МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования «Самарский национальный исследовательский   
университет имени академика С.П. Королева»

(Самарский университет)

|  |  |
| --- | --- |
| Институт | информатики, математики и электроники |
| Факультет | Информатики |
| Кафедра | геоинформатики и информационной безопасности |

КУРСОВОЙ ПРОЕКТ ПО ДИСЦИПЛИНЕ

«Технологии и методы программирования»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

«Разработка программных средств автоматического резервного копирования данных пользователя»

|  |  |
| --- | --- |
| Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)* | Ю.Д. Кропивенцев |
|  |  |
| Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)* | Е.В. Мясников |
|  |  |

САМАРА 2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования «Самарский национальный исследовательский   
университет имени академика С.П. Королева»

(Самарский университет)

|  |  |
| --- | --- |
| Институт | информатики, математики и электроники |
| Факультет | информатики |
| Кафедра | геоинформатики и информационной безопасности |

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

Студенту ***Кропивенцеву Юрию Дмитриевичу*** группы 6312

Тема проекта: ***«Разработка программных средств для автоматического копирования данных пользователя»***

|  |  |  |
| --- | --- | --- |
| Планируемые результаты освоения образовательной программы (компетенции) | Планируемые результаты практики | Содержание задания |
| ОПК-3 ‑ способность применять языки, системы и инструментальные средства программирования в профессиональной деятельности | Знать: основные виды программных средств, технологию разработки алгоритмов и программ и методы их отладки, основы объектно-ориентированного подхода к программированию  Уметь: работать с современными системами программирования, самостоятельно осваивать новые программные средства  Владеть: языками процедурного и объектно-ориентированного программирования, навыками разработки и отладки программ. | 1. Изучение существующих методов и видов копирования данных  2. Разработка алгоритма копирования  3. Изучение существующих программ по реализации резервного копирования  4. Изучение существующих программных средств для реализации алгоритма  5. Реализация разработанного алгоритма копирования  6. Разработка и реализация программного интерфейса пользователя, а также защиты от несанкционированного использования  7. Отладка и тестирование, и сборка программы |

Дата выдачи задания 5 сентября 2023 г.

Срок представления на кафедру отчета о практике 22 декабря 2023 г.

Руководитель курсового проекта

доцент каф. ГИиИБ, к.т.н. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Мясников

*(подпись)*

Задание принял к исполнению

студент группы № 6312-100503D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю.Д. Кропивнецев

*(подпись)*

РЕФЕРАТ

**Пояснительная записка к курсовому проекту:** ?? c., ?? рисунков, ?? таблиц, ?? источников, ?? приложений.

РЕЗЕРВНОЕ КОПИРОВАНИЕ, АЛГОРИТМ КОПИРОВАНИЯ, АВТОМАТИЧЕСКОЕ РЕЗЕРВНОЕ КОПИРОВАНИЕ, РАБОТА С ФАЙЛАМИ, ФАЙЛОВАЯ СИСТЕМА, АРХИВАЦИЯ, РАБОТА С АРХИВАМИ, ЛИЦЕНЗИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ *(5-15 слов)*

Цель работы – разработка и реализация алгоритма резервного копирования данных, с последующей имплементацией к приложению с графическим интерфейсом.

В ходе проделанной работы был сделан обзор существующих методов и решений в области резервного копирования информации, что послужило основой для написания собственного алгоритма. В ходе разработки были изучены существующие программные реализации алгоритмов на языке программирования Python, с дальнейшим применением в основном программном коде. В процессе проделанной работы был создан специальный вспомогательный модуль для работы с файлами, разгрузивший основной файл приложения. Разработанный продукт обладает интуитивно понятным интерфейсом, способствующим лучшему пользовательскому опыту. Реализована работа программы «в трее», большое количество времени было уделено отладке и защите приложения от поведения, не способствующего корректной работе файловой системы. В последних версиях был добавлен вход по логину и паролю, что способствует выполнение требования «программной защиты от несанкционированного использования». В результате работы имеется приложение, способное выполнять резервную копию файлов как в ручном, так и в автоматических режимах.

Содержание

[Содержание 4](#_Toc154092486)

[ВВЕДЕНИЕ 5](#_Toc154092487)

[1 МЕТОДЫ И АЛГОРИТМЫ РЕЗЕРВНОГО КОПИРОВАНИЯ 7](#_Toc154092488)

[1.1 Анализ методов резервного копирования 7](#_Toc154092489)

[1.2 Анализ существующих решений на рынке 9](#_Toc154092490)

[1.3 Изучение существующих программных средств для реализации алгоритма….. **Error! Bookmark not defined.**](#_Toc154092491)

[1.4 Разработка алгоритма… 12](#_Toc154092492)

[1.5 Выводы и результаты 16](#_Toc154092493)

[2 ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 17](#_Toc154092494)

[2.1 Выбор средств разработки и системных программных средств **Error! Bookmark not defined.**](#_Toc154092495)

[2.2 Требования к разрабатываемой программе 18](#_Toc154092496)

[2.3 Структура программы 19](#_Toc154092497)

[2.3.1 Основные модули 19](#_Toc154092498)

[2.3.2 Описание основных классов 19](#_Toc154092499)

[2.3.3 Описание основных структур данных **Error! Bookmark not defined.**](#_Toc154092500)

[2.3.4 Описание основных функций и процедур 19](#_Toc154092501)

[2.4 Описание интерфейса пользователя 21](#_Toc154092502)

[2.5 Контрольный пример и результаты тестирования 29](#_Toc154092503)

[2.6 Выводы и результаты 29](#_Toc154092504)

[3 ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ **Error! Bookmark not defined.**](#_Toc154092505)

[3.1 Данные для экспериментов **Error! Bookmark not defined.**](#_Toc154092506)

[3.2 Методика проведения эксперимента **Error! Bookmark not defined.**](#_Toc154092507)

[3.3 Результаты экспериментов **Error! Bookmark not defined.**](#_Toc154092508)

[3.4 Выводы и результаты **Error! Bookmark not defined.**](#_Toc154092509)

[ЗАКЛЮЧЕНИЕ 30](#_Toc154092510)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 31](#_Toc154092511)

[ПРИЛОЖЕНИЕ А 32](#_Toc154092512)

# ВВЕДЕНИЕ

В настоящее время мы почти каждый день имеем дело с далеко не самыми надежными носителями информации – крайне мало людей, не терявших важные данные из-за вышедшего из строя диска. Во избежание таких неприятных ситуаций были придуманы схемы копирования информации, примером может послужить легенда о правиле в компании Coca-Cola: оба создателя рецепта не должны летать в одном самолете. Так как сохранность информации актуальна и по сей день, резервное копирование информации обладает еще большим спросом, чем когда-либо.

На текущий момент времени на рынке представлено большое количество решений, предоставляющих большой спектр возможностей защитить свои данные. Копирование может быть представлено как загрузка на удаленный сервер хранения, что перекладывает ответственность на строну владельца сервиса, также архивация может производиться на внешний физический носитель, что позволяет вам непосредственно контролировать что происходит с вашей резервной копией. Большинство программ обладает сложным и перегруженным интерфейсом, что может вызвать затруднение у новых пользователей.

Целью данной работы является создать удобный и рабочий алгоритм резервного копирования, представленный в виде приложения для операционной системы Windows, обладающего простым и понятным интерфейсом, благодаря которому пользователь будет получать лучший опыт взаимодействия и решать задачи в области копирования информации. Для достижения поставленной цели были решены следующие задачи:

1. изучение видов резервного копирования и подходов к хранению копий, для последующей имплементации к проекту;
2. анализ существующих решений на рынке в целях получения общего представления об изучаемой сфере;
3. изучение и выбор программных средств и библиотек для написания авторской реализации полученных знаний;
4. разработка собственного алгоритма для последующей реализации;
5. реализация алгоритма копирования в виде программного кода на основе;
6. создание программы с графическим интерфейсом, обладающим простым и понятным дизайном;
7. отладка и тестирование созданного приложения.

# МЕТОДЫ И АЛГОРИТМЫ РЕЗЕРВНОГО КОПИРОВАНИЯ

## Анализ методов резервного копирования

Запрос на создание копии данных существовал задолго до появления компьютеров: второй экземпляр документов, летописи, те же конспекты – все это является дубликатом информации. С появлением цифровых устройств осуществлять подобные манипуляции вручную стало крайне долго и сложно в силу возросшего количества резервируемой информации. Решением стало «резервное копирование» – процесс, результатом которого является создание копии информации на носителе информации, например, на жестком диске или на удаленном сервере.

Резервное копирование преследует важную и простую цель: получение возможности быстро и недорого восстановить информацию в случае утери рабочей версии по какой-либо причине.

Система выполняющая роль резервного копирования должна отвечать нескольким требованиям:

* надежность хранения информации – обеспечение отказоустойчивости программных и технических средств;
* кроссплатформенность – возможность запускать систему резервного копирования на любом устройстве вне зависимости от операционной системы;
* простота эксплуатации – в первую очередь программой будут пользоваться обычные люди, что создает запрос на простой и понятный функционал;
* быстрое внедрение – развертывание системы должно происходить быстро, не требуя создания дополнительного окружения.

Разобравшись с требованиями и задачами, перейдем к описанию видов резервного копирования. Существует несколько видов резервного копирования:

* полное резервное копирование (full backup) – такое копирование, при котором выполняется полная копия всех данных, в основном затрагивается вся система и все файлы. Целесообразно производить раз в большое количество времени во избежание чрезмерной нагрузки на инфраструктуру и работу организации, а предыдущие копии сжимать и архивировать. Незаменим в случае необходимости восстановления системы.
* Дифференциальное резервное копирование (differential backup) – при таком типе копирования будет создана копия только изменившихся файлов с моментов последнего полного копирования. Положительно сказывается на процессе восстановления, особенно полезен при заражении файлов вирусами.
* Инкрементное копирование (incremental backup) – такой тип копирования отличается от двух вышеописанных подходов тем, что копирование происходит только тех файлов, которые были изменены с момента последней копии будь то полное копирование или дифференциальное. Последующее инкрементное копирование добавляет только изменившиеся файлы. Занимает меньше времени, однако требует большего времени восстановления из-за новых этапов: последовательное восстановление резервной копии, а после – инкрементных копий. Отличие инкрементного от дифференциального копирования в том, что поменявшиеся файлы добавляются на носитель независимо.
* Клонирование – копирование целого раздела со всем содержимым в другой раздел.
* Резервное копирование в виде образа – создание точной копии раздела или носителя, хранящейся в одном файле.
* Резервное копирование в режим реального времени – создание копии файлов, без остановки работы системы, процесс происходит «в фоне».
* Холодное копирование – копирование системы выполняется, когда пользователь не может вносить изменения в файлы, гарантируется согласованное состояние.
* Горячее копирование – такое копирование системы, при котором пользователи могут вносить изменения во время копирования, копия базы приводится в согласованное состояние путем использования журналов копирования по окончании основной операции резервного копирования.

Исходя из полученной информации было принято решение создать продукт, выполняющий задачи резервного копирования по типу полного резервного копирования и дифференциального резервного копирования в силу того, что предложенные подходы покрывают большинство случаев, а также отличаются простотой реализации. Также будет соблюдено требование кроссплатформенности и простоты эксплуатации.

## Анализ существующих решений на рынке

Убедившись в теоретическом аспекте вопроса, необходимо проверить, так ли нужен полноценный продукт покупателю, существует ли реальный запрос у рядового пользователя и у бизнеса, в частности.

Произведя анализ этой области рынка, было обнаружено множество решений, причем многие из них присутствуют на рынке уже далеко не первый год, а также, что радует, присутствует большое количество open-source проектов.

Проведем анализ наиболее известных и популярных решений:

* Acronis Cyber Protect Home Office (бывший Acronis True Image)

Программа для резервного копирования и восстановления данных, позволяющая создавать образы диска, резервные копии в ручном или автономном режиме, а онлайн хранилище позволяет дополнительно обезопасить данные. Поддерживается кроссплатформенность, запускать программное обеспечение (ПО) можно на операционных системах Microsoft Windows или MacOs или даже на мобильных устройствах с Android или Apple ios.

Отмечу интуитивный и приятный дизайн, возможность создания как онлайн, так и офлайн копии, а также возможность использования программы вне зависимости от операционной системы.

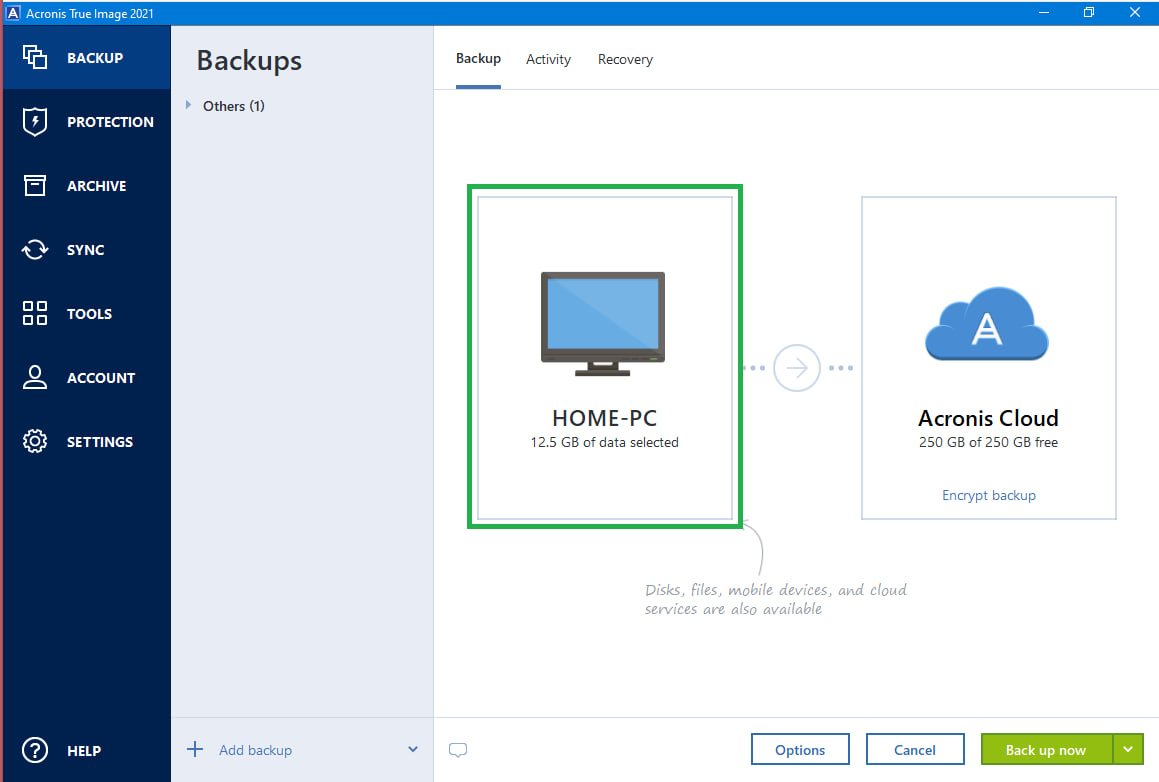


Рисунок 1 – Пример интерфейса программы Acronis True Image

* Time Machine (macOS) от Apple Inc

Программа, интегрированная в операционную систему Apple MacOs, позволяющая восстановить нужный вам файл или всю операционную систему в целом из нужного вам отрезка времени. Утилита работает как с облачным хранилищем, так и с физически подключенным к устройству. Обладая крайне простым и понятным функционалом, данная реализация имеет множество положительных отзывов от пользователей, потому что обладает крайне простым и понятным функционалом.

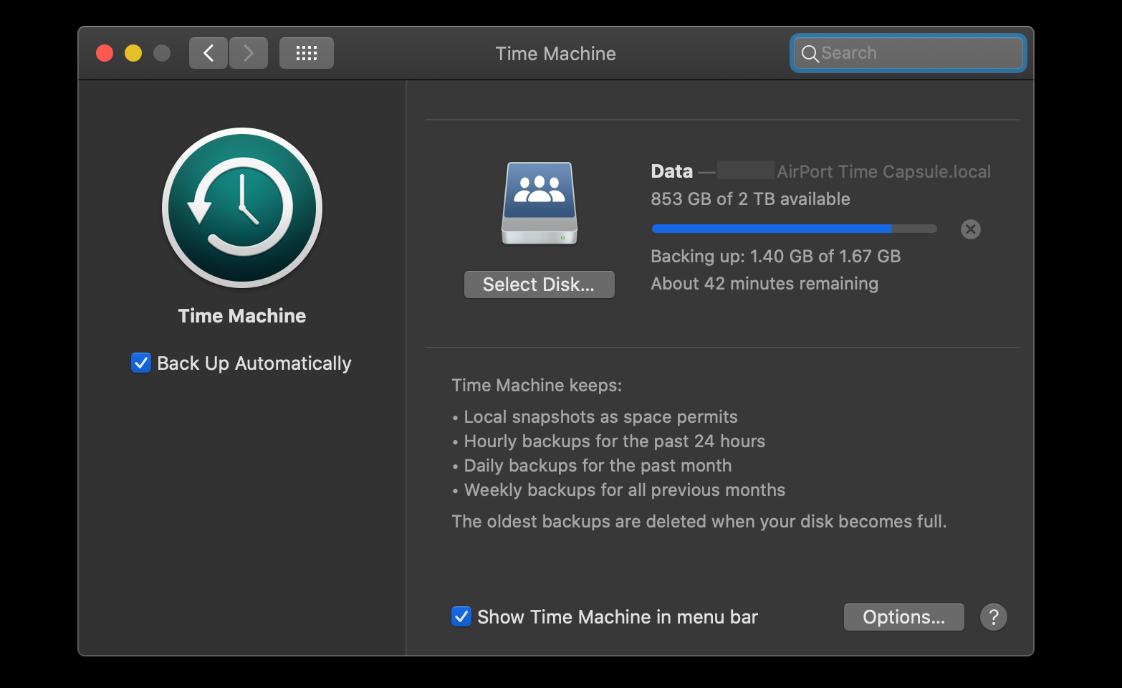


Рисунок 2 – Пример интерфейса Apple Time Machine

* Backup and Restore от Microsoft

Программа, являющаяся основной для создания резервной копии в операционной системе Windows 7. В ее функционал входит резервирование как файлов или папок, так и всей операционной системы в целом. Используется для восстановления системы. Копирование файлов и папок производится в двух режимах: полного и инкрементного копирований, а созданные копии хранятся в виде zip-архивов. А системы – в виде полноценного образа для последующего восстановления.



Рисунок 3 – Логотип Restore and Backup

Судя по рассмотренным вариантам, резервное копирование достаточно актуальная тема в силу того, что даже операционные системы предлагают собственные решения в этой области, а в финальную версию продукта войдут только лучшие решения.

## Разработка алгоритма

В момент написания алгоритма стояла задач написать алгоритм копирования, способный выполнять несколько задач:

* Модульность с возможностью быстрой модернизации
* Простота реализации
* Комфорт в использовании

Разработанные алгоримы: алгоритм согдания копии, алгоримат сраврения

Разработанный алгоритм создания резервной копии представлен на рисунке 4.

В начале алгоритма у пользователя запрашиваются входные данные: пути для копирования, список файлов, которые нужно игнорировать, и временной интервал копирования, то есть через какое время будет произведено следующее копирование, причем по умолчанию выбран ноль, а также тип копирования, по умолчанию выбора вариант, подразумевающий полное копирование.

Далее происходит проверка введенных данных: все ли пути существуют и верно ли введено время, если данная проверка не будет пройдена, пользователь увидит сообщение о том, что в процессе настройки произошел сбой и что необходимо ввести валидные данные. В случае выполнения условий алгоритм выполняется в штатном режим и введенные данные заносятся в соответствующие переменные.

Следующие несколько этапов связаны с установкой параметров: устанавливается интервал копирования и выбирается тип копирования.

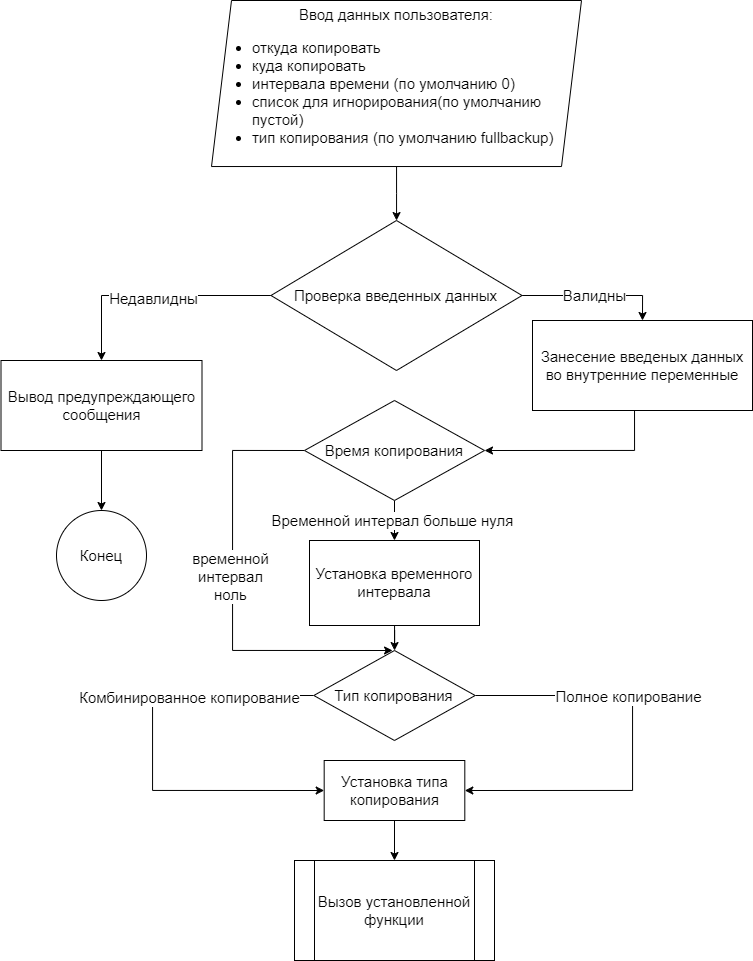


Рисунок 4 – Алгоритм копирования в виде блок схемы

Опишем алгоритмы копирования алгоритм полного копирования full\_backup представлен на рисунке 5

В начале алгоритма выполняется проверка на существование пути копируемой папки: если пути не существует – пользователь увидит сообщение об ошибке, а алгоритм завершится досрочно, в противном случае выполнение продолжится.

Далее проверяется количество уже существующих копий, если они существуют, то будет проведено сравнение последней копии с уже существующей, вариант в котором обе папки идентичны завершается установкой в соответствующем поле статуса последнего успешного копирования, а алгоритм начинается заново, если есть таймер и просто завершается иначе. Если же копий нет, то производится процедура создания нового имени для папки, с последующим копированием всех файлов. Следующие шаги аналогичны уже рассмотренным.

Комбинированное копирование аналогично алгоритму full\_backup, их различия заключаются в том, что рассматриваемый тип использует сразу и полный и дифференциальный виды копирования. Вместо того, чтобы каждый раз создавать копию всего раздела, программа меняет только изменившиеся файлы.

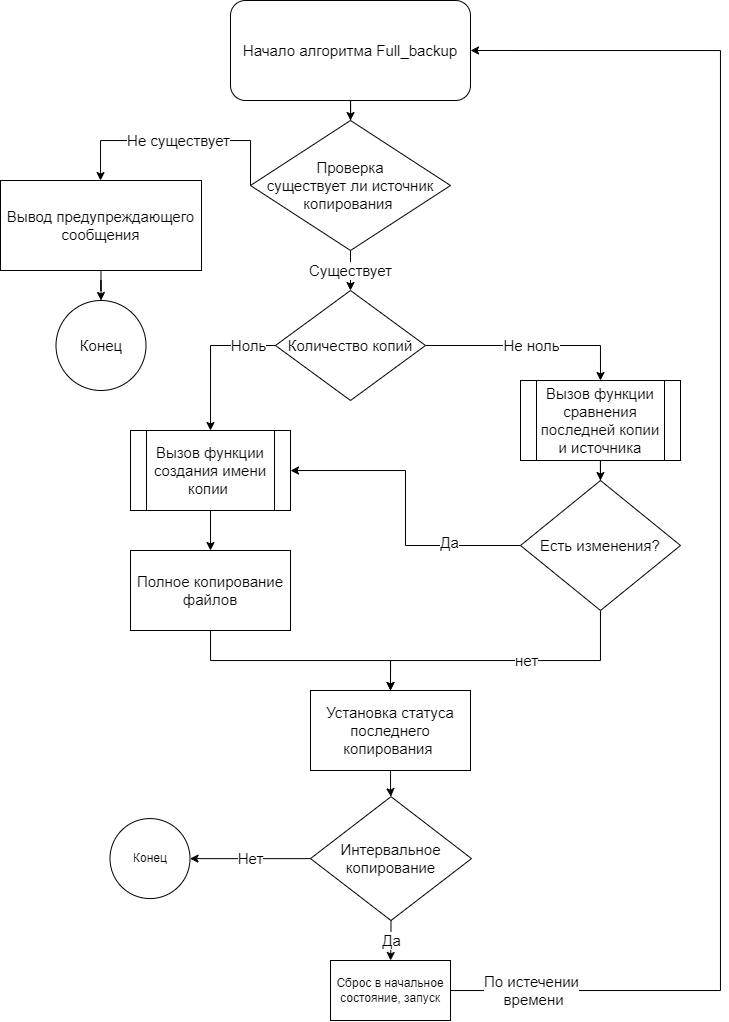


Рисунок 5 – Алгоритм полного копирования в виде блок схемы

## Выводы и результаты

В результате анализа видов резервного копирования, а также изучения существующих решений, было принято решение сосредоточиться на реализации полного и дифференциального видов резервного копирования. Основным видом послужит полное копирование, выполняющее роль основы всего проекта, а также алгоритм дифференциального копирования, чтобы разгрузить систему. Также будет реализован метод восстановления из резервной копии. Описанный функционал будет решать большинство задач, которые ставятся пользователем перед программой по автоматическому резервному копированию.

# ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## Выбор средств разработки и системы программных средств

Оценив количество работы, связанной с взаимодействием с файловой системой, необходимость создания современного и визуально приятного графического интерфейса и собственные требования к разработке, было принято решение в данном проекте использовать язык программирования Python. Такое решение обусловлено широким спектром возможностей при сравнительно меньших затратах усилий, причем качество разработки повышается за счет качественной документации и обилия материалов по языку.

Выбор библиотек оказался достаточно простым и быстрым, вместе с тем рассматриваемые экземпляры обладают отличной документаций и большим количеством примеров использования:

* Библиотека os

Модуль предоставляет удобный вариант использования функций, зависящих от операционной системы, такие как работа с файловой системой, чтение и запись в файл, создание и удаление папок. Библиотека будет использоваться для проверки, существует ли путь, для создания папок. К достоинствам отнесем хорошую документацию и большое количество материалов.

* Библиотека shutil

Предоставляется высокоуровневое взаимодействие с файловой системой, отличительной чертой являются функции:

shutil.copytree(src, dst, symlinks=False, ignore=None, copy\_function=copy2, ignore\_dangling\_symlinks=False, dirs\_exist\_ok=False)

shutil.rmtree(path, ignore\_errors=False, onerror=None, \*, onexc=None, dir\_fd=None)

* Библиотека zipfile

При рассмотрении существующих решений выяснилось, что целесообразно хранить устаревшие копии в виде zip файлов, данная библиотека предоставляет такую возможность: чтение, запись, проверка и создание zip-архивов.

* Библиотека PyQt5

Qt – это набор кроссплатформенных библиотек C++, которые реализуют высокоуровневые API. PyQt – адаптированная под язык программирования Python описанной библиотеки, позволяющая использовать этот язык как альтернативу C++. С помощью этого фреймворка будет создана графическая составляющая проекта – окно для ввода и вывода.

В качестве сред разработки были выбраны QtDesigner и Microsoft Visual Studio Code, такой выбор был сделан из-за удобства и скорости реализации поставленных задач в силу имеющегося опыта работы с этими программами

## Требования к разрабатываемой программе

Разрабатываемая программа должна отвечать требованиям, описанным в пункте 1.1, а также обеспечивать следующие функциональные требования:

* Выполнять копирование по заданным пользователем путям.
* Поддерживать возможность фильтра файлов.
* Реализовывать несколько видов резервного копирования.
* Поддерживать автоматическое копирование.
* Реализовывать возможность восстановления из копии.
* Поддерживать работу «трее».
* Поддерживать авторизацию пользователя.

А также соответствовать следующим нефункциональным требованиям:

* Иметь простой и понятный интерфейс
* Функционировать на операционной системе Windows
* Обеспечивать обратную связь с пользователем

## Структура программы

### Основные модули

Рассмотрим структуру получившейся программы:

* main.py – модуль, в котором содержится класс-приложение, наследуемое от класса PyQt5.QtWidgets.QMainWindow, отвечает за работу приложения: открытие окна, весь функционал и работу программы
* utils.py – вспомогательный модуль, содержит класс, используемый при работе с файлами, отвечает за копирование, сравнение, восстановления и проверку файлов в процессе работы основной программы
* app.ui – модуль создаваемый средой разработки графических интерфейсов QtDesigner, отвечает за визуальную составляющую приложения
* app.py – модуль создаваемый модулем PyUIC

### Описание основных классов

Созданный в main.py класс MainWindow наследуется от класса PyQt5.QtWidgets.QMainWindow и отвечает за весь функционал программы. Полями данных являются все поля ввода, функциональные элементы, а также переменные, отвечающие за хранение путей копирования, массив всех копий, таймеры.

Модуль utils.py в своем составе имеет класс InterfaceFileOperation, отвечающий за работу с файлами, в своем составе имеет множество различных методов таких как: копирование, сравнение, переименование и удаление файлов, также класс отвечает за вход пользователя в программу.

### Описание основных функций и процедур

Опишем методы, содержащиеся в вспомогательном модуле utils.py

Функция, отвечающая за настройку системы логирования (процесса занесения в файле выполненных действий):

def setup\_logging(self) -> None:

Две функции, отвечающие за вход пользователя в систему, первая функция загружает данные с сервера, используя вспомогательную библиотеку PyGithub, вторая уже возвращает результат проверки логина и пароля:

def load\_login(self):

"""fucntion for load hash from file on github"""

try:

g = Github()

user = g.get\_user(self.name)

for content in user.get\_repo("auto\_saver\_app").get\_contents("Passwords"):

if content.path.endswith(".txt"):

self.hash = content.decoded\_content.decode("utf-8")

except Exception as e:

self.logger.warning(f"Raised exception: {e} while load login")

def check\_user(self, login: str, password: str) -> bool:

"""Function to check users loging and password

@login: your login

@Password: your password

inside this function app requests file with hashed password

and then compart it with users password

"""

my\_hash = hashlib.md5((login+password).encode('utf-8')).hexdigest()

if self.hash:

return my\_hash == self.hash

try:

g = Github()

user = g.get\_user(self.name)

hash = ""

for content in user.get\_repo("auto\_saver\_app").get\_contents("Passwords"):

if content.path.endswith(".txt"):

hash = content.decoded\_content.decode("utf-8")

return my\_hash == hash

except Exception as e:

self.logger.warning(f"Raised exception: {e} login")

return False

Функция, отвечающая за создания наименования нового бекапа, на вход подаются строки источники, а также флаг, типа копирования:

def create\_name(self, src: str, dst: str, upd=False) -> str:

Функция, необходимая для создания архива, принимает на вход строку, возвращает путь до нового файла:

def ziping(self, src) -> str:

Функция для проверки существования папки:

def is\_dir(self, path) -> bool:

Функция восстановления из резервной копии, принимает на вход пути, а возвращает новый путь до папки источника:

def recover(self, src: str, dst: str) -> str:

Функция полного копирования, на вход подаются два пути, а также список-фильтр файлов, копирование которых не требуется выполнять, возвращает флаг завершения:

def full\_backup(self, src: str, dst: str, ignore=[]) -> bool:

Функция сравнения двух папок, на вход подаются два пути, а выход – флаг успешности операции:

def cmp\_folder(self, src: str, dst: str, ignore=[]) -> bool:

## Описание интерфейса пользователя

При включении программы, пользователь будет наблюдать уведомление о загрузке программы (рисунок 6)

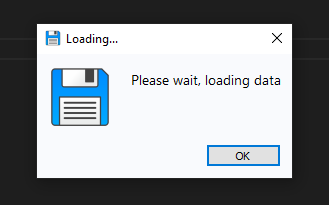


Рисунок 6 – Этап загрузки приложения

После нажатия на кнопку Ok, высветится следующее окно (Рисунок 7), на этом этапе пользователю необходимо ввести логин в поле 1 и пароль в поле 2, по окончании чего нажать кнопку 3, чтобы продолжить работу в программе, не предполагается, что пользователь будет нажимать на кнопки 4, 5 и 6, однако во всех трех случаях доступа в программу получить нельзя будет, в:

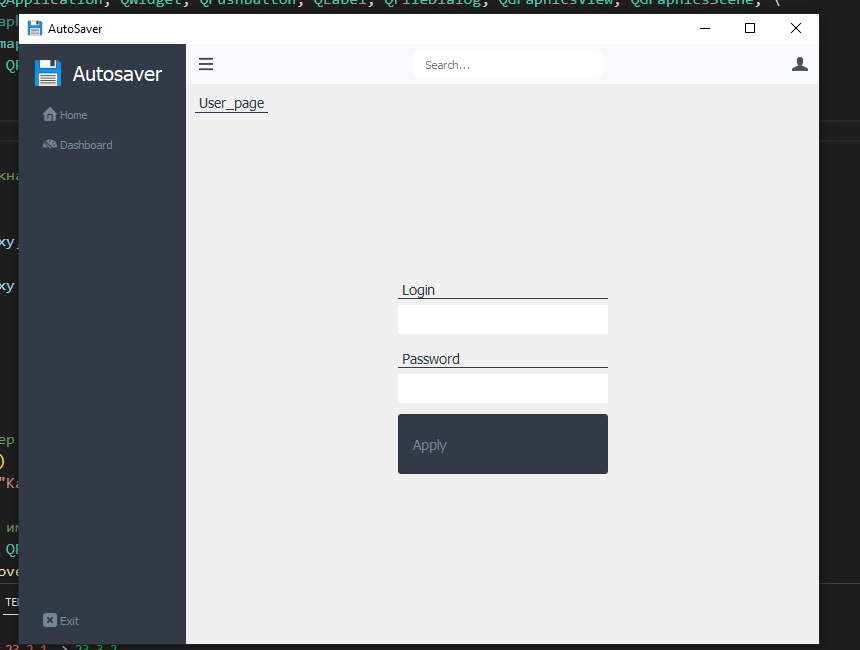


Рисунок 7 – Этап загрузки приложения

В случае 5 неудачных попыток ввести пароль пользователь получит уведомление (рисунок 8) о том, что программа завершает свою работу:

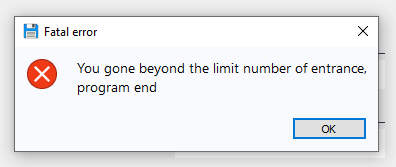


Рисунок 8 – Уведомление аварийного завершения работы

В случае успешного прохождения этапа авторизации перед пользователем появится окно, состоящее из двух вкладок (рисунок 9 и 10). На первом окне происходит вся работа в приложении: кнопки 1 и 2 отвечаю за ввод пользователем папки, пример ввода представлен на рисунке 11, а кнопка 3 отвечает за начало копирования в случае, если какие-либо из проверок не прошли, пользователь получит уведомление (рисунок 12) с отчетом о введенных данных. Кнопка 4 отвечает за сворачивание области слева в одно полоску с иконками. В окне настроек (рисунок 10) пользователю доступны: вводы временного интервала (кнопка 1), фильтр файлов, представленный в виде множества кнопок (указатель 2), а также режимы копирования (указатель 3). Блок применения настроек, состоящий из поля 4, разрешающий работу в свернутом режиме «в трее» и кнопка 5 применения изменений. Также пользователь может наблюдать иконку приложения «в трее» (рисунок 13). Чтобы перейти на нужную вкладку пользователь может воспользоваться полем 5 и кнопкой 6 на домашнем экране (рисунок 9), в случае неверного ввода будет высвечено окно ошибки (рисунок 14).

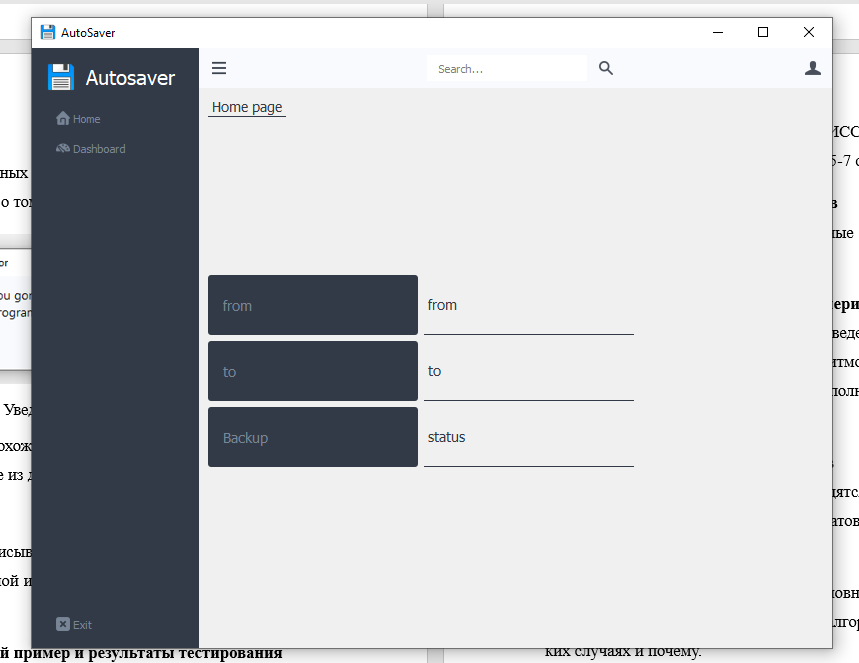


Рисунок 9 – домашнее окно программы

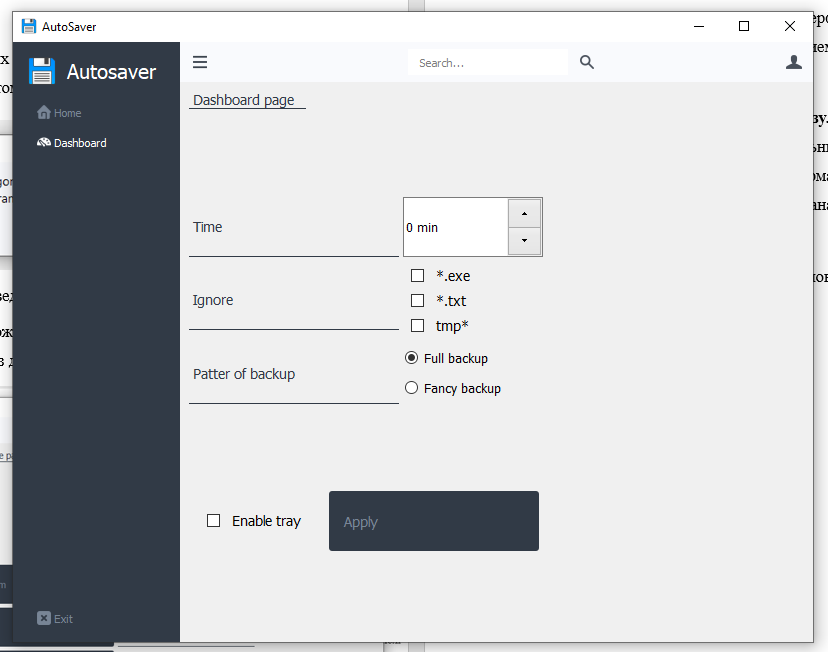


Рисунок 10 – Окно настроек программы

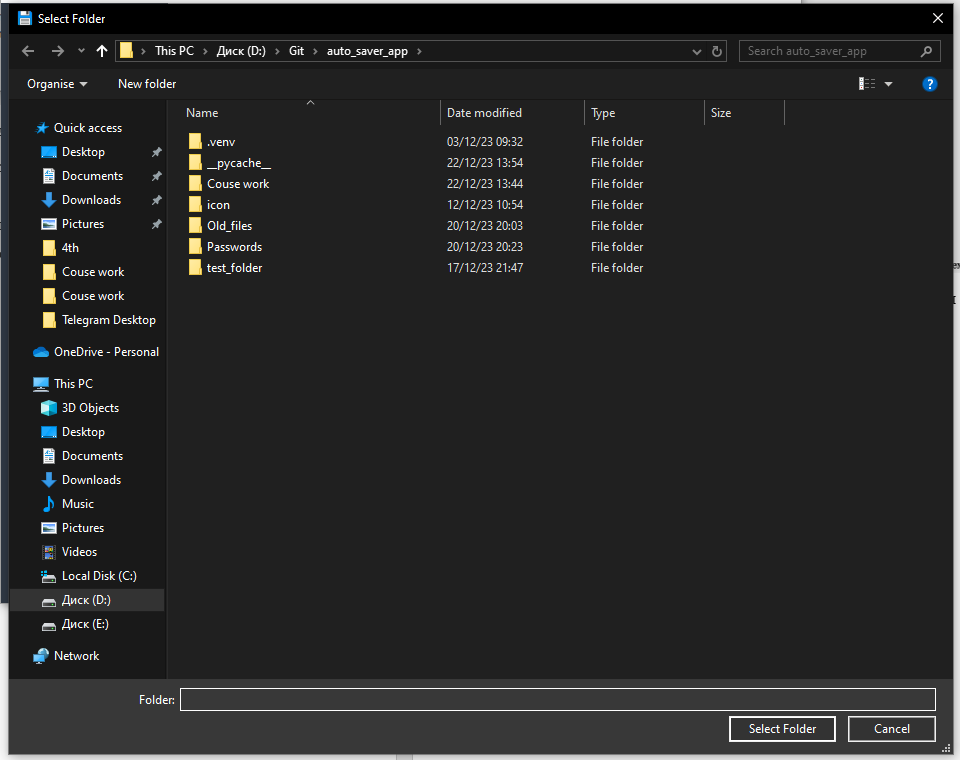


Рисунок 11 – Ввод пути

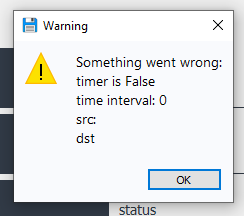


Рисунок 12 – Уведомление об ошибке ввода

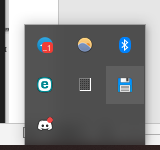


Рисунок 13 – Работа «в трее»

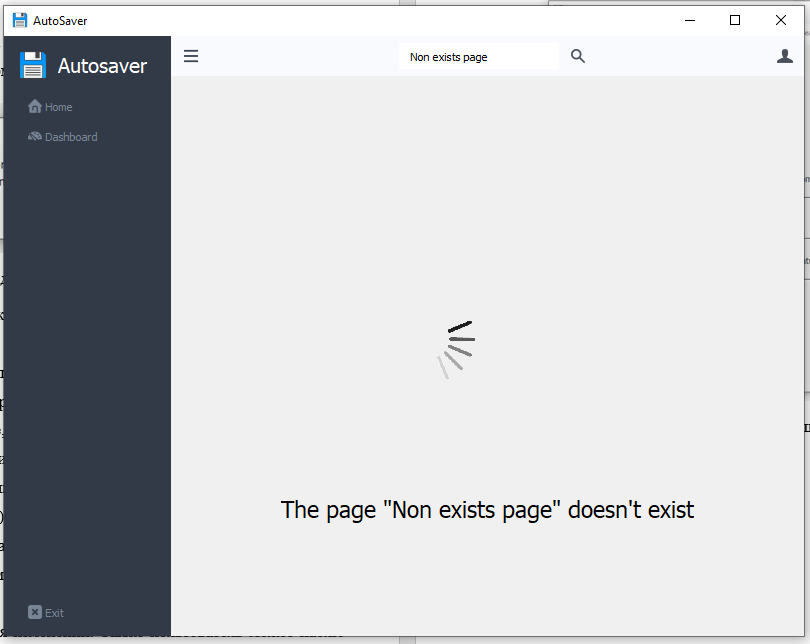


Рисунок 14 – Страница неправильного поиска

Когда пользователь совершит первое копирование окно 1 изменится: в области 1, 2 и 3 будет записана соответствующая информация, а также появится кнопка восстановления (кнопка 4), при нажатии на которую высветится меню восстановления (рисунок 16):

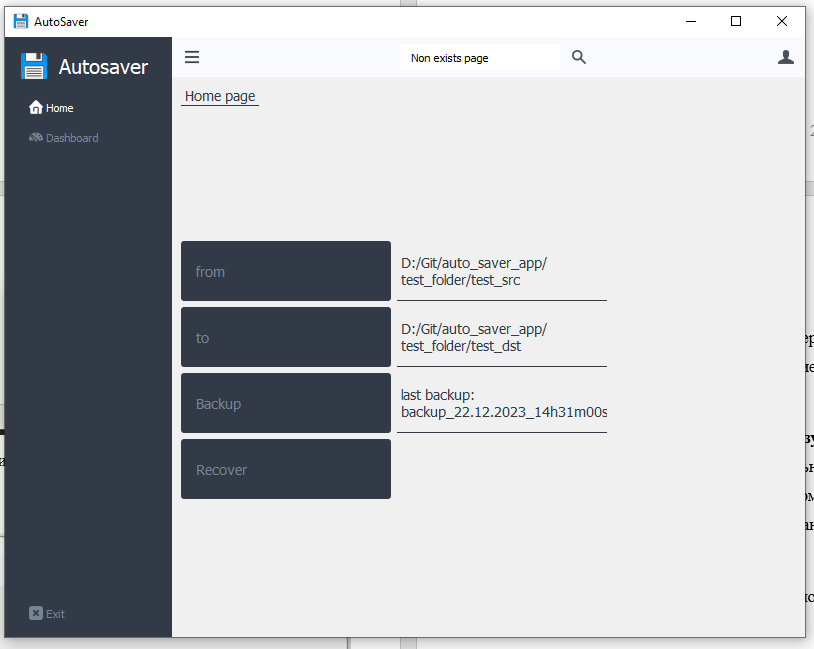


Рисунок 15 – Окно «после копирования»

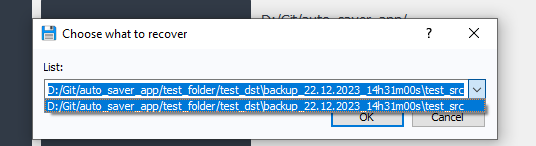


Рисунок 15 – Окно восстановления

## Контрольный пример и результаты тестирования

По мере разработки и тестирования были внедрены использованы такие методы отладки как логирование и функция дебагера в среде разработки Visual Studio Code.

## Выводы и результаты

С использованием документации и существующих решений было разработано и создано приложение, выполняющее функции автоматического резервного копирования, программа полностью выполняет поставленные перед ней задачи, и учтены все требования.

ЗАКЛЮЧЕНИЕ

В ходе реализации проекта были получены знания в области резервного копирования, изучены теоретические основы в области резервного копирования и поставлены первичные цели и задачи проекта. Далее был проанализированы уже существующие решения, по итогу чего стали окончательно понятны требования к создаваемому продукту.

Далее был сформулированы конкретные требования для создаваемой программой. После чего были выбраны технологии и методы, а необходимые инструменты для осуществления идей. Были рассмотрены существующие библиотеки и выбраны только самые надежные и проверенные решения.

По итогу работы имеет полностью рабочее и пригодное для использования приложение. Программа полностью удовлетворяет требованиям, которые были сформированы в начале проекта: простой и понятный интерфейс дополняется богатым функционалом, способствует положительному пользовательскому опыту.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Резервное копирование // Виды резервного копирования URL: https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B7%D0%B5%D1%80%D0%B2%D0%BD%D0%BE%D0%B5\_%D0%BA%D0%BE%D0%BF%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5 (Дата обращения: 18.12.2023).

2 List of backup software // List of backup software URL: https://en.wikipedia.org/wiki/List\_of\_backup\_software (Дата обращения:  18.12.2023).

3 Acronis Cyber Protect Home Office // Acronis Cyber Protect Home Office URL: https://ru.wikipedia.org/wiki/Acronis\_Cyber\_Protect\_Home\_Office (Дата обращения:  18.12.2023).

4 Time Machine (macOS) // Time Machine (macOS) URL: https://en.wikipedia.org/wiki/Time\_Machine\_(macOS) (Дата обращения:  18.12.2023).

5 Backup and Restore // Backup and Restore URL: https://en.wikipedia.org/wiki/Backup\_and\_Restore (Дата обращения:  18.12.2023).

6 filecmp – File and Directory Comparisonse // filecmp – File and Directory Comparisonse URL: https://docs.python.org/3/library/filecmp.html (Дата обращения:  18.12.2023).

7 os — Miscellaneous operating system interfaces // os — Miscellaneous operating system interfacese URL: https://docs.python.org/3/library/os.html (Дата обращения:  18.12.2023).

8 shutil — High-level file operations //shutil — High-level file operations URL:https://docs.python.org/3/library/shutil.html (Дата обращения:  18.12.2023).

ПРИЛОЖЕНИЕ А

Код программы

import sys

import typing

from PyQt5 import QtCore

from PyQt5.QtWidgets import QMainWindow, QApplication, QPushButton, QWidget

from PyQt5.QtCore import pyqtSlot, QFile, QTextStream

from PyQt5.QtGui import QMovie, QIcon

from app import Ui\_MainWindow

import typing

from PyQt5 import QtCore

from PyQt5.QtCore import QDir

from PyQt5.QtWidgets import QMainWindow, QWidget, QApplication, QShortcut, qApp, QMenu, QAction, QInputDialog, QLineEdit, QDesktopWidget, QDialog, QFileDialog, QMessageBox, QPushButton, QLabel, QVBoxLayout, QHBoxLayout, QSpinBox, QTimeEdit, QRadioButton, QCheckBox, QStatusBar, QSystemTrayIcon

from PyQt5.QtGui import QKeySequence

from PyQt5.QtCore import QTimer

from PyQt5 import uic

from functools import partial

import time

import os

import sys

from utils import InterfaceFileOperation

class MainWindow(QMainWindow):

def \_\_init\_\_(self) -> None:

super(MainWindow, self).\_\_init\_\_()

self.ui = Ui\_MainWindow()

self.ui.setupUi(self)

self.ui.icon\_only\_widget.hide()

self.ui.recovery\_button.setVisible(False)

self.ui.stackedWidget.setCurrentIndex(2)

self.ui.search\_btn.setVisible(False)

self.init\_UI()

# Function for searching

def on\_search\_btn\_clicked(self):

self.ui.stackedWidget.setCurrentIndex(3)

search\_text = self.ui.search\_input.text().strip()

if search\_text == "Home" or search\_text == "home":

self.ui.stackedWidget.setCurrentIndex(0)

return

if search\_text == "dashboard" or search\_text == "Dashboard":

self.ui.stackedWidget.setCurrentIndex(1)

return

if search\_text == "user page" or search\_text == "user":

self.ui.stackedWidget.setCurrentIndex(2)

return

if search\_text:

self.movie = QMovie("icon/2loading\_gif.gif")

self.ui.photo\_lable.setMovie(self.movie)

self.movie.start()

self.ui.label\_6.setText(

f"The page \"{search\_text}\" doesn't exist")

# Function for changing page to user page

def on\_user\_btn\_clicked(self):

print("on user")

self.ui.stackedWidget.setCurrentIndex(2)

# Change QPushButton Checkable status when stackedWidget index changed

def on\_stackedWidget\_currentChanged(self, index):

btn\_list = self.ui.icon\_only\_widget.findChildren(QPushButton) \

+ self.ui.full\_menu\_widget.findChildren(QPushButton)

for btn in btn\_list:

if index in [3, 3]:

btn.setAutoExclusive(False)

btn.setChecked(False)

else:

btn.setAutoExclusive(True)

# functions for changing menu page

def on\_home\_btn\_1\_toggled(self):

self.ui.stackedWidget.setCurrentIndex(0)

self.ui.stackedWidget.widget(0).setVisible(self.check)

def on\_home\_btn\_2\_toggled(self):

self.ui.stackedWidget.setCurrentIndex(0)

self.ui.stackedWidget.widget(0).setVisible(self.check)

def on\_dashboard\_btn\_1\_toggled(self):

self.ui.stackedWidget.setCurrentIndex(1)

self.ui.stackedWidget.widget(1).setVisible(self.check)

def on\_dashboard\_btn\_2\_toggled(self):

self.ui.stackedWidget.setCurrentIndex(1)

self.ui.stackedWidget.widget(1).setVisible(self.check)

def login(self):

login = self.ui.login\_line.text()

pas = self.ui.password\_line.text()

# if True:

if self.external.check\_user(login, pas):

# if True:

self.ui.stackedWidget.setCurrentIndex(0)

self.check = True

self.ui.search\_btn.setVisible(True)

self.ui.login\_button.setVisible(False)

else:

QMessageBox.warning(

self, "Warning", f"Wrong login or password", QMessageBox.Ok)

if self.count\_login > 4:

QMessageBox.critical(

self, "Fatal error", f"You gone beyond the limit number of entrance, \nprogram end", QMessageBox.Ok)

print(self.count\_login)

qApp.quit()

self.count\_login += 1

def init\_UI(self) -> None:

# tray

self.tray\_icon = QSystemTrayIcon(QIcon("icon/Logo.png"))

show\_action = QAction("Show", self)

quit\_action = QAction("Exit", self)

hide\_action = QAction("Hide", self)

self.backup\_action = QAction("backup", self)

show\_action.triggered.connect(self.show)

hide\_action.triggered.connect(self.hide)

quit\_action.triggered.connect(qApp.quit)

self.backup\_action.triggered.connect(self.setup\_backup)

tray\_menu = QMenu()

tray\_menu.addAction(show\_action)

tray\_menu.addAction(hide\_action)

tray\_menu.addAction(quit\_action)

tray\_menu.addAction(self.backup\_action)

self.tray\_icon.setContextMenu(tray\_menu)

self.tray\_icon.show()

self.dst = ""

self.src = ""

self.dst\_list = []

self.shortcut\_exit = QShortcut(QKeySequence('ctrl+q'), self)

self.shortcut\_exit.activated.connect(qApp.quit)

self.ui.from\_button.clicked.connect(

partial(self.\_\_input\_path, self.ui.from\_lable))

self.ui.to\_button.clicked.connect(

partial(self.\_\_input\_path, self.ui.to\_lable))

# self.ui.time\_holder = self.findChild(QTimeEdit, "time\_holder")

self.ui.time\_holder = self.findChild(QSpinBox, "timer\_holder")

self.ui.timer\_holder.setSuffix(" min")

# self.ui.time\_holder.setDisplayFormat("m (n)")

self.ui.time\_holder.setRange(0, 10)

self.ui.timer\_holder.setValue(0)

self.time\_for\_timer = 0

self.ui.time\_button.clicked.connect(self.update\_time)

self.check\_boxes = []

self.check\_boxes = self.ui.stackedWidget.findChildren(QCheckBox)

self.external = InterfaceFileOperation()

self.ui.tmp\_button.clicked.connect(self.setup\_backup)

self.setWindowTitle("AutoSaver")

self.setWindowIcon(QIcon("icon/logo.png"))

self.ui.recovery\_button.clicked.connect(self.setup\_recover)

self.count\_fancy\_backup = 0

self.repeat = 2

self.timer = QTimer()

self.timer.setInterval(1000)

self.ignore\_pattern = []

self.check = False

self.ui.password\_line.setEchoMode(QLineEdit.Password)

self.ui.login\_button.clicked.connect(self.login)

a = QMessageBox.about(self, "Loading...",

"Please wait, loading data")

self.external.load\_login()

self.count\_login = 0

self.show()

def \_\_clean\_array(self):

for i in self.dst\_list:

if not os.path.exists(i):

self.dst\_list.remove(i)

def \_\_zip\_array(self):

if len(self.dst\_list) >= 2:

for i in range(len(self.dst\_list)-1):

self.dst\_list[i] = self.external.ziping(self.dst\_list[i])

def \_\_backup\_\_(self):

if not self.external.full\_backup(self.src, self.dst\_list[-1], self.ignore\_pattern):

QMessageBox.warning(

self, "Warning", f"Something went wrong: \nbackup failed", QMessageBox.Ok)

if self.timer.isActive():

self.timer.stop()

self.dst\_list.remove(self.dst\_list[-1])

return False

return True

def backup(self):

# add backup option

if not self.external.is\_dir(self.src):

QMessageBox.warning(

self, "Warning", f"Something went wrong: \nsrc - {self.src} doesnt't exist", QMessageBox.Ok)

if self.timer.isActive():

self.timer.stop()

return

if not self.dst\_list and self.external.is\_dir(self.src):

self.dst\_list.append(self.external.create\_name(self.src, self.dst))

self.\_\_backup\_\_()

self.ui.stats\_lable.setText(

f"last backup: {os.path.basename(os.path.split(self.dst\_list[-1])[0])}")

return

if not self.external.cmp\_folder(self.src, self.dst\_list[-1], self.ignore\_pattern):

self.dst\_list.append(self.external.create\_name(self.src, self.dst))

self.\_\_backup\_\_()

self.ui.stats\_lable.setText(

f"last backup: {os.path.basename(os.path.split(self.dst\_list[-1])[0])}")

else:

print("equal")

self.ui.stats\_lable.setText(

f"last backup: {os.path.basename(os.path.split(self.dst\_list[-1])[0])}")

self.\_\_clean\_array()

self.\_\_zip\_array()

print("\_timer\_end\_")

def fancy\_backup(self):

# add backup option

if not self.external.is\_dir(self.src):

QMessageBox.warning(

self, "Warning", f"Something went wrong: \nsrc - {self.src} doesnt't exist", QMessageBox.Ok)

if self.timer.isActive():

self.timer.stop()

return

if not self.dst\_list and self.external.is\_dir(self.src):

self.dst\_list.append(self.external.create\_name(self.src, self.dst))

self.\_\_backup\_\_()

self.ui.stats\_lable.setText(

f"last backup: {os.path.basename(os.path.split(self.dst\_list[-1])[0])}")

return

if not self.external.cmp\_folder(self.src, self.dst\_list[-1], self.ignore\_pattern):

tmp\_name = ""

if self.count\_fancy\_backup > self.repeat:

self.dst\_list.append(

self.external.create\_name(self.src, self.dst))

self.count\_fancy\_backup = 0

self.\_\_backup\_\_()

else:

tmp = self.external.create\_name(self.src, self.dst, upd=True)

tmp\_name = self.dst[-1]

self.external.rename\_folder(tmp, self.dst\_list[-1])

self.dst\_list[-1] = tmp

self.count\_fancy\_backup += 1

if not self.\_\_backup\_\_():

self.dst\_list.append(tmp\_name)

self.ui.stats\_lable.setText(

f"last backup: {os.path.basename(os.path.split(self.dst\_list[-1])[0])}")

else:

print("equal")

self.\_\_clean\_array()

self.\_\_zip\_array()

print("\_timer\_end\_")

def update\_time(self):

# value = self.ui.time\_holder.time().toPyTime()

value = self.ui.timer\_holder.value()

self.ui.recovery\_button.setVisible(False)

self.time\_for\_timer = value\*60\*60 # tmp

def setup\_recover(self):

if self.dst\_list:

return\_recover\_src, ok = QInputDialog.getItem(

self, 'Choose what to recover', 'List:', self.dst\_list)

if ok and return\_recover\_src:

check\_return = self.external.recover(

return\_recover\_src, self.src)

if check\_return:

self.dst\_list.remove(return\_recover\_src)

self.dst\_list.append(check\_return)

self.\_\_clean\_array()

self.\_\_zip\_array()

self.ui.recovery\_button.setVisible(False)

else:

QMessageBox.warning(

self, "Warning", f"Something went wrong:\n path status exists {return\_recover\_src} - {self.external.is\_dir(return\_recover\_src)} \n{self.src} - {self.external.is\_dir(self.src)} ", QMessageBox.Ok)

else:

QMessageBox.about(self, "No recover option",

"There's nothing to recover")

def setup\_backup(self):

if self.timer.isActive():

self.timer.stop()

self.ui.tmp\_button.setText("Start")

self.backup\_action.setText("Start backup")

self.ui.recovery\_button.setVisible(True)

else:

self.ignore\_pattern.clear()

for i in self.check\_boxes:

if i.isChecked():

self.ignore\_pattern.append(i.text())

if self.time\_for\_timer >= 0 and self.external.is\_dir(self.ui.to\_lable.text()) and self.external.is\_dir(self.ui.from\_lable.text()):

self.src = self.ui.from\_lable.text()

self.dst = self.ui.to\_lable.text()

if not (self.src in self.dst or self.dst in self.src):

if self.time\_for\_timer == 0:

if self.ui.backup\_option\_1.isChecked():

self.backup()

if self.ui.backup\_option\_2.isChecked():

self.fancy\_backup()

self.ui.recovery\_button.setVisible(True)

else:

if self.ui.backup\_option\_1.isChecked():

self.timer.timeout.connect(self.backup)

if self.ui.backup\_option\_2.isChecked():

self.timer.timeout.connect(self.fancy\_backup)

self.timer.setInterval(self.time\_for\_timer)

self.timer.start()

print(f"time {self.time\_for\_timer} src {

self.src} dst {self.dst}")

self.ui.tmp\_button.setText("Stop")

self.backup\_action.setText("Stop")

else:

print(

f"(self.src in self.dst - {self.src in self.dst} or self.dst in self.src - {self.dst in self.src})")

QMessageBox.warning(self, "Warning", f"(self.src in self.dst - {

self.src in self.dst} or self.dst in self.src - {self.dst in self.src})", QMessageBox.Ok)

self.ui.stats\_lable.setText("Error")

else:

print('sad')

QMessageBox.warning(

self, "Warning", f"Something went wrong: \ntimer is {self.timer.isActive()} \ntime interval: {self.time\_for\_timer} \nsrc: {self.src} \ndst {self.dst}", QMessageBox.Ok)

print(

f"timer is {self.timer.isActive()} and it's because: {self.time\_for\_timer} or src {self.src} or dst {self.dst}")

def \_\_input\_path(self, label: QLabel) -> None:

"""service function"""

tmp = QFileDialog.getExistingDirectory(self, 'Select Folder')

if not tmp:

QMessageBox.warning(

self, "Warning", "please select dir", QMessageBox.Ok)

else:

label.setText(tmp)

# test close event

def closeEvent(self, event):

if self.ui.tray\_check\_box.isChecked():

event.ignore()

self.hide()

self.tray\_icon.showMessage(

"AutoSaver",

"Application was minimized to Tray",

QSystemTrayIcon.Information,

30

)

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

style\_file = QFile("style.qss")

style\_file.open(QFile.ReadOnly | QFile.Text)

style\_stream = QTextStream(style\_file)

app.setStyleSheet(style\_stream.readAll())

window = MainWindow()

window.show()

sys.exit(app.exec())

Код файла main.py

import os

import shutil

import glob

import logging

import filecmp

from datetime import datetime

import zipfile

import hashlib

from github import Github

import logging

class InterfaceFileOperation:

def \_\_init\_\_(self) -> None:

"""constructor

input: source of files

destination(where to save)

time(how often need to check updates)

"""

self.name = "c0de1sl1fe"

self.hash = ""

self.logger = self.setup\_logging()

def setup\_logging(self) -> None:

"""setup logging"""

logger = logging.getLogger("logger")

if not logger.handlers:

logger.setLevel(logging.INFO)

file\_handler = logging.FileHandler("autosaver.log")

file\_handler.setLevel(logging.DEBUG)

formatter = logging.Formatter(

"%(asctime)s - %(levelname)s - %(message)s")

file\_handler.setFormatter(formatter)

logger.addHandler(file\_handler)

return logger

def load\_login(self):

"""fucntion for load hash from file on github"""

try:

g = Github()

user = g.get\_user(self.name)

for content in user.get\_repo("auto\_saver\_app").get\_contents("Passwords"):

if content.path.endswith(".txt"):

self.hash = content.decoded\_content.decode("utf-8")

except Exception as e:

self.logger.warning(f"Raised exception: {e} while load login")

def check\_user(self, login: str, password: str) -> bool:

"""Function to check users loging and password

@login: your login

@Password: your password

inside this function app requests file with hashed password

and then compart it with users password

"""

my\_hash = hashlib.md5((login+password).encode('utf-8')).hexdigest()

if self.hash:

return my\_hash == self.hash

try:

g = Github()

user = g.get\_user(self.name)

hash = ""

for content in user.get\_repo("auto\_saver\_app").get\_contents("Passwords"):

if content.path.endswith(".txt"):

hash = content.decoded\_content.decode("utf-8")

return my\_hash == hash

except Exception as e:

self.logger.warning(f"Raised exception: {e} login")

return False

def create\_name(self, src: str, dst: str, upd=False) -> str:

"""create name with time of backup

@param src - is source of folder

@param dst - is destination for folder

return dst + name of folder src + str(backup\_ + time)

"""

now = datetime.now()

tmp = ""

if upd:

tmp = os.path.join(

dst, "backup\_upd\_" + str(now.strftime("%d.%m.%Y\_%Hh%Mm%Ss")))

else:

tmp = os.path.join(

dst, "backup\_" + str(now.strftime("%d.%m.%Y\_%Hh%Mm%Ss")))

tmp = os.path.join(tmp, os.path.basename(src))

self.logger.info(f"New name was created [{tmp}]")

return tmp

def ziping(self, src) -> str:

"""

@path - path to folder you want to archive

this function destroy existing path and return zip

return path to new archive or if path already leads to archive or doesn't exist return itself

"""

if not os.path.exists(src) or zipfile.is\_zipfile(src):

return src

new\_path = src

try:

# new\_path = shutil.make\_archive(os.path.join(path, f"{folder\_name} zipped"), 'zip', src)

new\_path = shutil.make\_archive(src, "zip", src)

self.logger.info(f"Create archive - {new\_path}")

except Exception as e:

self.logger.warning(f"Raised exception: {e} while archive")

new\_path = src

try:

shutil.rmtree(src)

self.logger.info(f"Complete removing tree - {src}")

except Exception as e:

self.logger.warning(f"Raised exception: {e} while removing dir")

return new\_path

def rename\_folder(self, new: str, old: str) -> bool:

try:

shutil.move(old, new)

self.logger.info(f"Folder {old} renamed to {new}")

except Exception as e:

self.logger.warning(f"Raised exception: {e} while rename")

return False

if os.path.exists(os.path.split(old)[0]):

try:

shutil.rmtree(os.path.split(old)[0])

return True

except Exception as e:

self.logger.warning(f"Raised exception: {e} while rename")

return False

return True

def is\_dir(self, path) -> bool:

"""

Check is folder or file exist

input: src: string

return: bool

"""

tmp = os.path.isdir(path)

self.logger.info(f"Complete is\_dir({tmp}) for {path}")

return tmp

def recover(self, src: str, dst: str) -> str:

if not os.path.exists(src) and not os.path.exists(dst):

self.logger.info(f"Raised exception because of src: {

src} or dst: {dst} doesn't exist")

return ""

tmp = src

if zipfile.is\_zipfile(src):

try:

tmp = src.replace(".zip", "")

shutil.unpack\_archive(src, tmp)

self.logger.info(f"Complete unpack archive {src} for recover")

except Exception as e:

self.logger.warning(f"Raised exception: {

e} while unpack archive {src} in recover")

try:

os.remove(src)

self.logger.info(f"complete remove {src} for recover")

except Exception as e:

self.logger.warning(f"Raised exception: {e} while remove {src}")

self.full\_backup(tmp, dst)

return tmp

def full\_backup(self, src: str, dst: str, ignore=[]) -> bool:

"""

Create a copy of scr into dst

input: @src: string

@dst: string

@ingnore: list of ignored files

return: bool

"""

if not os.path.exists(src):

self.logger.warning(f"{src} doesn't exist while full\_backup")

return False

try:

if ignore:

shutil.copytree(src, dst, ignore=shutil.ignore\_patterns(

\*ignore), dirs\_exist\_ok=True)

self.logger.info(f"complete copy {src} to {

dst} with ignore\_pattern")

else:

shutil.copytree(src, dst, dirs\_exist\_ok=True)

self.logger.info(f"complete copy {src} to {

dst} without ignore\_pattern")

return True

except Exception as e:

self.logger.warning(f"Raised exeption: {

e} while copy {src} to {dst}")

return False

def cmp\_folder(self, src: str, dst: str, ignore=[]) -> bool:

"""

Compare two directories recursively. Files in each directory are

assumed to be equal if their names and contents are equal.

@param src: First directory path

@param dst: Second directory path

@param ignore: list of patter to ignore

@return: True if the directory trees are the same and

there were no errors while accessing the directories or files,

False otherwise.

"""

if not os.path.exists(src) or not os.path.exists(dst):

self.logger.warning(f"cmp\_folder for {src} and {

dst} doens't complete")

return False

ignore\_list = []

if not ignore:

for pattern in ignore:

ignore\_left = [os.path.split(expanded)[1] for expanded in glob.glob(

os.path.join(src, pattern))]

ignore\_right = [os.path.split(expanded)[1] for expanded in glob.glob(

os.path.join(dst, pattern))]

ignore\_list.extend(ignore\_left)

ignore\_list.extend(ignore\_right)

self.dirs\_cmp = filecmp.dircmp(src, dst, ignore=ignore\_list)

else:

self.dirs\_cmp = filecmp.dircmp(src, dst)

if len(self.dirs\_cmp.left\_only) > 0 or len(self.dirs\_cmp.right\_only) > 0 or \

len(self.dirs\_cmp.funny\_files) > 0:

return False

(\_, mismatch, errors) = filecmp.cmpfiles(

src, dst, self.dirs\_cmp.common\_files, shallow=False)

if len(mismatch) > 0 or len(errors) > 0:

return False

for common\_dir in self.dirs\_cmp.common\_dirs:

new\_dir1 = os.path.join(src, common\_dir)

new\_dir2 = os.path.join(dst, common\_dir)

if not self.cmp\_folder(new\_dir1, new\_dir2):

return False

return True

if \_\_name\_\_ == '\_\_main\_\_':

a = InterfaceFileOperation()

Код файла utils.py