МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Вятский государственный университет»
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Лабораторный практикум
по дисциплине «Технологии программирования»

Выполнил студент группы ИВТ-32 _____ /Рзаев А. Э./
Проверил доцент кафедры ЭВМ_____ /Долженкова М.Л./

Киров 2017

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 Техническое задание

В первом разделе сформулировано техническое задание (ТЗ) для разработки программного обеспечения «QDDClient» - настольного клиента для Dropbox.

1.1 Обоснование темы разработки приложения

Основанием для разработки приложения является учебный план.

1.2 Краткая характеристика области применения

Программа предназначена к применению на домашних настольных ПК.

1.3 Назначение разработки

1.3.1 Функциональное назначение

Функциональным назначением программы является предоставление пользователю возможности управлять содержимым облачного хранилища Dropbox.

1.3.2 Эксплуатационное назначение

Программа должна эксплуатироваться на домашних настольных ПК пользователя. Особые требования к конченому пользователю не предъявляются.

1.4 Требования к программе

1.4.1 Требования к составу выполняемых функций

Программа должна 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.

1.4.2 Требования к организации входных данных

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

1.4.3 Требования к организации выходных данных

Требования к организации выходных данных не предъявляются.

1.4.4 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.

1.4.5 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.

1.4.6 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.

1.4.7 Требования к программным средствам, используемым программой

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

1.4.8 Специальные требования

Программа должна обеспечивать взаимодействие с пользователем посредством графического пользовательского интерфейса.

1.5 Требования к программной документации

Состав программной документации должен включать в себя:

1) Техническое задание.
2) Программу и методики испытаний.
3) Руководство пользователя.
4) Техническую документацию.

1.6 Стадии и этапы разработки

Разработка должна быть проведена в три стадии:

1) Разработка технического задания.
2) Рабочее проектирование.
3) Внедрение.

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания. На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1) Разработка программы.

2) Разработка программной документации.

3) Испытания программы.

На стадии внедрения должен быть выполнен этап разработки – подготовка и передача программы.

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1) Постановка задачи.

2) Определение и уточнение требований к техническим средствам.

3) Определение требований к программе.

4) Определение стадий, этапов и сроков разработки программы и документации на неё.

5) Согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.101-77 с требованием п. Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

1) Разработка, согласование, утверждение программы и методики испытаний.

2) Проведение приемо-сдаточных испытаний.

3) Корректировка программы и программной документации по результатам испытаний.

2 Анализ опорных точек зрения

Диаграмма идентификации точек зрения представлена на рисунке 1, диаграмма иерархии точек зрения представлена на рисунке 2.

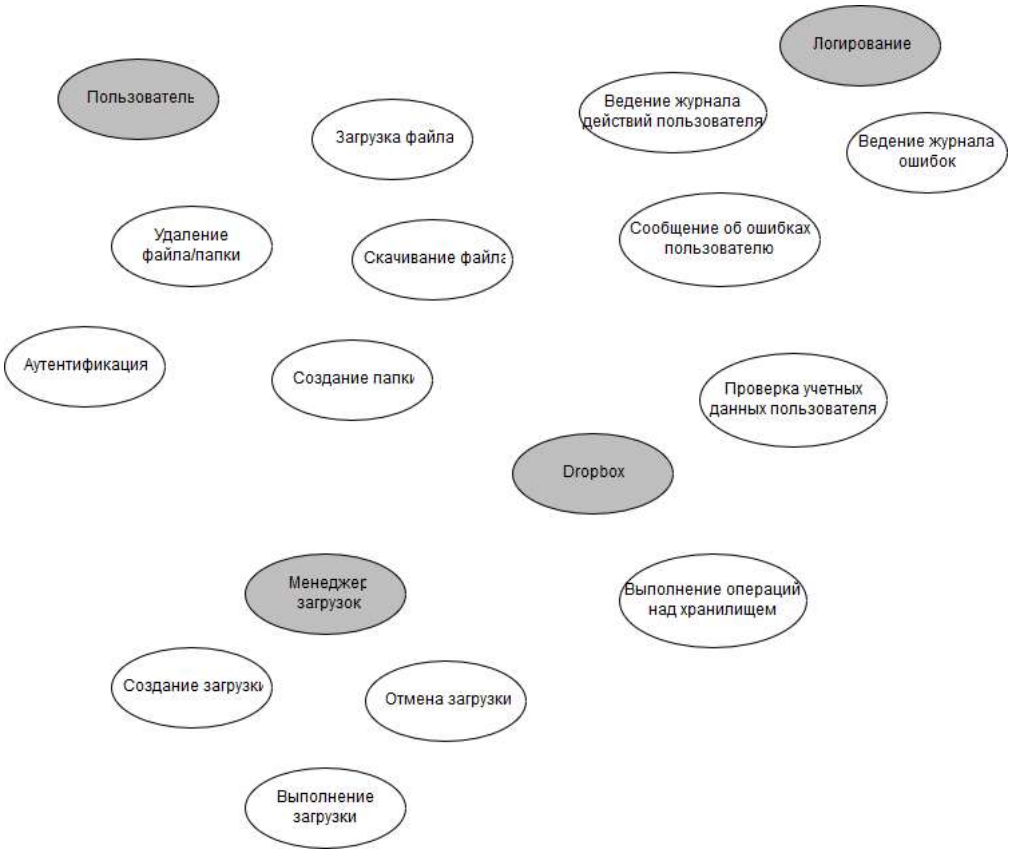

Рисунок 1 – Диаграмма идентификации точек зрения

Рисунок 2 – Диаграмма иерархии точек зрения

3 Разработка диаграммы переходов состояний

Диаграмма переходов состояний представлена на рисунке 3

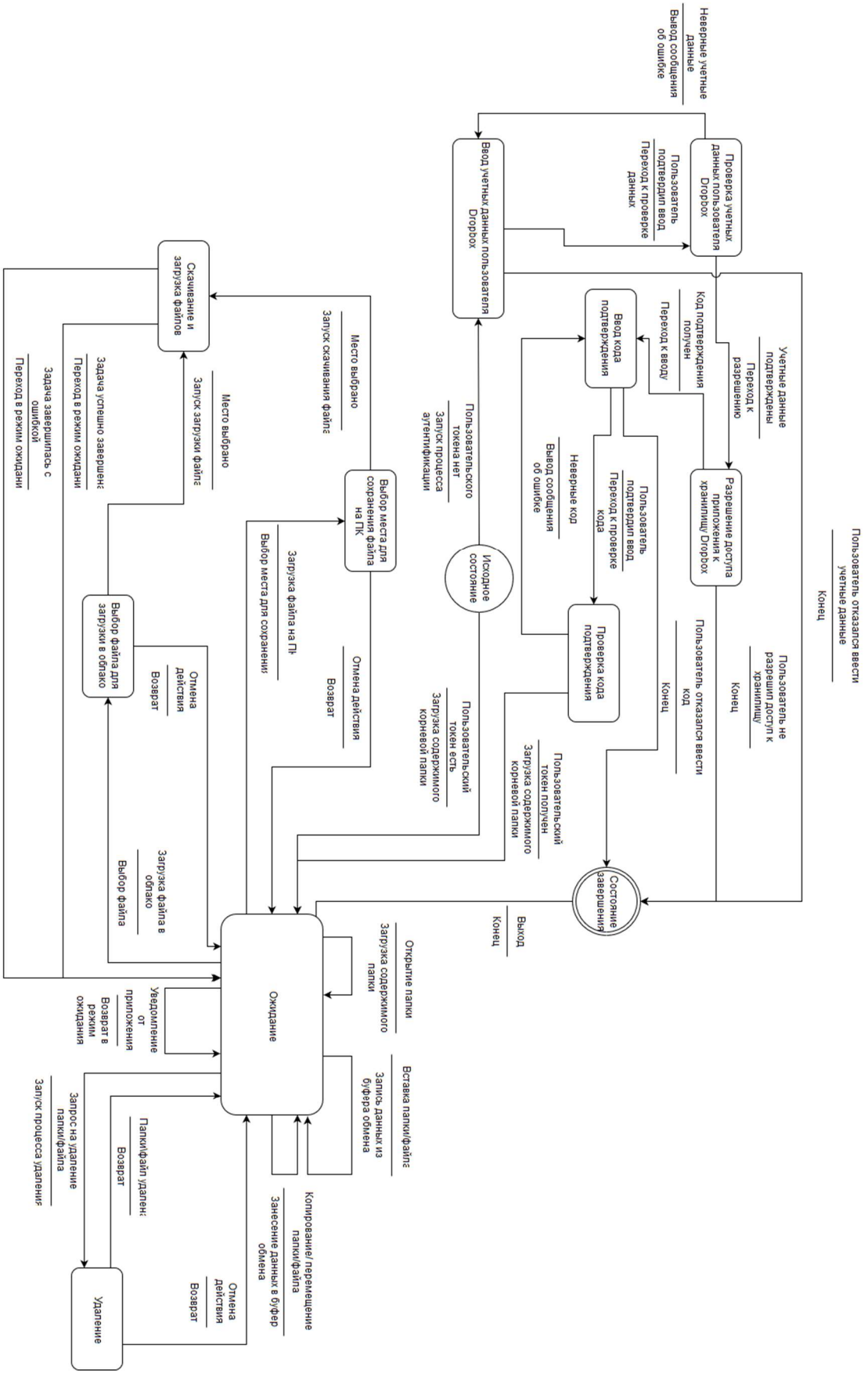

Рисунок 3 – Диаграмма переходов состояний

4 Разработка диаграммы потоков данных

Контекстная диаграмма потоков данных представлена на рисунке 4, детализирующая диаграмма потоков данных представлена на рисунке 5.

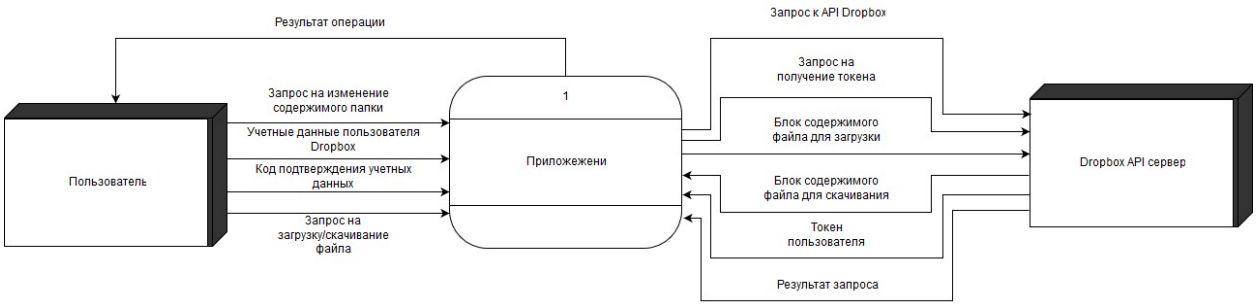

Рисунок 4 – Контекстная диаграмма потоков данных

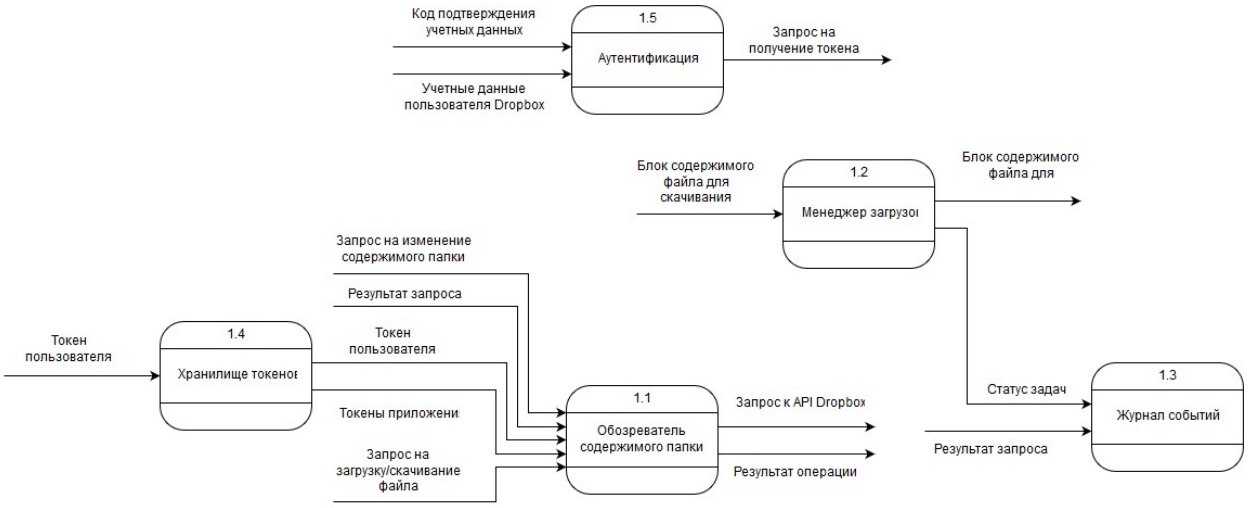

Рисунок 5 – Детализирующая диаграмма потоков данных

5 Диаграмма Насси-Шнейдермана

Диаграмма Насси-Шнейдермана алгоритма вставки файла/папки представлена на рисунке 6.

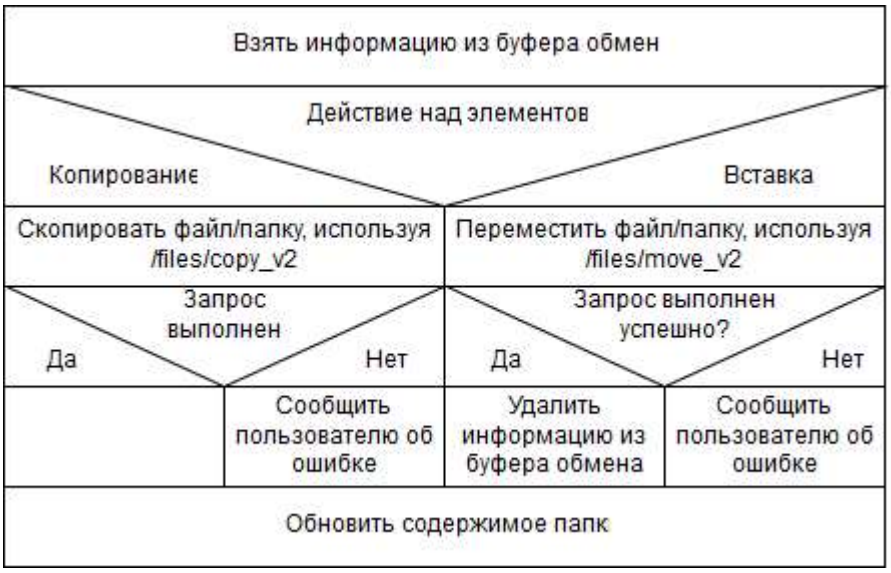

Рисунок 6 – Диаграмма Несси-Шнейдермана алгоритма вставки файла/папки

6 Flow-форма

Flow-форма алгоритма загрузки содержимого папки представлен на рисунке 7.

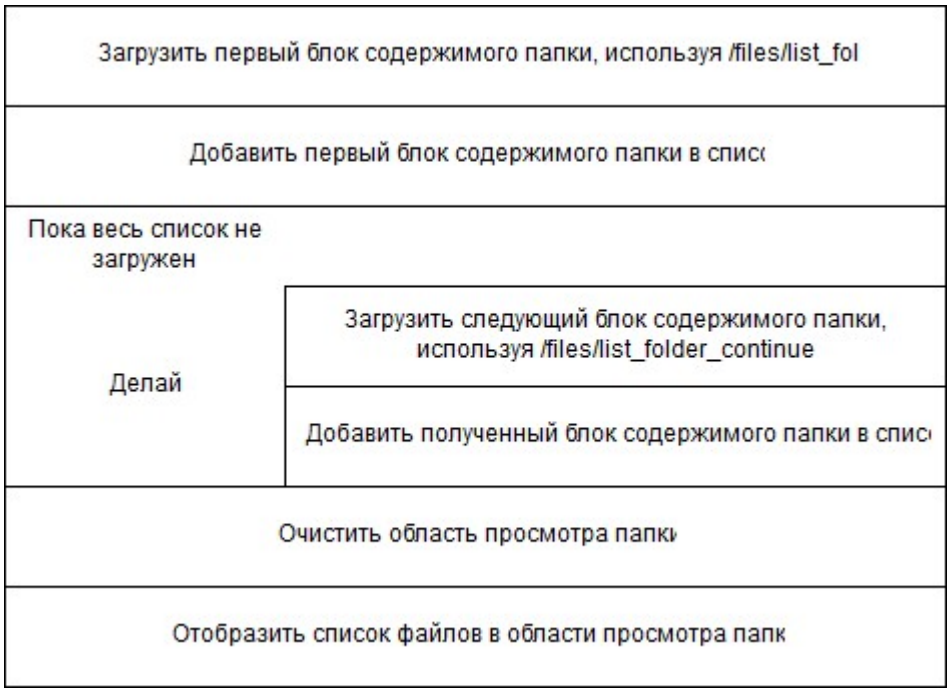

Рисунок 7 – Flow-форма алгоритма загрузки содержимого папки

7 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.

Экранные формы интерфейса пользователя представлены на рисунках 8-10.

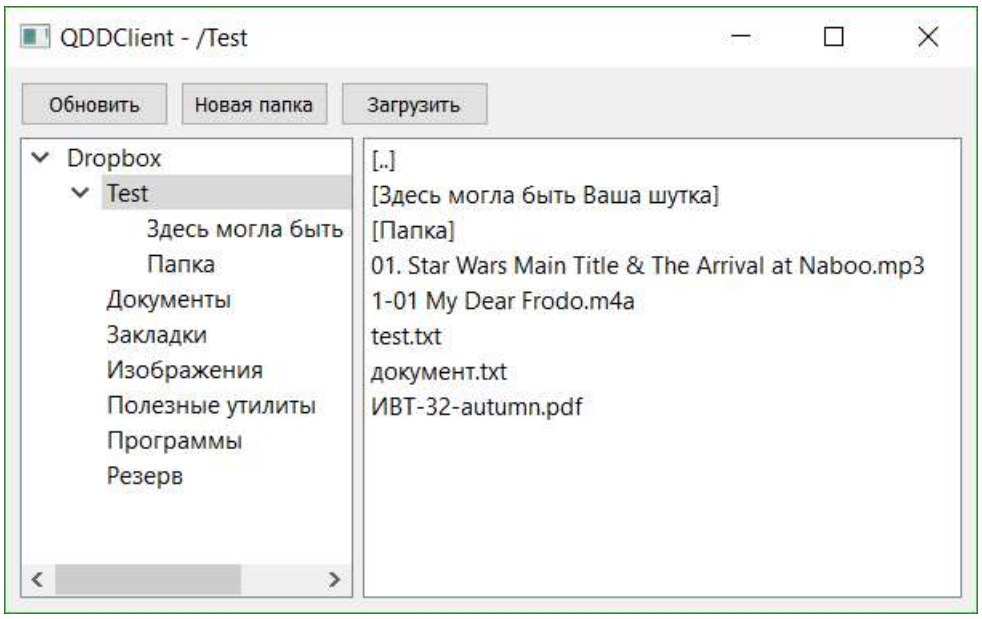

Рисунок 8 – Основной экран пользователя

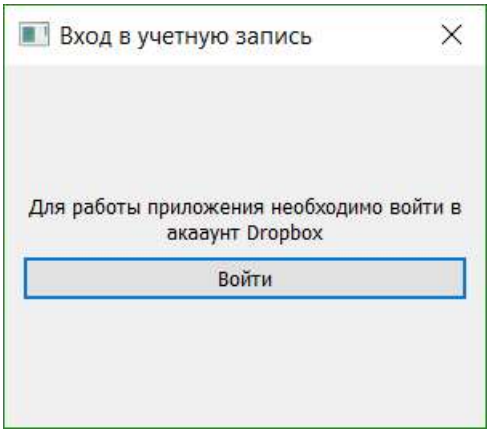

Рисунок 9 – Экран аутентификации пользователя

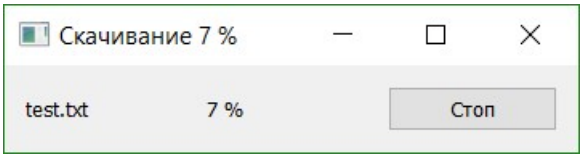

Рисунок 10 – Экран процесса выполнения задачи

8 UML диаграммы

Диаграмма вариантов использования представлена на рисунке 11, диаграмма последовательности представлена на рисунке 12, диаграмма состояний представлена на рисунке 13, диаграмма классов представлена на рисунке 14.

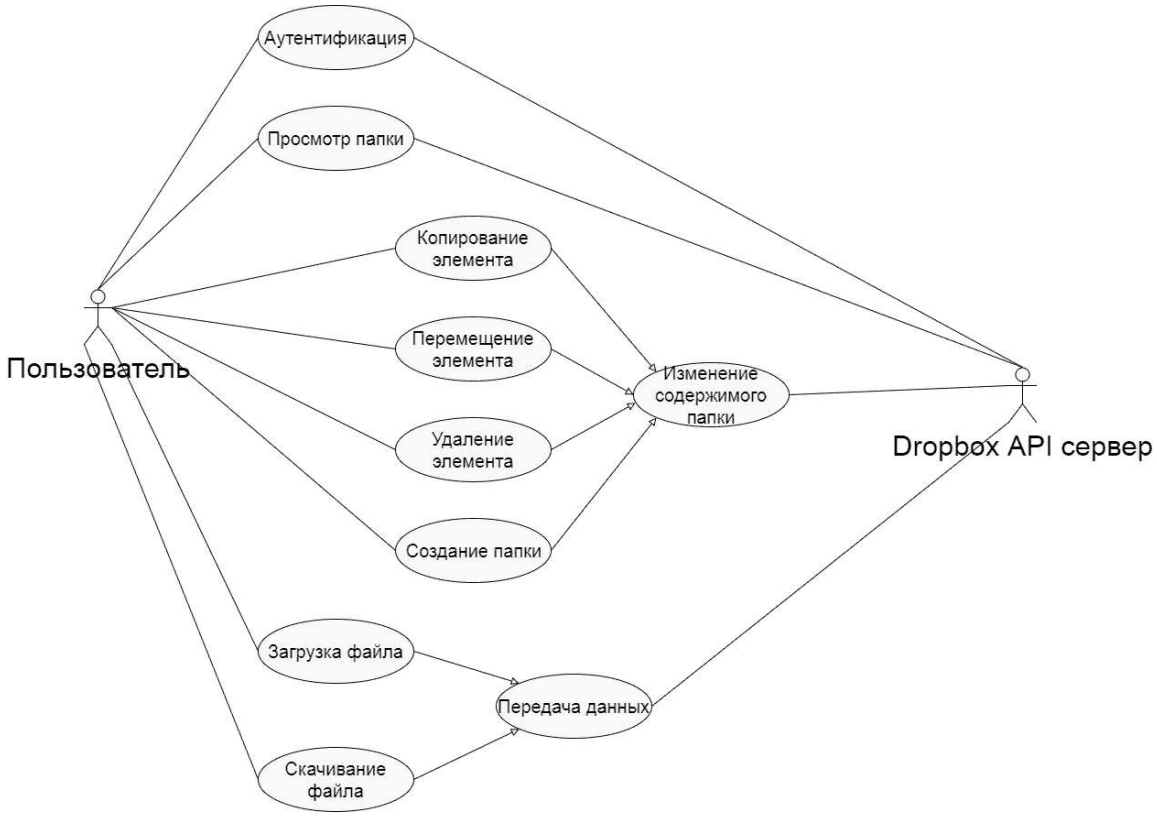

Рисунок 11 – Диаграмма вариантов использования

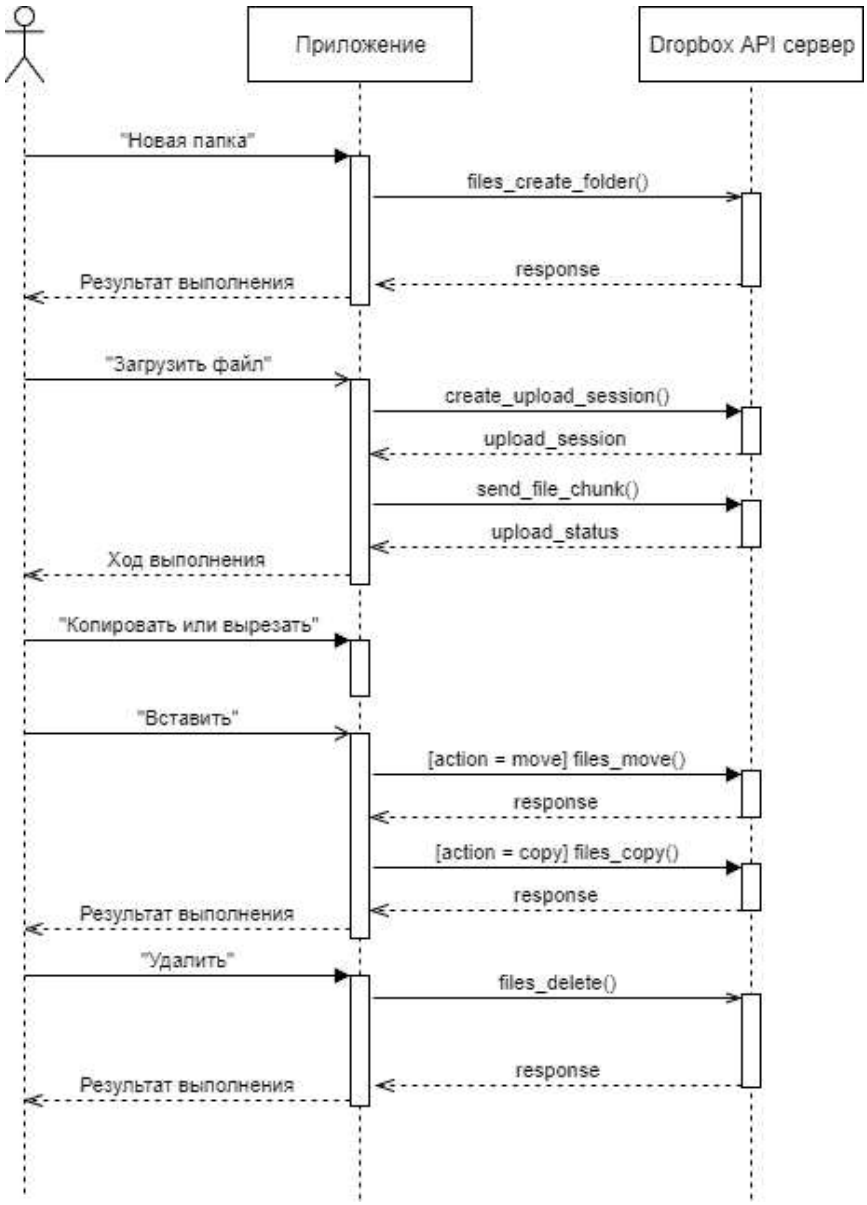

Рисунок 12 – Диаграмма последовательности

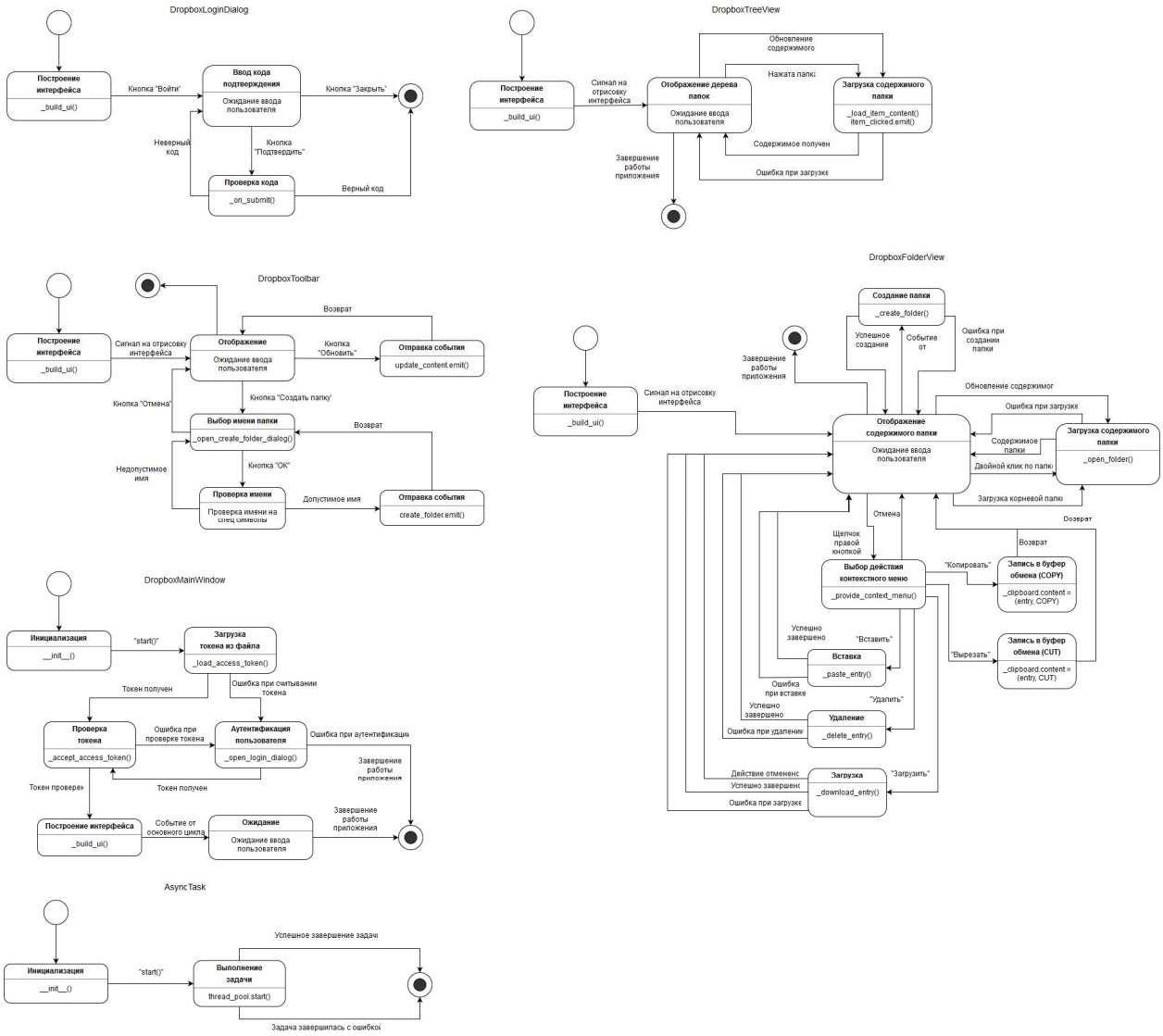

Рисунок 13 – Диаграмма состояний

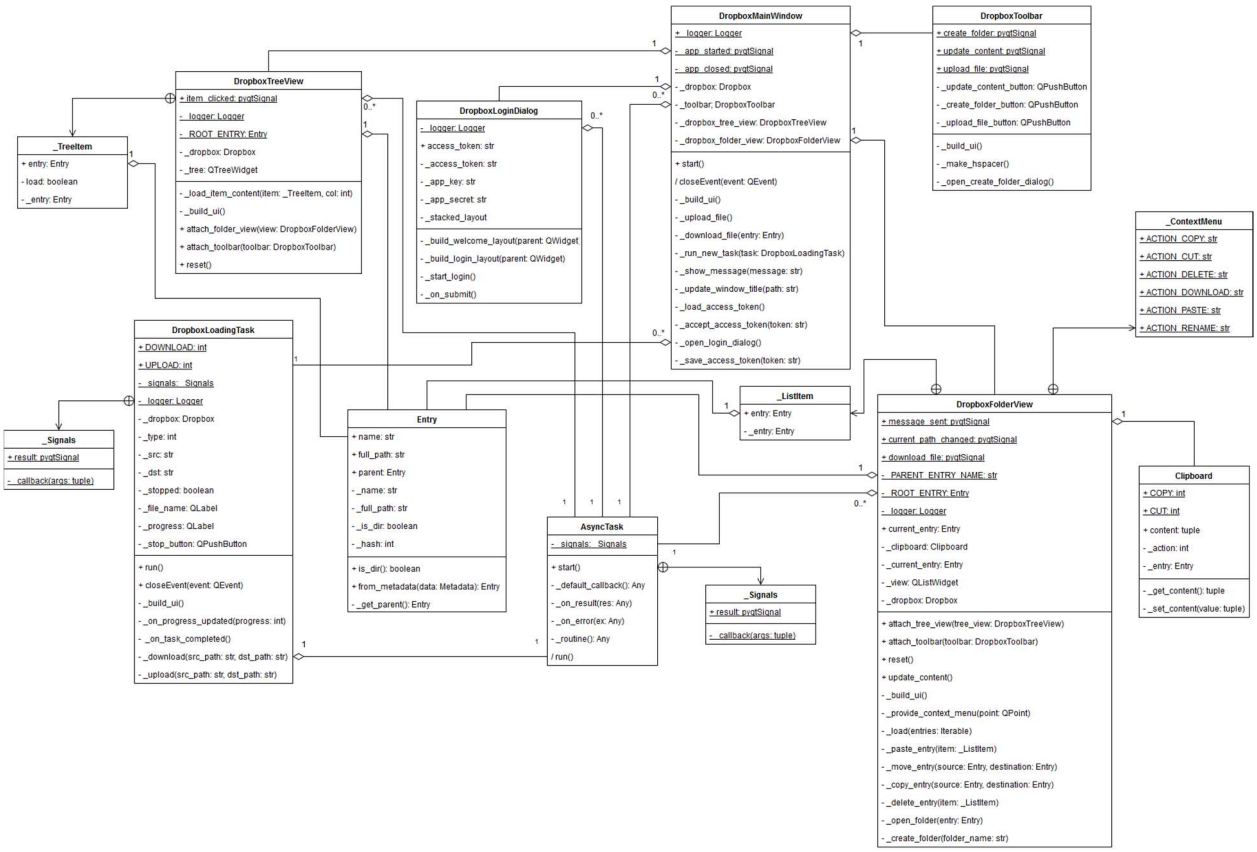

Рисунок 14 – Диаграмма классов

9 Тестирование программного обеспечения

9.1 Тестирование методом черного ящика

Тестирование методом черного ящика проводится для таких входных данных, как название новой папки.

Таблица 9.1 – Классы эквивалентности

Входные условия	Неправильные классы эквивалентности	Правильные классы эквивалентности
Ввод названия для новой папки	Строка, состоящая из пробельных символов (1) Строка, содержащая недопустимые символы (2)	Строка, с хотя бы одним непробельным символом, не содержащая недопустимые символы (3)

Перечисляются классы эквивалентности и соответствующие им тесты.

Таблица 9.2 – Классы эквивалентности и тесты

Номер класса эквивалентности	Тест
1	Строка состоит только из пробелов
2	В строке содержится хотя бы один недопустимый символ
3	В строке есть хотя бы один непробельный символ и нет недопустимых символов

Ожидаемый результат теста (1) – Выдать сообщение о том, что название не может состоять только из пробелов.

Ожидаемый результат теста (2) – Выдать сообщение о том, что название не может содержать недопустимые символы.

Ожидаемый результат теста (3) – Запуск процедуры создания папки.

Тестирование программной оболочки методом черного ящика представлено на рисунках 15 и 16.

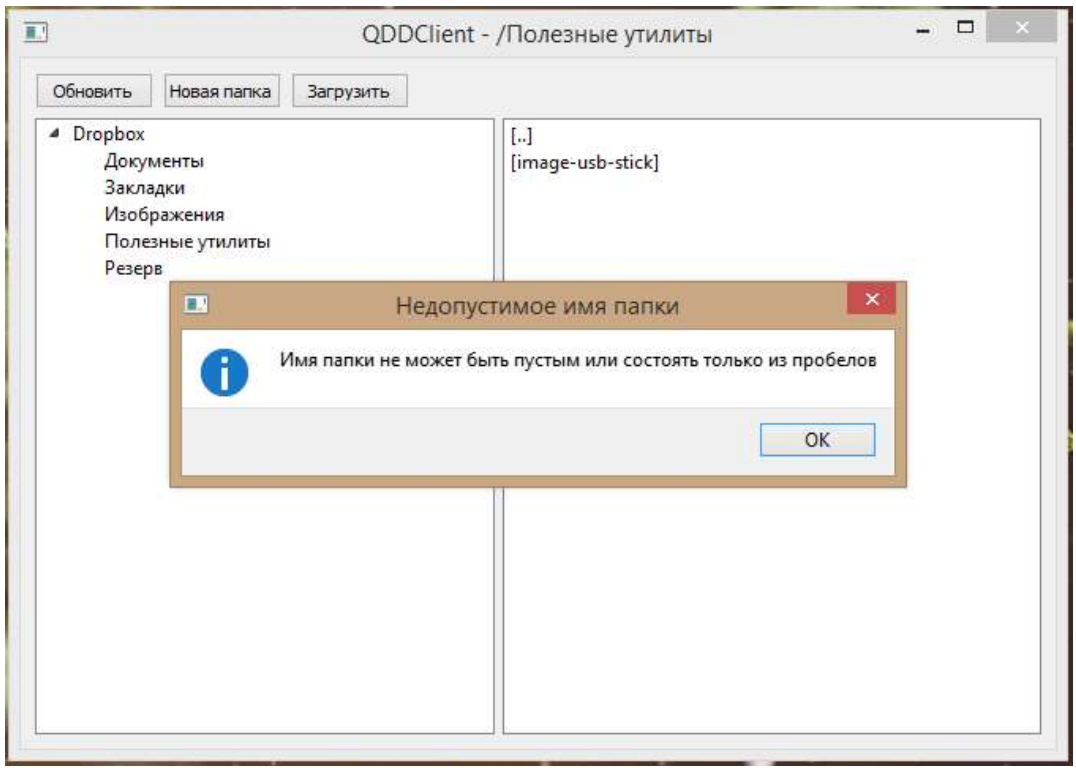

Рисунок 15 – Результат теста (1)

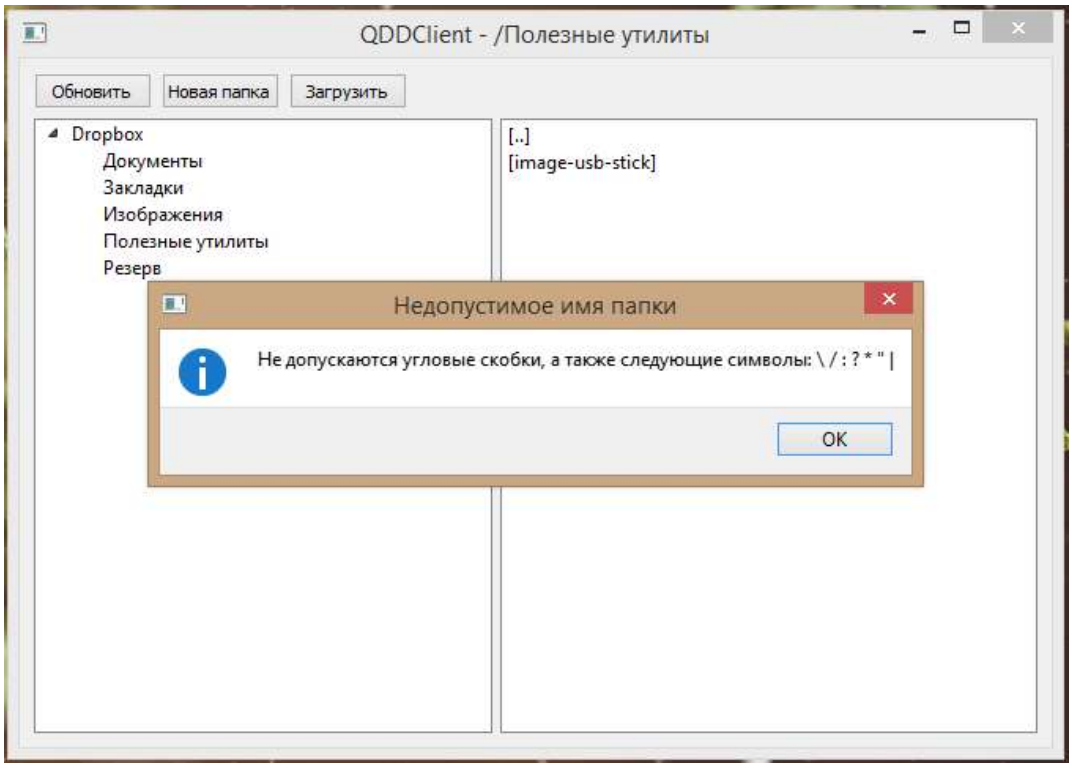

Рисунок 16 – Результат теста (2)

9.2 Выводы по тестированию методом черного ящика

В ходе тестирования методом черного ящика ошибок не выявлено.

9.3 Тестирование методом белого ящика

Составляются тесты методами покрытия операторов, покрытия решений и покрытия условий. Граф-схема алгоритма аутентификации пользователя представлена на рисунке 17.

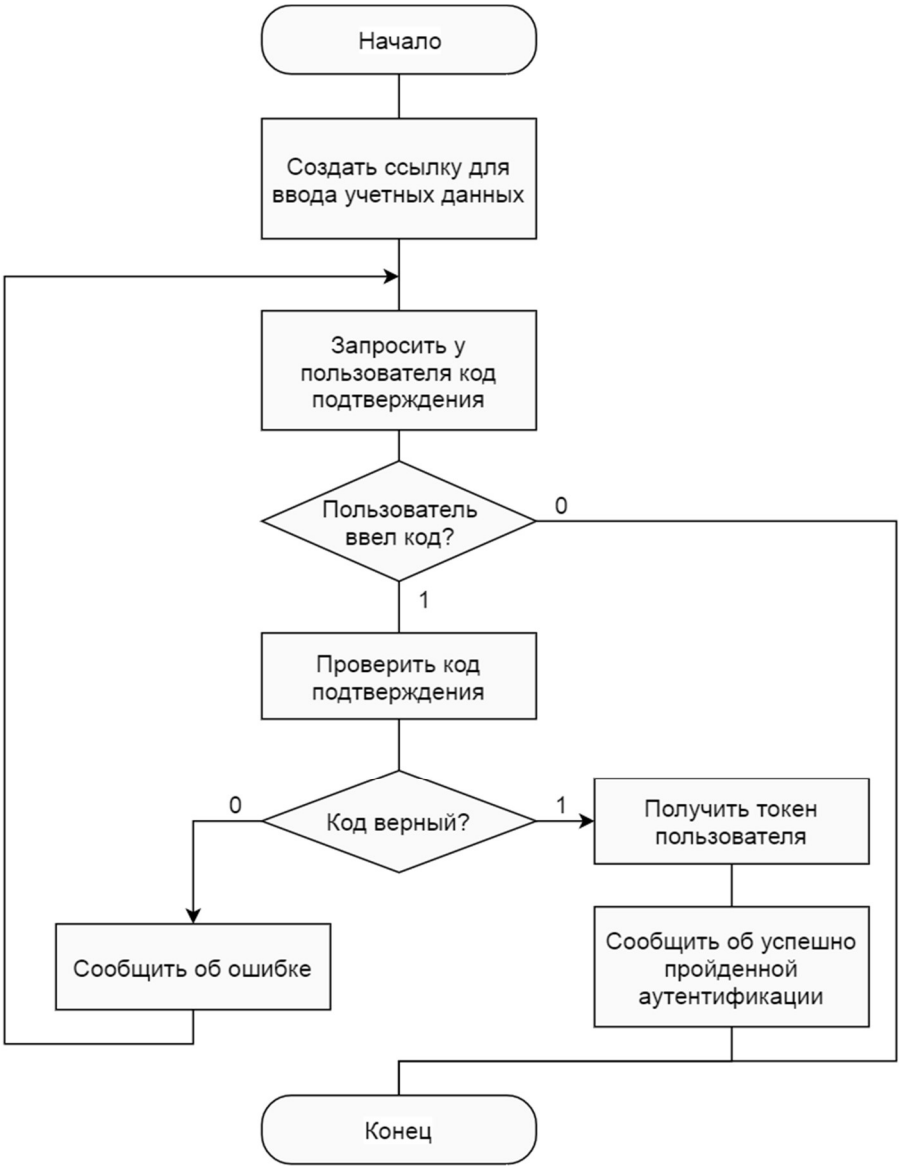

Рисунок 17 – Граф-схема алгоритма аутентификации пользователя

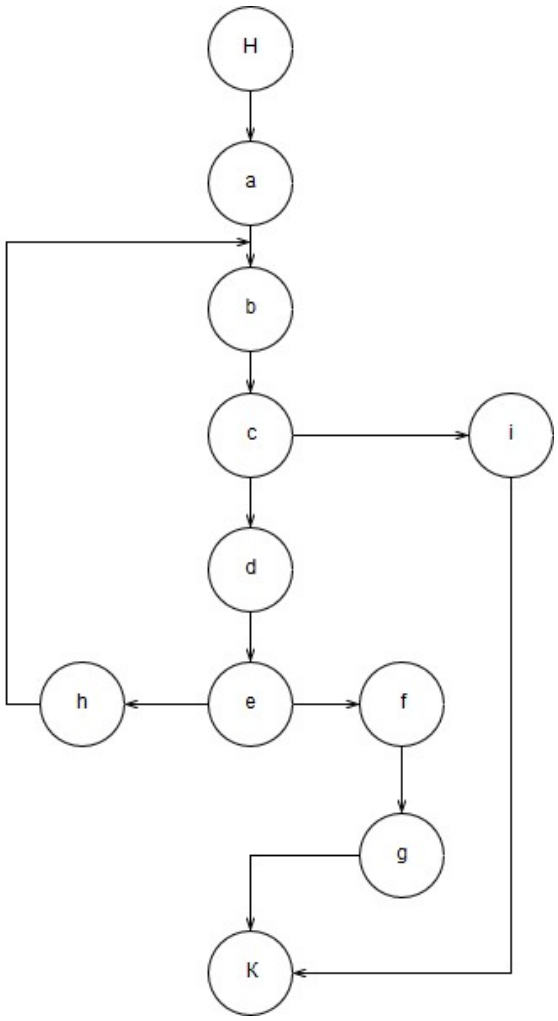

Рисунок 18 – Управляющий граф

Для метода покрытия операторов составляются следующие тесты:

1) Тест 1 – Запрос у пользователя кода, пользователь отказался ввести код, завершение работы приложения (a→b→c→i)

2) Тест 2 – Запрос у пользователя кода, ввод кода, проверка, выдача сообщения о неверном коде, очередной запрос кода у пользователя (a→b→c→d→e→h→b)

3) Тест 3 – Запрос у пользователя кода, проверка, получение токена, закрытие окна аутентификации (a→b→c→d→e→f→g)

Результаты тестирования приведены в таблице 11.3.

Таблица 9.3 – Результаты тестирования

№	Тест	Ожидаемый результат	Полученный результат	Результат тестирования
1	Запрос у пользователя кода, пользователь отказался ввести код, завершение работы приложения	Закрытие окна аутентификации, завершение работы приложения	Закрытие окна аутентификации, завершение работы приложения	Успешно пройдено
2	Запрос у пользователя кода, ввод кода, проверка, выдача сообщения о неверном коде, очередной запрос кода у пользователя	Выдача сообщения о неверном коде	Выдача сообщения о неверном коде	Успешно пройдено
3	Запрос у пользователя кода, проверка, получение токена, закрытие окна аутентификации	Закрытие окна аутентификации, открытие основного окна приложения	Закрытие окна аутентификации, открытие основного окна приложения	Успешно пройдено

Для метода покрытия условий тесты точно такие же.

9.4 Выводы по тестированию методом белого ящика

В ходе тестирования методом белого ящика ошибок не выявлено.

Приложение А

(обязательное)

Листинг процедуры создания новой папки

```
def _open_create_folder_dialog(self):
        while True:
            folder_name, accepted = QtWidgets.QInputDialog.getText(
                self, 'Создать новую папку', 'Введите имя:', text='Новая папка'
            )
            if not accepted:
                return

            folder_name = folder_name.strip()
            if any(char in folder_name for char in r'\/:?*"|<>'):
                QtWidgets.QMessageBox.information(
                    self,
                    'Недопустимое имя папки',
                    r'Не допускаются угловые скобки, а также следующие символы: \ / :
? * " |'
                )
            elif folder_name == '':
                QtWidgets.QMessageBox.information(
                    self,
                    'Недопустимое имя папки',
                    'Имя папки не может быть пустым или состоять только из пробелов'
                )
            else:
                self.create_folder.emit(folder_name)
                break
```

Приложение Б

(обязательное)

Листинг процедуры аутентификации пользователя

```
def _on_submit(self):
        def check_code() -> str:
            code = self._code_field.text().strip()
            result = self._flow.finish(code)

            return result.access_token

        def on_completed(token: str):
            self._access_token = token
            self.accept()

        def on_error(ex: Exception):
            self._logger.error('Invalid code')
            self._logger.error(ex)

            QtWidgets.QMessageBox.critical(self, 'Ошибка', 'Неверный код!')

            self._submit_button.setDisabled(False)
            self._submit_label.setDisabled(False)

        self._submit_button.setDisabled(True)
        self._submit_label.setDisabled(True)

        AsyncTask(
            check_code,
            on_completed,
            on_error
        ).start()
```