МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ УПРАВЛІННЯ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ Литвиненко О.Є.
«_____»_____2020 р.

ДИПЛОМНИЙ ПРОЕКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА
ЗА СПЕЦІАЛЬНІСТЮ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

Тема: «Програмний модуль автоматизованого налаштування директорій»

Виконавець: СП-425Б, Рабін Ігор Романович

Керівник: к.т.н. доцент Нечипорук Віталій Володимирович

Нормоконтролер: Тупота Євгеній Вікторович

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Напрям (спеціальність) 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ О.Є.Литвиненко

«_____» _____2020 р.

ЗАВДАННЯ

на виконання дипломного проекту

Рабіна Ігора Романовича
(П.І.Б. випускника)

1. Тема роботи: «Програмний модуль автоматизованого налаштування директорій»

затверджена наказом ректора від «06» лютого 2020 року №119/ст

2. Термін виконання проекту (роботи): з 18.05.2020 по 21.06.2020

3. Вихідні дані до проекту: Модуль автоматизації очистки робочого столу *DCleaner*, модулі *VCacheClean* і *TCacheClean*.

4. Зміст пояснювальної записки:
1) постановка завдання проектування та аналіз предметної області;
2) розробка діаграм та структури програмного модуля;
3) розробка алгоритмів, основних функцій та інтерфейсу.

5. Перелік обов'язкового ілюстративного матеріалу:
1) діаграма прецедентів;
2) діаграма кооперації;
3) програмний модуль (структурна схема);
4) головна функція (схема алгоритму);
5) інтерфейс програмного модуля.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Проаналізувати джерела, по темі дипломного проекту	18.05.20 – 19.05.20	
2	Проаналізувати тему автоматизації	20.05.20	
3	Проаналізувати засоби та методи автоматизації систем	21.05.20 – 22.05.20	
4	Провести аналіз аналогів-модулів автоматизації систем	23.05.20 – 24.05.20	
5	Розробити програмний модуль для очистки робочого стола	25.05.20 – 27.05.20	
6	Оцінити ефективність функціонування та додати ще декілька модулів для очистки кешу Telegram та Viber	28.05.20	
7	Розробити структуру та зміст пояснювальної записки	29.05.20 – 30.05.20	
8	Оформити пояснювальну записку	31.05.20 – 01.06.20	
9	Розробити та оформити презентаційні матеріали до пояснювальної записки	02.06.20 – 04.06.20	

7. Дата видачі завдання: 18.05.2020 р.

Керівник дипломного проекту: _____ Нечипорук В.В.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання: _____ Рабін І.Р
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Програмний модуль автоматизованого налаштування директорій», 41 сторінка, 15 рисунків, 14 літературних джерел, 1 додаток.

PYTHON, PYQT, INTERFACE, MODULE.

Об’єкт дослідження – процес автоматизації налаштування директорій.

Предмет дослідження – програмний модуль для автоматизованого налаштування директорій.

Мета дипломного проекту – розробити програмний модуль автоматизованого налаштування директорій для ОС Windows.

Методи дослідження – методи автоматизації процесів на персональному комп’ютері, влаштовані методи/бібліотеки мови програмування *Python*, застосування платформи *PyQT Designer*.

Рекомендації щодо використання результатів дипломного проекту – об’єкт та концепція проекту може бути використана в подальшому для адаптування програмного модуля під інші операційні системи.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ПК – персональний комп’ютер;

Дедлайн(*deadline*) – остаточний термін, дата здачі проекту;

ОС – операційна система;

ІС – інформаційна система;

ПЗ – програмне забезпечення;

GUI – графічний інтерфейс користувача;

DCleaner – модуль очистки робочого столу;

TCleaner – модуль очистки кешу *Telegram*;

VCleaner – модуль очистки кешу *Viber*.

ВСТУП

Щодня користувачі виконують дуже багато робіт, такі як: завантаження файлів та архівація данних, запуск деяких програм.

Навіть до теперішнього часу люди вручну удаляють старі файли, перевіряють свої жорсткі диски, інколи така робота займає не малий період часу.

У нинішній час автоматизація дуже обширна та актуальна тема, на яку треба звернути увагу, якщо Ви користувач, якому потрібно найбільш продуктивно проводити час за комп'ютером. Особливо, якщо ви працюєте, та вам не потрібно відволікатися на несуттєві програмні дії, щоб уникнути дальнішої неуспішності роботи щодо попередніх результатів.

Навіть дивлячись на всі ті плюси автоматизації, як кількість часу, яку Ви витрачаєте на рутинні дії, збільшення продуктивності роботи за ПК, є і мінуси автоматизації, які також можуть вплинути на деяких людей негативно.

Якщо в теорії автоматизувати всі процеси на комп'ютері, то прості користувачі не будуть знати функціоналу системи, що дуже погано, якщо буде потрібно вручну щось зробити в операційній системі без допоміжних модулів. Якщо ви працюєте бистріше за допомогою автоматизації та виконаєте роботу до дедлайну, на роботі вам можуть додати роботи, що не дуже приємно для вас, коли Ви хотіли виконати роботу та відпочити.

За останні років 20 автоматизація сильно розвинулася і тримає баланс між недоліками та перевагами.

Якщо Ви простий користувач, наприклад, на ОС «Windows» ви не зможете автоматизувати всі процеси, операційна система просто не дасть вам це зробити без злому системи та других утиліт, які могуть «влізти» в систему та отримати доступ до тих функцій, які були недоступні для простого користувача. З однієї сторони, це добре, що простому користувачу закривають доступ, щоб він не вліз, куди не треба. З другої – це погано, що користувачу не дають ті функції, які йому

потрібні. На других ОС, таких як «Linux», користувач може налаштувати все, що йому потрібно, без втручання програм, за належним знанням внутрішніх функцій.

Метою розробки власного программного модуля для автоматизованого налаштування директорій та очистки робочого столу були незручність і зайвий функціонал існуючих на цей момент подібних сервісів.

Цей модуль буде дуже корисним та актуальним для користувачів, які будуть їм користуватися.

Усе вище наведене дозволяє зробити висновок, що для підвищення ефективності роботи користувача в системі, слід автоматизовувати процеси взаємодії з системою, шляхом впровадження програмного модуля.

Об’єкт дослідження – процес автоматизації налаштування директорій.

Предмет дослідження – програмний модуль автоматизованого налаштування директорій.

Мета дипломного проекту – розробити програмний модуль автоматизованого налаштування директорій для ОС Windows.

Методи дослідження – методи автоматизації процесів на персональному комп’ютері, влаштовані методи/бібліотеки мови програмування *Python*, застосування платформи *PyQT Designer*.

Рекомендації щодо використання результатів дипломного проекту – об’єкт та концепція проекту може бути використана в подальшому для адаптування програмного модуля під інші операційні системи.

Про дальніший розвиток можна сказати, що цей проект буде мати модифікації, тому що є багато ідей, які хочеться проаналізувати та адаптувати у програмний модуль. Програми автоматизації роботи з системою вдосконалюються. Такі технології так і далі будуть в попиті, що з часом зробить їх ще краще.

РОЗДІЛ 1
ПОСТАНОВКА ЗАВДАННЯ ПРОЕКТУВАННЯ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз теми автоматизації процесів в ОС

На сьогодняшній день людям доводиться здійснювати багато дій. Наша повсякденність – зарядка, душ, сніданок, поїздки кудись. Багато людей розписує свій день по годинам чи навіть по хвилинам. Ви можете випадково забути про будь-що, або навмисне пропустити деякі звичайні дії.

Віртуальне життя комп’ютерних систем, на щастя, піддається автоматизації. Можна призначити якесь завдання на певний час, і воно обов'язково виконається. Шкода, що за допомогою будь-яких вищих сил не можна призначати автоматичну чистку зубів, гоління і інші, виснажливі процедури. Але те, що неможливо в реальному світі, вже давно реалізовано в комп'ютерах, в рамках спеціального класу програмного забезпечення, призначеного для автоматизації процесів.

Комп’ютеру можна доручити самостійне виконання самих різних операцій, таких як запуск додатків, перевірку і чистку системного реєстру оновлення антивірусних баз та інших потрібних даних, скачування файлів, перевірку, отримання і відправку електронної пошти, архівування даних, копіювання файлів, генеруванняроздрук документів і т.д. При цьому виконувати зазначені користувачем завдання ПК може в строго певні моменти: при кожному запуску ОС, при виключенні комп’ютера, щодня, щотижня і щомісяця в зазначені години і т.д., а також при виникненні деяких подій системи. Іншими словами, комп’ютер вміє виконувати багато дій самостійно, причому навіть у відсутності користувача – слід лише налаштувати ПК відповідним чином.

Автоматизація завдань, які виконуються на комп’ютері регулярно, дуже важлива з декількох точок зору.

По-перше, вона дозволить виконувати багато операції повсякденності майже або повністю без участі користувача і заощадить таким чином користувачу багато часу. ПК може на початку робочого дня автоматично запускати потрібні для роботи програми, самостійно відкривати деякі документи і папки, завантажувати пошту і т.п. В деяких випадках має сенс налаштувати комп'ютер на відстеження змін – файлів і каталогів на диску (зокрема, при змінах в налаштованому каталозі можна синхронізувати весь вміст директорії з іншою директорією) і деяких *Web*-сторінок (наприклад,посилати листом про наявність змін на контрольованих *Web*-сторінках) в Інтернеті. Особливий інтерес представляє автоматизація роботи за допомогою макросів, які містять послідовність записаних рухів, натискання миші і натискань клавіатури в будь-яких ОС-додатках і можуть бути відновлені у будь-який час за розкладом або при натисканні деякої комбінації клавіш. Таким чином, наприклад, можна швидко налаштовувати комп'ютер на конкретний вид діяльності, відкриваючи потрібні для цього програми або документи і виконуючи якісь часто повторювані операції.

По-друге, автоматизація деяких операцій, про необхідність регулярного проведення яких користувачі добре знають, але їх виконанням нехтують, зробить роботу за ПК набагато надійнішою і можна запобігти чимало стресових ситуацій. Приклади таких операцій – синхронізація, резервування, перевірка диска на наявність не дуже добрих компонентів, регулярна профілактична чистка диска і системного реєстру. І коли в массових компаніях за дані операції відповідають кваліфіковані компетентні адміністратори, то в маленьких офісах, а навіть і вдома користувачам доводиться самостійно налаштовувати та займатися вирішенням деяких завдань. Але, якщо довірити їх виконання комп’ютеру, то стабільність роботи значно збільшиться. Тоді, можна довірити комп’ютеру регулярно автоматично резервувати та навіть просто копіювати архів поштових повідомлень і папки з потрібними документами, наприклад раз в тиждень, за цей рахунок можливість їх втрати значно знизиться. Якщо налаштувати систему на

автоматичну комп'ютерну чистку, то на диску не буде непотрібних зайвих файлів, в реєстрі – неактуальних даних, а швидкість роботи комп’ютера буде стабільною.

1.2. Способи вирішення проблем автоматизації

Існує три способи вирішення проблеми автоматизації.

Першим способом буде обрати для виконання цільових завдань, що повторюються, ті додатки, у яких присутній в своєму складі внутрішній планувальник, що дозволить проводити певні дії за датою та часом. Вбудованим планувальником оснащені більшість додатків для резервування та синхронізації персональних даних, також для перевірки жорсткого диска на наявність вірусів чи шпигунських програм. Дуже часто виконання завдань за датою та часом є і у додатків для чистки диска від пошкоджених і непотрібних файлів та ліквідації старих записів в системному реєстрі. На роботу за датою та часом не дуже важко налаштувати завантажувачі, котрі мають завантажувати потрібні файли в автоматичному режимі, самостійно підключившись до Інтернету, отримати дані, а потім відключитись та вимкнути комп’ютер.

Другим способом – скористатися стандартними можливостями ОС, де є вбудований планувальник, який дозволить автоматично вмикати програмні файли при вімкненні системи, або відповідно до якоїсь дати та часу. Це дозволить автоматизувати ті операції, які регулярно виконуються за часом, як резервування диска, чистка його від різного файлового сміття, тестування жорсткого диска на предмет наявності неприємних программ-хробаків або шпигунських програм. При цьому зовсім не потрібно обов’язково створювати свою ціль для кожної операці, можна заздалегідь підготувати пакетний файл з потрібною послідовністю команд, який в майбутньому буде вмикати планувальник задач в деякий користувачем час.

Третім способом буде використання программ – планувальників завдань від сторонніх виробників, що відкриє користувачу набагато більше фунцій. Ці програми дозволяють вам здійснювати в автоматичному режимі найрізніші дії – від відкриття потрібних документів, додатків, папок, і до проведення різних

операцій з файловою системою для синхронізації директорій, архівування персональних даних, для відтворення деяких послідовностей натиснення клавіш і дій з мишею, завантаження файлів, та навіть для вимкнення ПК. Для виконання всіх завдань в автоматичному режимі зовсім не потрібно навичок програмування та вміння створювати пакетні файли. За допомогою включених в фоновому режимі планувальників можна автоматизувати процеси на візуальному рівні, просто вказавши системі на те, що і коли треба виконати та відразу визначивши параметри їх виконання. Планувальник допомагає набагато бистріше, надійніше та зручніше налаштувати систему на автоматичне виконання деяких завдань і заощадити користувачам багато часу.

1.3. Сучасні програми для автоматизації процесів

Сучасні програми для автоматизації мають багато цікавих концепцій та функцій.

Одні програми могуть автоматизувати кліки вашої миші за координатами по *X* та *Y*, інші – запускати програми або функції, які будуть налаштовувати ваш ПК для приємної роботи.

Наприклад така программа, як *AutoHotkey*, з її допомогою можна налаштувати автоматичний запуск необхідних програм і автоматизувати виконання необхідних вам операцій. Також, використовуючи *AutoHotkey*, можна налаштувати гарячі клавіші для клавіатури, мишки, сенсорних пристроїв, джойстиків і інших пристроїв введення.

Корисні функції програми на створенні поєднання гарячих клавіш не закінчується. З її допомогою можна створювати цілі макроси, в яких ви можете вказувати власні сценарії, які можуть автоматично виконувати найрізноманітніші завдання, в тому числі і найскладніші. Програма являє собою пакет, який запускає *.ahk* файли. В інтерфейсі користувача дана інструкція, як ці файли створювати, сам інтерфейс програми показаний на рис. 1.1.

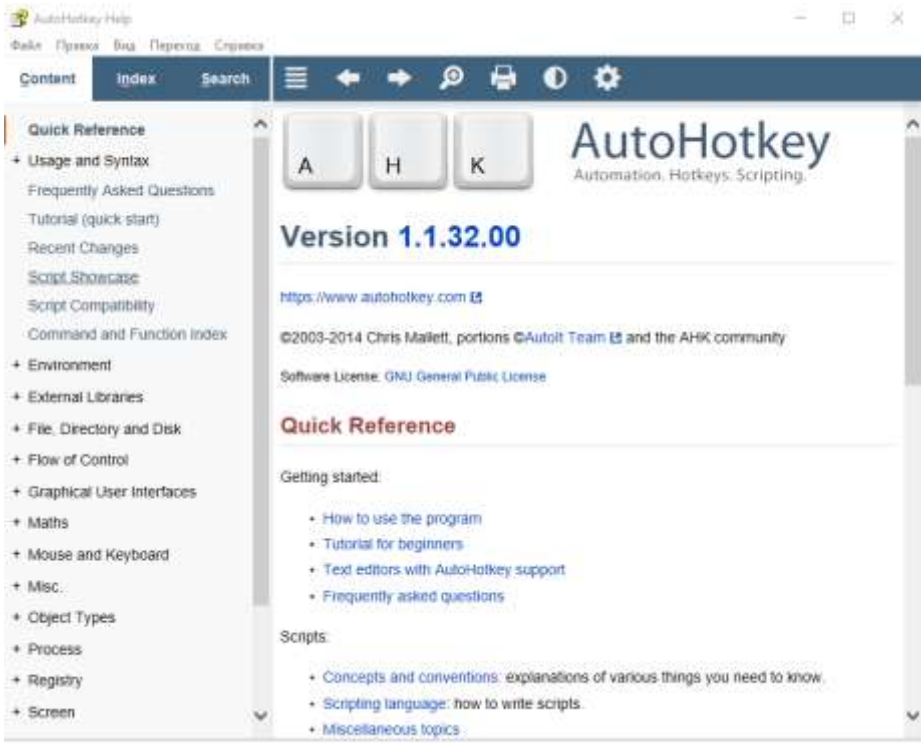

Рис. 1.1. Інтерфейс програми *AutoHotkey*

Наступна програма для огляду – *RoboTask.* За допомогою *RoboTask* користувач може автоматизувати такі речі, як відкриття декількох програм, редагування документів, надсилання електронних листів або вимкнення комп'ютера на основі тригерів. *RoboTask* дозволяє вибрати дії, які ви хочете автоматизувати, а потім відредагувати ці дії для кращої продуктивності.

Деякі з найпотужніших функцій *RoboTask* зарезервовані для преміум-версії додатка, який продається, хоча безкоштовна версія може легко задовольнити потреби більшості офісних працівників. *RoboTask* дозволяє автоматизувати повторювані завдання на вашому ПК з *Windows*, починаючи від простого запуску програм, до перевірки електронної пошти, переміщення чи резервного копіювання файлів до завантаження чи завантаження, надсилання електронної пошти та багато іншого. Програма дозволяє легко створювати прості завдання, а також дуже складні автоматизації, що включають умовні оператори *IF / ELSE,* цикли, спеціальні змінні та інші розширені параметри.

Інтерфейс цієї програми показаний на рис. 1.2.

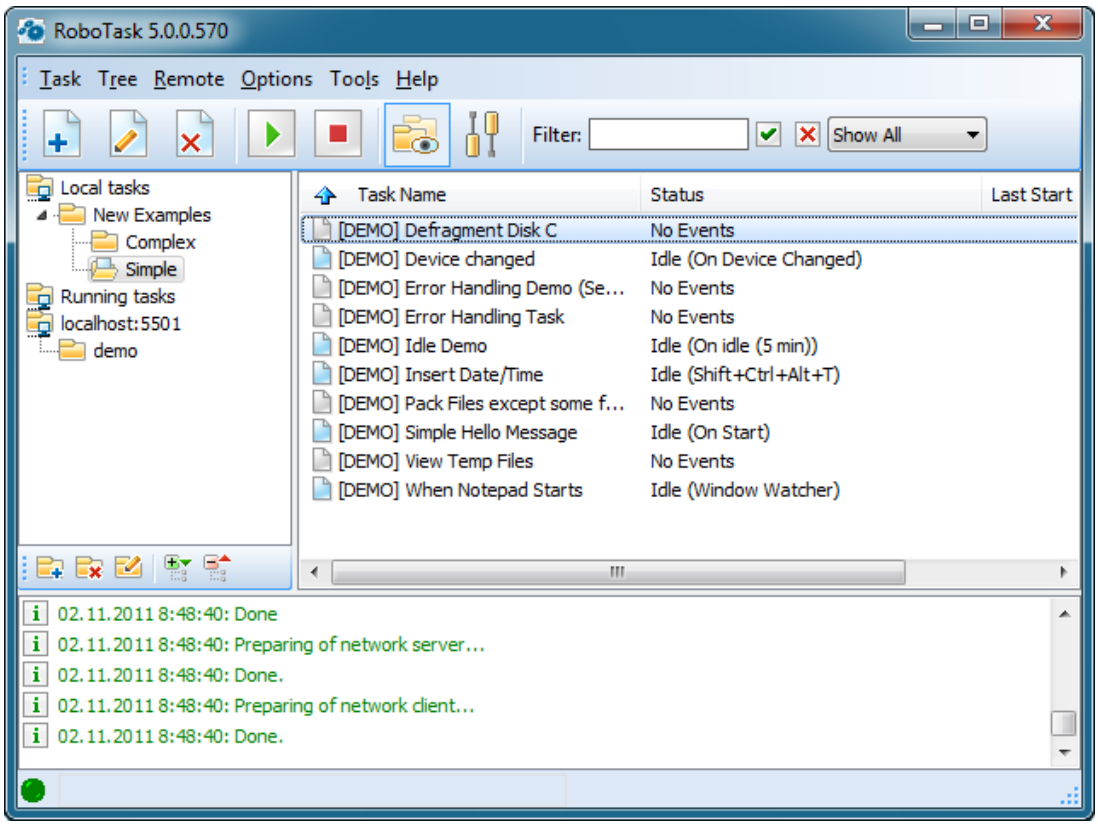

Рис. 1.2. Інтерфейс програми *RoboTask*

Також є така програма, як *TinyTask* – це додаток для автоматизації *Windows*, яки можна використовувати для запису та повторення дій. Використовуючи цей інструмент, ви можете записувати прості процеси та перетворювати його на невеликий макрос, який ви можете використовувати стільки разів, скільки вам потрібно, лише одним натисканням кнопки. Як випливає з назви, ця програма невелика - всього 35 КБ - але компактна. Більше того, він не вимагає сценаріїв при автоматизації процесів. Інтерфейс цього додатку дуже малий, але виконує дуже важливі функції, показаний на рис. 1.3.

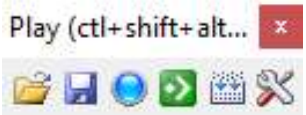

Рис. 1.3. Інтерфейс програми *TinyTask*

Загалом, всі ці програми дуже корисні для використання та спрощення процесів на комп’ютері. Якщо ви користуєтесь цими програмами та розібрались в функціоналі, то будьте впевнені, що Ви вже досвідчений користувач.

1.4. Постановка завдання проектування

Проаналізувавши багато джерел та програм-аналогів автоматизації процесів на ПК, постав вибір між мовами програмування, бібліотеками, інтерфейсами та методами реалізації модуля автоматизованого налаштування директорій. Ідея модуля полягала в тому, що при його вімкненні, модуль буде аналізувати корневу папку робочого столу на наявність файлів заданих заздалегіть форматів, та буде «чистити» робочий стіл методом перекладання їх в єдину папку за категоріями, такими як: тип файлу, рік створення ярлику та місяць створення ярлику.

Серед мов розглядалися *C++* та *Python*. Нарешті, проаналізувавши бібліотеки та реалізації, була обрана мова *Python*, за рахунок дуже зручних бібліотек для реалізації поставленої задачі.

Далі постав вибір між бібліотеками, які будуть використовуватися максимально продуктивно і компактно. Для створення цього модулю потрібні бібліотеки, які працюють з часом, датою та шляхами. Тоді було вирішено шукати подібні бібліотеки в стандартних та нестандартних бібліотеках *Python*. Далі буде перечислено ті бібліотеки, що були обрані для реалізації модуля.

Бібліотека *Sys*.

Ця бібліотека забезпечує доступ до деяких змінних, використовуваних або підтримуваних інтерпретатором, та до функцій, які сильно взаємодіють з ним. Вона завжди доступна.

Бібліотека *Pathlib*.

Ця бібліотека пропонує класи, що представляють шляхи файлової системи із семантикою, що підходять для різних операційних систем. Класи шляху розділені між чистими шляхами, які забезпечують обчислювальні операції без вводу чи виводу, і конкретними контурами, які успадковують від чистих контурів, але також забезпечують операції вводу та виводу. Вона створює конкретний шлях для платформи, на якій працює код.

Бібліотека *Time*.

Бібліотека, що надає різні функції, пов'язані з часом.

Бібліотека *os.path*.

Бібліотека, що реалізує деякі корисні функції в іменах шляхів. Для читання або запису файлів Параметри шляху можуть передаватися у вигляді рядків або байтів. Програмам рекомендується представляти імена файлів у вигляді символьних рядків (*Unicode*). На жаль, деякі імена файлів можуть не бути представленими як рядки в Unix, тому програми, яким потрібно підтримувати довільні імена файлів на Unix, повинні використовувати об'єкти байт для представлення імен шляхів. І навпаки, використання об'єктів байт не може представляти всі імена файлів у *Windows* (у стандартному кодуванні *mbcs*), отже, програми *Windows* повинні використовувати рядкові об’єкти для доступу до всіх файлів.

Бібліотека *shutil*.

Бібліотека *shutil* пропонує ряд операцій на високому рівні над файлами та колекціями файлів. Зокрема, передбачені функції, які підтримують копіювання та видалення файлів.

Бібліотека *Datetime*.

Datetime надає класи для керування датами та часом. У той час як арифметика дати та часу підтримується, фокус реалізації полягає в ефективному вилученні атрибутів для форматування виводу та маніпуляції.

Бібліотека *Watchdog*.

Ця бібліотека займається моніторінгом подій файлової системи.

З бібліотеками завершено, тепер треба було вибрати *GUI*, у *Python* є багато інтерфейсів, з якими можна працювати, але вибір постав між *tkinter* та *PyQT*.

Робота з tkinter ведеться безпосередньо через код, але у *PyQT* є середовище розробки, так званий *PyQT Designer*, в якому можна налаштувати вікно, додати всі компоненти, які тобі потрібні візуально, стилізувати, а потім вже експортувати файл *.gui* в файл *.py*.

Tkinter – графічна бібліотека на основі засобів *Tk* (широко поширена в світі *GNU / Linux* та інших *UNIX*-подібних систем, перенесена також і на *Microsoft Windows*). Входить в стандартну бібліотеку *Python*.

Переваги *Tkinter*:

– стисліcть;

Програми *Python*, що використовують *Tkinter*, можуть бути дуже короткими, почасти через потужність *Python*, а також завдяки *Tk*. Зокрема, прийнятні значення за замовчуванням визначаються для багатьох варіантів, що використовуються для створення віджета та його упаковки (тобто розміщення та показ).

– розширюваність.

Існує багато розширень *Tk*, і багато інших часто поширюються в Інтернеті. Будь-яке розширення доступне відразу з *Tkinter*, якщо не через розширення на *Tkinter*, то принайні через доступ *Tkinter* до мови *Tcl*.

Недоліки *Tkinter*:

– швидкість.

Є деяка стурбованість швидкістю *Tkinter*. Більшість викликів до *Tkinter* форматуються як команда *Tcl* (рядок) та інтерпретуються *Tcl*, звідки робляться фактичні виклики *Tk*. Це теоретичне уповільнення, спричинене послідовним виконанням двох інтерпретованих мов, рідко зустрічається на практиці.

Переваги *PyQT*:

– зручна міжпроцессорна взаємодія (сигнали / слоти);
– кросплатформенність на рівні вихідного коду. (Тут до ваших послуг *Windows, Linux, Mac, QNX, Android, IOS, WinRT*);
– низький поріг входження;
– безліч прикладів.

Недоліки *PyQT*:

– велика вага додатків. (Бібліотеки, в залежності від того, що ви використовуєте, будуть важити від 15 Мб і більше. Особливо погано для мобільних платформ);
– під не стандартні випадки складно знайти приклади.

Проаналізувавши всі «за» та «проти» був вибраний *PyQT*.

На основі зібраної інформації були поставлені задачі до проекту:

– оволодіти навичками *PyQT + Designer* та ознайомитися з документацією бібліотек;
– розробити навчальну програму, яка буде аналізувати робочий стіл на наявність ярликів та переносити їх у одну папку, в якій ці файли будут сортуватися по категоріям.

1.5. Висновки до розділу

У першому розділі було детально розглянуто та проаналізовано тему «Автоматизації процесів», виділено для себе ті деталі, які будуть потрібні для створення програми. Також було проаналізовано всі види вирішення проблем автоматизації, був проведений аналіз принципів та методологій, які використовуються для розробки таких модулів.

Була вибраний *GUI* із вище наведених прикладів, проаналізовані всі переваги та недоліки варіантів.

Результатом аналізу стало розуміння того, що на на теперішній час існує ще багато актуальних процесів у ОС, які хотілось би автоматизувати.

Наприкінець всього аналізу і отримання образу програми, яку треба реалізувати – була поставлена задача з покроковим поглибленням та вивченням документації бібліотек, функцій *PyQT Designer* та розробкою модуля автоматизованого налаштування директорій, який буде виконувати всі поставлені до нього завдання.

РОЗДІЛ 2
РОЗРОБКА ДІАГРАМ ТА СТРУКТУРИ МОДУЛЯ

2.1. Розробка діаграм та схем до програмного модуля

Діаграма прецедентів є одна з п'яти типів діаграм, що застосовуються в *UML* для моделювання динамічних аспектів системи (інші чотири типи – це діаграми класів, станів, послідовностей і кооперації). Діаграми прецедентів грають основну роль в моделюванні поведінки системи, підсистеми або класу. Кожна така діаграма показує безліч прецедентів, акторів і відносини між ними.

В програмному модулі існують дві сутності:

– «Користувач», який посилає запит на виконання функцій та очікує їх виконання;

– «*DCleaner*», який отримує функції і відповідає на запити.

Діаграму прецедентів показано на рис. 2.1.

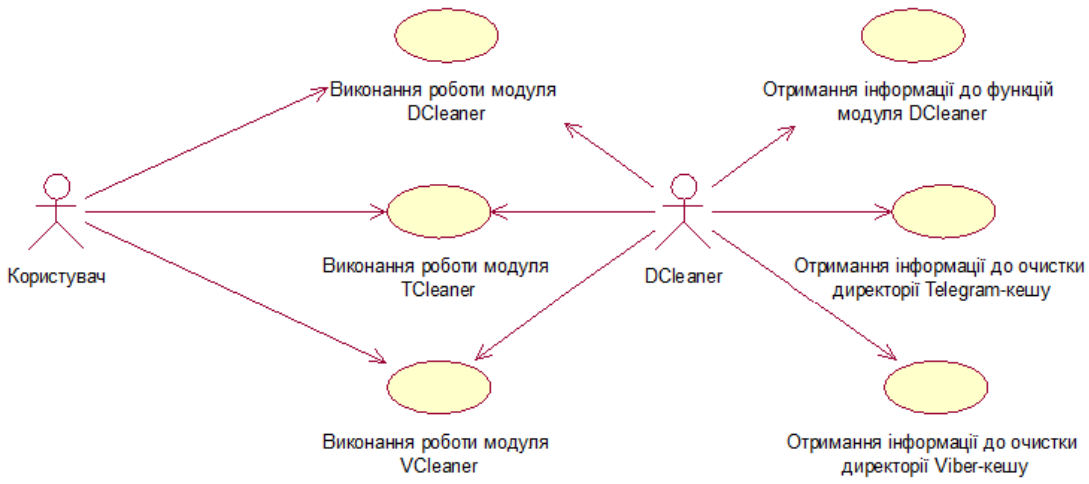

Рис. 2.1. Діаграма прецедентів

Діаграма класів – це набір статичних елементів моделі. Діаграми класів застосовуються при прямому проектуванні, тобто в процесі розробки нової системи, і інколи їх використовують для зворотнього проектування – описі існуючих і використовуваних систем. Інформація з діаграми класів безпосередньо відображається в вихідний код програми – в більшості існуючих інструментів UML-моделювання можлива кодогенерація для певної мови програмування (зазвичай *C* ++ або *Java*).

Система реалізована 3-ма класами:

– «*GUI*»(приймає запити від двох інших класів);

– «*DCleaner*» (задає функції та реалізацію модулю автоматизованого налашутвання);

– «*CacheCleaner*» (задає шлях та функції до очистки папок кешу).

Діаграма класів показана на рис. 2.2.

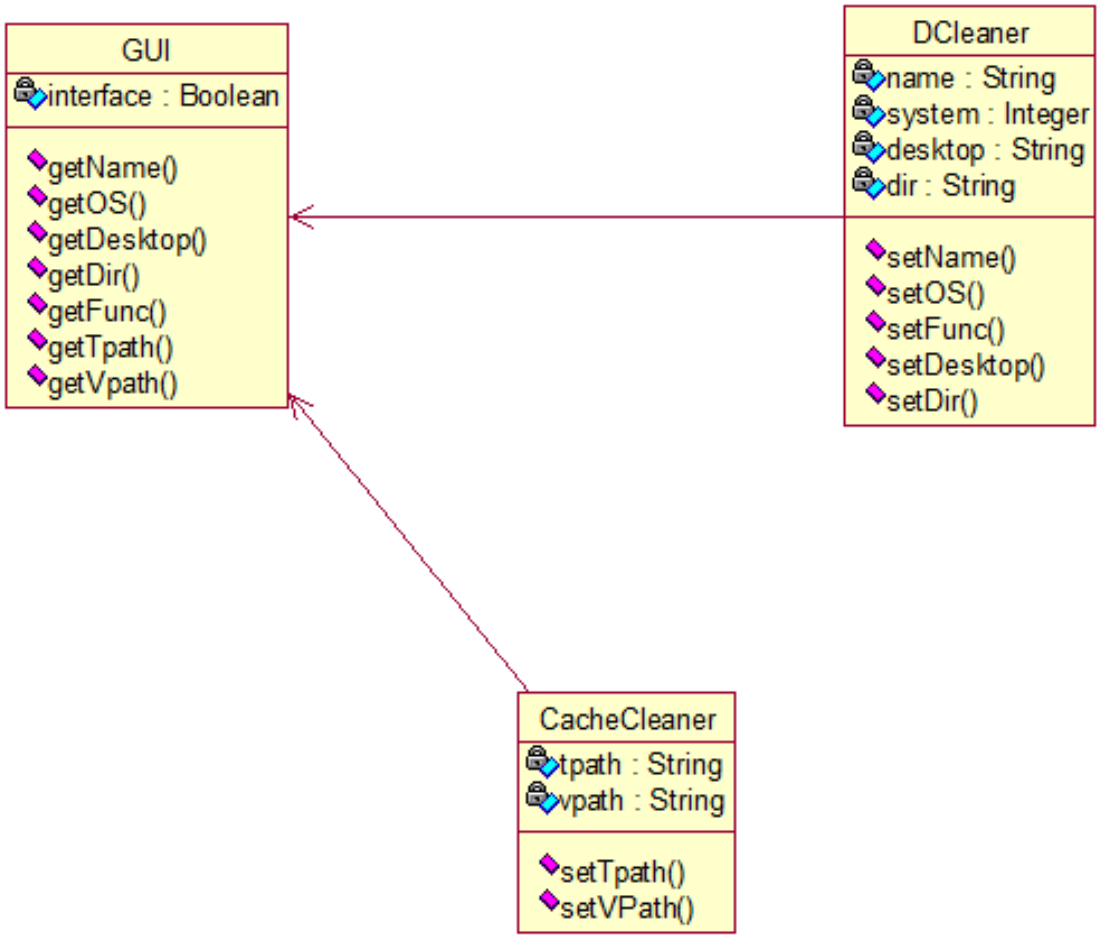

Рис. 2.2. Діаграма класів

У час роботи, *GUI* приймає функції двох модулів, обробляє їх, та працює як тригер, який буде виконувати отримані функції при виклику. Враховуючи те, яку функцію вибере користувач – різні сигнали будуть приходити на виклик фукнцій.

Тепер була поставлена мета реалізувати діаграму послідовності, яка показана на рис. 2.3.

Для моделювання взаємодії деяких об'єктів використовуються відповідні діаграми взаємодії. Взаємодії об’єктів можна розглядати в часі, і тепер для представлення тимчасових особливостей передачі і прийому повідомлень між об’єктами використовується діаграма послідовності. Взаємодіючі об’єкти обмінюються між собою інформацією.

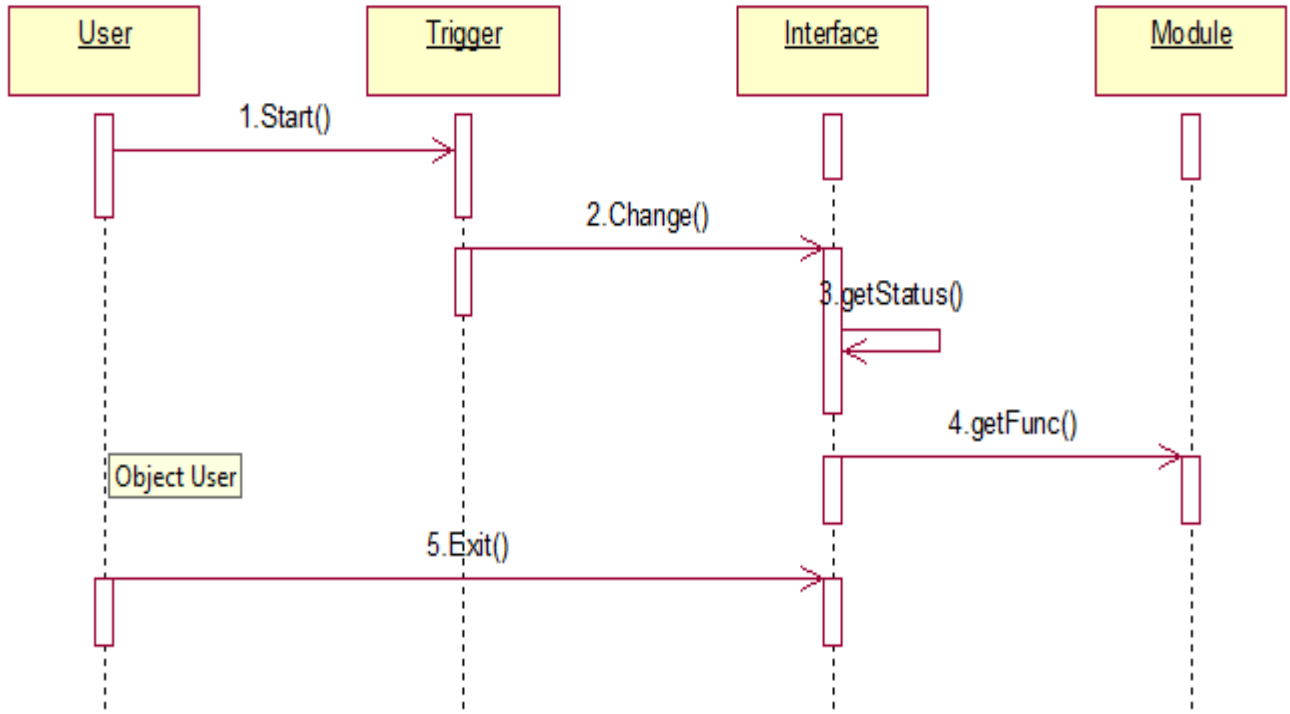

Рис. 2.3. Діаграма послідовності

На даній діаграмі бачимо, що є такі об’єкти, як *User, Trigger, Interface* та *Module*. Зараз буде пояснення всього, що показане, та як це працює.

Спочатку користувачу(*User*) потрібно відкрити програму, після того, як він її відкриває, перед ним постає вибір між функціями в реалізованій програмі, за допомогою кнопки(*Trigger*) він вмикає потрібну для себе функцію, і бачить анімацію натиснення на кнопку та вивід тексту(статусу роботи програми), яка

показує, що ця функція працює на інтерфейсі користувача(*Interface*), паралельно цей інтерфейс подає сигнал тому, чи іншому модулю(*Module*), що йому потрібна конкретна функція з нього. На цьому діаграма послідовності завершена. Наступною для аналізу та побудови була використана діаграма кооперації, яка слідує з діаграми послідовності. Наступну використану діаграму можна побачити на рис. 2.4.

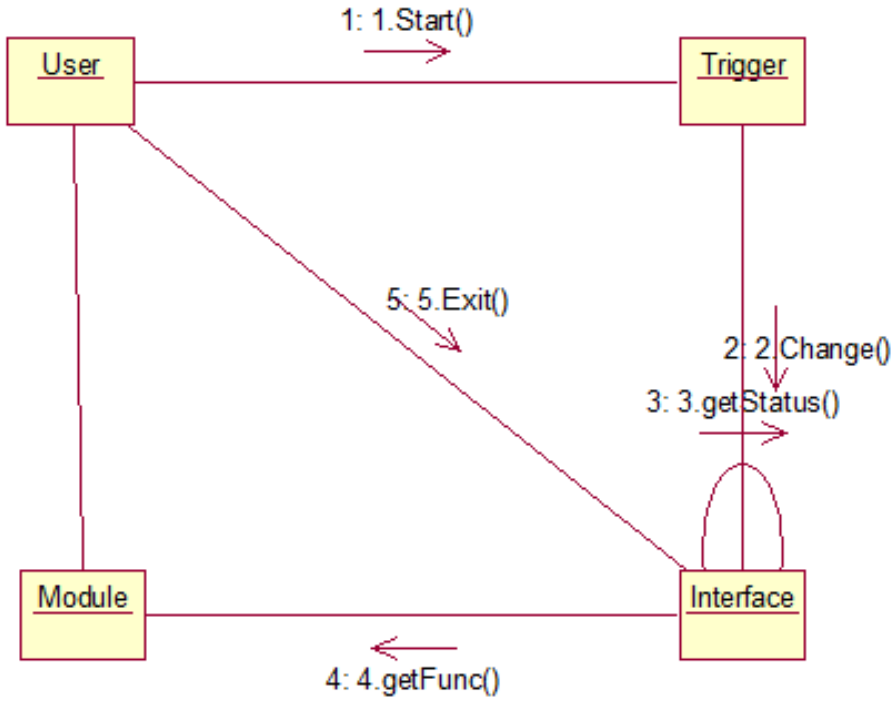

Рис. 2.4. Діаграма кооперації

Діаграма кооперації, як і діаграма послідовності показує нам ті ж об’єкти, але в другому візуалізованому стані.

На цій діаграмі кооперації у вигляді прямокутників беруть участь в взаємодії об'єкти, що містять ім'я об'єкту, його клас.

Як і на діаграмі послідовності, показують зв’язок між об’єктами у вигляді різних сполучних ліній з функціями.

Наступна використана діаграма була – діаграма станів, показана на рис. 2.5.

Найголовніше призначення діаграми станів – це описати деякі послідовності станів та переходів, які у сукупності характеризують поведінку об’єкта моделі напротязі його життєвого циклу. Діаграма станів представляє

поведінку сутностей, на основі специфікації їх реакції на сприйняття деяких конкретних подій.

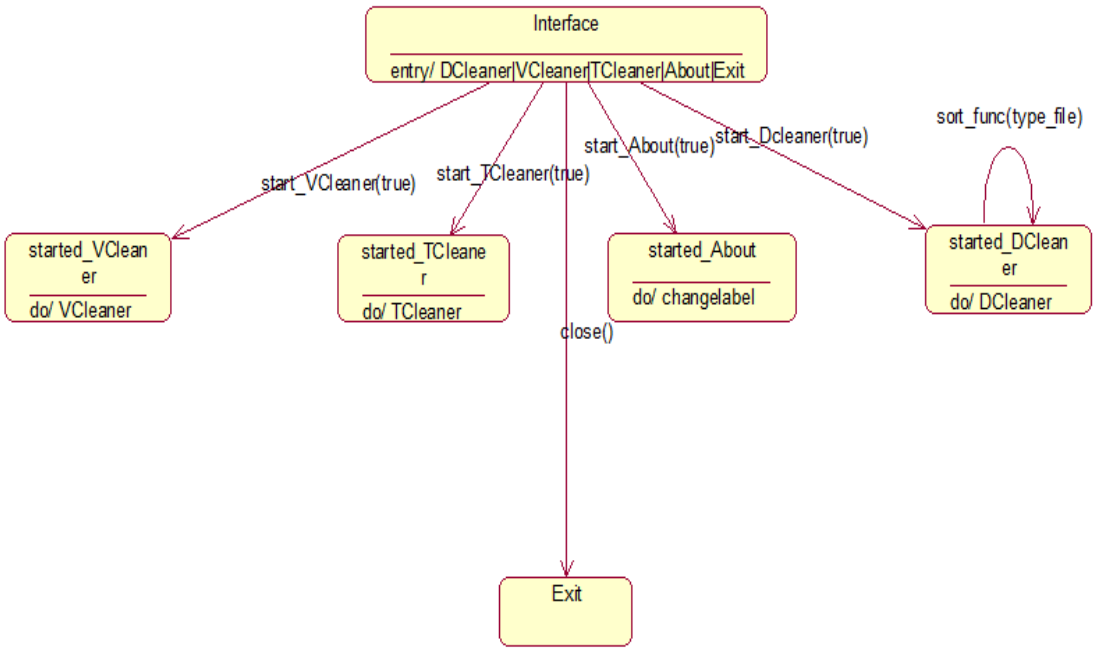

Рис. 2.5. Діаграма станів

Головне призначення цієї діаграми – описати можливі послідовності станів і переходів, які в збігі характеризують поводження елемента моделі протягом всього його життєвого циклу. Діаграма станів представляє динамічне поводження сутностей, на основі специфікації їхньої реакції на сприйняття деяких конкретних подій.

Наступна схема для розробки стала структурна схема.

Структурна схема – розробляється на початкових стадіях проектування і передує розробці схем інших типів. Структурна схема визначає основні модулі програми, взаємозв'язки між ними. Схема відображає принцип дії програми в найзагальнішому вигляді.

Весь програмний модуль складається з трьох частин, структурна схема показана на рис. 2.6.

– модуль очистки робочого столу *DCleaner* – це основний модуль програми, який виконує головну функцію, функцію автоматизованого налаштування робочого столу користувача;

– модуль очистки кешу *TCleaner* – додатковий модуль з функцією очистки директорії кешування «*Telegram*» по його стандартному шляху завантаження;
– модуль очистки кешу *VCleaner* – додатковий модуль з функцією очистки директорії кешування файлів «*Viber*» по його стандартному шляху завантаження.

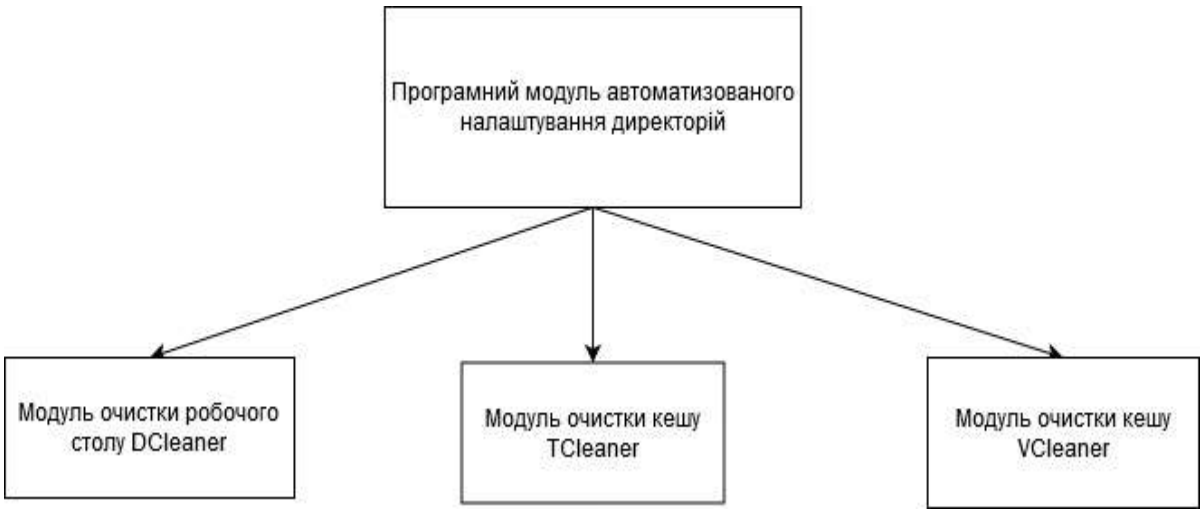

Рис. 2.6. Структурна схема програмного модулю

Це дуже важливо приділяти увагу до деталей модулю, так набагато простіше розуміти, з чим маєш справу і що за «це» або «те» відповідає.

2.2. Розробка структури інтерфейсу

Інтерфейс користувача(*GUI*) – це невід'ємна частина програмного продукту, без нього ніяк.

Насамперед треба зрозуміти, що ж таке «Інтерфейс», це «спосіб, яким ви виконуєте будь-яку задачу за допомогою будь-якого продукту, а саме здійснювані вами дії і те, що ви отримуєте у результаті»

Щоб у користувача не виникало проблем при використанні будь-якої програми, розумні люди візуалізують її функціональні можливості у вигляді зрозумілих елементів, кнопок, та інших візуальних елементів.

Спочатку треба зорієнтуватися в елементах майбутнього інтерфейсу. Для цього ми визначимо завдання, вивчимо предметну область, аудиторію, і те, як вона користується продуктами.

На цьому етапі було проаналізовано різні види інтерфейсів, їх вигляд, та виведено потрібні елементи для інтерфейсу програмного модуля.

Вирішено адаптувати та мінімізувати вікно для зручного користування та зробити конкретні кнопки для підключення основного модулю та додаткових, додати логотип до програмного продукту, також додати деякі інші кнопки, які будуть працювати конкретно з інтерфейсом користувача.

Основні елементи, які треба було розробити, були такі:

– «*Clean my desktop*» – підключає основний модуль «*DCleaner*», який виконує аналіз та автоматизовану очистку робочого столу;
– «*TCacheClean*» – підключає додатковий модуль очистки папки кешування файлів «*Telegram*» по стандартному шляху завантаження файлів;
– «*VCacheClean*» – підключає додатковий модуль очистки папки кешування файлів «*Viber*» по стандартному шляху завантаження файлів.

Додатковими кнопками, які треба було додати, були:

– «*Version*» – кнопка, яка буде показувати версію програми та ім’я розробника цих модулів;
– «*DirectCleanUp*» – кнопка, яка буде відкривати браузре провідника, щоб користувач вибрав, яку папку він хоче очистити;
– «*Finish and Exit*» – кнопка, яка буде завершувати роботу програми та закривати її.

На цьому етапі було вирішено завершити розробку функціональних елементів інтерфейсу.

Тепер треба було додати кольору до інтерфейсу, тому що всі кнопки та вікно в білому кольорі – це не дуже приємна картина для користувача.

Методом «проб та помилок» додано колір фону програми(сірий), колір елементів(блакитний) та колір тексту(чорний).

Тепер була вирішено виконати задачу, яка заключалася в розстановці об’єктів на вікні, та змінити розмір тексту та колір.

Надалі, у розділі 3 буде показано поетапно, як всі елементи підключалися.

2.3. Висновки до розділу

В даному розділі було спроектовано програмний модуль: розроблені схеми та діаграми до задачі, також розроблено концепт інтерфейса користувача.

Складалось проектування з розробки діаграм прецендентів, яка відображає взаємодію між сутностями та використанням варіантів їх можливостей, діаграму класів, для опису структури, методів, та параметрів програмного модулю, діаграму взаємодії, яка показує взаємодію деяких об’єктів, діаграму кооперацій, та діаграму станів. Була розроблена структурна схема, яка показує структуру програмного модулю, та функціональна схема, яка показує всі функції програмного модулю та процеси, зв’язуючись з структурною схемою.

У процесі поглиблення розроблених діаграм та схем вже було приблизно зрозуміло, як повинен виглядати продукт, але не вистачало всього одного, це інтерфейсу користувача.

При розробці прототипу інтерфейсу користувача було проаналізовано безліч програмних-додатків та вирішено, які елементи та функції потрібно було розробити для продукту.

РОЗДІЛ 3
РОЗРОБКА АЛГОРИТМІВ, ОСНОВНИХ ФУНКЦІЙ ТА ІНТЕРФЕЙСУ

3.1. Розробка алгоритмів основного та додаткових модулів

Основний модуль «*DCleaner*» в розробці потребував багатьох функцій та проектних рішень, блок-схема алгоритму модуля показана на рис. 3.1.

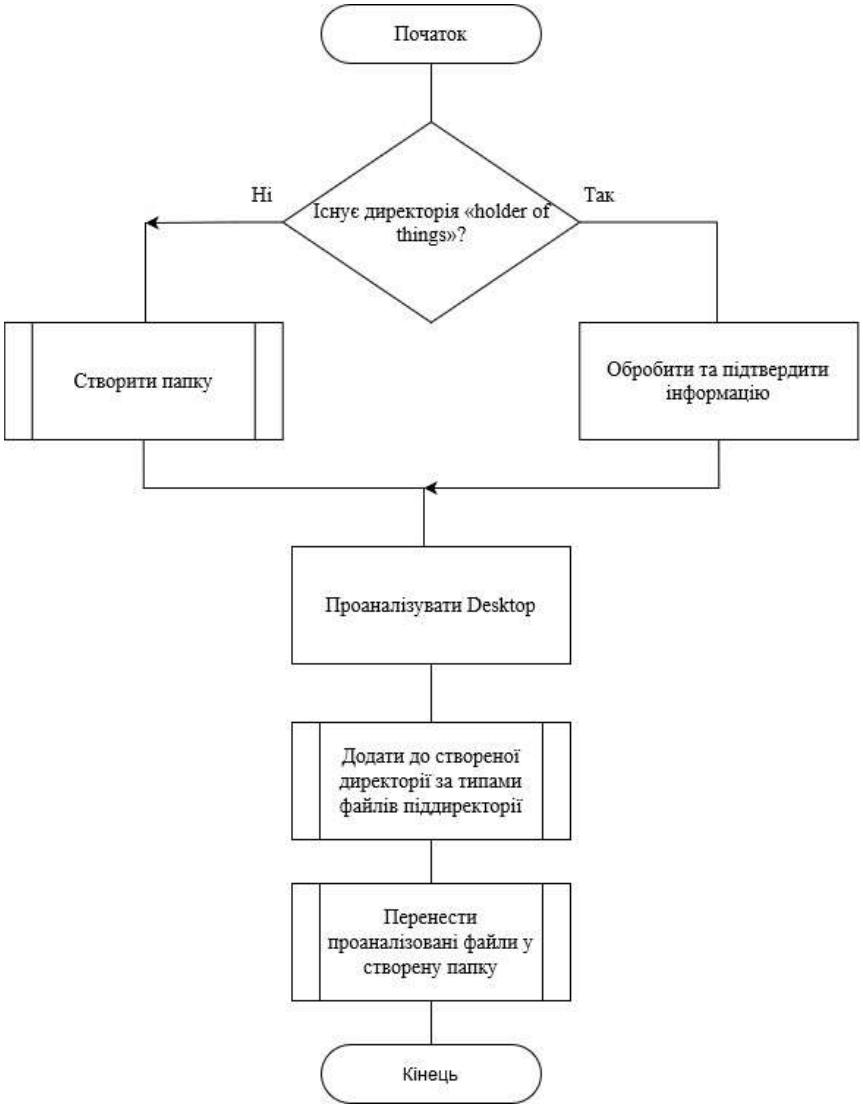

Рис. 3.1. Схема алгоритму основного модулю *DCleaner*

Покрокове пояснення роботи основного модулю «*DCleaner*»:

На блок-схемі показано, що під час запуску, модуль повинен проаналізувати, чи є на робочому столі директорія з найменуванням «*holder of things*», якщо ж її немає, то модуль повинен її створити.

Після створення цієї директорії або підтвердження, що вона вже існує – йде перехід до наступної дії, аналізу робочого столу.

Аналіз йде за параметрами типів файлів, які буде вказано в коді програмного модулю. Далі за даним аналізом буде додано піддиректорії за категоріями проаналізованих файлів робочого столу.

Останнім кроком в роботі програми стане перенесення цих файлів з робочого столу до нової директорії «*holder of things*» та конкретних піддиректорій, які були створені у процесі аналізу.

Наступним додатковим модулем, який треба було розробити, став модуль «*TCleaner*», показаний на рис. 3.2.

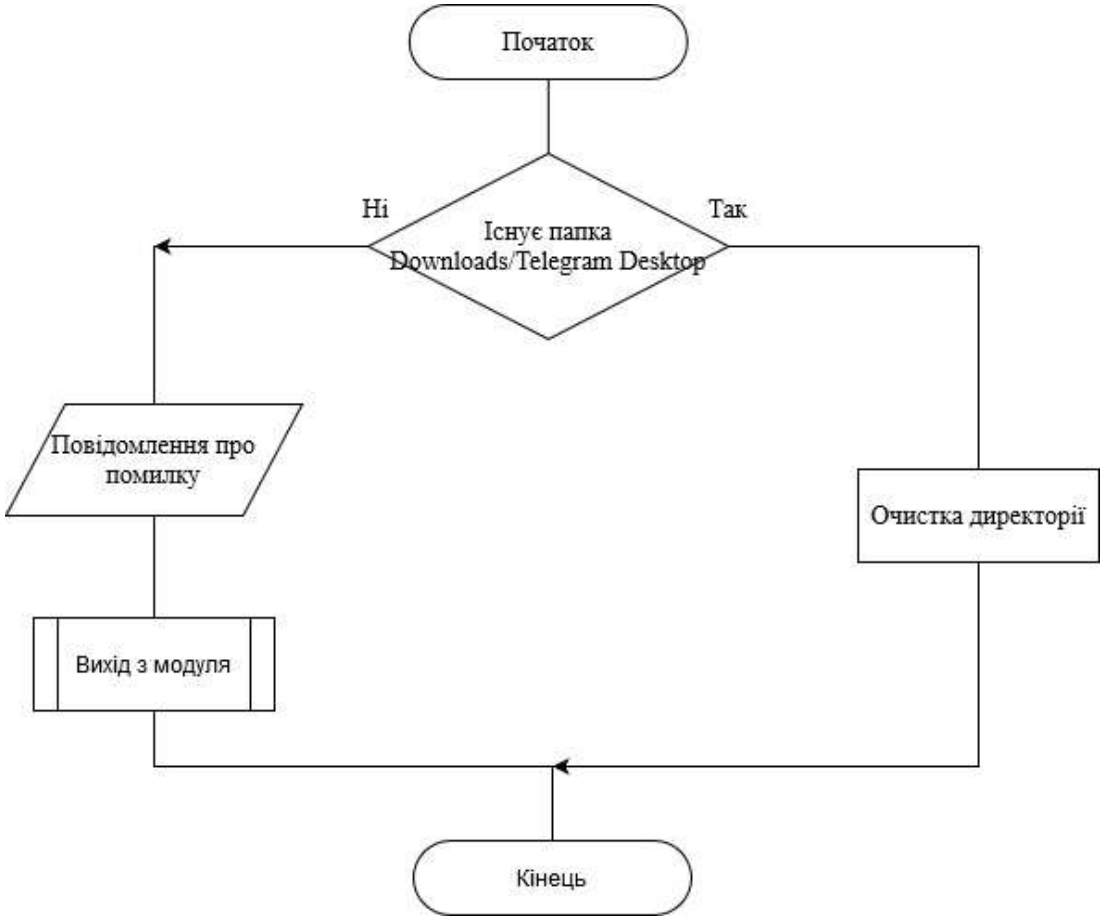

Рис. 3.2. Схема алгоритму додаткового модулю *TCleaner*

Покрокове пояснення роботи додаткового модулю «*TCleaner*»:

Під час запуску цього модуля, він перевіряє, чи існує директорія за стандартним шляхом кешування файлів «*Telegram*».

Якщо існує, то виконується повна очистка директорії.

Якщо ж такої директорії не існує, то на екран буде виведено помилку доступу до директорії та вихід з модулю.

Надалі було вирішено додати ще один аналоговий додатковий модуль «*VCleaner*», який буде виконувати ті ж самі функції, як і попередній модуль, але теперь буде аналізуватися стандартна папка кешування файлів «*Viber*», модуль показаний на рис. 3.3.

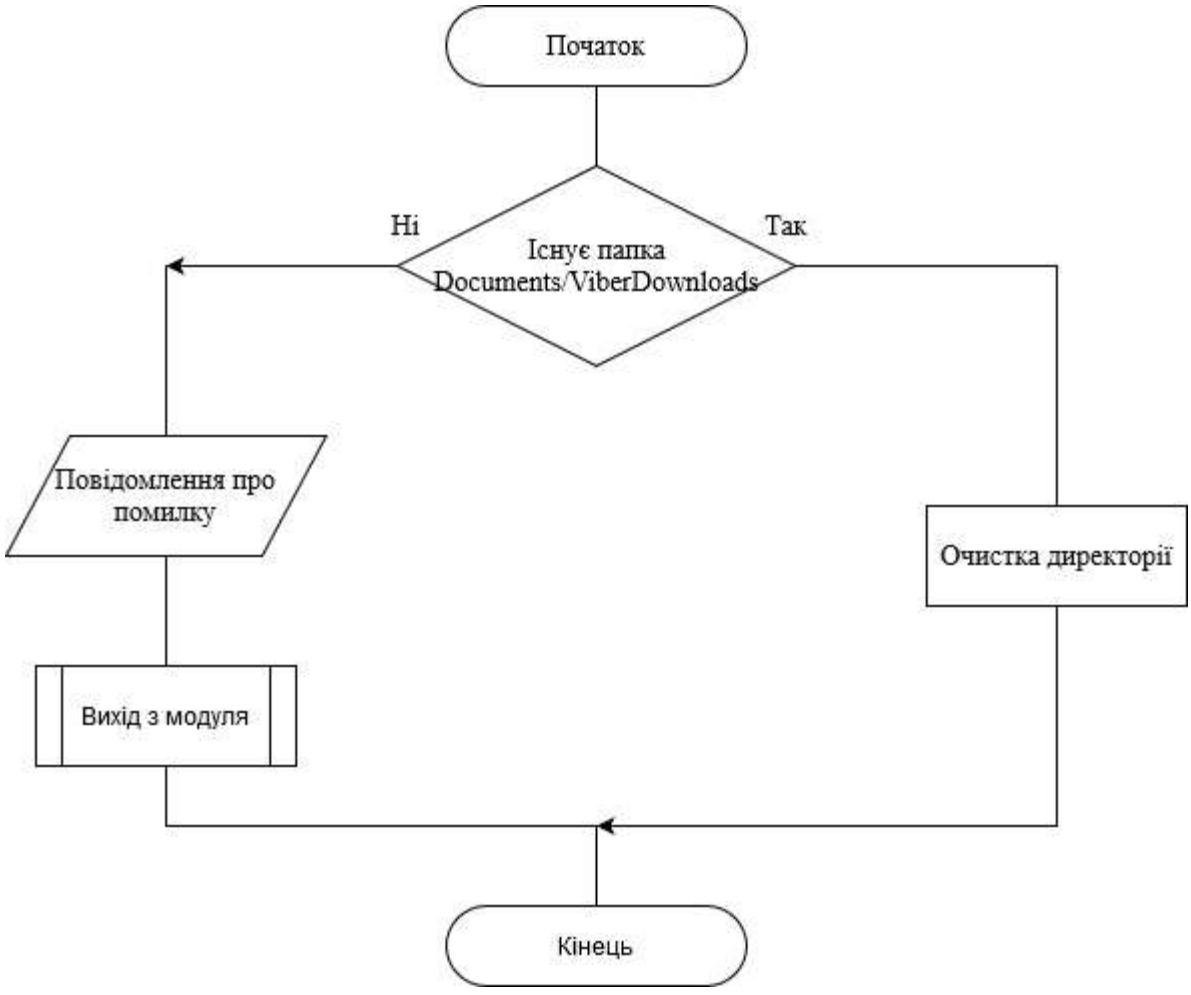

Рис. 3.3.Схема алгоритму додаткового модулю *VCleaner*

На цьому етапі блок-схеми основного модуля «*DCleaner*» та додаткових модулів «*TCleaner*» та «*VCleaner*» були розроблені в повному обсязі.

3.2. Розробка основних функцій

Спочатку треба було реалізувати роботу модуля «*DCleaner*». Для реалізації даної задачі було вирішено створити 3 *python*-файла. Першим файлом став *DCleaner.py*, який ініціалізував основні функції, задавав рекурсивність програмі та викликав додаткові функції з інших двух створених файлів реалізованого модулю, програмний код файлу *DCleaner.py*:

```
import sys
sys.path.append(".")
from watchdog.observers import Observer
from time import sleep
from pathlib import Path
from EventHandler import EventHandler

 __name__ == '__main__':
        w_path = Path.home() / 'Desktop'
        d_root = Path.home() / 'Desktop/DCleaner'
        e_handler = EventHandler(w_path=w_path, d_root=d_root)

observer = Observer()
observer.schedule(e_handler, f'{w_path}', recursive=True)
observer.start()

try:
        while True:
                sleep(60)
except KeyboardInterrupt:
        observer.stop()
 observer.join()
```

Наступним другим файлом став файл *EventHandler.py*, який виконує додаткові функції, такі як створення піддиректорій під рік та місяць створення файлу на робочому столі, також функції, яка аналізує, чи є вже файл з однаковою назвою, щоб автоматично не заміняти старі файли на нові, ця фунція буде інкрементувати цифру в кінці назви файлу, вихідний код файлу *EventHandler.py*:

```
import shutil
from pathlib import Path
from datetime import date
from watchdog.events import FileSystemEventHandler
from ext import ext_paths
def date_to_path(path: Path):
    d_path = path / f'{date.today().year}' / f'{date.today().month:02d}'
    d_path.mkdir(parents=True, exist_ok=True)
    return d_path
```

Функція, яка додає поточний рік / місяць до шляху призначення. Якщо шлях ще не існує, він створюється.

```
def r_file(source: Path, d_path: Path):
    if Path(d_path / source.name).exists():
        increment = 0
        while True:
            increment += 1
            new_name = d_path / f'{source.stem}_{increment}{source.suffix}'
            if not new_name.exists():
                return new_name
    else:
        return d_path / source.name
```

Ця функція перейменовує файл, щоб відобразити новий шлях. Якщо файл той самий та ім'я вже існує в цільовій папці, ім'я файла пронумеровується та збільшується до тих пір, поки ім'я файлу не стане унікальним (запобігає перезапису файлів).

Останній файл *ext.py* не менш важливий за інші у реалізації модулю, тому що саме він задавав категорії до типів файлів та найменування піддиректорій під це.

```
ext_paths = {
    # No name
    'noname':  'other/uncategorized',
    # audio
    '.aif':    'media/audio',
    '.cda':    'media/audio',
    '.mid':    'media/audio',
    '.midi':   'media/audio',
    '.mp3':    'media/audio',
    '.mpa':    'media/audio',
    '.ogg':    'media/audio',
    '.wav':    'media/audio',
    '.wma':    'media/audio',
    '.wpl':    'media/audio',
    '.m3u':    'media/audio',
```

І так далі, категорії та типи файлів, які було додано, це аудіо-файли, текстові файли, відео-файли, файли фото-формату, архіви, виконувані файли, файли шрифтів, таблиць, та системні файли.

Основний модуль «*DCleaner*» було реалізовано.

Тепер треба було перейти до реалізації додаткових модулів «*TCleaner*» та «*VCleaner*», для реалізації цих модулів були використані бібліотеки *os, shutil* та *Path*. Приклад вирішення задачі для очистки «*Telegram*»-кешу вписаний в *tcleaner.py*:

```
import os, shutil
from pathlib import Path
f = os.path.join(Path.home(), 'Downloads/Telegram Desktop')
for fn in os.listdir(f):
```

```
fp = os.path.join(f, fn)
try:
    if os.path.isfile(fp) or os.path.islink(fp):
        os.unlink(fp)
    elif os.path.isdir(fp):
        shutil.rmtree(fp)
except Exception as e:
    print('Failed to delete files %s. Because: %s' % (fp, e))
```

Все, що потрібно змінити в цьому коді для реалізації очистки «*Viber*»-кешу, це у коді замінити текст *'Downloads/Telegram Desktop'* на текст *'Documents/ViberDownloads'* ось так:

```
f = os.path.join(Path.home(),'Documents/ViberDownloads')
```

3.3. Розробка інтерфейсу

Для розробки інтерфейсу був використаний пакет *PyQT* та середа розробки *PyQT Designer*. Спочатку, у *PyQT Designer* були проаналізовані всі засоби для функціонування на робочому вікні. Потім було вирішено використати конкретні засоби, такі як «*PushButton*» та «*TextLabel*», які показані на рис. 3.4.

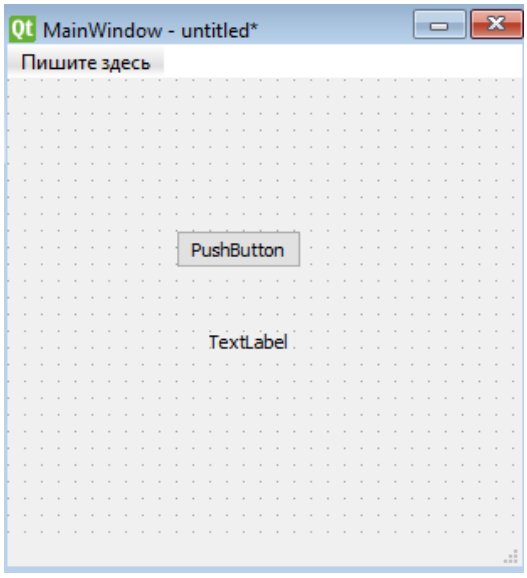

Рис. 3.4. Елементи *PushButton* та *TextLabel*

«*PushButton*» – це кнопка, якій розробник може в дальнішому додати якусь функцію, коли ви на неї натискаєте.

«*TextLabel*» – текст, який розробник може додавати та змінювати за допомогою додаткових функцій.

В розробці інтерфейсу було вирішено, що на ньому будуть присутні 6 кнопок «*PushButton*» та один «*TextLabel*», який буде змінюватися в процесі роботи програми, також будуть додані найменування основного вікна, логотипу вікна і картинки, яка буде явно показана на інтерфейсі.

Коли всі потрібні елементи вікна були додані та стилізовані, *GUI* був готовий для додання до кнопок функцій, було вирішено використати пакет стандартної бібліотеки *python – pyuic5*, який може зконвертувати файл з типом .ui у тип стандартного *.py* файлу. Все, що потрібно було зробити, це завантажити файл інтерфейсу до окремої директорій, війти у консоль *bash* у цій директорії, та прописати таку команду:

Pyuic5 –x ui.ui –o ui.py

Далі було перенесено експортований файл у папку проекту та почалося налаштування кожного елемента за допомогою програмного коду.

Трьом кнопкам треба було присвоїти функції окремих реалізованих модулів, а іншим – функції за синтаксисом *PyQT*:

– «*Clean my desktop*» – кнопка, яка підключає основний модуль та виклакає його роботу;
– «*TCacheClean*» – кнопка, яка підключає додатковий модуль «*TCleaner*» та виклакає його роботу;
– «*VCacheClean*» – кнопка, яка підключає додатковий модуль «*VCleaner*» та виклакає його роботу;
– «*DirectCleanUp*» – кнопка, яка дозволяє видалити вміст окремої вибраної директорії на комп’ютері;
– «*Version*» – кнопка, яка оновлює елемент «*TextLabel*» та дозволяє побачити на інтерфейсі версію продукту та ім’я розробника;
– «*Finish and Exit*» – кнопка, яка завершує роботу програми та вимикає її.

Отриманий результат розробки, можна спостерігати на рис. 3.5.

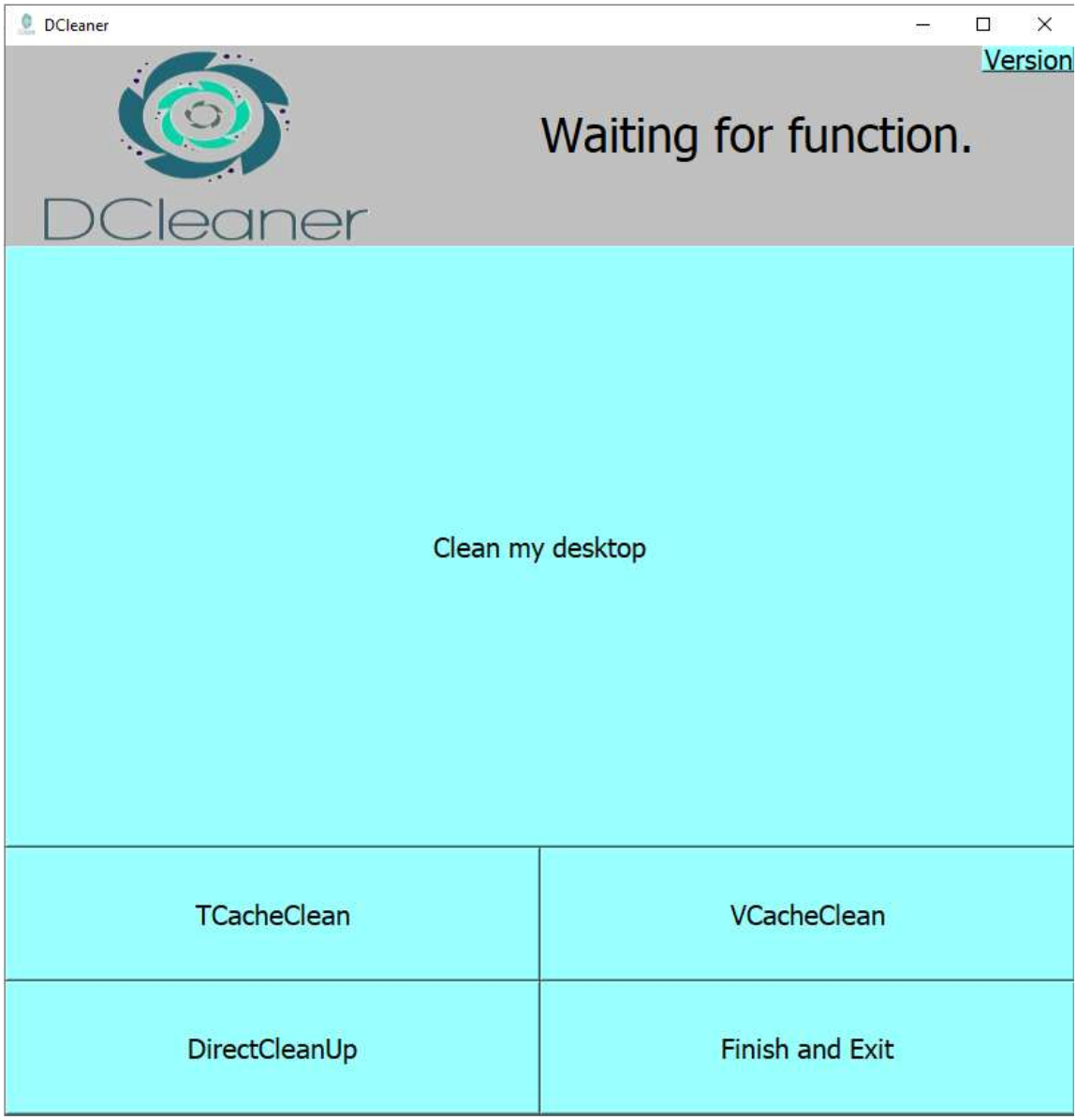

Рис. 3.5. Інтерфейс програмного модулю *Dcleaner*

3.4. Тестування ПЗ

Тестування ПЗ – дуже важлива дія перед тим, як випускати продукт у вільне користування. Під час перевірки можуть виявитися баги та дефекти в нестандартних ситуаціях. Є такі види тестування, як ручне та автоматизоване, ручне дозволяє самому виконувати кропітливе тестування окремих подій та сценарієв на візуальному фоні програми, а автоматизоване – вхідні дані та очікуваний результат.

Було вирішено використовувати ручне тестування та автоматизоване(*pytest*). Вирішено протестувати коректність функціонування, інсталяції, запуск на окремих ОС.

Перевірено функціонування кожної деталі інтерфейсу на коректнісь виконаних дій, все працює добре.

Перевірена інсталяція, пакет *DCleaner* потребує стільки місця, скільки задано у інсталяторі та ніяких файлів при інсталюванні не зникло.

Було перевірено запуск програми на ОС *Windows* та *macOS*. Все працює коректно на кожній системі. Помилок виявлено не було.

Всі тести програми пройшли успішно, программа готова до експлуатації користувачами, критичних помилок виявлено не було.

3.5. Збірка проекту

Для збірки проекту використовувалися пакети *Pyinstaller* та *NSIS*

Pyinstaller – бібліотека, яка може конвертувати проект, в якому наявні файли ру в один виконуваний файл з найменуванням .exe за допомогою команди в командному рядку *Windows*:

Pyinstaller --onefile -w dcleaner.py

NSIS – програма яка використовується для конвертації заархівованого проекту у установочний пакет.

Проект був зібраний, як показано на рис. 3.6, та готовий для використання користувачами.

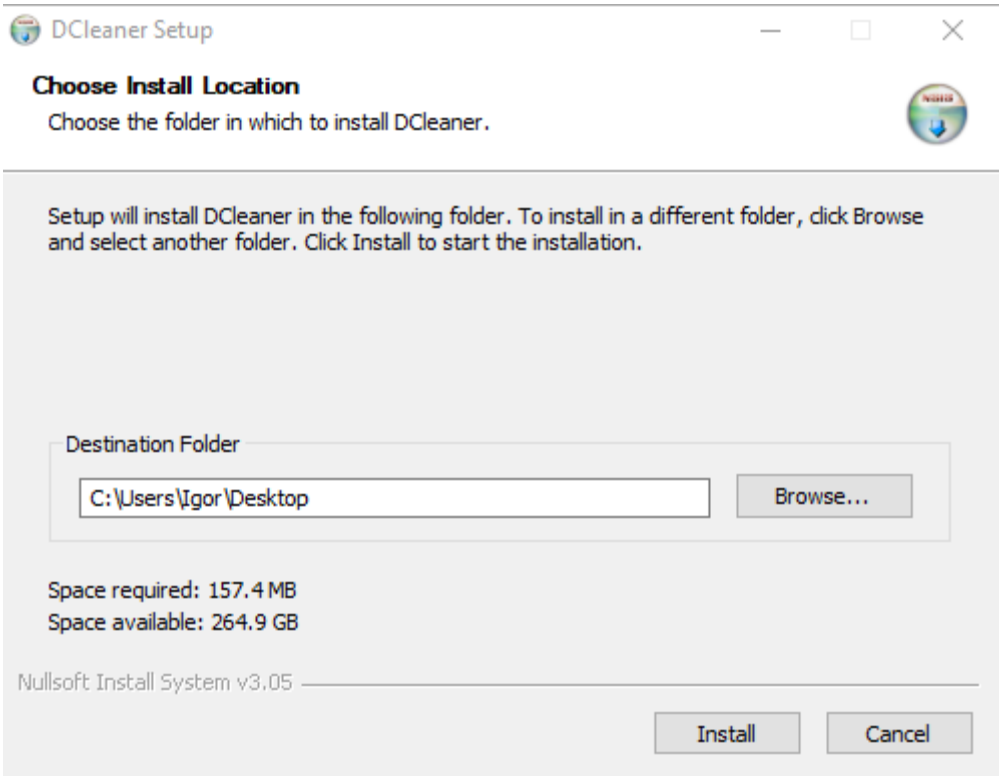

Рис. 3.6. Інсталятор програми *Dcleaner*

Для запуску програмного модулю на комп’ютері повинен бути мінімальний набір апаратних компонентів та наявність 160 МБ на жорсткому диску.

3.6. Висновки до розділу

В третьому розділі було розглянуто проаналізовано методології розробки блок-схем та модулів. Розроблено такі блок-схеми, як алгоритм основного модулю «*DCleaner*», алгоритм додаткового модулю «*TCleaner*» та алгоритм додаткового модулю «*VCleaner*». Далі був розроблений програмний код, основні функції до кожного з модулів. Також розроблено інтерфейс користувача та підключено його елементів до функціонування з модулями. Проведено ручне та автоматизоване тестування ПЗ, в якому не виявилось критичних помилок для роботи. Виконана збірка всього проекту за допомогою пакетів *Pyinstaller* та *NSIS*. Програма готова до використання.

ВИСНОВКИ

При розробці програмного модулю автоматизованого налашутвання директорій було проаналізовано багато інформації по темі автоматизації процесів пк та здобуто навички, такі як:

– програмування у мові *python*;
– оволодіння синтаксисом бібліотек мови *python*;
– робота з командним рядком *bash*;
– робота у середовищі розробки інтерфейсу *PyQT Designer*;
– робота з *pyinstaller*;
– робота з *NSIS*.

Для проектування та користування програмним додатком використана платформа *Windows*. Також користувачі можуть використовувати його на операційній системі *macOS*.

При виборі мови програмування для розробки було проаналізовано багато інформації та обрано мову *Python*.

Було детально розглянуто та проаналізовано тему «Автоматизації процесів», підчеркнуто для себе ті деталі, які будуть потрібні для створення програми, яка потрібна. Проаналізовано види вирішення проблем автоматизації, був проведений аналіз принципів та методологій, які використовуються для розробки таких модулів.

Для розробки інтерфейсу корстувача була вибрана бібліотека *PyQT* та середовище розробки інтерфейсів *PyQT Designer*. Поглиблено багато інформації про функції та синтаксис цих елементів розробки.

Було спроектовано програмний модуль: розроблені схеми та діаграми, такі як:

– діаграма прецендентів;
– діаграма класів;
– діаграма послідовності;

– діаграма кооперації;

– діаграма станів;

– структурна схема;

– функціональна схема.

Розроблено прототип інтерфейсу користувача, вирішено, які елементи та функції мають бути присутні.

Наступними були сформовані алгоритми до основних та додаткових функцій програми:

– алгоритм основного модулю *DCleaner*;

– алгоритм додаткового модулю *TCleaner*;

– алгоритм додаткового модулю *VCleaner*.

Проаналізувавши всі алгоритми, схеми та діаграми, наступними було розроблено основні та додаткові модулі у вигляді програмного коду:

– *DCleaner.py*;

– *EventHandler.py*;

– *Ext.py*;

– *TCleaner.py*;

– *VCleaner.py*.

Розглянувши приклади аналогових додатків для автоматизації було розроблено та стилізовано інтерфейс для найбільш зручного користування користувачами. Додано елементи до інтерфейсу, які підключають основні та додаткові функції програмного модулю.

Програмний модуль протестовано та критичних помилок при експлуатації не було виявлено.

Весь проект зібрано за допомогою додаткових утиліт *pyinstaller* та *NSIS*.

Практичне значення проектування полягає у наданні користувачу можливості більш продуктивно проводити час за комп’ютером, автоматизації процесів на ПК.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Learning Python, 5th Edition* [Електронний курс]. – 2013. – Режим доступу: https*://learning-python.com/about-lp5e.html* (дата звернення 22.05.2020). – Назва з екрана.

2. *Python tutorial* [Електронний курс]. – 2007. – Режим доступу: *https://habr.com/ru/post/31180/* (дата звернення 22.05.2020). – Назва з екрана.

3. *Learn python tutorial* [Електронний курс]. – 2007. – Режим доступу: *https://www.tutorialspoint.com/python/index.htm* (дата звернення 22.05.2020). – Назва з екрана.

4. *Datetime lib* [Електронний курс]. – 2020. – Режим доступу: *https://docs.python.org/3/library/datetime.html* (дата звернення 22.05.2020). – Назва з екрана.

5. *Watchdog tutorial* [Електронний курс]. – 2020. – Режим доступу: *https://pythonhosted.org/watchdog/* (дата звернення 22.05.2020). – Назва з екрана.

6. *Watchdog documentation* [Електронний курс]. – 2020. – Режим доступу: *https://pypi.org/project/watchdog/* (дата звернення 22.05.2020). – Назва з екрана.

7. *Path lib documentation* [Електронний курс]. – 2020. – Режим доступу: *https://docs.python.org/3/library/pathlib.html* (дата звернення 22.05.2020). – Назва з екрана.

8. *Time lib documentation* [Електронний курс]. – 2020. – Режим доступу: *https://docs.python.org/3/library/time.html* (дата звернення 22.05.2020). – Назва з екрана.

9. *Events in python* [Електронний курс]. – 2020. – Режим доступу: *https://www.python-course.eu/tkinter_events_binds.php* (дата звернення 22.05.2020). – Назва з екрана.

10. *Events in python* [Електронний курс]. – 2020. – Режим доступу: *https://www.python-course.eu/tkinter_events_binds.php* (дата звернення 22.05.2020). – Назва з екрана.

11. *Shutil lib* [Електронний курс]. – 2020. – Режим доступу: *https://docs.python.org/3/library/shutil.html* (дата звернення 22.05.2020). – Назва з екрана.

12. *PyQT tutorial* [Електронний курс]. – 2020. – Режим доступу: *https://www.tutorialspoint.com/pyqt/index.htm* (дата звернення 22.05.2020). – Назва з екрана.

13. *PyQT Designer tutorial* [Електронний курс]. – 2020. – Режим доступу: *https://pythonbasics.org/qt-designer-python/* (дата звернення 22.05.2020). – Назва з екрана.

14. *GUI tutorial* [Електронний курс]. – 2020. – Режим доступу: *https://opensource.com/resources/python/gui-frameworks* (дата звернення 22.05.2020). – Назва з екрана.

15. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63с.

16. ГОСТ 19.401-78. ЕСПД. Текст программы.

17. ГОСТ 19.402-78. ЕСПД. Описание программы.

18. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, програм данных и систем.

19. ДСТУ 3008-95 Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.

Додаток А

Вихідний код програмного модуля автоматизованого налаштування директорій «Dcleaner»

DCleaner.py:

```
import sys
sys.path.append(".")
from watchdog.observers import Observer
from time import sleep
from pathlib import Path
from EventHandler import EventHandler

 __name__ == '__main__':
        w_path = Path.home() / 'Desktop'
        d_root = Path.home() / 'Desktop/DCleaner'
        e_handler = EventHandler(w_path=w_path, d_root=d_root)

observer = Observer()
observer.schedule(e_handler, f'{w_path}', recursive=True)
observer.start()

try:
        while True:
                sleep(60)
except KeyboardInterrupt:
        observer.stop()
 observer.join(
```

EventHandler.py:

```
import shutil
from pathlib import Path
from datetime import date
from ext import ext_paths
def date_to_path(path: Path):
    d_path = path / f'{date.today().year}' / f'{date.today().month:02d}'
    d_path.mkdir(parents=True, exist_ok=True)
    return d_path
def r_file(source: Path, d_path: Path):
    if Path(d_path / source.name).exists():
        increment = 0
        while True:
            increment += 1
            new_name = d_path / f'{source.stem}_{increment}{source.suffix}'
            if not new_name.exists():
                return new_name
    else:
        return d_path / source.name
```

ext.py:

```
ext_paths = {
        # No name
        'noname':  'other/uncategorized',
        # audio
        '.aif':    'media/audio',
        '.cda':    'media/audio',
        '.mid': 'media/audio',
```

```
        '.midi':   'media/audio',
        '.mp3':    'media/audio',
        '.mpa':    'media/audio',
        '.ogg':    'media/audio',
        '.wav':    'media/audio',
        '.wma':    'media/audio',
        '.wpl':    'media/audio',
        '.m3u':    'media/audio',
    # text
        '.txt':    'text/text_files',
        '.doc':    'text/microsoft/word',
        '.docx':   'text/microsoft/word',
        '.odt ':   'text/text_files',
        '.pdf':    'text/pdf',
        '.rtf':    'text/text_files',
        '.tex':    'text/text_files',
        '.wks ':   'text/text_files',
        '.wps':    'text/text_files',
        '.wpd':    'text/text_files',
        # video
        '.3g2':    'media/video',
        '.3gp':    'media/video',
        '.avi':    'media/video',
        '.flv':    'media/video',
        '.h264':   'media/video',
        '.m4v':    'media/video',
        '.mkv':    'media/video',
        '.mov':    'media/video',
        '.mp4':    'media/video',
```

```
'.mpg':    'media/video',
'.mpeg':   'media/video',
'.rm':     'media/video',
'.swf':    'media/video',
'.vob':    'media/video',
'.wmv':    'media/video',
# images
'.ai':     'media/images',
'.bmp':    'media/images',
'.gif':    'media/images',
'.jpg':    'media/images',
'.jpeg':   'media/images',
'.png':    'media/images',
'.ps':     'media/images',
'.psd':    'media/images',
'.svg':    'media/images',
'.tif':    'media/images',
'.tiff':   'media/images',
'.cr2':    'media/images',
# internet
'.asp':    'other/internet',
'.aspx':   'other/internet',
'.cer':    'other/internet',
'.cfm':    'other/internet',
'.cgi':    'other/internet',
'.pl':     'other/internet',
'.css':    'other/internet',
'.htm':    'other/internet',
'.js':     'other/internet',
```

```
'.jsp':    'other/internet',
'.part':   'other/internet',
'.php':    'other/internet',
'.rss':    'other/internet',
'.xhtml':  'other/internet',
'.html':   'other/internet',
# compressed
'.7z':     'other/compressed',
'.arj':    'other/compressed',
'.deb':    'other/compressed',
'.pkg':    'other/compressed',
'.rar':    'other/compressed',
'.rpm':    'other/compressed',
'.tar.gz': 'other/compressed',
'.z':      'other/compressed',
'.zip':    'other/compressed',
# disc
'.bin':    'other/disc',
'.dmg':    'other/disc',
'.iso':    'other/disc',
'.toast':  'other/disc',
'.vcd':    'other/disc',
# data
'.csv':    'programming/database',
'.dat':    'programming/database',
'.db':     'programming/database',
'.dbf':    'programming/database',
'.log':    'programming/database',
'.mdb':    'programming/database',
```

```
'.sql':    'programming/database',
'.tar':    'programming/database',
'.xml':    'programming/database',
'.json':   'programming/database',
# executables
'.apk':    'other/executables',
'.bat':    'other/executables',
'.com':    'other/executables',
'.exe':    'other/executables',
'.gadget': 'other/executables',
'.jar':    'other/executables',
'.wsf':    'other/executables',
# fonts
'.fnt':    'other/fonts',
'.fon':    'other/fonts',
'.otf':    'other/fonts',
'.ttf':    'other/fonts',
# presentations
'.key':    'text/presentations',
'.odp':    'text/presentations',
'.pps':    'text/presentations',
'.ppt':    'text/presentations',
'.pptx':   'text/presentations',
# programming
'.c':      'programming/c&c++',
'.class':  'programming/java',
'.java':   'programming/java',
'.py':     'programming/python',
'.sh':     'programming/shell',
```

```
    '.h':      'programming/c&c++',
    # spreadsheets
    '.ods':    'text/microsoft/excel',
    '.xlr':    'text/microsoft/excel',
    '.xls':    'text/microsoft/excel',
    '.xlsx':   'text/microsoft/excel',
    # system
    '.bak':    'text/other/system',
    '.cab':    'text/other/system',
    '.cfg':    'text/other/system',
    '.cpl':    'text/other/system',
    '.cur':    'text/other/system',
    '.dll':    'text/other/system',
    '.dmp':    'text/other/system',
    '.drv':    'text/other/system',
    '.icns':   'text/other/system',
    '.ico':    'text/other/system',
    '.ini':    'text/other/system',
    '.lnk':    'text/other/system',
    '.msi':    'text/other/system',
    '.sys':    'text/other/system',
    '.tmp':    'text/other/system'
    }
```

TCleaner.py:

```
import os, shutil
from pathlib import Path
f = os.path.join(Path.home(), 'Downloads/Telegram Desktop')
```

```
for fn in os.listdir(f):
    fp = os.path.join(f, fn)
    try:
        if os.path.isfile(fp) or os.path.islink(fp):
            os.unlink(fp)
        elif os.path.isdir(fp):
            shutil.rmtree(fp)
    except Exception as e:
        print('Failed to delete files %s. Because: %s' % (fp, e))
```

VCleaner.py:

```
import os, shutil
from pathlib import Path
f = os.path.join(Path.home(), 'Documents/ViberDownloads')
for fn in os.listdir(f):
    fp = os.path.join(f, fn)
    try:
        if os.path.isfile(fp) or os.path.islink(fp):
            os.unlink(fp)
        elif os.path.isdir(fp):
            shutil.rmtree(fp)
    except Exception as e:
        print('Failed to delete files %s. Because: %s' % (fp, e))
```

Ui.py:

```
from PyQt5 import QtWidgets, QtCore, QtGui
from PyQt5.QtCore import QCoreApplication
```

```
from PyQt5.QtWidgets import QApplication, QMainWindow, QInputDialog,
QFileDialog, QVBoxLayout, QLabel
from PyQt5.QtGui import QIcon, QPixmap
import os, shutil
import sys
import signal
sys.path.append(".")
from pathlib import Path
import time


from watchdog.observers import Observer

from EventHandler import EventHandler


class MyWindow(QMainWindow):
    def open_dialog_box(self):
        folder = QFileDialog.getExistingDirectory(self, ("Open Directory"),
                                    "/home",
                                    QFileDialog.ShowDirsOnly)
        for filename in os.listdir(folder):
            file_path = os.path.join(folder, filename)
            try:
                if os.path.isfile(file_path) or os.path.islink(file_path):
                    os.unlink(file_path)
                elif os.path.isdir(file_path):
                    shutil.rmtree(file_path)
            except Exception as e:
```

```
            print('Failed to delete %s. Reason: %s' % (file_path, e))

    def sub(self):

        import subprocess
        subprocess.Popen(["C:/Users/Igor/Desktop/desktop_cleaner-
master/desktop_cleaner/venv/Scripts/python", "cleandesk.py"])


    def telegram(self):
        import subprocess
        subprocess.Popen(["C:/Users/Igor/Desktop/desktop_cleaner-
master/desktop_cleaner/venv/Scripts/python", "telegram.py"])

    def viber(self):
        import subprocess
        subprocess.Popen(["C:/Users/Igor/Desktop/desktop_cleaner-
master/desktop_cleaner/venv/Scripts/python", "viber.py"])

    def __init__(self):
        super(MyWindow,self).__init__()
        self.initUI()

    def button_start(self):
        self.label.setText("Cleaning your desktop...")
        self.label.setGeometry(QtCore.QRect(400, 65, 400, 100))
        self.update()

    def telegram_start(self):
```

```
        self.label.setText("Telegram Cache cleared.")
        self.label.setGeometry(QtCore.QRect(400, 65, 400, 100))
        self.update()

    def viber_start(self):
        self.label.setText("Viber Cache cleared.")
        self.label.setGeometry(QtCore.QRect(400, 65, 400, 100))
        self.update()

    def versions_start(self):
        self.label.setText("v0.1 by Rabin Igor")
        self.label.setGeometry(QtCore.QRect(400, 65, 400, 100))
        self.update()

    def searcher(self):
        self.open_dialog_box()



    def initUI(self):
        self.setGeometry(200, 200, 800, 800)
        self.setWindowTitle("DCleaner")
        self.setWindowIcon(QIcon('logos.png'))
        self.setStyleSheet("background-color: rgb(192, 192, 192);")


        self.imagePath = "C:/Users/Igor/Desktop/desktop_cleaner-
master/desktop_cleaner/log.png"
        self.label1 = QtWidgets.QLabel(self)
```

```
        self.label1.setGeometry(0, 0, 300, 150)

        self.image = QtGui.QImage(self.imagePath)
        self.pixmapImage = QtGui.QPixmap.fromImage(self.image)
        self.label1.setPixmap(self.pixmapImage)
        self.label1.setScaledContents(True)

        #self.imagePath1 = "C:/Users/Igor/Desktop/desktop_cleaner-
master/desktop_cleaner/version.png"
        #self.label2 = QtWidgets.QLabel(self)
        #self.label2.setGeometry(500, 0, 500, 100)



        #self.image1 = QtGui.QImage(self.imagePath1)
        #self.pixmapImage1 = QtGui.QPixmap.fromImage(self.image1)
        #self.label2.setPixmap(self.pixmapImage1)
        #self.label2.setScaledContents(True)







        self.label = QtWidgets.QLabel(self)
        self.label.setText("Waiting for function.")
        #self.label.move(0,110)
        self.label.setGeometry(QtCore.QRect(400, 15, 400, 100))
        self.label.setStyleSheet("color: black;""font-size:36px;")
```

```
        self.b1 = QtWidgets.QPushButton(self)
        self.b1.setText("Clean my desktop")
        self.b1.setGeometry(QtCore.QRect(0, 150, 800, 450))
        self.b1.setStyleSheet("QPushButton""{""font-size:20px;""background-color:
rgb(153,255,255);""}""QPushButton::pressed""{"
                              "background-color : lightblue;"
                              "}")
        self.b1.clicked.connect(self.sub)
        self.b1.clicked.connect(self.button_start)

        self.b3 = QtWidgets.QPushButton(self)
        self.b3.setText("TCacheClean")
        self.b3.setGeometry(QtCore.QRect(0, 600, 400, 100))
        self.b3.setStyleSheet("QPushButton""{""font-size:20px;""background-color:
rgb(153,255,255);""}""QPushButton::pressed""{"
                              "background-color : lightblue;"
                              "}")
        self.b3.clicked.connect(self.telegram)
        self.b3.clicked.connect(self.telegram_start)

        self.b4 = QtWidgets.QPushButton(self)
        self.b4.setText("VCacheClean")
        self.b4.setGeometry(QtCore.QRect(400, 600, 400, 100))
        self.b4.setStyleSheet("color: rgb(219,2,2);")
        self.b4.setStyleSheet("QPushButton""{""font-size:20px;""background-color:
rgb(153,255,255);""}""QPushButton::pressed""{"
                              "background-color : lightblue;"
                              "}")
```

```
        self.b4.clicked.connect(self.viber)
        self.b4.clicked.connect(self.viber_start)

        self.b5 = QtWidgets.QPushButton(self)
        self.b5.setText("DirectCleanUp")
        self.b5.setGeometry(QtCore.QRect(0, 700, 400, 100))
        self.b5.setStyleSheet("QPushButton""{""font-size:20px;""background-color:
rgb(153,255,255);""}""QPushButton::pressed""{"
                              "background-color : lightblue;"
                              "}")
        self.b5.clicked.connect(self.open_dialog_box)

        self.b2 = QtWidgets.QPushButton(self)
        self.b2.setText("Finish and Exit")
        self.b2.setGeometry(QtCore.QRect(400, 700, 400, 100))
        self.b2.setStyleSheet("background-color: rgb(153,255,255);""font-size:20px;")
        self.b2.clicked.connect(self.close)



        self.b6 = QtWidgets.QPushButton(self)
        self.b6.setText("Version")
        self.b6.setGeometry(QtCore.QRect(730, 0, 70, 20))
        self.b6.setStyleSheet("background-color: rgb(153,255,255);""font-size:20px;")
        self.b6.clicked.connect(self.versions_start)

    def update(self):
        self.label.adjustSize()

def window():
```

```
    app = QApplication(sys.argv)
    win = MyWindow()
    win.show()
    sys.exit(app.exec_())

window()
```