Для отображения списка содержимого был задействован класс QListWidget, в качестве элементов списка используется класс _ListItem, производный от класса QListWidgetItem. Класс _ListItem содержит дополнительное поле entry, хранящее информацию о папке (файле), представляемой данным элементом.

Для изменение содержимого папки путем копирования, вставки, перемещения или удаления файлов и папок, а также для выбора файла для скачивания используется контекстное меню, реализованное в виде класса _ContextMenu. Меню вызывается путем щелчка правой кнопкой мыши по

области просмотра. Данное меню предоставляет следующие пять действий:

1) «Скачать» - Запускает процедуру скачивания файла (создает сигнал download_file). Неактивно, если пользователь вызвал контекстное меню, либо щелкнув по пустой области, либо щелкнув по папке.
2) «Вырезать» - Помещает выбранную папку (файл) в буфер обмена с кодом действия CUT. Неактивно, если пользователь вызвал контекстное меню, щелкнув по пустой области.
3) «Копировать» - Помещает выбранную папку (файл) в буфер обмена с кодом действия COPY. Неактивно, если пользователь вызвал контекстное меню, щелкнув по пустой области.
4) «Вставить» - Выполняет копирование или перемещение элемента из буфера обмена (вызывает метод _paste_entry). Место куда копируется (перемещается) файл зависит от того куда пользователь щелкнул, вызывая контекстное меню. Если была выбрана папка, то место назначения – выбранная папка, если был выбран файл или щелчок был сделан по пустой области, то место назначения – текущая папка. Действие неактивно, если буфер обмена пуст.
5) «Удалить» - Выполняет удаление выбранной папку или файла (вызывает метод _delete_entry). Неактивно, если пользователь вызвал контекстное меню, щелкнув по пустой области.

В методе _provide_context_menu выполняется создание и отображение контекстного меню, получение выбранного пользователем действия. Также происходит выполнение следующих действий: «Копировать», «Вырезать», «Скачать».

Операция вставки выполняется в методе _paste_entry, где извлекается из буфера обмена содержимое и определяется место назначения. В зависимости от действия (CUT или COPY) вызывается метод _move_entry

или _copy_entry. Методы _move_entry и _copy_entry идентичны за исключением двух вещей:

- В методе _move_entry используется запрос Dropbox API /files/move_v2, а в _copy_entry – /files/copy_v2.
- После успешного перемещения буфер обмена очищается.

Все запросы к API выполняется в отдельном потоке, используя AsyncTask. При возникновении ошибок пользователь получит сообщение в строке уведомлений. В случае если место назначения – текущая папка, то вызывается метод update_content. Если в месте назначения уже имеется элемент с таким же названием, то к нему автоматически будет приписан суффикс ‘(N)’, где N – количество совпадений.

Процедура удаления папки (файла) выполняется в методе _delete_entry. Пользователь подтверждает или отменяет выполнение операции с помощью диалогового окна. В случае положительного ответа в фоновом режиме создается запрос к Dropbox API /files/delete_v2. В случае успешного выполнения запроса элемент, соответствующий удаленной папке (файлу) удаляется из списка содержимого. В случае неудачи пользователь получит сообщение в строке уведомлений.

Для того чтобы открыть какую-либо папку, пользователь должен сделать двойной щелчок по элементу списка, представляющего нужную папку. При этом создается сигнал itemDoubleClicked класса QListWidgetItem, на который подписан метод _open_folder. В качестве параметра передается описание папки (Entry) элемента списка. Загрузка содержимого папки выполняется в фоновом режиме, используя запрос /files/list_folder. При успешном завершении запроса текущий список содержимого очищается, добавляются новые элементы на основе полученного списка переменных Entry. Также создается сигнал current_path_changed с параметром, содержащим путь к текущей папке. Если текущая папка не является

корневой, то добавляется элемент, представляющий родительскую папку. Информация о том, какая папка сейчас открыта, хранится в поле current_entry. В случае неудачи пользователь получит сообщение в строке уведомлений.

Также метод _open_folder подписан на сигнал item_clicked модуля дерева папок. В качестве параметра передается описание папки (Entry), которую нужно открыть.

Привязка к сигналам дерева папок (item_clicked) выполняется в методе attach_tree_view.

Предусмотрена возможность реагирования на сигнал update_content панели инструментов, при этом обновляется содержимое текущей папки методом update_content, который в свою очередь вызывает _open_folder с параметром _current_folder.

Также модуль реагирует на сигнал create_folder, вызывая при этом метод _create_folder с параметром, содержащим имя новой папки. Папка создается с помощью запроса /files/create_folder_v2. Ситуация с конфликтом имен разрешается точно так же, как и при вставке папки (файла). При успешном выполнении запроса обновляется содержимое текущей папки. В случае ошибки пользователь получит сообщение в строке уведомлений.

Привязка к сигналам панели инструментов (update_content и create_folder) выполняется в методе attach_toolbar.

Также предусмотрена возможность загрузки содержимого папки, выбранной в обозревателе дерева папок, реагируя на сигнал item_clicked методом _open_folder.

Данный модуль реализован в виде класса DropboxFolderView.

3.3.4 Модуль панели инструментов

Модуль панели инструментов предоставляет три кнопки, выполняющие следующие действия:

1) «Обновить» - Запускает процедуру обновления содержимого текущей папки и/или дерева папок (создает сигнал update_content)
2) «Новая папка» - Запускает процедуру ввода имени для новой папки (вызывает метод _open_create_folder_dialog)
3) «Загрузить» - Запускает процедуру загрузки файла в облачное хранилище (создает сигнал upload_file)

Для ввода имени новой папки пользователю предоставляется диалоговое окно ввода. Dropbox API накладывает следующие ограничения на имена папок:

- Имя не должно быть пустой строкой или состоять только из символов пробела
- Имя не должно содержать угловые скобки, а также символы: ‘\’, ‘/’, ‘:’, ‘?’, ‘*’, ‘"’, ‘|’

Если хотя бы одно из этих ограничений не соблюдено, то пользователь получит сообщение, в котором будет написано какое правило было нарушено, а также предложение заново ввести имя. Если введенное имя удовлетворяет ограничениям, то создается сигнал create_folder с параметром, содержащим имя новой папки.

Данный модуль реализован в виде класса DropboxToolbar.

3.3.5 Модуль загрузки и скачивания файлов

Модуль разработан таким образом, что информация о каждой задаче отображается в отдельном информационном окне DropboxLoadingTask. В качестве параметров конструктора передается тип задачи (скачивание или загрузка), путь к скачиваемому или загружаемому файлу, путь к месту назначения.

Метод run обеспечивает запуск задачи. В зависимости от ее типа вызывается метод _download (скачивание) или _upload (загрузка).

В методе _download в фоновом режиме (используется AsyncTask) выполняется запрос Dropbox API /files/download. При успешном выполнении возвращается ответ на HTTP-запрос, тело которого – файл, который нужно скачать. После открывается дескриптор файла на локальном ПК с помощью функции open в двоичном режиме. Скачивание выполняется блоками по 4096 байт. Перед записью очередного блока проверяется флаг _stop, сигнализирующий о том, что процесс необходимо прервать. Если этот флаг равен True, то выполнение функции скачивания немедленно прерывается. В противном случае блок байт записывается в файл на ПК, вычисляется процентное соотношение скачанного количества байт к размеру всего файла в байтах – процент выполнения задачи. Для обновления индикатора вызывается метод _on_progress_updated, которому в качестве параметра передается процентное соотношение. При успешном завершении задачи вызывается метод _on_task_completed, при ошибке выдается информационное сообщение и также вызывается _on_task_completed.

В методе _upload в фоновом режиме выполняется запрос Dropbox API /files/upload_session_start. При успешном выполнении запроса возвращается идентификатор временного файла в облачном хранилище. После открывается дескриптор файла на локальном ПК с помощью функции open в двоичном режиме. Загрузка выполняется блоками по 1 мегабайту с помощью запроса /files/upload_session_append. Перед отправкой очередного блока проверяется флаг _stop, сигнализирующий о том, что процесс необходимо прервать. Если этот флаг равен True, то выполнение функции скачивания немедленно прерывается. В противном случае блок байт записывается в файл в облачном хранилище, вычисляется процентное соотношение загруженного количества байт к размеру всего файла в байтах – процент выполнения задачи. Для обновления индикатора вызывается метод _on_progress_updated, которому в качестве параметра передается процентное соотношение. При успешном завершении задачи вызывается метод _on_task_completed, при ошибке

выдается информационное сообщение и также вызывается _on_task_completed.

Методы _on_progress_updated и _on_task_completed вызываются точно так же, как и в AsyncTask вызываются обработчики результата и ошибок. Для этого был разработан класс _Signals, аналогичный классу _Signals в AsyncTask.

В методе _on_task_completed булевой переменной _finished присваивается значение True и вызывается метод close, закрывающий информационное окно.

Пользователь может прервать выполнение задачи либо закрыв окно, либо нажав на кнопку «Стоп». В обоих случаях будет вызван метод closeEvent, в котором переменной _stop присваивается значение True.

Отображаемые сведения о задаче:

- Выполняемая операция: скачивание или загрузка
- Название файла, который скачивается (загружается)
- Индикатор выполнения (в процентах)

3.3.6 Модуль аутентификации пользователя

Модуль реализован в виде специального диалогового окна, реализованного в виде класса DropboxLoginDialog. Для работы модуля необходимы специальные токены приложения (APP_KEY и APP_SECRET), которые передаются в качестве параметров конструктора. Данные токены используются для создания с помощью Dropbox API ссылки, по которой пользователь должен перейти в веб-браузере. На загруженной странице необходимо ввести, если нужно, учетные данные от аккаунта Dropbox и разрешить доступ приложения к облачному хранилищу. После этого будет выдан код подтверждения, который необходимо ввести в соответствующее поле в диалоговом окне и подтвердить ввод.

Обработка введенного кода выполняется в методе _on_submit, где в фоновом режиме проверяется код и в случае успешной проверки выдается токен пользователя. Полученный токен записывается в поле access_token, после этого создается сигнал accepted. Если токен не удалось проверить, то пользователю выведется сообщение с описанием ошибки. Пользователь может отказаться от аутентификации, закрыв диалоговое окно. В этом случае создается сигнал rejected.

3.4 Разработка графического интерфейса пользователя

При анализе реализаций графических интерфейсов пользователя в существующих решениях были выявлены следующие общие черты:

- Основное пространство страницы занимает содержимое папки, представленное в виде списка или сетки.
- Слева имеется небольшая область переходов. У Google Диска рядом с областью перехода расположено дерево папок.
- Сверху отображается полный (если возможно) путь к папке, а также кнопки выполняющие различные функции: копирование, перемещение, создание папки, загрузка файлов.

3.4.1 Прототипирование интерфейса

Исходя из проведенного анализа интерфейсов было принято решение разработать три экрана:

- Экран аутентификации. Здесь пользователь должен предоставить доступ к своему облачному хранилищу.
- Основной экран. На данном экране собран весь функционал приложения за исключением процессов скачивания и загрузки файлов, и аутентификации.
- Экран процесса выполнения задачи. Небольшое информационное

окно на каждую задачу скачивания или загрузки файлов. Помимо информации о процессе выполнения, также есть возможность прервать задачу.

Черновой прототип интерфейса представлен на рисунке 7.

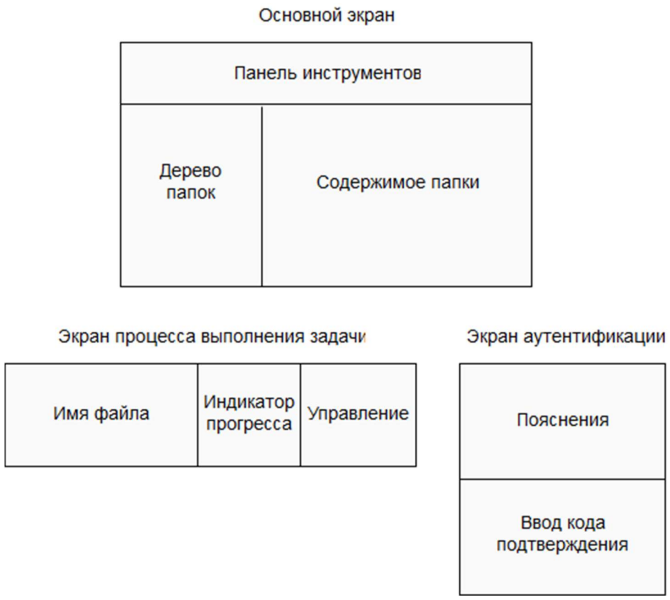


Рисунок 7 – Черновой прототип интерфейса

На экране аутентификации в области «Пояснения» будет отображен текст с ссылкой и указаниями по дальнейшим действиям для прохождения аутентификации. В области «Ввод кода подтверждения» будет расположено поле для ввода кода и кнопка для его проверки.

Область «Панель инструментов» основного экрана включает в себя следующие кнопки:

– «Обновить» - для обновления дерева папок и содержимого текущей папки.
– «Новая папка» - для запуска процесса создания новой папки.

– «Загрузить» - для запуска процесса загрузки файла в облачное хранилище.

В «Дереве папок» содержится древовидная структура из папок облачного хранилища.

В области «Управление» окна процесса выполнения задачи расположена кнопка «Стоп», прерывающая работу задачи.

Финальный прототип интерфейса представлен на рисунке 8.

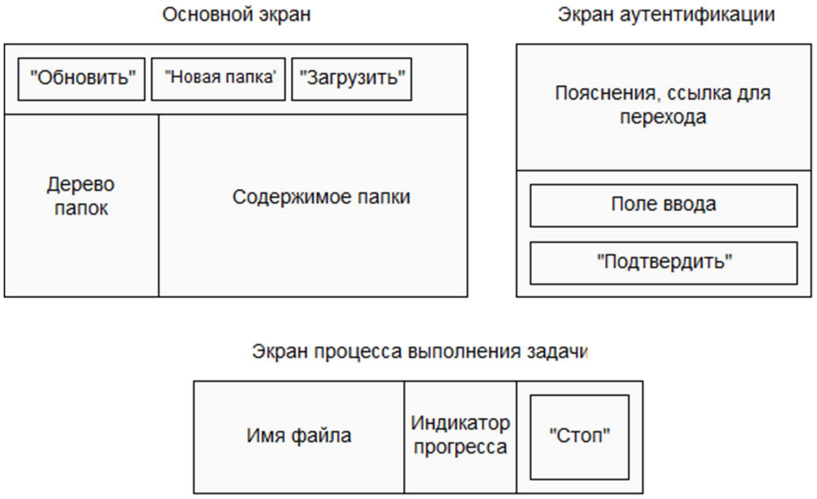


Рисунок 8 – Финальный прототип интерфейса

3.4.2 Оформление экранов

Экранные формы представлены на рисунках 9-11.

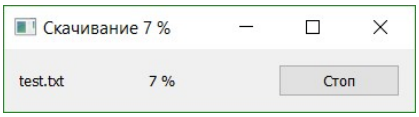


Рисунок 9 – Экран процесса выполнения задачи

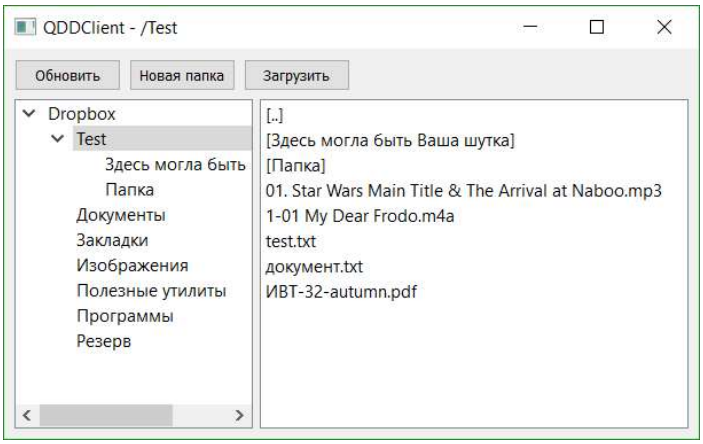


Рисунок 10 – Основной экран приложения

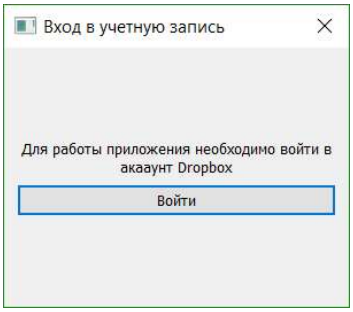


Рисунок 11 – Экран аутентификации

На основе разработанной структуры приложения, алгоритмов функционирования была выполнена программная реализация приложения. Диаграмма разработанных классов представлена в приложении А. Особо следует отметить разработанный класс AsyncTask, с помощью которого выполнялись файловые операции и практически все обращения к Dropbox API.

Заключение

В ходе выполнения курсового проекта было разработано программное обеспечение «QDDClient». На этапе анализа был выполнен обзор существующих облачных хранилищ, на его основе было составлено техническое задание к разрабатываемому приложению. В ходе проектирования была разработана модульная структура приложения, схемы алгоритмов функционирования приложения и диаграмма переходов состояний. При проектировании решались проблемы, связанные с особенностями работы сервиса Dropbox, была выполнена программная реализация модулей приложения, а также был разработан графический интерфейс пользователя. Результат этапа проектирования – диаграмма классов, черновой и финальный прототип интерфейса, оформление всех экранов.

В качестве направления дальнейшего развития приложения можно выбрать расширение функционала приложения, в том числе: отображение сведений о файлах (размер, тип, дата последнего изменения), создание ссылок доступа к файлам и папкам, добавление многопользовательского режима, реализация функции «Drag and Drop», возможность приостановки задач скачивания и загрузки файлов.

Перечень сокращений

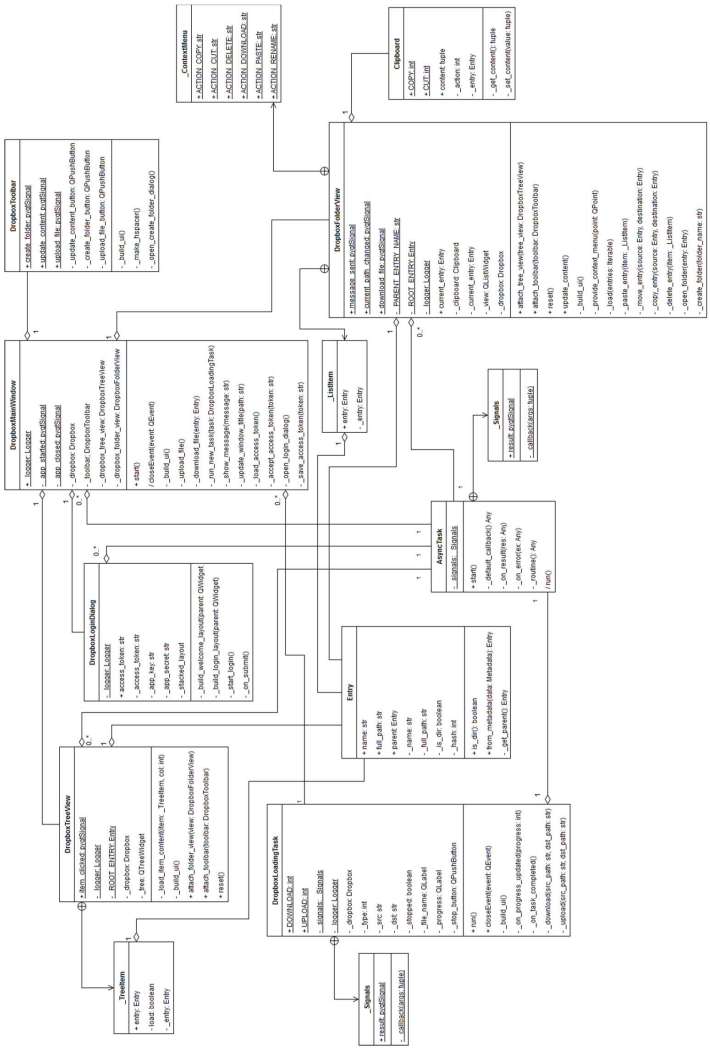
ПК – персональный компьютер

API – application programming interface, интерфейс прикладного программирования

Приложение А

(обязательное)

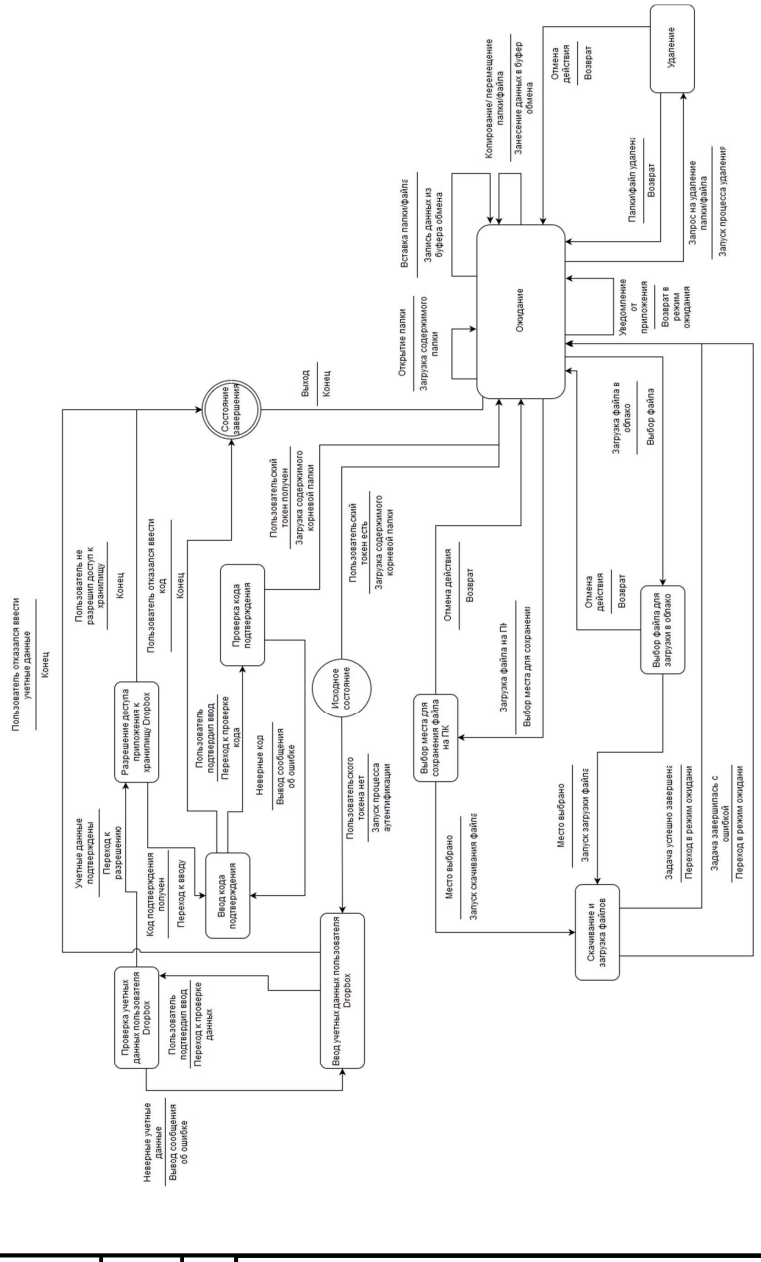
Диаграмма классов



Приложение Б

(обязательное)

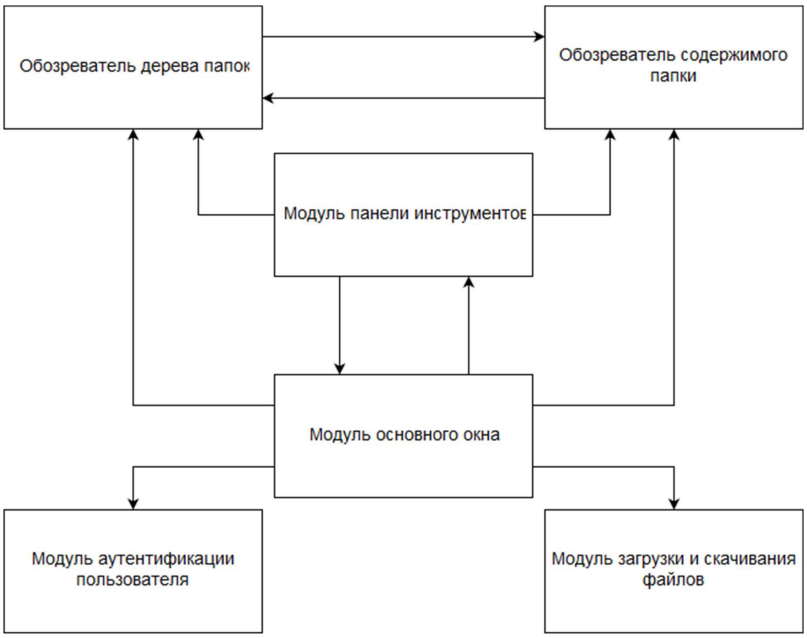
Диаграмма переходов состояний



Приложение В

(обязательное)

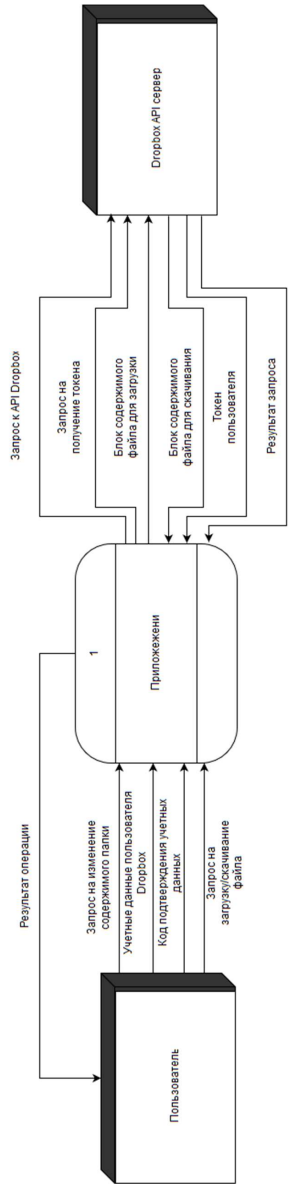
Модульная структура приложения



Приложение Г

(обязательное)

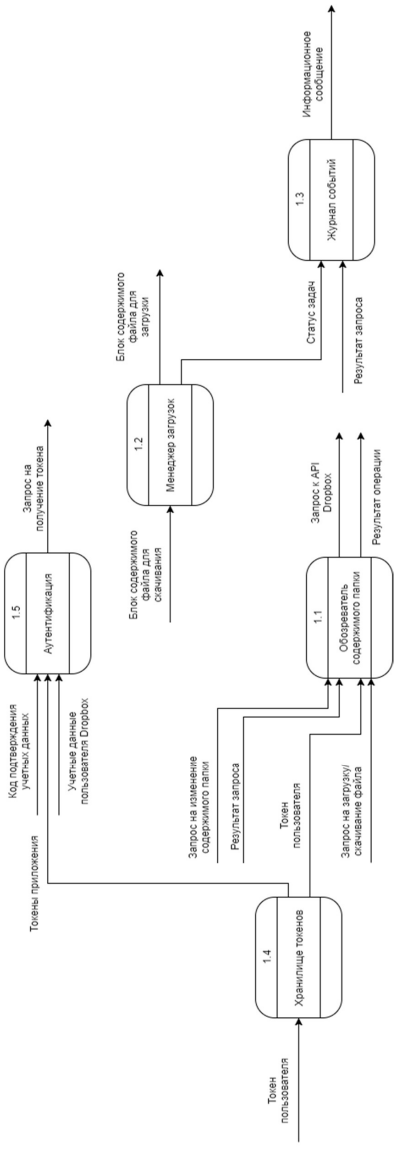
Контекстная диаграмма потоков данных



Приложение Д

(обязательное)

Детализирующая диаграмма потоков данных



Приложение Е

(справочное)

Библиографический список

1 PyQt5 Reference Guide [Электронный ресурс]. – Режим доступа: http://pyqt.sourceforge.net/Docs/PyQt5. – PyQt 5.9.2 Reference Guide. – (Дата обращения: 01.11.2017).

2 Qt Documentation [Электронный ресурс]. – Режим доступа: http://doc.qt.io/qt-5/reference-overview.html. – Qt Reference Pages | Qt 5.10. – (Дата обращения: 01.11.2017).

3 Dropbox for Python [Электронный ресурс]. – Режим доступа: http://dropbox-sdk-python.readthedocs.io/en/stable/. – Dropbox for Python Documentation – Dropbox for Python 8.4.1 documentation. – (Дата обращения: 01.11.2017).

4 Source code for requests.models [Электронный ресурс]. – Режим доступа: http://docs.python-requests.org/en/master/_modules/requests/models/ – requests.models – Requests 2.18.4 documentation. – (Дата обращения: 01.11.2017).

5 Python documentation [Электронный ресурс]. – Режим доступа: https://docs.python.org/3.5/ – 3.5.4 Documentation. – (Дата обращения: 01.11.2017).