КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

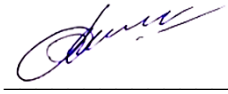
Кваліфікаційна робота
на здобуття ступеня магістра
за спеціальності 122 «Комп'ютерні науки»
на тему:

РОЗРОБКА СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ У СФЕРІ УПРАВЛІННЯ КАДРАМИ

Виконав студент 2-го курсу магістратури
Нікіта ШУЛЯК

(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Олексій ТКАЧЕНКО



(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент 

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теорії та технології програмування

«_08_» _____травня_____ 202_3_ р.,
протокол №_16_

Завідувач кафедри
Микола НІКІТЧЕНКО _____

(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 61 сторінок, 12 ілюстрацій, 14 джерел посилань.

УПРАВЛІННЯ КАДРАМИ, СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ, АВТОМАТАЗИЦІЯ ПРОЦЕСІВ, RPA, C#, .NET, MYSQL, EXCEL, OUTLOOK, POWERBI DESKTOP, СЕРВІСИ.

Об’єктом дослідження є сфера управління кадрами на підприємстві.

Предмет дослідження - інформаційні системи автоматизації бізнес-процесів у сфері управління кадрами.

Мета роботи полягає у практичному ознайомлені з методами та засобами нової технології RPA (Robotic Process Automation) та безпосередньої реалізації програмного засобу для розв’язування задачі ефективного процесу прийняття рішень у сфері управління кадрами засобами RPA.

Методи дослідження: загальнонаукові, об’єктно-орієнтоване програмування, емпіричний.

Інструменти розробки: середовище розробки Visual Studio 2022, мова програмування C#, СУБД MySQL.

Результати роботи: реалізовано систему, яка допомагає керівництву компанії приймати рішення про роботу працівників і команди у вигляді генерації звіту про їх успішність.

Система може застосовуватись у компаніях, які спеціалізуються на технологіях RPA і їх впровадженнях.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ІТ - інформаційні технології;

БД - база даних;

ПК - персональний комп’ютер;

ПЗ - програмне забезпечення;

AI - (Artificial Intelligence) – штучний інтелект;

IDE - (Integrated Development Environment) - інтегроване середовище розробки;

SQL - (Structured Query Language) - мова структурованих запитів;

OS - (Operation System) - операційна система;

RPA - (Robotic Process Automation) - автоматизація бізнес процесів;

OCR - (Optical Character Recognition) – оптичне розпізнавання символів;

СУБД - система управління базами даних;

UI - (User Interface) – інтерфейс користувача;

DI - (Dependency Injection) – методологія розділення відповідальності;

AWS - (Amazon Web Services) – публічне хмарне сховище, яке підтримує компанія Amazon.

АСУК - автоматизовані системи управління кадрами;

SOLID - об’єктно-орієнтована методологія проектування.

ВСТУП

Актуальність теми кваліфікаційної роботи. Нині використання інформаційних технологій та інструментів є необхідною та невід'ємною складовою діяльності будь-якого підприємства, зокрема для оцінки та визначення структури бізнес-процесів.

Розвиток інформаційних технологій надає підприємствам більші можливості для збору та обробки даних, однак це також викликає потребу в потужніших інструментах для виконання різноманітних задач. Наприклад, впровадження систем відслідковування та аналізу даних, дозволяє підприємствам підвищити ефективність своєї діяльності та приймати обґрунтовані рішення на основі фактів.

Однак, це також створює нові виклики для бізнесу, який повинен впроваджувати складніші технології та розробляти власні методики аналізу та використання даних, щоб бути конкурентоспроможним.

Сучасний облік кадрів вимагає використання новітніх технологій та методик для отримання звітів. Основна мета цього процесу полягає у зборі, обробці та аналізі даних про кадровий склад організації, щоб забезпечити ефективне управління персоналом.

Однією з новітніх методик отримання звітів є використання автоматизованих систем управління кадрами (АСУК). Ці системи забезпечують повну автоматизацію процесів управління персоналом, включаючи збір та обробку даних про кадровий склад. Вони дозволяють отримувати звіти у режимі реального часу, що значно полегшує прийняття рішень.

Більшість українських підприємств зосереджують свою увагу на розробці і впровадженні новітніх технологій та методів управління, що сприяє розвитку схожих розробок. Цей процес є дуже важливим для українського бізнесу, оскільки він дозволяє підприємствам стати більш конкурентоспроможними на міжнародному ринку. Тому, актуальність даної теми буде зберігатися в найближчі

десятиліття, оскільки постійний розвиток технологій створює нові можливості для підприємств, а також вимагає постійного оновлення та удосконалення вже існуючих розробок.

Об’єктом дослідження даної роботи є сфера управління кадрами на підприємстві.

Предмет дослідження - інформаційні системи автоматизації бізнес-процесів у сфері управління кадрами.

Мета роботи полягає у практичному ознайомлені з методами та засобами нової технології RPA (Robotic Process Automation) та безпосередньої реалізації програмного засобу для розв’язування задачі ефективного процесу прийняття рішень у сфері управління кадрами засобами RPA.

Для досягнення бажаної цілі вирішуються наступні задачі:

- аналіз предметної області;
- огляд існуючих програмних продуктів;
- моделювання предметної області з використанням об'єктно-орієнтованого підходу;
- вибір середовища розробки;
- програмна реалізація із написанням сервісів для роботи з Excel (вхідні дані), Outlook (нотифікація, зчитування листів та отримання прикріплених до них файлів), SQL (зберігання, накопичення, аналіз даних) та PowerBi (генерування динамічних репортів).

Можливі сфери застосування. Розробка, яка наведена у дипломній роботі рекомендується до впровадження у компаніях, які спеціалізуються на технологіях RPA.

Результати даної роботи були представлені на науково-практичній конференції студентів і аспірантів «Теоретичні та прикладні аспекти розробки комп’ютерних систем 2023». [1]

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ «СИСТЕМИ АВТОМАТИЗАЦІЇ БІЗНЕС-ПРОЦЕСІВ У СФЕРІ УПРАВЛІННЯ КАДРАМИ»

1.1. Майбутнє роботизації: переваги та виклики впровадження RPA

З 2020 року в Кремнієвій долині однією з найпопулярніших професій стала RPA, оскільки венчурні капіталісти інвестують у цю категорію величезні суми. Причина проста: кількість постачальників програмного забезпечення зростає із приголомшливою швидкістю.

Дійсно, згідно з нещодавнім опитуванням LinkedIn, RPA є другою категорією кар’єр, яка швидко розвивається (у 2019 році вона зросла на 40%). У звіті зазначається: «Кар'єра в галузі робототехніки може сильно відрізнятися між програмними (software) та апаратними (hardware) ролями, і наші дані показують, що інженери, які працюють як з віртуальними, так і з фізичними ботами, стають дедалі популярнішими». [2]

Термін RPA досить неоднозначний і інтерпретувати його можна по різному. Головною причиною цього є те, що він був придуманий у 2012 році, коли категорія все ще розвивалася. Наприклад, слово "роботизований" не стосується фізичного робота - натомість йдеться про програмного робота (або бота), який може автоматизувати дії людини на робочому місці. Навіть слово "процес" не є особливо описовим. Кращою альтернативою було б "завдання", які є окремими елементами дій, що є частиною процесу. Бот може бути доставлений через хмарне середовище або через програмне забезпечення, яке можна завантажити.

Що ж таке RPA? Власне кажучи, RPA передбачає використання ботів, які виконують набір певних дій або завдань, наприклад, таких, як наведені нижче:

- вирізання та вставка інформації з одного додатка в інший;
- відкриття вебсайту та вхід у нього;
- відкриття електронної пошти та вкладень;
- читання/запис бази даних;
- витяг вмісту з форм або документів;

- використання обчислень і робочих процесів.

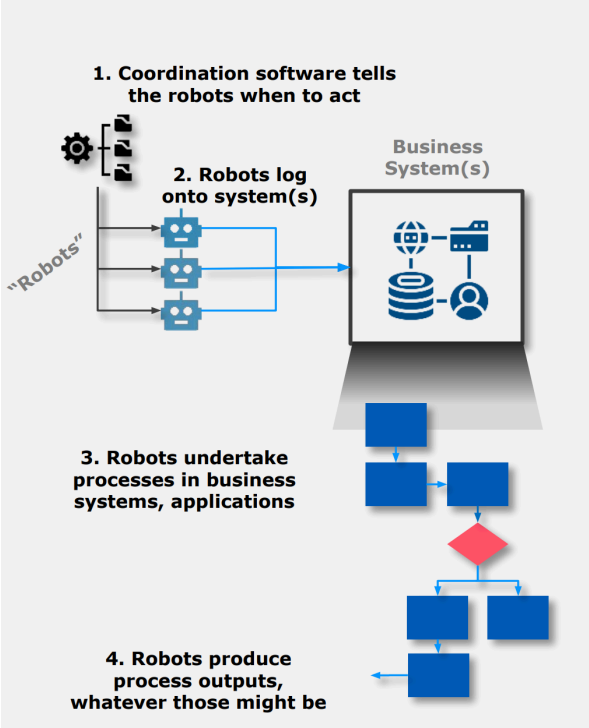


Рисунок 1. Загальний робочий процес RPA

Такі речі можуть звучати дещо буденно та спрощено. Але в цьому і вся суть. RPA зосереджується на тих завданнях, які є марною тратою часу для працівників (вони можуть витрачати час на складніші задачі). Використання слова «автоматизація» в RPA дійсно є влучним. Саме вона лежить в основі функціональності напрямку.

Питання, яке виникає у більшості людей «в чому різниця між RPA та іншими формами автоматизації? Чи не є ця технологія просто як макрос в Excel?». Насправді це не є зовсім так. Перш за все, макрос призначений лише для певної програми. Але з RPA система може бути використана практично для будь-чого: чи то на ПК, чи навіть на мейнфреймі. Далі, RPA може записувати дії людини, щоб допомогти створити автоматизацію. Деякі системи використовують складні технології, такі як AI, щоб допомогти в цьому. Нарешті, RPA-платформа стане цінним сховищем знань про те, як виконується робота в організації. Це може дати

уявлення про те, як покращити робочі процеси та процеси, що можуть сприяти подальшому підвищенню ефективності.

Існують різні типи підходів до RPA. Частково це пов'язано з тим, що технологія продовжує розвиватися. Постачальники також шукають способи переосмислити RPA, щоб допомогти їм виділитися на ринку.

На найвищому рівні, ми можемо розділити підходи на наступні:

1) RPA з наглядом (Attended) (який можна назвати роботизованою автоматизацією робочого столу або RDA) - це перша форма RPA, яка з'явилася приблизно у 2003 році. RPA з наглядом означає, що ПЗ постачає співпрацю з людиною для виконання певних завдань. Яскравим прикладом може бути кол-центр, де представник може доручити системі RPA пошук інформації під час розмови з клієнтом.

2) RPA без нагляду (Unattended) - ця технологія була другим поколінням RPA. За допомогою RPA без нагляду можна автоматизувати процес без участі людини - тобто, бот запускається, коли відбуваються певні події, наприклад, коли клієнт надсилає інвойс на електронну пошту.

3) Інтелектуальна автоматизація процесів або IPA (її також можна назвати когнітивним RPA) - це останнє покоління технології RPA, яка використовує штучний інтелект, щоб дозволити системі навчатися з часом (прикладом може бути інтерпретація документів, таких як рахунки-фактури). Завдяки цьому втручання людини може бути ще меншим, оскільки програмне забезпечення RPA використовуватиме власні знання та судження для прийняття рішень.

Важливо розуміти ці відмінності, оскільки деякі системи RPA можуть спеціалізуватися на певному підході. [3]

Основною перевагою RPA є операційна ефективність, яка сприяє підвищенню якості обслуговування, скороченню часу на виконання замовлення і зниженню витрат.

Нище наведемо ключові переваги RPA:

- **Швидкість**: RPA виконує завдання в 4-10 разів швидше, ніж людина, дозволяючи персоналу краще організовувати свою роботу.
- **Надійність**: боти RPA виконують лише ті дії, які на них покладені, вони ніколи не помиляються, не прораховують і не мають залежності від настрою (за умови правильних вхідних даних і бізнес-правил, вихідні дані будуть правильними).
- **Скорочення витрат і повернення інвестицій (ROI)**: боти дешевші, швидші, доступні 24/7 і можуть підвищити продуктивність і якість даних, що призводить до зниження операційних витрат, а, отже, до більшої користі для громади. Більшість організацій повідомляють про скорочення витрат на 20-30% і 30-50% рентабельності інвестицій у проєкти RPA.
- **Можливість аудиту**: боти збирають інформацію про все, що вони роблять, що дозволяє проводити повну ретроспективну перевірку кожної здійсненої ними операції.
- **Продуктивність**: доступні 100% часу 24/7 - ботам ніколи не потрібно буде спати, вони виконуватимуть свою роботу, коли це необхідно.
- **Легке впровадження**: боти працюють з наявними додатками та системами організації, що дозволяє прискорити цифрову трансформацію.
- **Гнучкість**: ботів легко планувати і призначати для автоматизації після їх створення. Гнучкість ботів забезпечує перевагу у вигляді доповнюваності ботів. Вони можуть бути відносно швидко оновлені, якщо вимоги до процесу змінюються.
- **Задоволеність працівників**: передаючи роботам рутинні завдання, працівники зосереджуються на тому, що люди роблять найкраще (думають, приймають рішення, виробляють і створюють). Це підвищує стійкість персоналу - більше часу на трансформаційну роботу та впровадження нових методів роботи.
- **Зменшення плинності кадрів**: Підвищення рівня задоволеності персоналу призводить до зменшення відтоку кадрів в організаціях. Компанії

все частіше зосереджуються на цьому як на головній перевазі, яку вони прагнуть отримати від RPA. [4]

1.2. RPA навички та технології

Традиційний підхід до IT-систем полягає у використанні локальних технологій. Це означає, що компанія купує і встановлює власне обладнання та програмне забезпечення у власному дата-центрі.

Деякі з переваг включають в себе наступні:

- компанія має повний контроль над усім. Це особливо важливо для регульованих галузей, які вимагають високого рівня безпеки та конфіденційності;
- завдяки локальному програмному забезпеченню можна краще налаштувати рішення відповідно до унікальних потреб та IT-політик компанії.

Однак локальна технологічна модель також має серйозні проблеми. Однією з найбільших є вартість, яка часто передбачає великі затрати. Крім того, існує постійна потреба в технічному обслуговуванні, модернізації та моніторингу. Загалом, це означає, що IT-відділ може витрачати цінний час на непрофільну діяльність. І, нарешті, використання точкових додатків, таких як Excel, може призвести до фрагментації середовища, в якому стає важко централізувати дані через велику кількість файлів, розпорошених по всій організації.

Через це компанії розглядають інший підхід - власне, хмарні обчислення. Компанія Salesforce.com створила один з перших бізнес-додатків у даній галузі, що дозволив користувачам використовувати програмне забезпечення через веббраузер. Головна ідея полягала у тому, щоб зробити програмне забезпечення доступним для широкого кола користувачів, полегшити його придбання та використання, а також знизити вартість інсталяції, обслуговування та постійних оновлень. Замість традиційного продажу програмних пакетів на компакт-дисках, які встановлювалися від 6 до 18 місяців та потребували значних витрат на апаратне забезпечення та мережу, вони запропонували модель під назвою "хмарних обчислень", де програмне

забезпечення надавалося як послуга через Інтернет. Компанії мали змогу сплачувати щомісячну абонентську плату за послуги, які вони використовують, а ці послуги негайно надавалися їм у хмарі.

Звісно, жодна технологія не є ідеальною, і хмарне програмне забезпечення також має свої недоліки, які важливо враховувати:

- зменшення контролю над платформою може призвести до появи більшої кількості вразливостей щодо безпеки та може порушити конфіденційність даних;
- збої можуть відбуватися, і вони можуть бути дуже дорогими та руйнівними для підприємств, які покладаються на надійність хмарних послуг;
- вартість хмарних обчислень не завжди є дешевою. Наприклад, однією з найбільших скарг на Salesforce.com є висока вартість їхніх послуг.

Існують різні підходи до хмарних сховищ, такі як:

- публічна хмара, до якої можна отримати доступ з віддалених серверів, таких як AWS, Salesforce.com та Microsoft. Цей підхід дозволяє використовувати багатокористувацьку інфраструктуру, що забезпечує економію від масштабу;
- приватна хмара, коли компанія володіє центром обробки даних. Цей підхід дає більший контроль і безпеку, але не забезпечує економію від масштабу;
- гібридна хмара, яка поєднує в собі публічну та приватну хмари. Наприклад, публічна хмара може використовуватися для менш критичних функцій, тоді як приватна - для більш важливих.

Щодо впливу хмарної технології на RPA, слід зазначити, що вона може стикатися з розподіленими додатками, що ускладнює роботу платформи, особливо якщо компанія використовує власні програми на хмарному сервісі.

Більшість платформ RPA починали свій шлях як локальне програмне забезпечення і зазвичай не переходили на хмарну інфраструктуру до недавнього часу. Це може здатися дивним, оскільки хмарні обчислення вже існують протягом

деякого часу і здається, що це стандартний підхід для компаній, що займаються розробкою програмного забезпечення. Але для RPA створення хмарних систем не є легкою задачею, оскільки потребуються глибокі інтеграції з багатьма програмами та середовищами.

У деяких випадках можна завантажити локальну систему RPA в хмарний сервіс, такий як AWS. Хоча це має свої переваги, воно не є нативним для хмарної інфраструктури. Це пояснюється тим, що все одно потрібно оновлювати та підтримувати програмне забезпечення.

Загалом, для розвитку в галузі RPA необхідно мати широкий спектр компетенцій, включаючи такі області технологій, як:

- вебтехнології,
- мови програмування та платформи низького коду,
- оптичне розпізнавання символів (OCR),
- робота з базами даних,
- інтерфейси прикладного програмування (API),
- штучний інтелект (ШІ),
- когнітивна автоматизація,
- методології розробки програмного забезпечення, такі як Agile, Scrum, Kanban і Waterfall,
- DevOps (це підхід до розробки та впровадження програмного забезпечення, що поєднує в собі розробку (Dev) та операції (Ops). DevOps спрямований на автоматизацію процесів розробки та впровадження програмного забезпечення з метою забезпечення швидкості, надійності та безпеки).

Важливо пам'ятати, що для успішної роботи в цій галузі необхідно постійно оновлювати свої знання та навички.

1.3. Готові системи і рішення, для впровадження RPA

Blue Prism, UiPath та Automation Anywhere є провідними гравцями на ринку RPA-рішень.

Blue Prism - світовий лідер у галузі інтелектуальної автоматизації, що покращує роботу співробітників на підприємствах. Blue Prism допомагає організаціям підвищити операційну ефективність та гнучкість, виконуючи автоматизацію завдань для окремих працівників. Blue Prism також забезпечує аналітичні засоби для моніторингу та відстеження процесів.

Основні компоненти функціоналу Blue Prism наступні:

- Blue Prism має вбудоване середовище розробки, яке дозволяє розробляти, тестувати та налагоджувати ботів RPA. Це середовище містить набір інструментів, які дозволяють розробникам створювати складні логічні взаємодії між ботами та іншими системами. Розробка відбувається на високо кодовому рівні, де потрібно перетягувати блоки із діями та впорядковувати їх.
- Рішення надає можливість створювати ботів RPA, які можуть взаємодіяти з іншими програмами та API, виконувати рутинні та повторювані завдання, збирати, обробляти та передавати дані між системами.
- Blue Prism має додаткові інструменти, які дозволяють розширювати можливості платформи. Наприклад, це можуть бути інструменти для моніторингу, аналізу та відстеження роботи ботів, їх продуктивності та коректності, інтеграції з іншими системами.
- Blue Prism має систему керування правами доступу, яка дозволяє встановлювати рівень доступу до функціоналу та інформації для різних груп користувачів.
- Рішення має можливість відстежувати та аналізувати продуктивність ботів RPA із наданням функціоналу для візуалізації цих даних в звітах та дашбордах. [5]

UiPath - це інший провідний RPA-постачальник, який дозволяє підприємствам автоматизувати процеси бізнесу за допомогою цифрових роботів. UiPath забезпечує візуальний інтерфейс користувача для програмування роботів та має ряд інструментів для автоматизації процесів, зокрема засоби роботизації процесів, інструменти аналітики та інструменти для забезпечення безпеки.

UiPath Orchestrator - це централізована система керування ботами RPA, яка надає інструменти для керування, моніторингу та аналізу роботи ботів. Orchestrator дозволяє забезпечувати безперервну роботу процесів, автоматизувати завдання керування ботами та управляти ресурсами.

Основні функції Orchestrator UiPath:

- Orchestrator надає інструменти для керування ботами, такі як додавання/видалення ботів, запуск/зупинка ботів, призупинення/відновлення завдань тощо.
- Orchestrator дозволяє створювати та розподіляти завдання між ботами. Завдання можуть бути призначені для виконання на конкретному боті або на будь-якому вільному боті.
- Orchestrator надає інструменти для розкладування завдань на певний час або для виконання в певний день.
- Orchestrator дозволяє моніторити роботу ботів та перевіряти стан завдань в режимі реального часу. Інформація про стан завдань та ботів може бути відображена в зручній для користувача формі, такій як таблиці, діаграми, дашборди.
- Orchestrator забезпечує захист від несанкціонованого доступу до системи та даних, включаючи систему автентифікації та авторизації, засновану на ролях та правах доступу.
- Orchestrator надає інструменти для аналізу та відстеження продуктивності ботів, які дозволяють стежити за виконанням завдань та генерувати звіти про роботу ботів. [6]

Automation Anywhere - це інноваційна RPA-платформа, яка дозволяє автоматизувати бізнес-процеси та забезпечує безперервність роботи підприємств.

Особливості Automation Anywhere:

- Automation Anywhere надає візуальний редактор, який дозволяє користувачам створювати автоматизовані робочі процеси шляхом перетягування та випадання блоків коду.
- Automation Anywhere побудована на розподіленій архітектурі, що дозволяє працювати з різними додатками та системами в різних місцях.
- Automation Anywhere має розширюваність для створення власних модулів та розширень.
- Automation Anywhere надає інструменти для керування та моніторингу ботів RPA. Менеджер ботів дозволяє керувати ботами, перевіряти їх стан та відслідковувати їх виконання завдань.
- Automation Anywhere має бібліотеку макрофункцій, що дозволяє користувачам створювати більш складні автоматизовані процеси з використанням широкого спектру функцій та можливостей.
- Automation Anywhere має можливості для автоматизації важких та складних завдань, таких як операції з даними, обробка документів, збір та аналіз даних тощо.
- Automation Anywhere дозволяє працювати з різними форматами файлів та різними додатками.
- Automation Anywhere забезпечує безпеку від несанкціонованого доступу до системи та даних, включаючи систему автентифікації. [7]

Всі три RPA-постачальники дозволяють підприємствам автоматизувати бізнес-процеси за допомогою цифрових роботів. Кожен з цих постачальників має свої переваги та особливості.

Blue Prism відомий своєю масштабованістю, надійністю та безпекою. Його система базується на .NET, що дозволяє розробникам використовувати різноманітні мови програмування, включаючи C#, VB.NET, Python та інші. Крім того, Blue Prism має вбудовані засоби для безпеки та захисту даних.

UiPath, з іншого боку, відомий своєю простотою та легкістю використання. Його система базується на .NET та має велику кількість вбудованих компонентів та

інструментів, які дозволяють швидко та легко програмувати роботів. Крім того, UiPath має безліч підключень до різних систем та програм, що дозволяє легко інтегрувати його з існуючими процесами бізнесу.

Automation Anywhere забезпечує широкі можливості для автоматизації процесів бізнесу. Його система базується на власному мовному середовищі, що дозволяє програмістам створювати складні роботи з високим рівнем налаштування. Крім того, Automation Anywhere має велику кількість вбудованих компонентів та інструментів, які дозволяють швидко та легко програмувати роботів.

Усі три постачальники RPA мають інтелектуальні засоби для розпізнавання та обробки даних, забезпечують розширення можливостей за допомогою API, підтримують різні моделі та формати даних, мають інструменти для аналізу даних та забезпечення безпеки.

Вибір конкретного RPA-постачальника залежить від потреб і вимог бізнесу. При виборі постачальника RPA необхідно враховувати такі чинники, як рівень складності та кількість процесів, які необхідно автоматизувати, тип інтерфейсу користувача, наявність необхідних інтеграцій з існуючими системами та програмним забезпеченням, рівень безпеки та надійності, вартість та доступність технічної підтримки.

Окрім Blue Prism, UiPath та Automation Anywhere, на ринку існує ще багато інших постачальників RPA, таких як WorkFusion, Kofax, NICE, OpenSpan, і т.д. Кожен з них має свої переваги та особливості, тому необхідно ретельно досліджувати різні варіанти, щоб знайти той, який найкраще відповідає вашим потребам.

Загалом, RPA є потужним інструментом для автоматизації рутинних та повторюваних бізнес-процесів, що дозволяє зменшити ризик помилок та збільшити ефективність та продуктивність роботи співробітників. Вибір постачальника RPA повинен бути здійснений на основі конкретних потреб вашого бізнесу, але незалежно від цього, впровадження RPA може допомогти зменшити навантаження на працівників та покращити якість роботи бізнесу в цілому.

Один з найбільших недоліків рішень Blue Prism, UiPath, Automation Anywhere - це велика вартість ліцензій. Ці продукти є досить дорогими для великих та середніх компаній, особливо якщо вони потребують використання більшого числа ботів або користувачів.

Також варто враховувати, що вартість ліцензій може збільшуватися залежно від потреб користувача. Наприклад, якщо потрібні додаткові функції або додаткова підтримка, то вартість може значно збільшитися.

Крім того, велика вартість ліцензій може стати перешкодою для менших компаній або стартапів, які можуть потребувати рішень з автоматизації процесів, але не можуть дозволити собі великі витрати на придбання ліцензій.

Таким чином, хоча продукти Blue Prism, UiPath, Automation Anywhere можуть мати багато переваг, велика вартість ліцензій може бути недоліком для багатьох компаній та користувачів, особливо для тих, хто потребує додаткових функцій та підтримки.

Саме у цьому аспекті, реалізація, яка наведена у цій роботі, має найбільшу перевагу. Використання відкритих бібліотек дозволяє знизити витрати на розробку та підтримку роботів, оскільки вони можуть бути використані безкоштовно або за невелику плату. Крім того, відкриті бібліотеки часто оновлюються та доповнюються спільнотою розробників, що дозволяє використовувати оновлення та нові функції без додаткових витрат на розробку.

1.4. Process Mining та застосування технології RPA у сфері управління кадрами

Проблема впровадження RPA постала перед багатьма бізнесами. З того часу, як компанії зрозуміли усю потужність RPA, виникало питання – які саме процеси слід автоматизовувати в першу чергу? Чи існує якийсь засіб, який міг би знаходити місця для автоматизації, обраховувати ефективність і надавати аналітичний звіт запровадження технології RPA? Усе це вирішує Process Mining.

Отже, давайте поглянемо на процес інтелектуального аналізу на високому рівні. В основі цього лежить лог (log – журнал подій), який являє собою

інформацію, що генерується, коли програма або мережа виконує дію. Це може бути щось на зразок наступного:

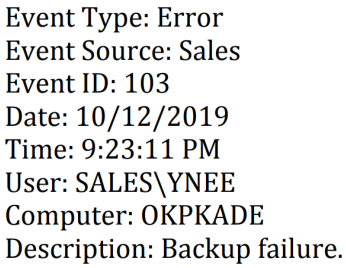


Рисунок 2. Приклад логу

Process Mining аналізує ці логи (яких можуть налічуватись мільйони), щоб виконувати наступні дії:

Виявлення: процес розглядається на основі «як є», що виконується за допомогою альфа-алгоритму або навіть методів, які застосовуються в соціальних мережах. Незалежно від методу, ви можете побачити поточний стан робочих процесів, і ця інформація може бути використана для створення візуальних репрезентацій, які можуть суттєво змінити спосіб розуміння та оптимізації процесів. Це безумовно може привести до нових відкриттів та відкрити вам очі щодо вузьких місць та проблем в організації, які можуть бути виявлені швидко.

Відповідність: це включає в себе налаштування моделі для процесів, а потім надання можливості аналізу процесу знаходити відхилення. Також будується дерево рішень, яке розглядає безліч шляхів. Використовуючи відповідність, ви зможете отримати ще краще розуміння та оптимізацію процесів.

Аналітика: після того, як дані будуть зібрані, інтелектуальний аналіз процесів може розглядати «майбутні» процеси. Це базується на дослідженні основних причин робочих процесів. У результаті ви можете отримати уявлення про те: Чи витрачається занадто багато часу на певні види діяльності чи партнерів? Чи достатньо навчання в окремих сферах? Чому один відділ ефективніший за інший?

Важливо відзначити, що process mining не є статичним. Навпаки, технологія постійно аналізує логи. Іншими словами, є постійні вдосконалення та ідеї.

Одним із готових рішень Process Mining, яке користується популярністю, виступає Celonis Process Mining [8]. Воно дозволяє підприємствам виявляти, аналізувати та оптимізовувати свої бізнес-процеси.

Celonis Process Mining використовує машинне навчання та інші технології для автоматичного виявлення та візуалізації процесів. Це дозволяє підприємствам побачити свої процеси з іншого ракурсу та знайти способи для покращення їх продуктивності. Крім того, Celonis Process Mining може автоматично створювати звіти, давати рекомендації та вести моніторинг ефективності процесів у режимі реального часу.

Узагалі, Celonis Process Mining допомагає компаніям збільшити продуктивність, знизити витрати та покращити якість своїх продуктів та послуг. Це робить її корисним рішенням для будь-якої компанії, яка бажає покращити свої бізнес-процеси.

Технологія RPA може бути дуже корисною для автоматизації багатьох рутинних процесів, що пов'язані з управлінням кадрами в організації. Для прикладу, RPA може допомогти автоматизувати процеси ведення списку працівників, управління їх контактними даними, відслідковування робочого часу, підготовку звітів про кадровий склад і багато іншого.

Одна з головних переваг використання RPA в управлінні кадрами полягає в тому, що це може допомогти знизити кількість помилок, пов'язаних з ручним введенням даних, що може призвести до серйозних наслідків, таких як невідповідність звітності чи виплата неправильної зарплати. Автоматизовані процеси RPA можуть бути налаштовані для виконання задач точно й швидко, що зменшує можливість помилок.

Крім того, використання RPA в управлінні кадрами може зменшити витрати на робочу силу, оскільки автоматизовані процеси можуть замінити повторювані процеси, що раніше виконувались людьми. Заощаджені кошти можуть бути використані на інші цілі, що дозволяє підвищити ефективність роботи організації в цілому.

У сфері прийняття рішень, RPA може бути корисним інструментом для обробки даних і автоматичного аналізу кадрових звітів, що може допомогти виробляти більш обґрунтовані рішення. Наприклад, RPA може автоматично аналізувати дані про відсутність на роботі та перевіряти, чи є звіти про відсутність офіційно затвердженими, що зменшить ризик помилок і забезпечить більш точне прийняття рішень.

Крім того, за допомогою RPA можна виконувати аналіз робочої продуктивності та оцінювати результати працівників, що може допомогти виробляти об'єктивніші рішення про підвищення зарплати, переведення на нову посаду та інше.

Нарешті, RPA може бути корисною для підвищення ефективності процесів навчання та розвитку працівників. Наприклад, автоматизовані процеси можуть бути налаштовані для проведення онлайн-тестування та оцінки знань працівників, що допоможе визначити їхні сильні та слабкі сторони та розробити індивідуальні плани навчання.

Узагалі, застосування технології RPA може допомогти забезпечити більш точне та ефективне управління кадрами в організації, зменшити кількість помилок, збільшити ефективність роботи та знизити витрати на робочу силу.

РОЗДІЛ 2. СТВОРЕННЯ ТА ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Загальна архітектура системи

Структура архітектури RPA-рішення для системи підтримки прийняття рішень в сфері управління кадрами включає такі складові:

1) **Підготовка даних**: цей етап включає збір та обробку даних про кадри, що включають інформацію про наймання, звільнення, відпустки, час витрачений на проекти та інші відомості про кадровий склад. Для цього можна використовувати різні інструменти, такі як Excel, бази даних та системи управління кадрами. В нашому випадку, для збору і оброки даних використовуватиметься Excel файли та база даних SQL.

2) **Роботизація процесів**: на цьому етапі створюються боти, які виконують певні рутинні процеси, пов'язані з обробкою даних кадрового складу. Реалізація, наведена в цій роботі, об'єднує кілька окремих бізнес-процесів, таких як завантаження даних, їх обробка, аналіз, форматування та створення звіту, і надсилання результату роботи відповідальній особі.

3) **Інтеграція з системами управління кадрами**: на цьому етапі боти, створені на попередньому етапі, інтегруються з існуючими системами управління кадрами. Це дозволяє ботам автоматично отримувати необхідні дані та виконувати рутинні завдання на основі цих даних. В нашому випадку, інтеграції не здійснювалось, оскільки існуючої системи управління кадрами в компанії не існувало, тому ми запровадили свою.

4) **Моніторинг та оптимізація**: цей етап включає моніторинг роботи ботів та виявлення можливих помилок та проблем. Це дозволяє швидко виявляти та виправляти помилки та вдосконалювати роботизовані процеси. Оптимізація процесів забезпечує покращення продуктивності та ефективності виконання завдань.

5) **Звітність**: на цьому етапі забезпечується створення звітів та аналітичних даних про роботу ботів та їх результативність. Це дозволяє

відстежувати та аналізувати продуктивність системи та вносити необхідні зміни для її покращення.

2.2. Реалізація бази даних та опис її таблиць

В даній реалізації використовується СУБД MySQL. Вона є безкоштовною та відкритою системою управління базами даних, яка широко використовується при різноманітних типах розробок та забезпечує надійність, продуктивність та складні можливості управління даними.

В нашому випадку, використання сторонніх застосунків для управління сервером не вимагається, оскільки кількість таблиць є невеликою і нам достатньо вбудованого функціоналу.

Для того, щоб збирати і зберігати усю необхідну інформацію, яка стосується інформації про робітників компанії, була розроблена база даних “Personnel Management Database”. Вона містить в собі 2 основні таблиці, які надають загальну інформацію про робітників та їх діяльність (назва команди, спеціалізація, проект, витрачений час на проект у конкретний день тощо).

Розглянемо таблиці детальніше. Перша таблиця – “Base Employee Info” (англ. Загальна Інформація про Робітника) призначена для зберігання даних про робітника. Фактично, це будь-яка людина, яка працює (чи колись працювала) в організації, не залежно від її посади чи ролі.

Таблиця 2.2.1 – Опис атрибутів таблиці “Base Employee Info”

Атрибут	**Тип**	**Опис**
Employee_Id	int	Первинний ключ, відповідає унікальному номеру працівника
Employee_Name	varchar(100)	Ім’я (повне)
Employee_Birth	date	Дата народження
Employee_Education	varchar(150)	Ступінь освіти

Employee_Specialization	varchar(100)	Спеціалізація освіти
Employee_Start_Date	date	Дата початку роботи у компанії
Employee_End_Date	date	Дата звільнення з компанії
Employee_Position	varchar(50)	Посада працівника
Employee_Team	varchar(10)	Команда
Employee_Gender	varchar(10)	Стать

По кожному робітнику заповнюється таблиця “Employee Time Info” (англ. Інформація про Затрачений Час Робітника). В ній зберігаються дані про те, як робітник витрачає свій робочий час. Вона пов’язана із таблицею “Base Employee Info” через атрибут “Employee_Id”.

Таблиця 2.2.2 – Опис атрибутів таблиці “Employee Time Info”

Атрибут	**Тип**	**Опис**
Employee_Id	int	Первинний ключ, відповідає унікальному номеру працівника
Employee_Position	varchar(40)	Посада працівника
Employee_Team	varchar(10)	Команда
Project	varchar(255)	Назва проекту
ProjectDate	date	Дата, в яку робітник працював над проектом
HoursSpent	int	Кількість часу, яку витратив робітник в день роботи над проектом (від 1 до 8 годин)

2.3. Опис процесу роботи системи

Система підтримки прийняття рішень у сфері управління кадрами будувались по основним принципам технології RPA, що в свою чергу не вимагає UI реалізації. Основна мета виконання – інформування керівництва про успішність робітників у вигляді звіту, на основі якого керівництво може приймати рішення відносно працівників.

В цілому, процес можна поділити на 6 блоків.

У першому етапі проводиться перевірка середовища, щоб переконатись у наявності всіх необхідних ресурсів та файлів конфігурації для подальшої роботи процесу.

Далі відбувається обробка електронної пошти в Outlook, де зчитуються всі вхідні листи та визначається лист, який потрібно опрацювати.

Третій та четвертий етапи передбачають безпосередню обробку даних, включаючи опрацювання вхідних даних, запис та збереження даних у базі, аналіз та запис даних у файли.

П’ятий етап включає створення звіту про успішність робітників.

У шостому етапі здійснюється інформування керівництва та надсилання звіту на їх поштові адреси.

Детальну реалізацію кожного із блоків можна побачити у наступному розділі.

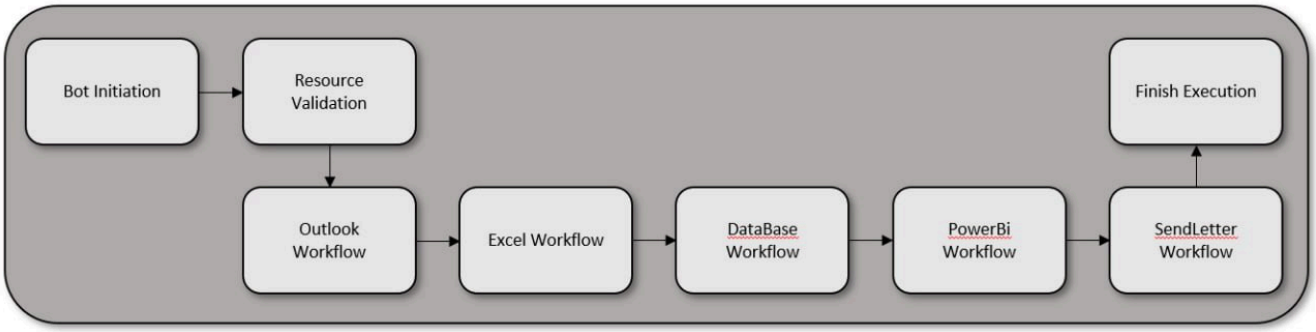


Рисунок 2.3.1. Схематичне зображення процесу

2.4. Системні вимоги та програмні забезпечення

Для повноцінного функціонування системи необхідно у середовищі існування мати наступний перелік встановлених програм:

1) Ліцензійний Microsoft Office 365. У даній роботі головними програмними забезпеченнями з Microsoft Office 365 виступають Excel та Outlook. Excel є потужним інструментом для ведення Timesheets, тобто графіків робочого часу співробітників компанії. Це дозволяє дисципліновано контролювати час, витрачений на роботу, та допомагає враховувати час на різні проекти і завдання. Сьогодні велика кількість компаній здійснює ведення робочого часу співробітників компанії у Excel документах, оскільки це зручно і дозволяє впроваджувати автоматизацію. Outlook - це програма, яка може бути використана для роботи з електронною поштою в компанії.

2) PowerBi Desktop - безкоштовна програма візуалізації даних, яка дозволяє швидко і легко створювати інтерактивні звіти та панелі управління для аналізу даних з різних джерел, таких як Excel, бази даних, хмарні сервіси тощо. Power BI Desktop дозволяє компаніям більш ефективно аналізувати та використовувати свої дані, що може призвести до зниження витрат та збільшення прибутку. [9]

3) Встановлене та налаштоване СУБД MySQL.

Дозволена OS – Windows. Для роботи достатньо версії Windows Server 2019.

2.5. Використане IDE та мова програмування C#

Програмна реалізація системи була розроблена в середовищі програмування Visual Studio 2022 мовою програмування C#. Вся логіка бізнесу реалізована у вигляді бібліотек, які використовують платформу .NET 6 [10]. У даному проекті використання платформи .NET 6 має численні переваги порівняно з .NET Framework. Наведемо декілька з них:

- .NET 6 пропонує вищу продуктивність, що дозволяє додаткам працювати швидше та більш ефективно.
- .NET 6 надає розробникам доступ до нових функцій та можливостей, таких як удосконалена робота з веб-додатками, масштабованість, безпека та інші.
- Крос-платформена підтримка: .NET 6 підтримує більшість операційних систем, що дозволяє розробляти крос-платформені додатки для різних платформ.
- Платформа .NET 6 є продуктом з відкритим вихідним кодом, що доступний для розробників.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ПРИЙНЯТТЯ РІШЕНЬ

3.1. Вибір методології та моделі життєвого циклу

При програмній реалізації системи було дотримано основних принципів методології SOLID [11], яка є однією з ключових концепцій об'єктно-орієнтованого програмування. Вона допомагає розробникам створювати більш ефективний, масштабований та зручний для розуміння код.

Причиною обрання цієї методології пов’язано із тим, що реалізація повинна дотримуватись принципу відкритості/закритості, тобто, класи повинні бути відкритими для розширення, але закритими для змін.

Для розробки системи було вибрано модель життєвого циклу водоспадного або каскадного типу, оскільки вона забезпечує чітку дисципліну та описані кроки розробки на кожній стадії проекту. Ця модель гарантує виключення ризиків недооцінки часу виконання проекту та має просту систему визначення затрат.

Однак, недоліком даної моделі є те, що для переходу на наступну стадію розробки потрібно мати повністю коректний результат попередньої стадії. Тому вона не підходить для великих проектів, де можливі зміни вимог та важливість прототипування та швидкої ітерації.

Модель життєвого циклу водоспадного або каскадного типу підходить для створення невеликих систем, таких як роботизована автоматизація, де всі вимоги до бізнес-процесу попередньо створені, зафіксовані та не підлягають змінам в майбутньому. Якщо необхідно змінити кроки процесу у майбутньому, потрібно провести аналіз ризиків та можливо створити новий допоміжний бізнес-процес.

3.2. Розробка окремих блоків роботи системи

3.2.1. Необхідні вхідні дані

Для коректної роботи системи вимагається наступне:

- Наявність файлу конфігурації, в якому прописані усі необхідні поля і значення.
- Створений файл формату Excel по шаблону.
- Створений PowerBi файл, який працює із Excel темплейт-файлом.
- Лист на пошті, із визначеною темою листа (бажана тема листа зазначається в файлі конфігурації) та прикріпленим до нього файлом.

Поля, які повинен містити файл конфігурації наведені в Додатку 1.

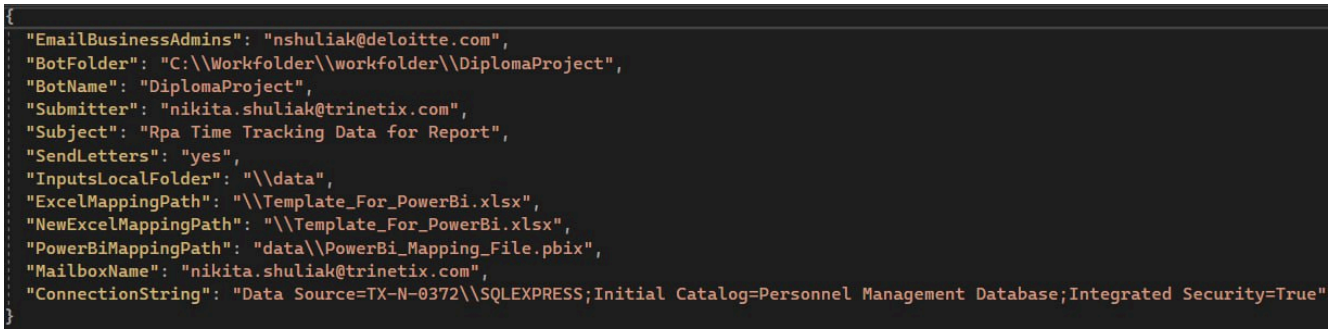


Рисунок 3.2.1.1. Приклад заповненого файлу конфігурації

Один з важливих етапів створення звіту у Power BI - це підготовка даних. Одним із ключових файлів для зберігання та обробки даних є шаблонний файл Excel. В цьому файлі необхідно мати чітку структуру даних. Після цього можна імпортувати ці дані в Power BI та провести розширену обробку та аналіз даних. В результаті отримується звіт з різноманітними візуалізаціями, що допоможе зробити виведення даних більш зрозумілим та наглядним.

Однією з важливих функцій бота є робота з електронною поштою. Завдання полягає у виявленні непрочитаних листів від визначеного в конфігураційному файлі відправника, перевірці відповідності теми листа та наявності вкладеного Excel файлу із даними. Ці дані містять інформацію про витрачений час робітниками на

конкретні проекти за певний період, а також запланований час, який був виділений компанією на ці проекти. Після знаходження відповідного листа, бот здійснює імпорт даних з Excel файлу, обробку даних та генерацію звіту. Таким чином, бот сприяє автоматизації процесу аналізу витраченого часу на проекти та допомагає компанії ефективно управляти своїми ресурсами.

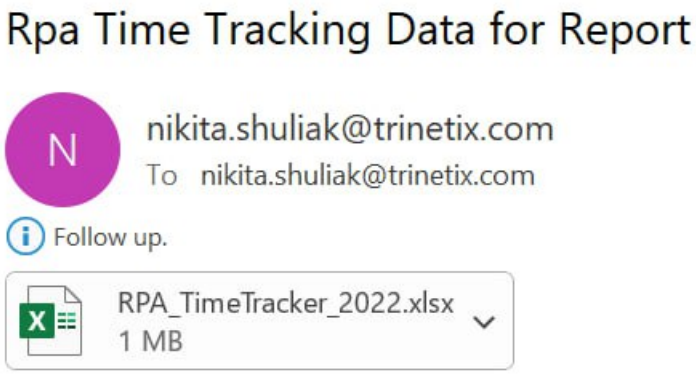


Рисунок 3.2.1.3. Приклад вхідного листа

3.2.2. DI, головний клас та етап валідації

Реалізація системи використовує методологію DI, яка дозволяє розділити відповідальність за створення та залежності об'єктів від класів, що їх використовують. Замість того, щоб клас самостійно створював об'єкти, він отримує їх через конструктор або властивість, що дозволяє йому декларувати свої залежності.

Метод, який конфігурує використання DI в системі наведено в Додатку 2.

Головний клас системи виглядає наступним чином:

```
class Program
  {
    static void Main(string[] args)
    {
      Bot bot = null;
      try
      {
```

```
            InitialSetup();
            IHost host = BuildHost(args);
            bot = host.Services.GetService<Bot>();
            bot.Run();
        }
        catch (Exception ex)
        {
            Console.WriteLine($"High level exception - <{ex.Message}>");
            Log.Error(ex, "High level exception");
        }
        finally
        {
            System.Threading.Thread.Sleep(15 * 1000);
            bot?.FinishExecution();
        }
    }
}
```

Створення екземпляру класу "Bot" та виклик його методу "bot.Run()" є дією, що ініціює запуск системи. В нашій реалізації бота ми використовуємо каскадну модель, яка передбачає послідовне виконання етапів. Першим кроком є валідація системи, під час якої проводиться перевірка наявності всіх необхідних файлів, доступ до них та до ресурсів.

Важливою метою валідації є переконання, що бот може працювати ефективно та безперебійно. Під час виконання цього кроку, перевіряється правильність ініціалізації бота та його конфігурації.

Якщо в процесі валідації буде виявлено будь-які проблеми, то процес роботи бота буде зупинено, адже недоступність необхідних файлів та ресурсів може призвести до помилок та втрати даних.

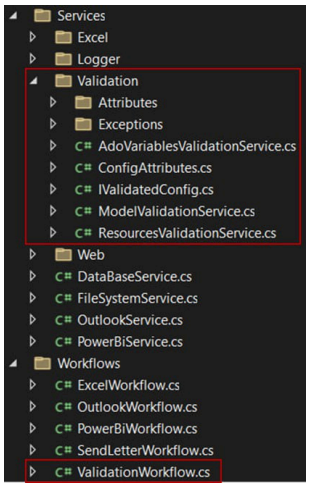


Рисунок 3.2.2.1. Етап валідації

3.2.3. Етап роботи із Outlook

Був створений сервіс, який був спрямований на полегшення роботи з додатком Outlook. Його головна мета - надання універсальних методів, які можна використовувати в подальшій роботі з цим додатком.

Для ініціалізації роботи з Outlook була використана відкрита бібліотека Interop.Outlook [12], що надає зручний доступ до багатьох функцій цього додатку. Ця бібліотека дозволяє здійснювати операції з електронною поштою, календарем, контактами та іншими компонентами Outlook з використанням .NET-коду.

Interop.Outlook дозволяє створювати екземпляри Outlook Application та Namespace, які потрібні для роботи з додатком Outlook. Outlook Application - це головний об'єкт, який управляє додатком Outlook. Namespace - це об'єкт, який дозволяє отримувати доступ до різних компонентів додатку Outlook, таких як папки електронної пошти, календарі, контакти та завдання.

Отримавши екземпляри Outlook Application та Namespace, можна використовувати їх для взаємодії з додатком Outlook та здійснювати різноманітні операції з електронною поштою.

Було реалізовано методи, які надають універсальний функціонал для роботи з додатком Outlook. Зокрема, сервіс дозволяє відправляти листи, зберігати їх у вигляді чернетки, створювати нові листи, відмічати листи як прочитані (і навпаки), зберігати прикріплені файли до листів, перевіряти доступність папок у мейлбоксі,

знаходити папки у мейлбоксі рекурсивно за їхнім іменем, зчитувати усі листи з папки з різними властивостями та прикріплювати файли до листів.

Приклад реалізації одного із методів наведено нижче:

```
public List<MailItem> GetReceivedItemsFromMailbox(string senderEmails,
string subject, string mailbox, string folderName, bool includeRead, bool
includeUnread, string receivedEarliest, string receivedLatest)
    {
      List<MailItem> mails = new List<MailItem>();
      Items messages = null;
      MAPIFolder folder = GetFolderInMailbox(mailbox, folderName);
      _outlookInboxFolderItems = folder.Items;
      StringBuilder builder = new StringBuilder();
      builder.Append("@SQL=");
      if (includeRead & !includeUnread)
        builder.Append("urn:schemas:httpmail:read=1 AND ");
      if (includeUnread & !includeRead)
        builder.Append("urn:schemas:httpmail:read=0 AND ");
      if (subject.Trim() != "")
        builder.Append(String.Format("urn:schemas:httpmail:subject LIKE
'%{0}%' AND ", subject.Trim()));
      if (receivedEarliest.Trim() != "")
        builder.Append(String.Format("urn:schemas:httpmail:datereceived>=
'{0}' AND ", Convert.ToDateTime(receivedEarliest)));
      if (receivedLatest.Trim() != "")
        builder.Append(String.Format("urn:schemas:httpmail:datereceived<=
'{0}' AND ", Convert.ToDateTime(receivedLatest)));
      if (builder.ToString() != "@SQL=")
      {
        string filterCondition = builder.ToString().TrimEnd(new char[] { '
', 'A', 'N', 'D' });
        messages = _outlookInboxFolderItems.Restrict(filterCondition);
      }
      else
        messages = _outlookInboxFolderItems;

      foreach (Object it in messages)
      {
        if (it is MailItem)
```

```
            mails.Add((MailItem)it);
        }
        return mails;
    }
```

За допомогою цих методів можна автоматизувати рутинні задачі, пов'язані з роботою з електронною поштою і ефективно взаємодіяти з додатком Outlook, що дозволяє збільшити продуктивність роботи.

Під час виконання етапу роботи з додатком Outlook система перевіряє наявність непрочитаного листа у зазначеному мейлбоксі, який має певного відправника та певну тему, і прикріплений до нього файл Excel. Якщо сервіс не знайде такого листа, то керівництво буде повідомлене про помилку, яка виникла на етапі роботи з Outlook. Результатом виконання етапу є збережений файл Excel у папку, зазначену в файлі конфігурацій.

3.2.4. Етап роботи із Excel та БД, проведення аналізу даних

Як вже зазначалось, реалізацію було побудовано таким чином, щоб в подальшій роботі можна було користуватись сервісами, які працюють з окремими додатками. Це підкреслює значущість даної роботи і її масштабованість. Такий підхід забезпечує гнучкість використання системи, оскільки користувачі можуть обирати тільки ті функції, які їм потрібні, та інтегрувати їх з іншими додатками і сервісами, які вже використовуються.

Робота із Excel та БД реалізовувалась по такому ж принципу. Розпочнемо із Excel.

Для реалізації сервісу роботи із додатком Excel використовувалась відкрита бібліотека Interop.Excel [13]. Бібліотека Interop.Excel дозволяє взаємодіяти з Excel з програм на .NET, що дозволяє легко і швидко створювати та редагувати Excel-файли.

За допомогою DI було створено зручну структуру сервісу, яка дозволяє створювати окремі моделі для валідації та перевірки наявності необхідних елементів у файлі Excel при його відкритті. Це досягнуто за допомогою створення

окремих екземплярів класів "FileInfo" та "FileDto", які містять загальну інформацію про відкритий файл та зчитані з нього дані відповідно. Крім того, файл розбивається на окремі листи "SheetInfo" та "SheetDto", які містять загальну інформацію про листи та їх зчитані дані відповідно.

"FileInfo" та "SheetInfo" заповнюються із файлу конфігурації "ExcelConfig", що дозволяє заздалегідь визначити очікуваний файл, його листи та назви колонок. Якщо файл є шаблонним, то це спрощує валідацію та перевірку наявності необхідних елементів у ньому. Класи "FileInfo" та "SheetInfo" успадковують поля із іменами файлу та листів, а також метод отримання назв усіх листів файлу від класів "BaseFileInfo" та "BaseSheetInfo".

Якщо ми не можемо передбачити структуру файлу, який потрібно зчитати, то ми використовуємо клас BaseSheetOutData, який автоматично заповнюється даними з обраного листа відкритого файлу.

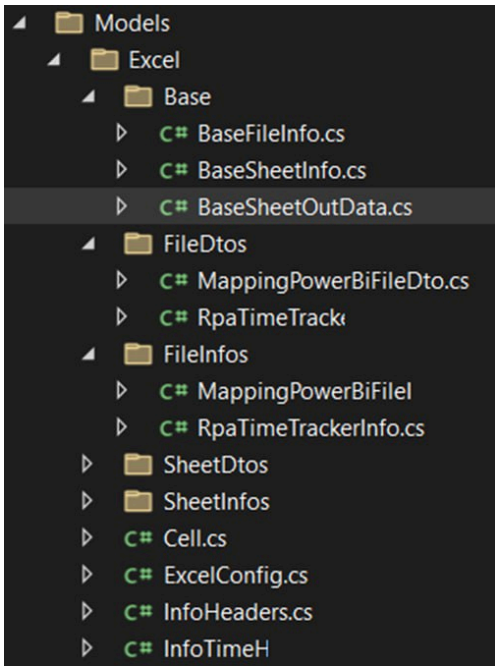


Рисунок 3.2.4.1. Вигляд реалізованої моделі для сервісу роботи із Excel

Як і сервіс для роботи з додатком Outlook, сервіс Excel має універсальні методи для роботи із Excel файлами. Проте його реалізація дещо відрізняється, оскільки функціонал Excel набагато ширший ніж Outlook. Тому був додатково створений клас ExcelFileManager, який містить в собі сервіс Excel і доповнює його.

Наступні універсальні методи були розроблені: ініціалізація класів Application та Workbook для роботи з Interop.Excel; відкриття та закриття файлів з можливістю очікування завантаження файлу, оскільки Excel-документи можуть містити значну кількість даних; зчитування всіх назв листів; отримання індексу рядка, де розташовуються заголовки таблиці (заголовки можуть бути впорядковані або містити пропущені значення, для цього є окремий метод); безпосередній пошук заголовків таблиці на листі та отримання індексів їх колонок; очищення листа; отримання імені колонки, маючи її індекс (перетворення з 1 на A); зчитування даних листа в двовимірний масив; зчитування унікальних значень з колонки таблиці; знаходження підпослідовності в послідовності заголовків і повернення їх індексів; знаходження наступної порожньої клітинки після значення; запис даних в клітинку з будь-якими стилями, що вказуються в параметрах методу; фільтрація назв листів, щоб отримати список листів, які нас цікавлять; зчитування всіх існуючих заголовків на листі; знаходження індексів заголовків у вигляді бібліотеки; знаходження «якірного» елементу (унікального елемента даних листа, від якого можна будувати пошук інших елементів); валідація файлу та листів файла (перевірка на наявність необхідних елементів); перевірка, чи файл не захищений паролем.

Ці методи значно полегшують розробку програм, що взаємодіють з додатком Excel, тому що вони розв'язують більшість типових задач. Використання цих методів дозволяє зосередитись на розробці більш складних функцій, замість витрат на вирішення проблем базового функціоналу Excel. Це забезпечує ефективну розробку ботів, які використовують Excel для зберігання і обробки даних. Застосування цих методів зменшує ризик виникнення помилок та забезпечує більш високу якість коду.

Реалізацію кількох методів наведено в додатках 3-6.

На початку роботи блоку Excel, бот перевіряє і валідує файл, отриманий на попередньому етапі. Файл містить інформацію про робітників, їх затрачений час на проекти у вигляді годин, зазначених у таблиці. Також вони зазначають неактивні години, відпустки та лікарняні. Ці дані знаходяться на листі “Time Tracking”.

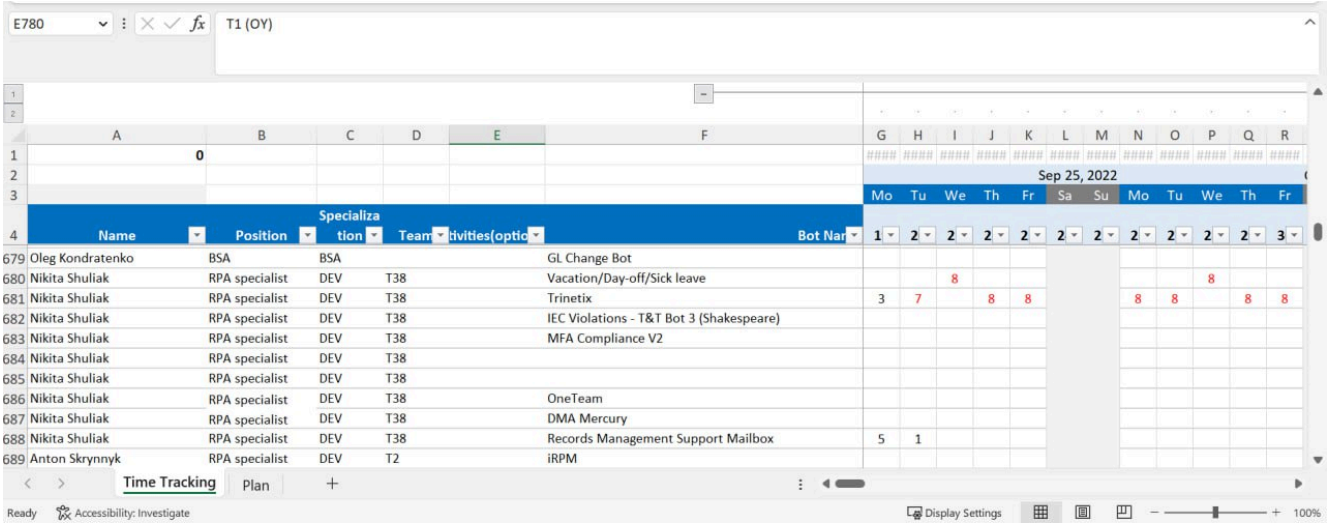


Рисунок 3.2.4.2. Вигляд листа Time Tracking

Також файл містить інший лист під назвою “Plan”, в якому знаходиться інформація про заплановані години, які повинні були бути витрачені на той чи інший проект.

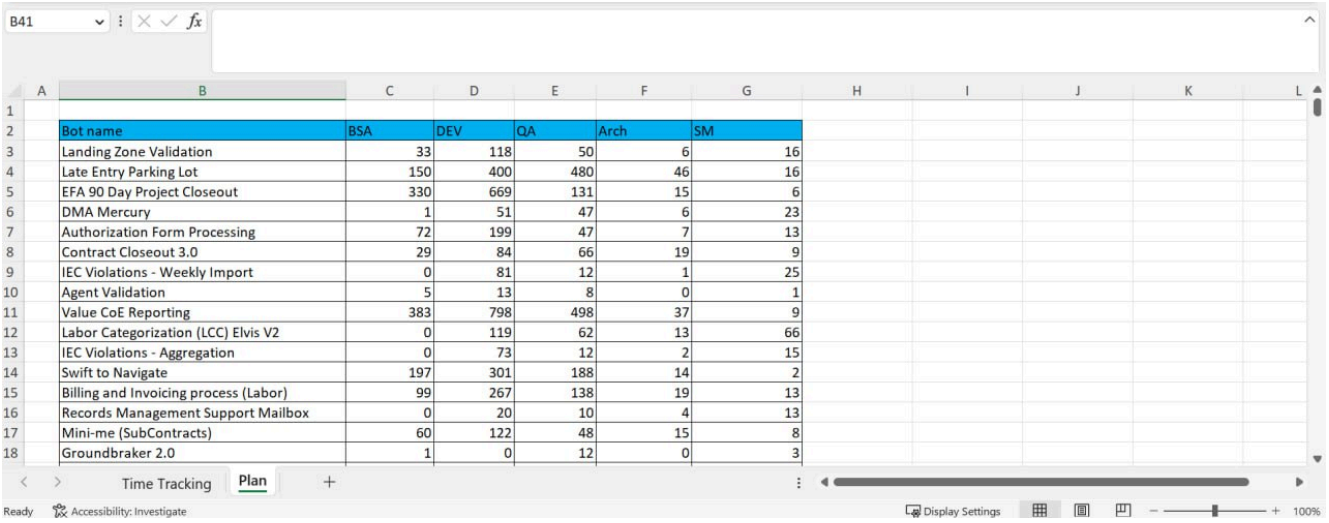


Рисунок 3.2.4.3. Вигляд листа “Plan”

Після проведення перевірки валідності, дані про затрачений час працівників обробляються та конвертуються у зручний формат для подальшого аналізу. Інформація про час зберігається в базі даних, яка опрацьовується окремим сервісом. Основна функція цього сервісу - забезпечити зв'язок з наявною базою даних, зчитувати інформацію з таблиць, зберігати її та виконувати запити з можливістю отримання аналітичних даних.

Приклад отримання даних із БД через запит у сервісі наведений в додатку 7.

Після того, як всі необхідні дані були отримані, бот розпочинає процес запису цих даних в зручному форматі «Excel_Mapping_File». У цьому файлі зібрана інформація про кожного зі співробітників, затрачений час, приналежність до конкретної команди, порівняльні дані та інші важливі деталі.

Для забезпечення оптимального процесу заповнення окремих листів файлу, були створені окремі "Worksheet Managers" - менеджери робочих аркушів, які відповідають за логіку заповнення кожного окремого листа. Це дозволяє боту автоматизувати процес запису даних та забезпечити їх точність та консистентність в усьому файлі.

```
// write into excel mapping file user and user time infos
foreach (var el in _rpaTimeTrackerDto.TimeTrackingDto.SpecialistInfo)
{
    string userSheetName = el.Key + " Info";
    string userTimeSheetName = el.Key + " Time Info";
    _userInfoWorksheetManager.ProcessUserWorksheet(userSheetName, el.Value);
    _userTimeInfoWorksheetManager.ProcessUserTimeWorksheet(userTimeSheetName, el.Value);
}
// write into excel mapping file plan and fact info
_mappingPlanWorksheetManager.ProcessPlanWorksheet();
_mappingFactWorksheetManager.ProcessFactWorksheet();
_teamAnalysedDataWorkflow.ProcessTeamAnalysedDataWorksheet(dbData.Item1);
_teamAnalysedNotWorkedDataWorkflow.ProcessTeamAnalysedNotWorkedDataWorksheet(dbData.Item2);
_genderDataWorksheetManager.ProcessGenderDataWorksheet(dbData.Item3);
_excelFileManager.CloseWorkbook(true);
```

Рисунок 3.2.4.4 Процес запису даних у шаблонний файл

Результатом даного блоку виступає існуючий і заповнений даними шаблонний файл, який передається для створення звіту у PowerBi Desktop.

3.2.5. Етап створення звіту в додатку PowerBi Desktop

Якісний звіт - це важлива складова успішного управління персоналом. Звіт допомагає керівництву швидко зрозуміти, як розвивається їх бізнес, побачити ключові показники та зробити висновки на основі цих даних. Графічне відображення даних є дієвим інструментом, що дозволяє більш ефективно і точно аналізувати та візуалізувати результати. Звіти мають бути якісними, адже на їх

основі приймаються стратегічні рішення щодо планування бюджету. Успішне управління персоналом вимагає постійного моніторингу та аналізу даних, що дозволяє швидко реагувати на зміни та вносити необхідні корективи в стратегію розвитку бізнесу.

Реалізований бот має можливість автоматично генерувати звіт про роботу персоналу за визначений період часу. Бот також здійснює аналіз ефективності кожного з робітників та команд, їх динаміки та надає інформацію про те, наскільки вдало дотримуються заплановані терміни виконання проектів. Крім того, бот порівнює середню активність робітників в залежності від їх статі, що допомагає зрозуміти, як різні гендерні групи працюють в компанії та наскільки результативно вони виконують свої завдання. У результаті наш бот надасть корисну інформацію для вдосконалення стратегії управління персоналом та покращення продуктивності компанії.

На жаль, Power BI Desktop не надає готових бібліотек або NuGet пакетів для прямої інтеграції з мовою програмування C#. Для вирішення цієї проблеми існує підхід у RPA технології, який орієнтується на UI автоматизації. Оскільки Power BI Desktop має графічний інтерфейс користувача, тому ми можемо взаємодіяти з ним використовуючи бібліотеку FlaUI для автоматизації взаємодії з додатком [14]. Це означає, що ми можемо написати скрипти на мові програмування C#, які будуть емулювати дії користувача, такі як кліки на кнопки, введення даних та інші. FlaUI є відкритим джерелом і розповсюджується під ліцензією Apache 2.0. Документація та приклади використання бібліотеки також доступні для користувачів.

За допомогою бібліотеки FlaUI було розроблено окремий сервіс, який включає універсальні методи роботи з додатком PowerBi Desktop. Ці методи дозволяють відкривати та під’єднуватись до додатку, розпізнавати та знаходити його елементи, натискати клавіші, розпізнавати спливаючі вікна та зчитувати з них дані, а також закривати та зберігати звіти.

Реалізації декількох корисних методів взаємодії із PowerBi Desktop наведені в додатках 8-11.

Результатом виконання блоку є готовий звіт із усіма проаналізованими даними.

Приклад готових елементів звіту наведено на Рисунках 3.2.5.1 та 3.2.5.2. Power BI Desktop дозволяє створювати інтерактивні звіти, які можна динамічно змінювати за допомогою вибору різних елементів. При виборі конкретного елементу, звіт автоматично перефокусовується на дані, які пов'язані з цим елементом, що дозволяє користувачам отримувати більш детальну інформацію про дані та їх зв'язки.

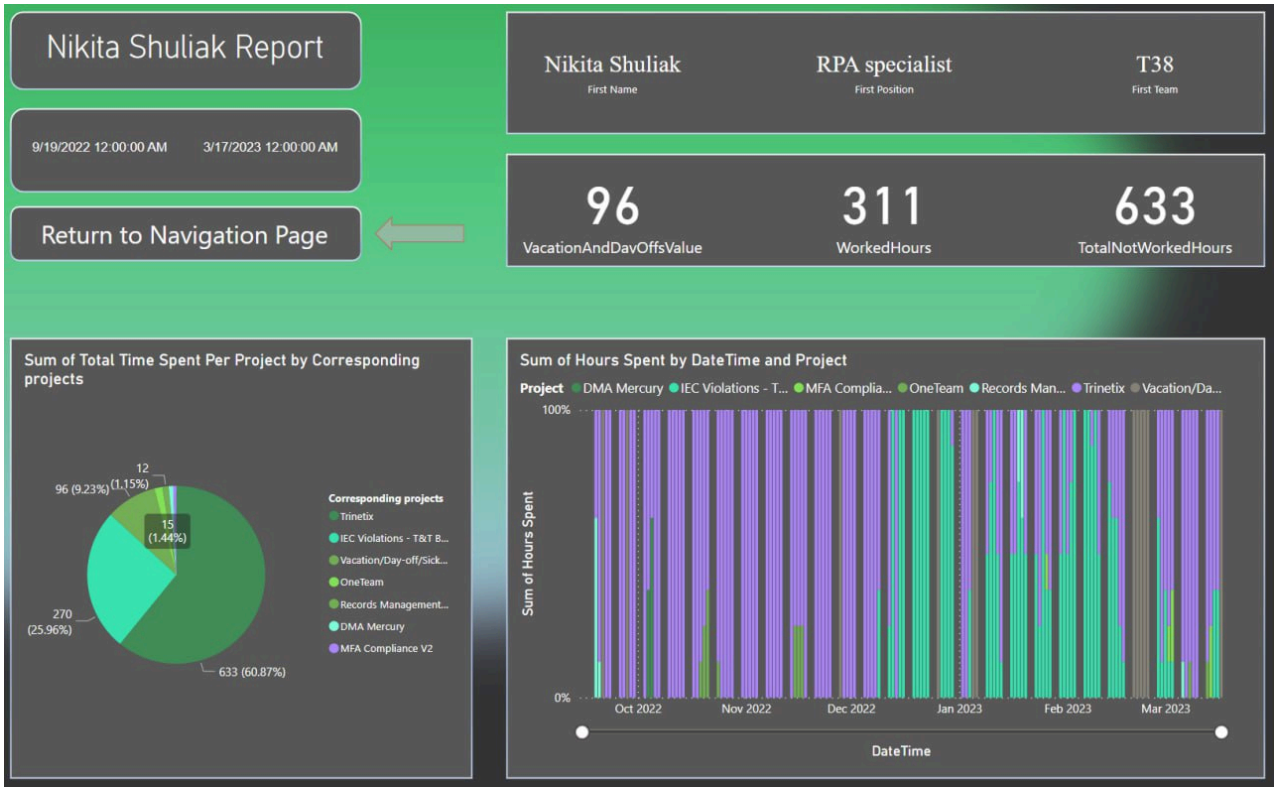


Рисунок 3.2.5.1. Аналіз одного із робітників за період

Ефективність команд можна розглядати як один із ключових показників продуктивності організації. Це важливо не тільки для успішної реалізації проектів, але й для забезпечення мотивації та задоволеності робітників, що в свою чергу позитивно впливає на загальну ефективність бізнесу. На Рисунку 3.2.5.2 можна знайти детальний аналіз ефективності команд у розрізах місяців і років.

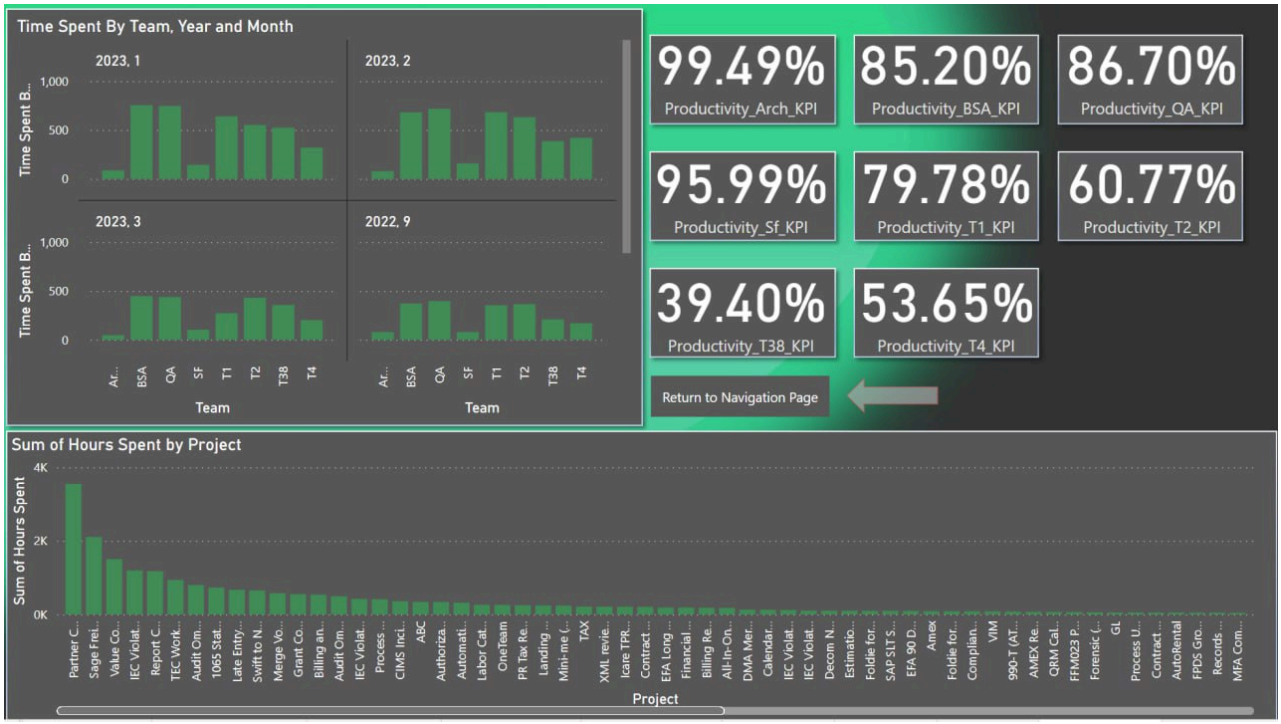


Рисунок 3.2.5.2. Ефективність команд та динаміка ефективності

3.2.6. Надсилання листа із результатом роботи керівництву

Заключним етапом роботи системи є інформування керівництва про роботу бота. Цей етап є не менш важливим, ніж попередні, оскільки інформування керівництва про роботу бота має на меті забезпечити розуміння ефективності та користі, яку може принести бот для бізнесу.

Під час інформування керівництва необхідно представити результати роботи бота та зробити висновки щодо його ефективності.

В цілому, в нашій реалізації є два типи листа: перший вказує про коректне відпрацювання бота і містить в собі прикріплений файл звіту, інший – інформує керівництво про помилку яка сталась на одному із етапів роботи бота і детальний опис помилки.

Процес створення та відправки листа використовує реалізований і зазначений раніше сервіс Outlook.

Реалізація відправки листів наведена нижче:

```
public void SendResultLetter()
    {
```

```
            _logger.LogInformation("Start sending result letter.");
            var resultLetter = _outlookService.CreateMailItem("Rpa
TimeTracker Result", "Please, download the attachment with report.",
_botExecutionConfig.Submitter );
            _outlookService.AddAttachment(resultLetter,
String.Concat(_botExecutionConfig.BotFolder,
_botExecutionConfig.InputsLocalFolder,
_botExecutionConfig.PowerBiMappingPath));
            Thread.Sleep(5000);
            _outlookService.Send(resultLetter);
            Thread.Sleep(2000);
            _logger.LogInformation($"Result letter successfully sended to
{_botExecutionConfig.Submitter}.");
          }
          public void SendErrorLetter(string errMsg)
          {
            _logger.LogInformation("Start sending Error letter.");
            var resultLetter = _outlookService.CreateMailItem("Rpa
TimeTracker Bot Faced An Error", $"Error message: {errMsg}",
_botExecutionConfig.Submitter);
            _outlookService.Send(resultLetter);
            Thread.Sleep(2000);
            _logger.LogInformation($"Error letter successfully sended to
{_botExecutionConfig.Submitter}.");
          }
```

ВИСНОВКИ

Сучасний облік кадрів вимагає використання новітніх технологій та методик для отримання звітів. Основна мета цього процесу полягає у зборі, обробці та аналізі даних про кадровий склад організації, щоб забезпечити ефективне управління персоналом.

Під час виконання роботи був проведений аналіз предметної області даної тематики, розглянуті існуючі рішення автоматизації даного процесу. Було зазначено їх головний недолік у вигляді високої вартості ліцензії цих рішень.

Створена програма надає можливість повністю покривати бажаний функціонал та розв’язує задачі ефективного процесу прийняття рішень у сфері управління кадрами засобами RPA. Також було реалізовано універсальні сервіси роботи із такими додатками як: Excel, Outlook, MySQL та PowerBi Desktop. Це спрощує майбутнє вдосконалення даної системи та подальші створення ботів.

Як результат, можна виділити наступні позитивні сторони даного рішення:

- Гнучкість використання;
- Достатня функціональна потужність;
- Легкість впровадження;
- Низькі вимоги до ресурсів;
- Масштабованість та доповнюваність.

У перспективні подальшого вдосконалення планується створення UI інтерфейсу та специфікації із документом контрольного списку впровадження із усіма необхідними коментарями.

Наявність перспектив є необхідною мотивацією для подальшої роботи над вдосконаленням і продовженням представленої реалізації в найближчому майбутньому.

Варто зазначити, що розроблена система зараз перебуває на етапі затвердження і тестування в компанії і дані, які слугували для створення звіту і над якими проводився аналіз були реальними даними компанії. Результат системи вже

дозволяє робити висновки на рахунок кадрів і команд компанії, а також відображає їх слабкі сторони, які раніше було складно помітити.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шуляк Н. Використання технології RPA у розробці системи підтримки прийняття рішень у сфері управління кадрами: матеріали наук.-практ. конф. студентів і аспірантів «Теоретичні та прикладні аспекти розробки комп'ютерних систем 2023» [Електронний ресурс]. Режим доступу: http://econference.nubip.edu.ua/index.php/taacsd/2023/paper/view/2804
2. Emerging LinkedIn Jobs Report U.S. 2020. [Електронний ресурс]. Режим доступу: https://business.linkedin.com/content/dam/me/business/en-us/talent-solutions/emerging-jobs-report/Emerging_Jobs_Report_U.S._FINAL.pdf
3. Tom Taulli The Robotic Process Automation Handbook / Tom Taulli
4. RPA Guidance. [Електронний ресурс]. Режим доступу: https://transform.england.nhs.uk/media/documents/RPA-Guidance-May-22.pdf
5. Офіційний сайт Blue Prism [Електронний ресурс]. Режим доступу: https://www.blueprism.com/
6. Офіційний сайт UiPath [Електронний ресурс]. Режим доступу: https://www.uipath.com/
7. Офіційний сайт Automation Anywhere [Електронний ресурс]. Режим доступу: https://www.automationanywhere.com/
8. Офіційний сайт Celonis [Електронний ресурс]. Режим доступу: https://www.celonis.com/
9. Офіційний сайт PowerBi [Електронний ресурс]. Режим доступу: https://www.powerbitiles.com/
10. Офіційний сайт .Net6 [Електронний ресурс]. Режим доступу: https://dotnet.microsoft.com/en-us/
11. The Importance of SOLID Design Principles [Електронний ресурс]. Режим доступу: https://www.bmc.com/blogs/solid-design-principles/
12. Офіційний сайт Interop.Excel [Електронний ресурс]. Режим доступу: https://learn.microsoft.com/en-us/dotnet/api/microsoft.office.interop.excel?view=excel-pia

13. Офіційний сайт Interop.Outlook [Електронний ресурс]. Режим доступу: https://learn.microsoft.com/en-us/dotnet/api/microsoft.office.interop.outlook?view=outlook-pia
14. Бібліотека FlaUI [Електронний ресурс]. Режим доступу: https://github.com/FlaUI/FlaUI

ДОДАТОК 1

Таблиця Додатку 1. Опис полів, які повинен містити файл конфігурації

Поле	**Опис**
EmailBusinessAdmins	Пошти керівників
BotFolder	Шлях до папки бота
BotName	Назва бота
Submitter	Пошта, від кого буде отримуватись файл для обробки (для безпеки)
SendLetters	Надсилати лист чи створювати драфт листа в папці (поле для тестування коректності роботи)
InputsLocalFolder	Папка, яка містить в собі дані з усіма необхідними файлами
ExcelMappingPath	Шлях до файлу шаблону Excel (утворюється шляхом склеювання із полем BotFolder)
NewExcelMappingPath	Шлях до заповненого файлу шаблону Excel (утворюється шляхом склеювання із полем BotFolder та InputsLocalFolder)
PowerBiMappingPath	Шлях до файлу шаблону PowerBi (утворюється шляхом склеювання із полем BotFolder та InputsLocalFolder)
MailboxName	Назва мейлбоксу, з яким буде працювати бот (їх може бути декілька)
ConnectionString	Рядок тексту, який використовується для підключення до бази даних

ДОДАТОК 2

Налаштування DI

```
private static IHost ConfigureServices(IConfigurationRoot config)
    {
      return Host.CreateDefaultBuilder()
        .ConfigureLogging((context, logging) =>
        {
          logging.ClearProviders();
          logging.AddSerilog(dispose: true);
        })
        .ConfigureServices((context, services) =>
        {
          services
            .AddSingleton((serviceProvider) => config)
            .AddHttpClient()
            .RemoveAll<IHttpMessageHandlerBuilderFilter>()
            .Configure<BotExecutionConfig>(config)
            .Configure<RpaLoggerConfig>(config)
            .AddSingleton<ValidationWorkflow>()
            .AddSingleton<ResourcesValidationService>()
            .AddSingleton<AdoVariablesValidationService>()
            .AddSingleton<OutlookWorkflow>()
            .AddSingleton<OutlookService>()
            .AddSingleton<ExcelWorkflow>()
            .AddSingleton<ExcelFileManager>()
            .AddSingleton<ExcelService>()
            .AddSingleton<RpaTimeTrackerExcelService>()
            .AddSingleton<TimeTrackingWorksheetManager>()
            .AddSingleton<PlanWorksheetManager>()
            .AddSingleton<RpaTimeTrackerDto>()
            .AddSingleton<UserInfoWorksheetManager>()
            .AddSingleton<UserInfoWorksheetManager>()
            .AddSingleton<UserTimeInfoWorksheetManager>()
            .AddSingleton<MappingPlanWorksheetManager>()
            .AddSingleton<MappingFactWorksheetManager>()
            .AddSingleton<TeamAnalysedDataWorkflow>()
            .AddSingleton<TeamAnalysedNotWorkedDataWorkflow>()
            .AddSingleton<GenderDataWorksheetManager>()
            .AddSingleton<PowerBiWorkflow>()
            .AddSingleton<PowerBiService>()
            .AddSingleton(new UiPBIConnector(new UIA3Automation()))
            .AddSingleton<FileSystemService>()
            .AddSingleton<SendLetterWorkflow>()
            .AddSingleton<DataBaseService>()
            .Configure<ExcelConfig>(config.GetSection("Excel"))
            .Configure<PowerBiConfig>(config.GetSection("powerBi"))
```

```
            .Configure<DatabaseConfig>(config.GetSection("database"))
#if DEBUG
            .AddSingleton<ILogEventSink, RpaLoggerMockService>()
#else
                        .AddSingleton<ILogEventSink, RpaLoggerService>()
#endif
            .AddSingleton<Bot>();
        })
        .Build();
    }
```

ДОДАТОК 3

Реалізація відкриття файлу Excel із перевірками його на валідність.

```
public Workbook OpenWorkbook(string filepath)
        {
            if (!File.Exists(filepath))
            {
                throw new Exception($"file {Path.GetFileName(filepath)} does not exist.");
            }

            Workbook workbook;
            try
            {
                Log.Debug("Opening file - <{file}>", Path.GetFileName(filepath));
                workbook = ExcelApplication.Workbooks.Open(filepath, Type.Missing, false,
Type.Missing, "", Type.Missing,
                    Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing,
Type.Missing, Type.Missing, Type.Missing, Type.Missing);
                if (workbook.ReadOnly)
                {
                    throw new Exception($"file {Path.GetFileName(filepath)} is read only");
                }
                Log.Debug("Opening file - <{file}> finished",
Path.GetFileName(Path.GetFileName(filepath)));
            }
            catch (Exception ex)
            {
                if (ex.Message.Contains("read only"))
                    throw new Exception($"file {Path.GetFileName(filepath)} is locked
(cannot be edit)");
                if (ex.Message.Contains("file format or file extension is not valid"))
                    throw new Exception($"file {Path.GetFileName(filepath)} is broken
(cannot be read)");
                if (ex.Message.Contains("The password"))
                    throw new Exception($"file {Path.GetFileName(filepath)} is password
protected");
                throw;
            }
            currentWorkbook = workbook;
            return workbook;
        }
```

ДОДАТОК 4

Знаходження хедерів на листі файлу Excel

```
public int GetMultiRowHeadersLastRowIndex(IEnumerable<string> headers, object[,] data, out
List<string> missingHeaders, bool startsWith = false, bool contains = false)
        {
            missingHeaders = new List<string>();
            int headersRowIndex = -1;
            int columnsCount = data.GetLength(1);
            int rowsCount = data.GetLength(0);
            List<string> foundHeadersRowScope = new List<string>();

            for (int rowIndex = 1; rowIndex <= rowsCount; rowIndex++)
            {
                List<string> missingHeadersRowScope = new List<string>();
                List<string> rowValues = new List<string>();
                // collect current row values
                for (int columnIndex = 1; columnIndex <= columnsCount; columnIndex++)
                {
                    rowValues.Add(Convert.ToString(data[rowIndex, columnIndex]));
                }
                // check headers match
                foreach (string header in headers)
                {
                    if (rowValues.Where(rowValue => IsRowValueMatchHeader(rowValue, header,
startsWith, contains)).Count() != 0)
                    {
                        foundHeadersRowScope.Add(header);
                        missingHeadersRowScope.RemoveAll(h => h.Equals(header));
                    }
                    else if (rowValues.Where(rowValue => IsRowValueMatchHeader(rowValue,
header, startsWith, contains)).Count() == 0)
                    {
                        missingHeadersRowScope.Add(header);
                    }
                }
                // if all headers found -> break, else get max missing headers
                if (foundHeadersRowScope.Count == headers.ToList().Count)
                {
                    headersRowIndex = rowIndex;
                    missingHeaders.Clear();
                    break;
                }
                else
                {
                    if (!missingHeaders.Any() || missingHeadersRowScope.Count <
missingHeaders.Count)
                    {
                        missingHeaders = missingHeadersRowScope;
                    }
                }
            }
            return headersRowIndex;
        }
```

ДОДАТОК 5

Зчитування листа файлу Excel у двовимірний масив

```
public object[,] ReadWorksheetToArray(Worksheet worksheet)
        {
            var lastColumnIndex = 0;
            var lastRowIndex = 0;
            object[,] data = null;
            try
            {
                lastColumnIndex = worksheet.Cells.Find("*",
System.Reflection.Missing.Value, System.Reflection.Missing.Value,
                    System.Reflection.Missing.Value, XlSearchOrder.xlByColumns,
XlSearchDirection.xlPrevious,
                    false, System.Reflection.Missing.Value,
System.Reflection.Missing.Value).Column;
                lastRowIndex = worksheet.Cells.Find("*", System.Reflection.Missing.Value,
System.Reflection.Missing.Value,
                    System.Reflection.Missing.Value, XlSearchOrder.xlByRows,
XlSearchDirection.xlPrevious,
                    false, System.Reflection.Missing.Value,
System.Reflection.Missing.Value).Row;
            }
            catch (Exception ex)
            {
                Log.Error(ex, "Read worksheet array failed");
            }
            if (lastColumnIndex == 1 & lastRowIndex == 1)
                data = worksheet.get_Range("A1", GetExcelColumnName(lastColumnIndex) +
(lastRowIndex + 1)).Value2 as object[,];
            else if (lastColumnIndex > 0 & lastRowIndex > 0)
                data = worksheet.get_Range("A1", GetExcelColumnName(lastColumnIndex) +
lastRowIndex).Value2 as object[,];
            return data;
        }
```

ДОДАТОК 6

Валідація листа файлу Excel

```
public StatusInfo CheckSheets(BaseFileInfo baseFileInfo)
        {
            if (SheetsInfo.Count == 0)
                return new StatusInfo(false, $"No sheets found");
            //check all sheets found
            IEnumerable<string> requiredSheets = baseFileInfo.GetAllSheetNames();
            List<string> notFoundSheets = new List<string>();
            foreach (string requiredSheet in requiredSheets)
            {
                if (!SheetsInfo.Any(sheetInfo => sheetInfo.SheetName.Trim().ToLower() ==
requiredSheet.Trim().ToLower()))
                {
                    notFoundSheets.Add(requiredSheet);
                }
            }
            if (notFoundSheets.Any())
            {
                Log.Debug($"Not found sheets - <{string.Join(", ", notFoundSheets)}>");
                return new StatusInfo(false, $"Not found sheets - <{string.Join(", ",
notFoundSheets)}>");
            }
            Log.Debug($"Sheets to get data from - <{string.Join(", ", SheetsInfo.Select(x
=> x.SheetName))}>");
            foreach (var sheetInfo in SheetsInfo)
            {
                if (IsSheetProtected(sheetInfo.SheetName))
                {
                    sheetInfo.IsValid = false;
                }
            }
            if (GetCountValidSheets() == 0)
            {
                return new StatusInfo(false, $"All sheets are locked");
            }
            else if (GetCountValidSheets() < SheetsInfo.Count)
            {
                return new StatusInfo(false, $"Sheet(s) <{string.Join("; ",
SheetsInfo.Where(x => !x.IsValid).Select(x => x.SheetName))}> are locked");
            }
            Log.Debug($"Valid sheets to get data from - <{string.Join(", ",
SheetsInfo.Where(x => x.IsValid).Select(x => x.SheetName))}>");
            return new StatusInfo(true, string.Empty);
        }
```

ДОДАТОК 7

Отримання аналітичних даних по командах у розрізі місяця з БД через запит

```
List<TeamAnalysedData> lstOfTeamAnalysedData = new();
        cmd = new SqlCommand(_databaseConfig.GetAnalysedDataCrossMonthQuery,
_sqlConnection);
        using (SqlDataReader reader = cmd.ExecuteReader())
        {
          while (reader.Read())
          {
            int year_ = (int)reader["year_"];
            int month_ = (int)reader["month_"];
            string project_ = (string)reader["Project"];
            string team_ = (string)reader["team_"];
            int hoursspent_ = (int)reader["hoursspent_"];
            int numberofpeople_ = (int)reader["numberofpeople_"];
            lstOfTeamAnalysedData.Add(new TeamAnalysedData
            {
              CorrespondedYear = year_,
              CorrespondedMonth = month_,
              Project = project_,
              Team = team_,
              HoursSpent = hoursspent_,
              NumberOfPeopleWhichWorkedForProject = numberofpeople_
            });

          }
        }
```

ДОДАТОК 8

Відкриття додатку PowerBi Desktop

```
public bool OpenApp(string path)
        {
            try
            {
                try
                {
                    _application = Application.Launch(path);
                    Log.Debug($"Application ProcessId: {_application.ProcessId}");
                    Log.Debug($"Application name: {_application.Name}");
                }
                catch (Exception e)
                {
                    Log.Warning($"Fail open Power BI application by Launch and try to
search application by Fla UI. Message: {e.Message}");
                    try
                    {
                        Thread.Sleep(60000);
                        _application = Application.Attach(path);
                    }
                    catch (Exception e1)
                    {
                        Log.Warning($"Fail open Power BI application by Attach to all path.
Message: {e1.Message}");
                        try
                        {
                            _application = Application.Attach(_application.Name);
                        }
                        catch (Exception e2)
                        {
                            Log.Warning($"Fail open Power BI application by Attach to
application name. Message: {e2.Message}");
                            throw e2;
                        }
                    }
                }
                for (int i = 0; i < 24; i++)
                {       // wait 4 minutes
                    Thread.Sleep(10*1000);
                    try
                    {
                        WinPowerBi =
_application.GetMainWindow(_uiPBIConnector.GetAutomationBase(), TimeSpan.FromSeconds(5));
                        Log.Debug($"{i}. WinPowerBi is null: {WinPowerBi == null},
WinPowerBi.Name is supported: {WinPowerBi?.Properties?.Name?.IsSupported}, WinPowerBi.Name:
{WinPowerBi?.Name}");
                        Log.Debug($"MainWindow name: {WinPowerBi.Name}");
                    }
                    catch (Exception e)
                    {
                        Log.Debug($"Fail log. MainWindow name: {e.Message}");
                        continue;
                    }
                    if (!string.IsNullOrWhiteSpace(WinPowerBi?.Name))
                    {
                        _uiPBIConnector.SetWindowState(WinPowerBi,
WindowVisualState.Maximized);
                        for (int j = 0; j < 20; j++)
                        {
                            Thread.Sleep(10000);
                            WinPowerBi =
_uiPBIConnector.GetMainAppWindowByEndsWithName(_powerBiConfig.PowerBiMain);
                            var previousPages =
_uiPBIConnector.GetElementByName(WinPowerBi, _powerBiConfig.PreviousPages,
SearchScope.FirstDescendant, ControlType.Button);
                            if (previousPages != null)
```

```
                            {
                                Thread.Sleep(10000);
                                return true;
                            }
                        }
                    }
                }
                try
                {
                    _uiPBIConnector.SetWindowState(WinPowerBi,
WindowVisualState.Maximized);
                    for (int j = 0; j < 20; j++)
                    {
                        Thread.Sleep(10000);
                        WinPowerBi =
_uiPBIConnector.GetMainAppWindowByEndsWithName(_powerBiConfig.PowerBiMain);
                        Thread.Sleep(5000);
                        Log.Debug($"WinPowerBi Name: " + WinPowerBi.Name);
                        var previousPages = _uiPBIConnector.GetElementByName(WinPowerBi,
_powerBiConfig.PreviousPages, SearchScope.FirstDescendant, ControlType.Button);
                        Thread.Sleep(5000);
                        Log.Debug($"previousPages: " + previousPages);
                        if (previousPages != null)
                        {
                            Thread.Sleep(10000);
                            return true;
                        }
                    }
                }
                catch (Exception e)
                {

Log.Warning($"Fail open Power BI application. Fail SetWindowState. Message: {e.Message}");
                }
            }
            catch (Exception e)
            {
                Log.Warning($"Fail open Power BI application. Message: {e.Message}");
            }
            return false;
        }
```

ДОДАТОК 9

Пошук елементів PowerBi Desktop за їх типом

```
public AutomationElement[] GetElementsByType(AutomationElement parent, SearchScope scope,
ControlType controlType)
        {
            AutomationElement[] elements = Enumerable.Empty<AutomationElement>().ToArray();
            if (parent == null)
            {
                return elements;
            }
            try
            {
                ConditionFactory conditionFactory = new
ConditionFactory(_automation.PropertyLibrary);
                switch (scope)
                {
                    case SearchScope.Descendants:
                        elements = parent.FindAll(TreeScope.Descendants,
conditionFactory.ByControlType(controlType));
                        break;
                    default:
                        Log.Warning("Unknown SearchScope enum type when invoke
GetElementsByName method");
                        break;
                }
                if (elements.Any())
                {
                    Log.Debug("Elements found, type - <{type}>", controlType);
                }
                else
                {
                    Log.Debug("Elements were not found, type - <{type}>", controlType);
                }
                return elements;
            }
            catch (Exception ex)
            {
                Log.Warning("Get elements failed - <{ex}>", ex);
                return elements;
            }
        }
```

ДОДАТОК 10

Взаємодія із елементом PowerBi Desktop

```
public bool Interact(AutomationElement element, InteractionType interactionType, string
value = null)
        {
            if (element == null)
            {
                Log.Warning("Element to interact with is <null>");
                return false;
            }
            SetForeground(element);
            try
            {
                string name = String.Empty;
                try
                {
                    name = element.Name;
                }
                catch (Exception ex)
                {
                    Log.Debug("Get name for element - <{name}> failed, exception - <{ex}>",
name, ex);
                }
                ControlType type = ControlType.Unknown;
                try
                {
                    type = element.ControlType;
                }
                catch (Exception ex)
                {
                    Log.Debug("Get ControlType for element - <{name}> failed, exception -
<{ex}>", type, ex);
                }
                switch (interactionType)
                {
                    case InteractionType.Select:
                        element.Patterns.SelectionItem.Pattern.Select();
                        Log.Debug("Element <{name}>, type - <{type}> selected", name,
type);

                        break;
                    case InteractionType.Toggle:
                        element.Patterns.Toggle.Pattern.Toggle();
                        Log.Debug("Element <{name}>, type - <{type}> toggled", name, type);
                        break;
                    case InteractionType.Invoke:
```

```
                        element.Patterns.Invoke.Pattern.Invoke();
                        Log.Debug("Element <{name}>, type - <{type}> invoked", name, type);
                        break;
                    case InteractionType.SetValue:
                        if (value != null)
                        {
                            element.Patterns.Value.Pattern.SetValue(value);
                            Log.Debug("Element <{name}>, type - <{type}>, set value -
<{value}>", name, type, "****");
                        }
                        else
                        {
                            Log.Warning("Value to set is <null> when invoke Interact
method");
                        }
                        break;
                    case InteractionType.Expand:
                        element.Patterns.ExpandCollapse.Pattern.Expand();
                        Log.Debug("Element <{name}>, type - <{type}> expanded", name,
type);
                        break;
                    case InteractionType.Collapse:
                        element.Patterns.ExpandCollapse.Pattern.Collapse();
                        Log.Debug("Element <{name}>, type - <{type}> collapsed", name,
type);
                        break;
                    default:
                        Log.Warning("Unknown InteractionType enum type when invoke Interact
method");
                        break;
                }
                return true;
            }
            catch (Exception ex)
            {
                Log.Warning("Interact with element failed - <{ex}>", ex);
                return false;
            }
        }
```

ДОДАТОК 11

Пошук елементів PowerBi Desktop за їх іменем

```
public AutomationElement GetElementByName(AutomationElement parent, string name,
SearchScope scope, ControlType controlType)
        {
            AutomationElement search = null;
            if (parent == null)
            {
                return search;
            }
            try
            {
                ConditionFactory conditionFactory = new
ConditionFactory(_automation.PropertyLibrary);
                switch (scope)
                {
                    case SearchScope.FirstChild:
                        search =
parent.FindFirstChild(conditionFactory.ByName(name).And(conditionFactory.ByControlType(cont
rolType)));
                        break;
                    case SearchScope.FirstDescendant:
                        if (controlType == ControlType.Unknown)break;
                        search = parent.FindFirst(TreeScope.Descendants,
conditionFactory.ByName(name).And(conditionFactory.ByControlType(controlType)));
                        break;
                    default:
                        Console.WriteLine("Unknown SearchScope enum type when invoke
GetElementByName method");
                        break;
                }
                if (search != null){}
                else
                {
                    Console.WriteLine("Element was not found - <" + name + ">, type - <" +
controlType + ">");
                }
                return search;
            }
            catch (Exception ex)
            {
                Console.WriteLine("Get <" + name + "> element failed - <" + ex + ">");
                return search;
            }
        }
```