Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ДЛЯ ФОРМУВАННЯ ЗВІТНОСТІ ЗА КОЛЕКЦІЯМИ ДОКУМЕНТІВ
SOFTWARE IMPLEMENTATION OF AN APPLICATION FOR GENERATING REPORTS ON DOCUMENT COLLECTIONS

Виконав: студент(ка) 4 курсу, групи КНТ-117
Спеціальності 121 Інженерія програмного забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Таряник М.В.
(прізвище та ініціали)
Керівник Гофман Є.О.
(прізвище та ініціали)
Рецензент Скрупський С.Ю.
(прізвище та ініціали)

2021

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____С.О. Субботін
“____” _____2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Тарянику Максиму Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація застосунку для формування звітності за колекціями документів. Software Implementation of an Application for Generating Reports on Document Collections

керівник проєкту (роботи) Гофман Євгеній Олександрович, к.т.н.,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від “ 30 ” березня 2021 року № 103
2. Строк подання студентом проєкту (роботи) травень 2021 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Гофман Є.О., доцент		
Нормоконтролер	Липовець М.В., асистент		

7. Дата видачі завдання “ 1 ” лютого 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2-5 тижні	Розділи 1, 2
3	Розробка архітектури програми	6 тиждень	Розділ 2
4	Розробка програми	7-9 тижні	Розділ 3
5	Тестування та експериментальне дослідження	10 тиждень	Розділ 4
	програми		
6	Оформлення пояснювальної записки та документів	11 тиждень	Додатки
	до неї. Нормоконтроль та рецензування		
7	Захист роботи	12 тиждень	

Студент(ка) _____ Таряник М.В.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ Гофман Є.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до випускної дипломної роботи бакалавра: 96 с., 32 рис., 4 табл., 4 дод., 18 джерел.

ДОКУМЕНТООБІГ, МОНІТОРИНГ, ЗВІТНІСТЬ, КОЛЕКЦІЯ ДОКУМЕНТІВ, ПРОГРАМНА РЕАЛІЗАЦІЯ, ПРОЄКТУВАННЯ, РОЗРОБКА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об’єкт дослідження – процес формування звітності за колекціями документів.

Предмет дослідження – методи формування звітності за колекціями документів.

Метою роботи є програмна реалізація застосунку для формування звітності за колекціями документів.

Матеріали, методи та технічні засоби: об’єктно-орієнтоване програмування, мова програмування C#, середовище розробки Microsoft Visual Studio.

Результати. Розроблено автоматизовану систему у вигляді десктопного застосунку для формування звітності за колекціями документів.

Практична цінність роботи полягає в частковій автоматизації процесів документообігу, що сконцентровані на аналізі та видобутку даних із колекцій документів для подальшої звітності. Це забезпечується за рахунок використання системи у вигляді програмного застосунку, що може формувати звітність за колекціями документів.

Висновки. Розроблено програмне забезпечення, що реалізує формування звітності за колекціями документів для співробітників структурних підрозділів установ.

Галузь використання – працівники структурних підрозділів установ та підприємств.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 96 pages, 32 figures, 4 tables, 4 appendixes, 18 sources.

DOCUMENT MANAGEMENT, MONITORING, REPORTING, DOCUMENT COLLECTION, SOFTWARE IMPLEMENTATION, DESIGN, DEVELOPMENT, SOFTWARE.

The object of research is the process of generating reports on document collections.

The subject of the research is methods of generating reports on document collections.

The purpose of this work is to implement a software application for generating reports on document collections.

Materials, methods, and technical tools: Object-Oriented Programming, The C# programming language, and the Microsoft Visual Studio development environment.

Results. An automated system has been developed in the form of a desktop application for generating reports on document collections.

The practical value of the work lies in the partial automation of document management processes, which are focused on analyzing and extracting data from document collections for further reporting. This is ensured by using the system in the form of a software application that can generate reports on document collections.

Conclusions. Software has been developed that implements reporting on collections of documents for employees of structural divisions of institutions.

Field of usage: employees of structural divisions of institutions and enterprises.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

GUI – графічного інтерфейсу користувача;

MVC – Model-view-controller;

UML – Unified Modeling Language;

ЗФЗКД – застосунок формування звітності за колекціями документів;

ОС – операційна система

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

СЕД – системи електронного документообігу.

ВСТУП

Багато обчислень, пов’язаних із повсякденною діяльністю людини, доцільно виконувати в табличному вигляді. До таких обчислень належать, скажімо, бухгалтерські розрахунки, облік обороту матеріалів і продукції на заводі, товарів на складі, різні інженерні і статистичні розрахунки. У вигляді таблиць можна оформляти ділові документи: рахунки, накладні, відомості тощо. Взагалі зображення даних у вигляді прямокутних таблиць є надзвичайно зручним і звичним.

Розвиток програмного забезпечення комп’ютерів вплинув і на галузь табличних обчислень. Для оперування табличними даними є сучасні програми, названі електронними таблицями.

Електронна таблиця – це програма, що моделює на екрані двовимірну таблицю, яка складається з рядків і стовпців. Основним призначенням електронної таблиці є введення даних до комірок й обробка їх за формулами. Електронна таблиця є універсальним засобом для автоматизації розрахунків над табличними даними.

За допомогою табличних процесорів можна не тільки автоматизувати розрахунки, а й ефективно проаналізувати їхні можливі варіанти. Змінюючи значення одних даних, можна спостерігати за змінами інших, що залежать від них. Такі розрахунки виконуються швидко і без помилок, надаючи користувачу за лічені хвилини велику кількість варіантів розв’язання задачі.

Тому важливим завданням є розроблення застосувань для Microsoft Excel, які дозволяють безпосередньо працювати з даними, організовуючи їх зручним чином та виконувати пошук даних за визначеними фільтрами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Електронний документообіг. Переваги та специфічні нюанси

Діловодство – комплекс заходів з організації документообігу підприємства або організації. Діловодство – це термін, застосовуваний в конторській практиці для позначення формального набору правил роботи з документами [1].

Електронний документ – документ, створений за допомогою засобів комп'ютерної обробки інформації, який може бути підписаний електронним підписом і збережений на машинному носії у вигляді файлу відповідного формату [1].

Документообіг – система та процес створення, інтерпретації, передачі, прийому та архівування документів, а також контролю за їх виконанням і захисту від несанкціонованого доступу [1].

Електронний документообіг – сукупність автоматизованих процесів по роботі з документами, представленими в електронному вигляді, з реалізацією Концепції «безпаперового діловодства» [1]-[2].

Види електронного документообігу [1]-[2]:

- виробничий документообіг;
- управлінський документообіг;
- архівна справа (сукупність процедур архівного документообігу);
- кадровий документообіг (процедури кадрового обліку);
- бухгалтерський документообіг;
- складський документообіг;
- секретне і конфіденційне діловодство;
- технічний і технологічний документообіг.

Очевидно, що систем документообігу може бути стільки ж, скільки існує видів діяльності, як наслідок, інформаційні системи, що автоматизують приватні види документообігу, розвиваються у напрямку масовості [1]-[2].

1.1.1 Переваги електронного документообігу

Економія часу: службовці витрачають менше часу на пошук паперових документів [3]. Завдяки центральній базі даних, регулярно створюються резервні копії файлів, завдяки чому виключається можливість того, що документ буде безповоротно втрачений, якщо його забудуть в літаку, випадково або навмисно знищать або ж просто згине в офісному безладді. Абсолютно виключається втрата часу на пошуки файлів і документів, яких, з якоїсь причини, не виявилося на своєму місці [4].

Більш адекватне використання фізичного простору і техніки: цінні квадратні метри, зайняті зайвими серверами та іншими пристроями для зберігання документів можуть бути звільнені. Залежно від статусу та актуальності інформації, документи і файли можуть безпечно видалятися після закінчення терміну їх зберігання. Управління даними не тільки допомагає відповідати корпоративним нормам, але і сприяє більш адекватному використанню місця для зберігання [3]-[4].

Підвищення прозорості внутрішньої роботи підприємства: системи електронного документообігу (СЕД) дозволяють керівникам спостерігати за статусом документа, протягом усіх етапів його узгодження та затвердження. На додаток до цього, СЕД дозволяє моментально і легко викликати не тільки запитуваний файл, але також і повний звіт про те, хто його створив, Хто мав до нього доступ і хто його редагував [3]-[4].

Ведення особистої історії кожного файлу і супутньої документації: СЕД дозволяють централізовано управляти взаємовідносинами з клієнтами і постачальниками [5]. Наприклад, достатньо лише одного клацання миші, щоб викликати всі необхідні документи, які містять вимоги, пов'язані з різними типами взаємин між організацією і зовнішніми суб'єктами [5].

Більше гнучкості щодо фізичного місцезнаходження співробітників: завдяки можливостям електронного доступу та комунікацій, службовці отримують можливість працювати віддалено. І навіть перебуваючи в одному

і тому ж географічному місці, службовцям більше не потрібно чекати, поки паперові копії файлів будуть пересилатися з сусіднього офісу [3]-[5].

Підвищення безпеки інформації та документів: як уже згадувалося, центральна база даних дозволяє робити резервні копії документів, завдяки чому знижується ризик випадкової або навмисної втрати файлів. При цьому, менше часу витрачається на пошуки необхідного документа, якщо його місцезнаходження з якоїсь причини змінилося [6].

Зниження витрат на роздруківку, Поштові марки, конверти і пересилання: паперові документи, які пересилаються між відділами або Постачальниками, можуть пересилатися в електронному вигляді [7].

Підвищення рівня задоволеності службовців і керівників: оптимізація щоденних завдань дозволяє співробітникам отримувати більше задоволення від робочого процесу. Звільнення співробітників від таких, часто нудних завдань, як обробка накладних, дозволяє їм присвятити себе іншій діяльності. У той же час, керівники відділів отримують більше можливостей контролювати роботу своїх підлеглих. Зрештою, деякі організації можуть виявити, що заощаджені кошти дозволяють їм вийти на новий бізнес-рівень [6]-[7].

1.1.2 Завдання систем електронного документообігу

Забезпечення ефективного управління за рахунок автоматичного контролю виконання, прозорості діяльності всієї організації на всіх рівнях [3]-[7].

Підтримка системи контролю якості, що відповідає міжнародним нормам [3]-[7].

Підтримка ефективного накопичення, управління та доступу до інформації та знань. Забезпечення кадрової гнучкості за рахунок більшої формалізації діяльності кожного співробітника і можливості зберігання всієї передісторії його діяльності [3]-[7].

Протоколювання діяльності підприємства в цілому (внутрішні службові розслідування, аналіз діяльності підрозділів, виявлення «гарячих точок» в діяльності) [3]-[7].

Оптимізація бізнес-процесів і автоматизація механізму їх виконання і контролю [3]-[7].

Виключення паперових документів з внутрішнього обороту підприємства. Економія ресурсів за рахунок скорочення витрат на управління потоками документів в організації [3]-[7].

Виключення необхідності або істотне спрощення і здешевлення зберігання паперових документів за рахунок наявності оперативного електронного архіву [3]-[7].

1.1.3 Критерії вибору системи електронного документообігу

Вимоги за обсягом зберігання. Необхідно вибрати систему документообігу, що підтримує ієрархічне структурне зберігання. Цей механізм зберігає найбільш активно використовувані дані на найбільш швидких, але і найбільш дорогих носіях, в той час як рідше використовувана інформація автоматично переноситься на повільні і дешеві носії [6]-[7].

Наявність формалізованих процедур, що вимагають підтримки їх виконання і автоматизації контролю (підготовки документів певного типу, виконання стандартних функцій організації і т.п.) [6]-[7].

Необхідність автоматизації адміністративного управління організацією. Ступінь складності організаційної структури.

Наявність територіально розподілених підрозділів. Цей фактор накладає певні вимоги до віддаленого доступу, до реплікації даних і т.п. [6]-[7].

Наявність паперового архіву великого обсягу. Деякі системи документообігу поставляються з уже інтегрованими підсистемами масового введення документів [6]-[7].

Наявність не задовольняє поточним потребам системи документообігу [6]-[7].

Необхідність у розвиненій маршрутизації документів, в управлінні потоками робіт. Як продовження цієї необхідності потреба в підтримці довільних бізнес-процесів, можливо працюють спільно з прикладними системами підтримки цих процесів [6]-[7].

Вимоги щодо термінів зберігання документів. При великих термінах зберігання (десятки років) варто серйозно подумати про організацію паралельного архіву на мікрофільмах [6]-[7].

Вимоги до «відкритості», розширюваності системи. Можливість інтеграції з існуючими інформаційними системами та використання наявного обладнання [6]-[7].

Необхідність зберігання зображень документів. Використання в організації специфічних форматів зберігання документів. Необхідність підтримки інженерних і конструкторських завдань, інших особливостей діяльності підприємства [6]-[7].

Необхідність розвинених засобів пошуку інформації. Повна підтримка системою мов наявних в організації документів [6]-[7].

Вимоги до безпеки (шифрування, організація доступу, і т.п.). Можливість використання вже наявних в інформаційній інфраструктурі організації механізмів доступу в системі документообігу.

Вимоги щодо відповідності певним стандартам: внутрішнім, галузевим, ГОСТ, міжнародним стандартам з контролю якості, рівню організації зберігання інформації [6]-[7].

1.2 Колекції документів на підприємствах та в установах

Електронні колекції відносяться до виду електронних інформаційних ресурсів, які в даний час активно розробляються бібліотечно-інформаційними установами [1], [5]-[6]. Створення електронних колекцій

дозволяє їм забезпечити збереження наявних унікальних документів (раритетних видань, рукописів, краєзнавчих документів тощо), акумулювати розрізнені по структурних підрозділах (а іноді і по різних установах, організаціях, особистих фондах) матеріали; розширити склад інформаційних ресурсів, використовуваних при організації інформаційно-бібліотечного обслуговування користувачів [1], [5]-[6].

Аналіз існуючих визначень поняття «електронна колекція» дозволяє виявити відмінні ознаки, властиві електронним колекціям, до числа яких відносяться [1], [5]-[6]:

– цілісність зборів;

– взаємозв'язок представлених у колекції документів / даних за певними ознаками (змістовним, формальним);

– повнота відображення предмета колекції за тривалий період його розвитку;

– єдині принципи відбору матеріалу;

– подання документів в цифровій формі;

– загальний інтерфейс колекції;

– можливість багатоаспектного пошуку елементів колекції.

Слід зауважити, що для позначення електронної колекції розробниками використовується різна лексика: «електронна колекція», «електронний архів», «мультимедійний літопис», «літопис міста», «фотолітопис» та ін. [1], [5]-[6].

Електронна колекція документів – цілісна сукупність документів, об'єднаних за однією або кількома ознаками, представлена в цифровій формі з єдиним інтерфейсом і володіє засобами навігації і пошуку її елементів [1], [5]-[6].

Оскільки бібліотечно-інформаційні установи переважно об'єднують в рамках колекцій документи, далі будемо розглядати електронні колекції документів [1], [5]-[6].

1.3 Порівняння існуючих аналогів

На сьогоднішній день на просторах Інтернету існує достатньо багато систем, що схожі за функціоналом та загалом надають програмне забезпечення (ПЗ) для автоматизації та діджиталізації внутрішніх процесів документообігу [8]-[10]. Проте слід зазначити, що більшість з них досить різняться за основним та фокусом роботи.

Так деякі сервіси зосереджені на повній автоматизації процесів документообігу: від створення шаблонів або документів до їх архівування.

Інші ж сервіси надають можливість колективно опрацьовувати електронні документи, проте їх фокус зосереджено саме на захисті документів та процесу роботи над ними через використання електронних цифрових підписів.

Третя ж група зосереджена на частковій обробці окремо розроблених та оновлювальних документах підприємства або установ. Тобто просто надає систему для менеджменту цих документів в локальній мережі компанії.

1.3.1 Сервіс «Docsvision»

Рішення «Docsvision» дозволяє значно прискорити всі процеси з обліку та підготовки документів в компанії [8]. Налаштовані маршрути, сценарії виконання, шаблони і автовідповіді – система сама проводить всю попередню обробку документів. Знижується навантаження на секретарів, діловодів, співробітників Служби канцелярії. В рази скорочуються терміни кожного етапу виконання документів [8].

При розміщенні документа в системі він проходить автоматичну реєстрацію: створюється картка, формується назва і номер згідно з різними правилами нумерації [8]. Частина полів заповнює система, в інші – дані передаються з довідників, для чого достатньо ввести перші літери значення. Це знижує ймовірність помилки введення даних через людський фактор [8].

Швидкість обробки документів часто безпосередньо впливає на оперативність роботи всієї організації [8]. Рішення «Docsvision» надає користувачам інструменти, що допомагають зробити роботу більш структурованою, швидкою і комфортною в зручному інтерфейсі, який можна адаптувати під себе. Переведення діловодства в електронний вигляд зменшує трудовитрати фахівців з ведення і контролю документообігу, виключає ризик втрати документів або дублювання обробки одного запиту [8].

На рисунку 1.1 наведено приклад використання програми «Docsvision».

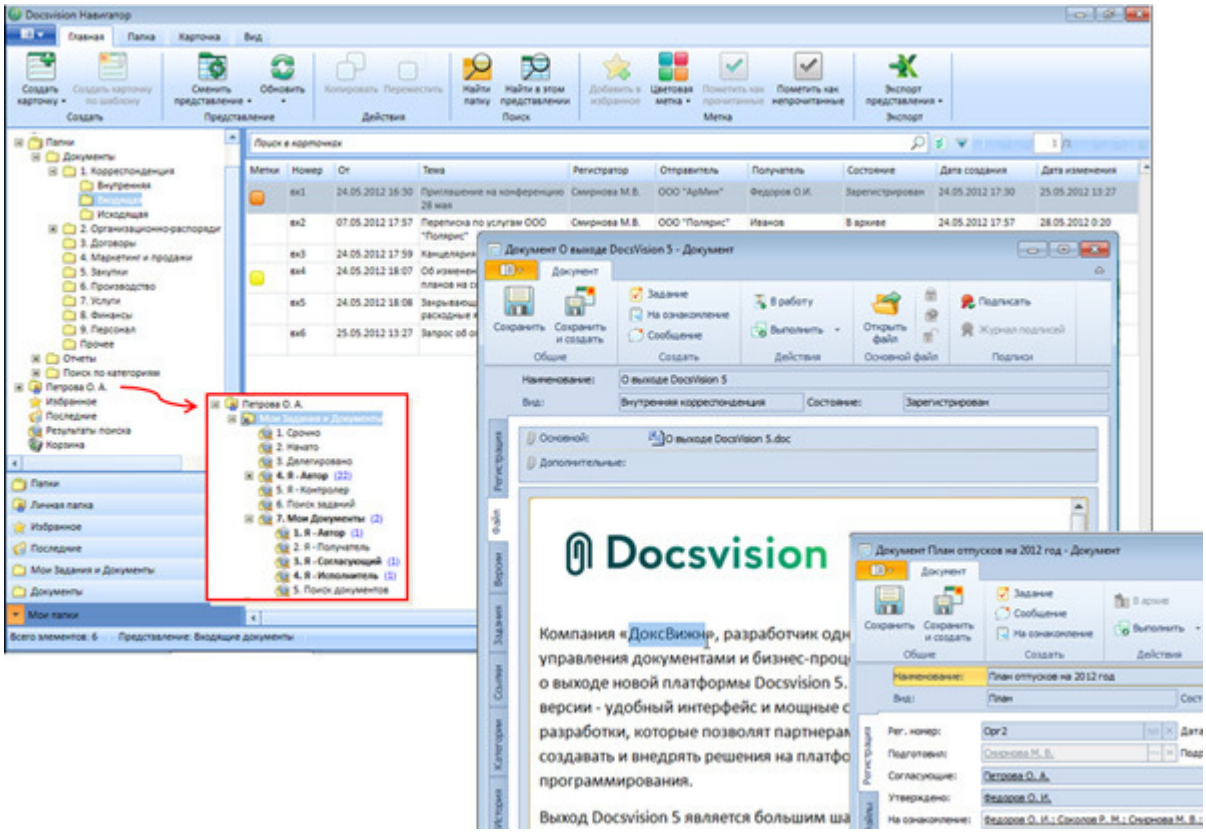


Рисунок 1.1 – Приклад використання програми «Docsvision»

1.3.2 Система «ВЧАСНО»

Система «ВЧАСНО» пропонує перевести в електронний режим наступні документи [9]:

– договори, додаткові угоди;

- рахунки і акти надання послуг;
- видаткові накладні;
- товарно-транспортні накладні;
- службові листи; д
- окументи внутрішнього документообігу.

Щоб зав'язати компанію з сервісом ВЧАСНО необхідні доступ до Інтернету і електронний цифровий підпис [9]. Згодом компонувальник запрошує виробити пару кроків: зареєструватися в сервісі і зафіксувати співробітників; познайомитися з приписами використання і ознайомити з ними співробітників; оповістити партнерів і контрагентів про перехід на електронний документообіг; підписувати і виконувати документи контрагентам [9].

«ВЧАСНО» пропонує три різних види тарифікації, що значно підвищує ймовірність підбору раціонального в підневільності через незвичайностей бізнесу [9].

«Початковий» – безоплатний тариф, користуючись яким можливо підписувати і відправляти як вхідні, так вихідні документи, приберігати документи до 3 років. Так, ймовірність тлумачити документи ніби внутрішнього документообігу, аналогічно спільно з зовнішніми контрагентами [9].

«Професійний» тариф значно дорожчий, але надає більші (крім зорієнтованих у початковому тарифі) здібності у варіанті внутрішніх погоджень, додаткових параметрів і сценаріїв документів (угруповання після поставлених ознак, наприклад), моральний оборот і ймовірність п'ятирічного збереження електроних документів у хмарному сховищі [9].

Тариф «інтеграція «Розцінок», ще дорожчий. Даний розцінок дозволяє діяти з обліковою налагодженістю підприємства, дозволяє підписувати і виконувати укладаються і вихідні документи; тлумачити їх якщо необхідно як з співробітниками, так і з партнерами [9].

На рисунку 1.2 наведено приклад використання програми «ВЧАСНО».

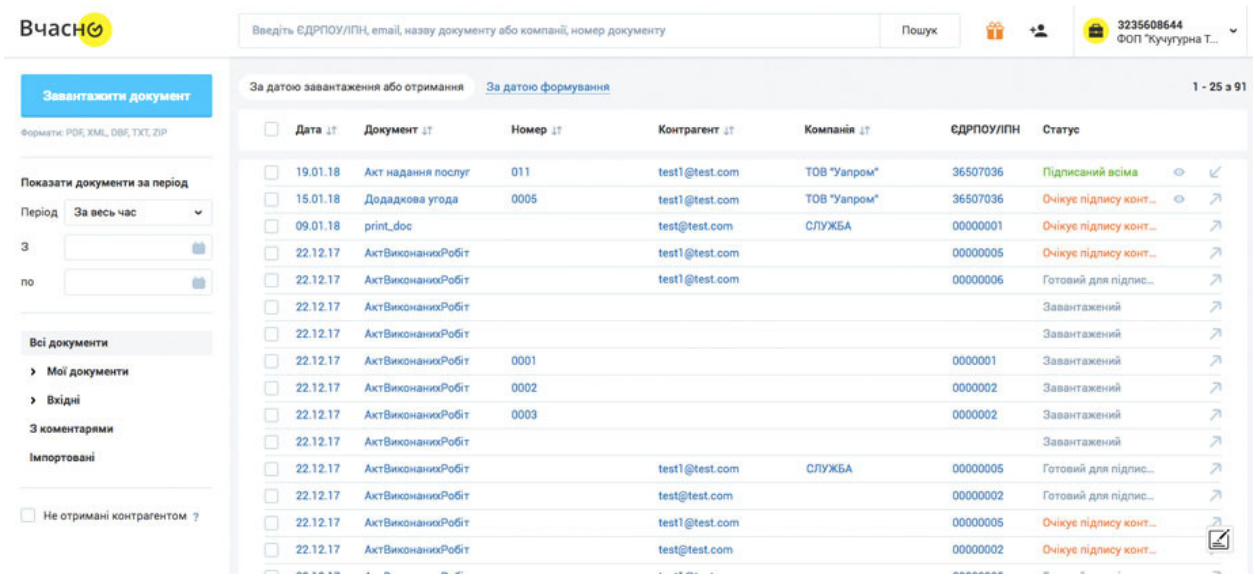


Рисунок 1.2 – Приклад використання програми «ВЧАСНО»

1.3.3 Сервіс «Document.Online»

Інший визнаний актуальний сервіс електронного документообігу – «Document.Online» [10]. Сервіс підтримує співпацю з усіма акредитованими центрами сертифікації ключів в Україні і позиціонується ніби єдина програма для внутрішнього і зовнішнього документообігу. Користуватися ним можливо з кожних девайсів: стаціонарних і мобільних пристроїв в усіх куточках світу. Сервіс «Document.Online» кроссфплатформний, поставленою для користувача [10].

Архітектура «Document.Online» дозволяє створювати, редагувати та оновляти документи з готових шаблонів. Крім цього, є шанс розглядати завантажені, створені та оновлені документи з відображенням інформації про підписанта [10]. Крім цього, користувачу доступний функціонал з обміну документами і установка маршрутів для документації, що дає можливість реалізувати процес узгодження і підпису документа як всередині підприємства, так і з контрагентами, та сповістити іншим співробітникам про

узгодження і підписання. «Document.Online» надає налагодженість контролю над документообігом: модифікування статусу документа після виконання дій, попередження про завершення маршруту документа і подібне [10].

На рисунку 1.3 наведено приклад використання програми «Document.Online».

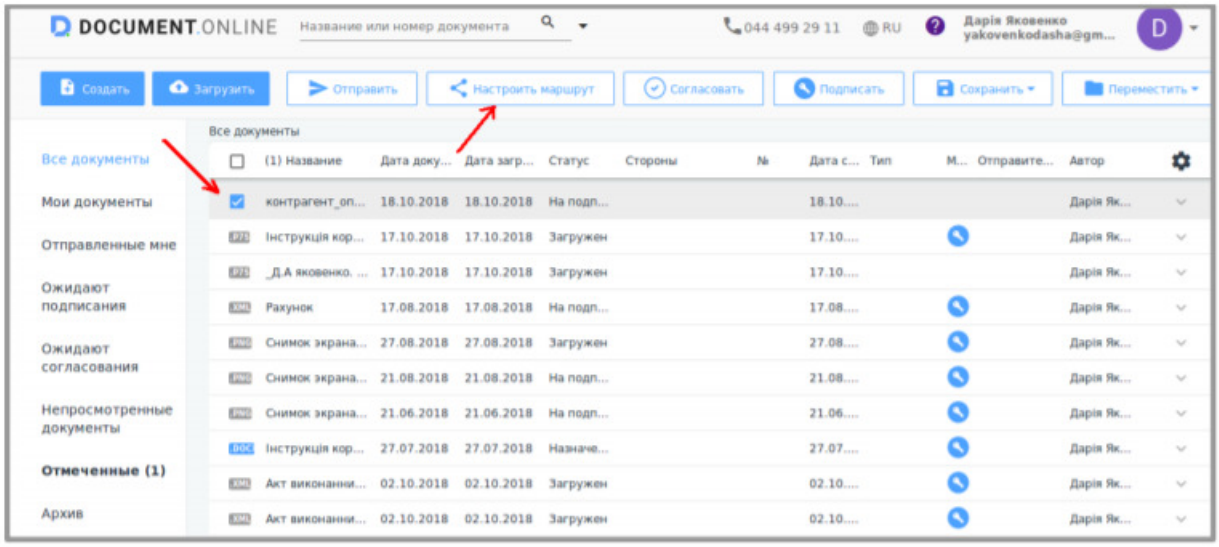


Рисунок 1.3 – Приклад використання програми «Document.Online»

1.3.4 Порівняння існуючих аналогів

Проведемо порівняння за певними критеріями, хід якого представимо у вигляді таблиці (табл. 1.1). У таблиці будуть порівнюватися попередні аналоги із системою, що розробляється – застосунок формування звітності за колекціями документів (ЗФЗКД). Серед критеріїв виділимо наступні:

– вартість використання – чи є система платною, чи є безкоштовні версії системи, а також підтвердження додаткового функціоналу, що доступний виключно в платній версії;

– підтримка роботи над колекціями документів – чи підтримує систему роботи над колекціями документів;

– внутрішнє ранжування документів – чи є в системі ранжування документів за важливістю;

– збереження архівної інформації – чи зберігає система певний архів документів з якого можливо відновити файл.

Таблиця 1.1 – Порівняння існуючих аналогів

Критерій порівняння	Docsvision	ВЧАСНО	Document. Online	ЗФЗКД
Вартість використання	Висока	Є безкоштовна версія	Висока	Безкоштовно
Підтримка роботи над колекціями документів	Відсутня	Відсутня	Відсутня	Підтримує
Внутрішнє ранжування документів	Відсутнє	Відсутнє	Відсутнє	Підтримує
Збереження архівної інформації	Підтримує	Тільки в платній версії	Підтримує	Підтримує

З результатів порівняння можна зробити висновки, що більшість схожих систем відрізняються досить великою вартістю використання. І хоча в деяких аналогів є безкоштовна версія її функціонал значно обмежений та урізаний. Більш того, більшість систем зосередилися саме на підтримці та верифікації документів за електронним цифровим підписом і не надають функціоналу саме для робот із колекціями документів для формування звітності або фільтрації інформації в колекціях. Також слід відзначити відсутність ранжування документів в більшості систем, що може ускладнити роботу із потоком документів. Надання функціоналу архівування в більшості аналогів вимагає додаткових витрат для купівлі відповідного тарифу.

Тож можна засвідчити, що наразі є актуальною задача розробки системи для формування звітності за колекціями документі для підвищення рівня автоматизації в системах електронного документу обігу для підприємств та установ. Це дозволить пришвидшити внутрішні процеси структурних підрозділів та зменшити ризики помилок пов’язаних із людським фактором втручання.

1.4 Постановка завдання

За результатами аналізу проблемної області та дослідження існуючих аналогів для вирішення проблемних питань, сформулюємо постановку задачі.

Метою є програмна реалізація застосунку для формування звітності за колекціями документів. Серед функціональних вимог визначимо:

- реєстрація користувачів;
- вивід інформації про категорії даних;
- перегляд інформації;
- пошук інформації;
- додавання інформації;
- видалення інформації;
- оформлення звіту з обраної інформації;
- вивід створених або збережених звітів на друк.

Система буде розроблена у вигляді десктопного застосунку, що гарантує простоту розгортання на робочих станціях користувачів.

Розглянемо вимоги до розроблюваної системи за їх типом [11]-[12].

1.4.1 Функціональність системи

Програма повинна дозволяти користувачам виконувати наступні функції:

а) завантажувати файлу Microsoft Excel;

б) читати та записувати дані в файли Microsoft Excel;

в) виконувати пошук вмісту, використовуючи фільтри.

Також додатково слід розглянути визначення функціональних вимог за категоріями загальних та специфічних вимог, як представлено у таблиці 1.2.

Таблиця 1.2 – Функціональні вимоги до системи за категоріями

Загальні	Специфічні
Забезпечення створення електронних документів (сканування, імпорт)	Створення дискусій щодо документів
Можливість додавання коментарів до документів	Порівняння змісту документів
Контроль виконання документів	Коментування змісту за допомогою виділення в тексті «червоним олівцем»
Забезпечення звітності та аналізу	Масове завантаження документів в систему
Забезпечення друку електронних документів, метаданих	Імпорт документів
Колективна обробка документів	Публікація певних видів документів
Відправка повідомлень і оповіщень користувачам системи	
Зберігання та класифікація документів	
Пошук документів	

1.4.2Вимоги до інтерфейсу

Для забезпечення ергономічності, зручності на зрозумілості інтерфейс розроблюваного ПЗ повинен:

– бути зручним та інтуїтивно-зрозумілим;

– при першому користуванні продуктом користувач отримує підказки;

– пароль відображається зірочками.

1.4.3Не функціональні вимоги до продукту

Для нормального функціонування ПЗ необхідно виконання технічних умов.

Вимоги до клієнтської частини:

– IBM-сумісний персональний комп’ютер (ПК) під управлінням операційної системи (ОС) з лінійки Windows не старше 7 версії;

– вільне місце на накопичувачі – від 1 Гб;

– оперативна пам’ять стандарту DDR3 об’ємом від 4 Гб;

– мережева карта.

1.3 Висновки за розділом 1

Даний розділ містить у собі аналіз існуючих систем, які дозволяють автоматизувати певні етапи в процесі електронного документообігу підприємств та установ.

Таким чином, на підставі проведеного аналізу засобів, методів та технологій для формування звітності за колекціями документів, було виділено основні функціональні можливості, які повинна мати програмна реалізація застосунку для формування звітності за колекціями документів.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

У попередньому розділі було проведено аналіз засобів та систем, що дозволяють автоматизувати та налаштувати процеси електронного документообігу, розглянуто основні можливості таких засобів. Виходячи з проведеного аналізу, було сформульовано основні можливості, якими повинна оперувати система, що проєктується.

Що стосується технічних та програмних можливостей, то система повинна забезпечувати імпорт даних про студентів з таблиць у форматі *.xls, виконувати їх обробку, та передавати відповідні дані у документ *.doc для формування відповідних документів стосовно випускників та студентів навчального закладу. Система для такої взаємодії розроблена із використанням мови програмування C#, базуючись на вбудованих класах мови, що дозволяють працювати з офісними додатками – Microsoft.Office.Interop.Excel, Microsoft.Office.Interop.Word.

Даний розділ містить опис питань, які стосуються безпосередньо структури системи, що розробляється, вибору середовища проєктування, а також визначаються основні та специфічні архітектурні рішення, які має система.

2.1 Вибір середовища проєктування

2.1.1 Вибір мови програмування

На сьогоднішній момент мова програмування C# один з найпотужніших, швидко розвиваються і затребуваних мов в IT-галузі. На даний момент на ньому пишуться найрізноманітніші програми: від невеликих десктопних програмок до великих веб-порталів і веб-сервісів, які обслуговують щодня мільйони користувачів [13]-[14].

C# вже не молода мова і вся платформа. Net вже пройшов великий шлях. Перша версія мови вийшла разом з релізом Microsoft Visual Studio.Net

в лютому 2002 року. Поточною версією мови є версія C# 9.0, яка вийшла 10 листопада 2020 року разом з релізом.Net 5 [13]-[14].

C# є мовою з Сі-подібним синтаксисом і близький в цьому відношенні до C++ і Java. Тому, якщо Ви знайомі з одним з цих мов, то оволодіти C# буде легше [13]-[14].

Завдяки неабиякій потужності мови C#, на нього впав вибір розробників движка Unity. Сьогодні є одним з топових движків для ігор на Windows. Випуск і активне використання движка пішли на руку C#, який став ще популярнішим [13]-[14].

Після базового вивчення C# для чайників, можна відразу перейти до поглибленого вивчення бібліотек і суміжних технологій. Більшість сучасних проєктів пишуться з їх залученням, на чистому C# працюють рідко [13]-[14].

Якщо порівняти C# з іншими мовами, його можна назвати досить молодим, хоча за плечима вже пройдено чималий шлях. Вперше повноцінна версія мови з'явилося після виходу Microsoft Visual Studio. Net, подія датується лютим 2002 року. На сьогодні актуальна версія – C# 7.0, випущена 7.03.2017 паралельно з Visual Studio 2017 [13]-[14].

C# є об'єктно-орієнтованим і в цьому плані багато перейняв у Java і C++. Наприклад, C# підтримує поліморфізм, успадкування, перевантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і розширюваних додатків. І C# продовжує активно розвиватися, і з кожною новою версією з'являється все більше цікавих функціональностей, як, наприклад, лямбди, динамічне зв'язування, асинхронні методи і т.п. [13]-[14].

Так як C# – це проєкт Microsoft, то у нього все в порядку з підтримкою і з бібліотеками. У світі C# бібліотеки є практично для всього, в тому числі і для роботи з нейромережами і машинним навчанням — ML.NET [13]-[14].

Це означає, що його можна використовувати всі можливості нейронів в застосунках і об'єднувати їх за допомогою одного і того ж мови програмування [13]-[14]. А так як C# – кроссплатформна мова, то машинне навчання можна вбудувати практично у що завгодно, від мобільних застосунків до електроніки [13]-[14].

Проведемо порівняння мов програмування для реалізації мобільного застосунку, результати порівняння представимо в табличній формі (табл. 2.1). У якості критеріїв будемо використовувати наступні:

– наявність літератури та простота вивчення – сукупний критерій для визначення спільноти розробників та популярності мови програмування;

– стандартні бібліотеки шаблонів – наявність таких бібліотек може спростити розробку;

– кроссплатформність – чи може розроблене ПЗ бути кроссплатформним;

– мультипарадигмність – використання декількох парадигм програмування може спростити розробку;

– сумісність з пакетом Microsoft Office – чи підтримує мова програмування механізми та бібліотеки для кращої роботи із файлами пакетів Microsoft Office;

– простота розроблення графічного інтерфейсу користувача (GUI) – наскільки просте розроблення GUI при роботі із мовою програмування.

–

Таблиця 2.1 – Порівняння мов програмування

	C#	Python	Java
Наявність літератури та простота вивчення	+ +	+ +	+ +
Стандартні бібліотеки шаблонів	+	+	+
Кроссплатформність	±	+	+ +
Мультипарадигмність	+ +	+ +	±
Сумісність з пакетом Microsoft Office	+ +	±	±
Простота розроблення GUI	+ +	+	+

За результатами порівняння можна зробити висновок, що C# залишається кращою альтернативою, адже має і розширену спільноту, і набір розширень, що спрощують розробку застостуснків, що працюють із файлами пакету Microsoft Office. Для написання нашого застосунку за обраною темою розробки цілком достатньо вибрати мову C#.

2.1.2 Роль платформи .NET Framework

Microsoft .NET Framework – це програмна платформа, розроблена компанією Microsoft. Вона може бути інстальована на комп'ютерах з операційною системою Microsoft Windows. .NET Framework містить велику бібліотеку класів, які містять готові запрограмовані рішення для типових завдань, а також віртуальну машину для виконання програм, написаних спеціально для цієї платформи [13]-[14].

Бібліотека базових класів .NET (Base Class Library) надає широкий діапазон функцій для програмування користувацького інтерфейсу, доступу до даних, веб-аплікацій, мережевих систем, шифрування даних тощо. Ці класи використовуються прикладними програмістами, які, поєднуючи їх із власним кодом, створюють програми [13]-[14].

Основні принципи та переваги .NET [13]-[14]:

– взаємодія з іншими програмами. Оскільки часто необхідно пов'язати функціональність .NET із програмами, які виконуються поза середовищем .NET, платформа надає можливості для взаємодії з сторонніми компонентами: COM-компонентами – за допомогою InteropServices, Native-кодом – за допомогою механізму P/Invoke.

– спільне середовище виконання (Common Language Runtime, CLR). Всі .NET-програми виконуються у спеціальній віртуальній машині. Завдяки цьому можна очікувати від всіх програм однакової поведінки в плані керування пам'яттю, безпеки та обробки помилок;.

– незалежність від мови програмування. Платформа .NET містить так звану спільну систему типів (Common Type System, CTS), яка специфікує всі допустимі типи даних, якими може оперувати CLR, і описує, яким чином вони можуть чи не можуть взаємодіяти один з одним. Завдяки цьому бібліотеки класів, написані для CLR однією з мов програмування, можуть бути використані в проєктах, що розробляються на іншій мові;

– переносимість. .NET Framework спроєктовано таким чином, що вона теоретично незалежна від платформи, а отже, може бути міжплатформною. Програми для .NET не компілюються відразу в машинний код, а тільки в проміжне представлення коду, яке називається Common Intermediate Language (CIL). При запуску програми на цільовій платформі трансляцією проміжного коду в машинний код займається JIT(Just In Time)-компілятор. Тому .NET-програма може виконуватись на будь-якій архітектурі, для якої реалізовано середовище виконання .NET відповідної версії. В даний момент вже існує реалізація .NET для Unix-подібних платформ (проєкт Mono під керівництвом компанії Novell);

– «Збір сміття» – автоматичне звільнення пам’яті, зайнятої об’єктами, що більше не використовуються. Ця можливість звільняє програміста від відповідальності за виділення ділянок пам’яті та їх вивільнення [13]-[14].

Платформа .NET разом з середовищем розробки Microsoft Visual Studio та мовою програмування C# надають широкі засоби розробки офісних застосувань на різних рівнях.

2.1.3 Вибір середовища розробки

За визначенням фірми Microsoft, Microsoft Visual Studio – це продукт, призначений для розробки заснованих на мові XML розподілених Web-сервісів і застосунків, які забезпечать доступ до інформації в будь-якому місці, в будь-який час і через будь-який пристрій. Коротко перерахуємо цілі,

які переслідувала команда розробників, починаючи роботу над новою версією засобу розробки [15]:

- полегшити створення XML Web-сервісів і Web-додатків;
- максимально збільшити продуктивність розробників;
- розширити функціональність мов програмування;
- забезпечити підтримку всього циклу створення корпоративних додатків.

Розглянемо функціональні новинки в кожній категорії. В цілому версія відрізняється від попередньої наступними ключовими функціями [15].

Підтримка Web-хостингу. З появою версії Beta 2 ряд компаній почали забезпечувати хостинг. Net-застосунків і Web-сервісів, що дозволяє розробникам впроваджувати зстосунки і робити їх доступними для мільйонів користувачів у всьому світі [15]. Більш того, Microsoft надає безкоштовну ліцензію на впровадження рішень, створених за допомогою даної версії Visual Studio. NET [15].

Можливість завантаження компонентів сторонніх фірм. Пошук компонентів сторонніх фірм у версії Beta 2 полегшений. Для доступу до груп новин існує вкладка Online Community, а вкладка Downloads забезпечує доступ до додаткових компонентів і прикладів, доступних для завантаження [15].

Наявність засобів для розробки корпоративних додатків. До складу Microsoft Visual Studio. Net Enterprise Architect входять засоби для планування, аналізу, дизайну, компонування і налагодження корпоративних додатків. У новій версії продукту є засоби моделювання додатків на мові UML 1.2, а також для моделювання баз даних [15].

Порівняльна характеристика найпоширеніших середовищ розробки для мови C# наведена у таблиці 2.2. Серед характеристик для порівняння виділімо:

– кроссплатформність – чи є можливість встановити середовище розробки на різних ОС;

– підтримка та оновлення – як швидко оновлюється середовище в контексті нових стандартів мов програмування;

– засоби для роботи із файлами пакету Microsoft Office – наявність чи відсутність вбудованих засобів для роботи із файлами пакету Microsoft Office;

– підтримка репозиторіїв – чи є вбудований механізм роботи із репозиторіями.

За результатами порівняння перевагу було надано саме Microsoft Visual Studio.

Таблиця 2.2 – Порівняння середовищ розробки

	Microsoft Visual Studio	Code::Blocks
Кроссплатформність	+	+
Підтримка та оновлення	+	±
Засоби для роботи із файлами пакету Microsoft Office	+	±
Підтримка репозиторіїв	+	±

2.2 Загальна структура додатку

Для побудови архітектурного каркасу ПЗ буде використано шаблон проєктування Model-view-controller (MVC). Ця схема використання декількох шаблонів проєктування, за допомогою яких модель даних програми, користувацький інтерфейс і взаємодія з користувачем розділені на три окремих компоненти [16].

View – відповідає за збір даних від користувачів в різних формах та обробку їх у відповідності зі специфікацією.

Controller – обробляє дані, отримані від користувача для передачі на наступний рівень.

Model – сукупність правил, принципів, залежностей поведінки об'єктів предметної області. Це реалізація предметної області в інформаційній системі.

На рисунку 2.1 показано, який вигляд має архітектура ПЗ на основі предметної області з використанням шаблону проєктування MVC.

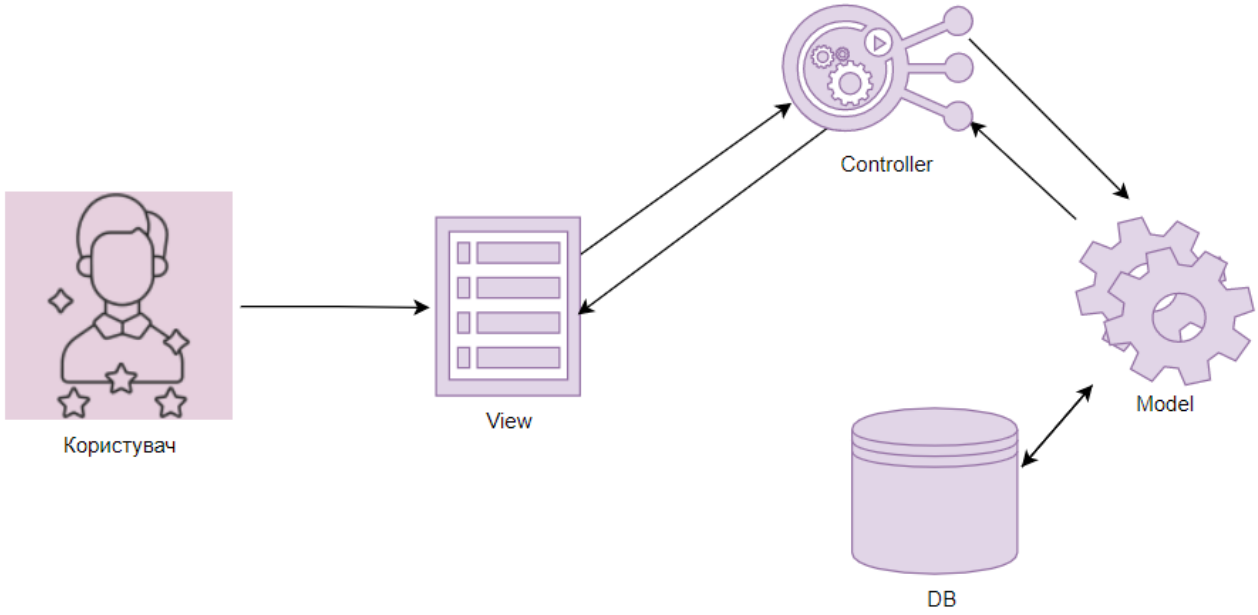


Рисунок 2.1 – Архітектура системи «ЗФЗКД» на основі предметної області

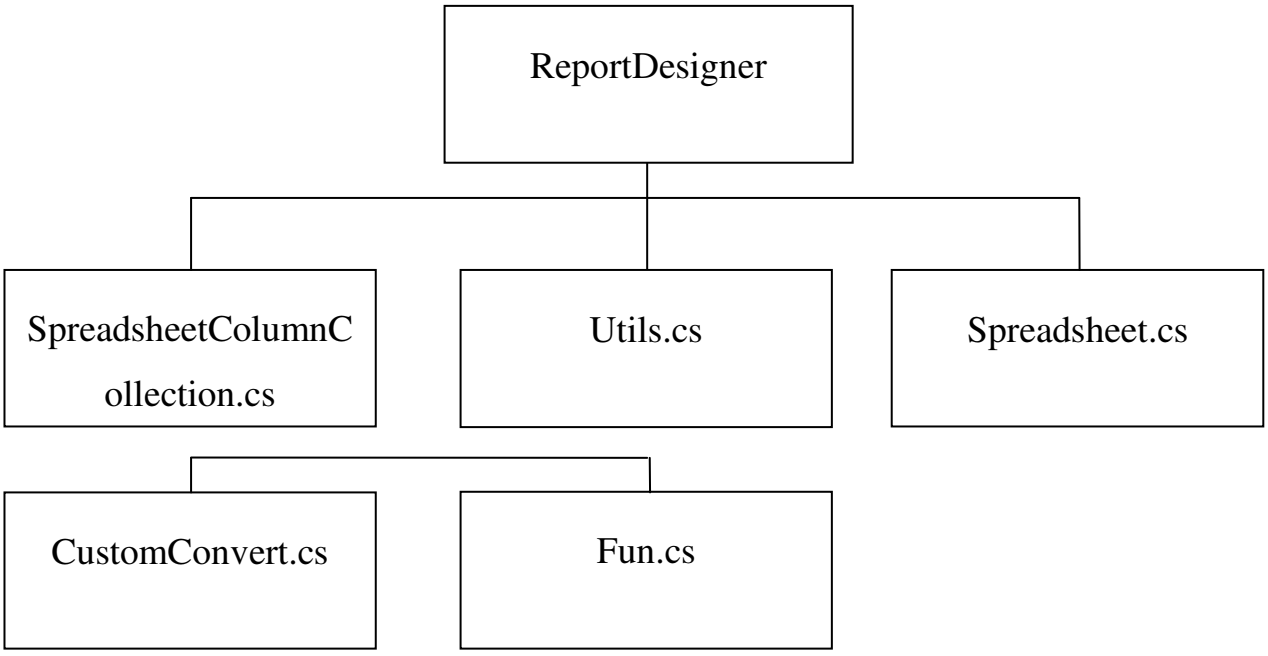


Рисунок 2.2 – Структурна схема програми

2.3 Висновки за розділом 2

Таким чином, архітектура програмного засобу дозволяє вирішувати задачі, що були поставлені перед системою, пов’язані із переглядом, пошуком даних про студентів, а також виведенням інформації на друк. Засоби, що використовуються, забезпечують реалізацію функцій програми.

Аналізуючи обрану структуру програмного засобу, можна зробити висновок, що вона є лаконічною, але, разом з тим, досить працездатною, використовує ресурси ПК раціонально. Також програмний засіб є нескладним у використанні, обране середовище програмування дозволяє у майбутньому розширити та доповнити можливості та інтерфейс системи.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

Програму реалізовано у вигляді окремого застосування, що дозволяє використовувати дані та засоби Microsoft Excel у разі необхідності, при цьому використовуючи спроєктоване рішення для представлення таблиць у вигляді баз даних.

Для функціонування програми визначаються вхідні дані, що задають файл, в якому зберігається таблиця заданої структури, та безпосередньо сама структура даних.

Для доступу до даних, що зберігаються в файлах Microoft Excel, використано засоби Microsoft Office API компонентів .NET Framework. Для реалізації відповідних функцій використано бібліотеку Microsoft.Office.Interop.Excel.

3.1 Проєктування системи на основі предметної області

На попередньому етапі було сформовано архітектуру системи на основі шаблону розробки MVC. Тепер розглянемо основні функціональні вимоги до системи у вигляді діаграми прецедентів, що продемонстрована на рисунку 3.1 [17]-[18].

3.2 Проєктування на основі логіки взаємодії

Діаграми станів зображають всі можливі стани, в яких може знаходитись конкретній об’єкт, а також зміни стану об’єкту, які відбуваються в результаті впливу деяких подій на цей об’єкт [17]-[18].

На рисунку 3.2 зображено діаграму станів користувача у системі «ЗФЗКД». На ній показано переходи між станами користувача.

Рисунок 3.1 – Діаграма прецедентів

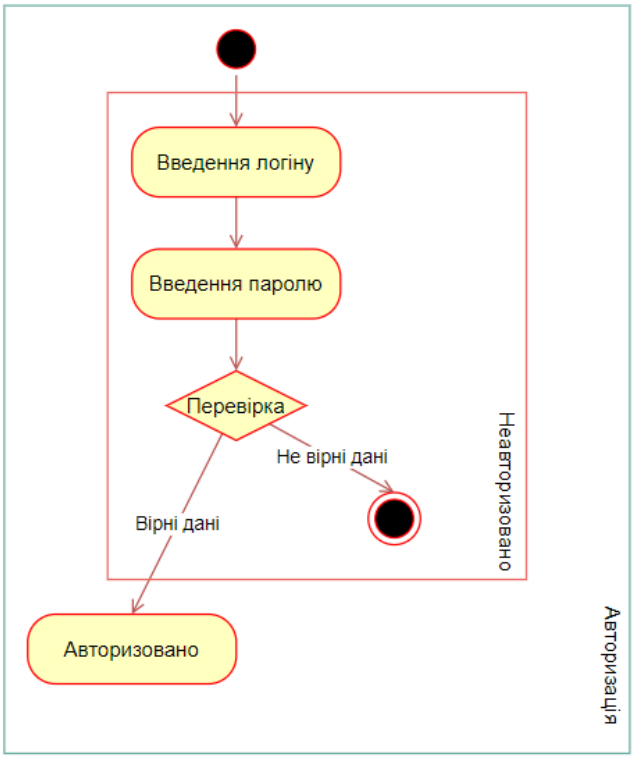


Рисунок 3.2 – Діаграма станів

Діаграма діяльності показує переходи між видами діяльності [17]-[18]. Модель видів діяльності може подавати в графічній формі потік подій для прецеденту.

Діаграма діяльності переходу користувачів до системи показана на рисунку 3.3.

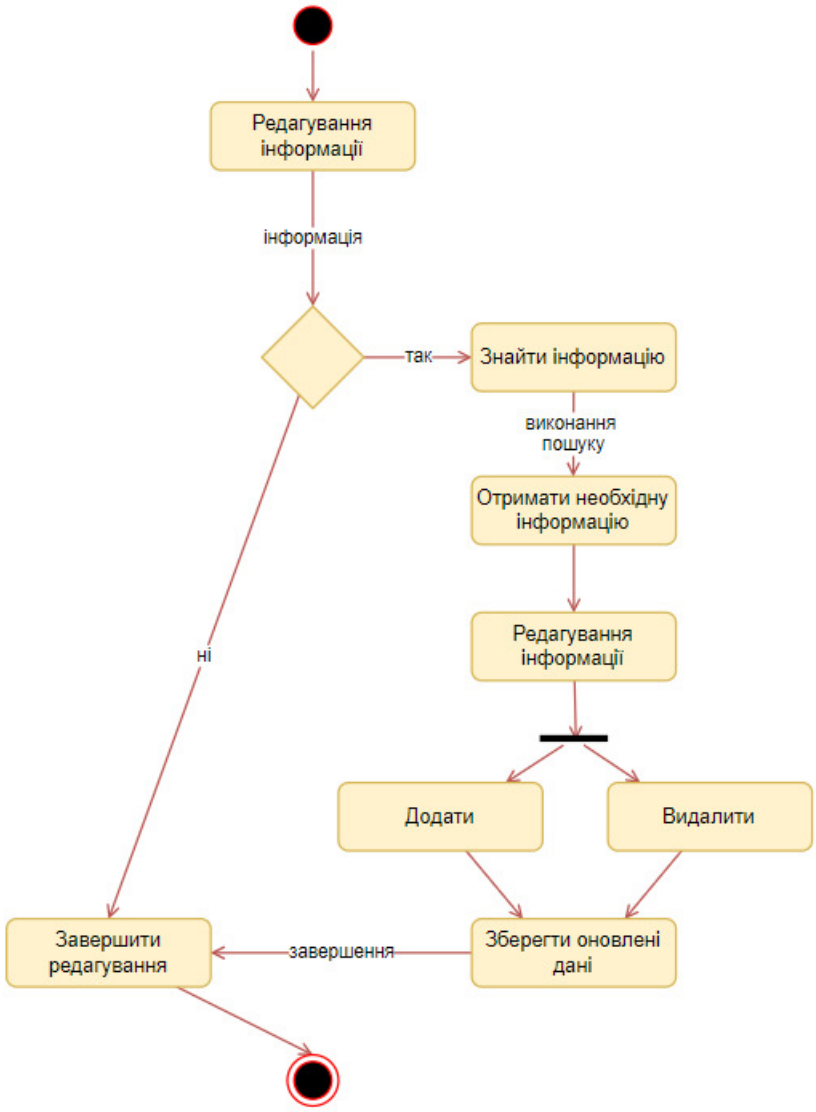


Рисунок 3.3 – Діаграма діяльності: редагування інформації

Діаграма кооперацій призначена для специфікації структурних аспектів взаємодії [17]-[18]. Головна особливість діаграми кооперації полягає в можливості графічно уявити не тільки послідовність взаємодії, але і всі структурні відносини між об’єктами, що беруть участь в цій взаємодії.

На рисунку 3.4 зображено діаграму кооперацій, на якій показано взаємодію користувача та системи, коли виконуються операції: редагування, оновлення, пошук інформації, формування звіту, реєстрації користувача. Всі інші операції у системі здійснюються аналогічним чином.

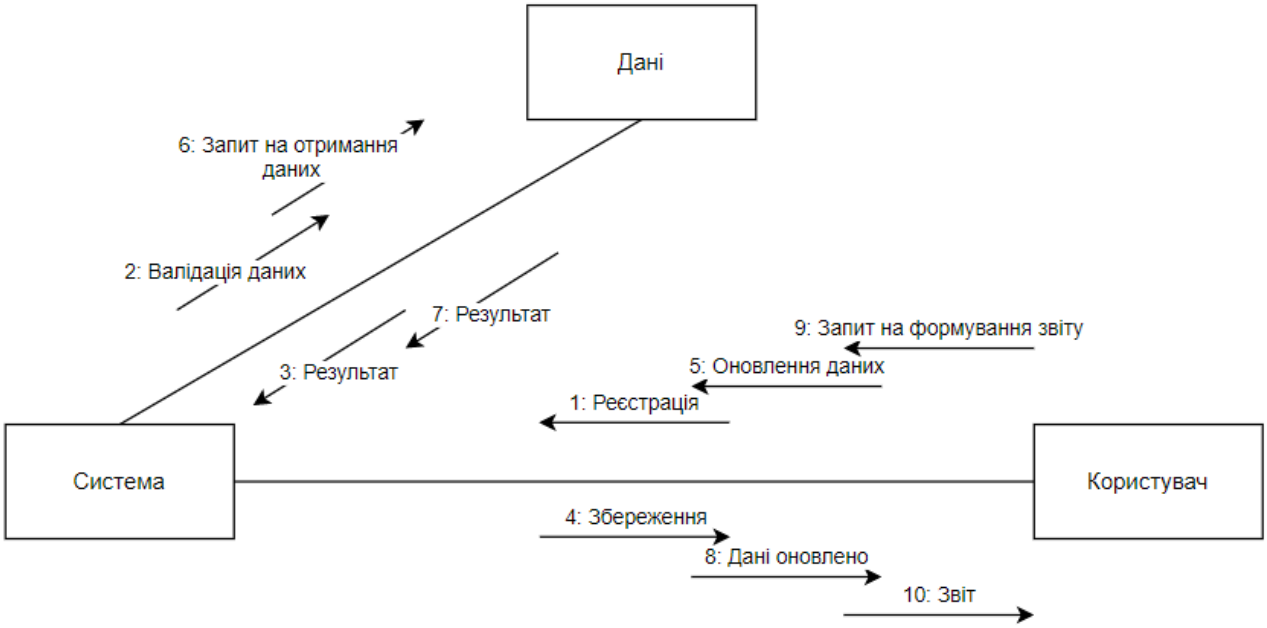


Рисунок 3.4 – Діаграма кооперації «Взаємодія користувача з системою»

3.3 Опис модулів розроблюваної системи

Програма ЗФЗКД включає наступні модулі:

– модуль SpreadsheetColumnCollection, що надає функції роботи з представленням окремих колонок;

– модуль Utils, що виконує доступ до даних у форматі XML;

– модуль Spreadsheet, що надає функції роботи з представленням окремих аркушів;

– модуль CustomConvert;

– модуль Fun, що надає функції роботи безпосередньо з застосуванням Microsoft Excel.

На рисунку 3.5 зображені модулі розроблюваної системи.

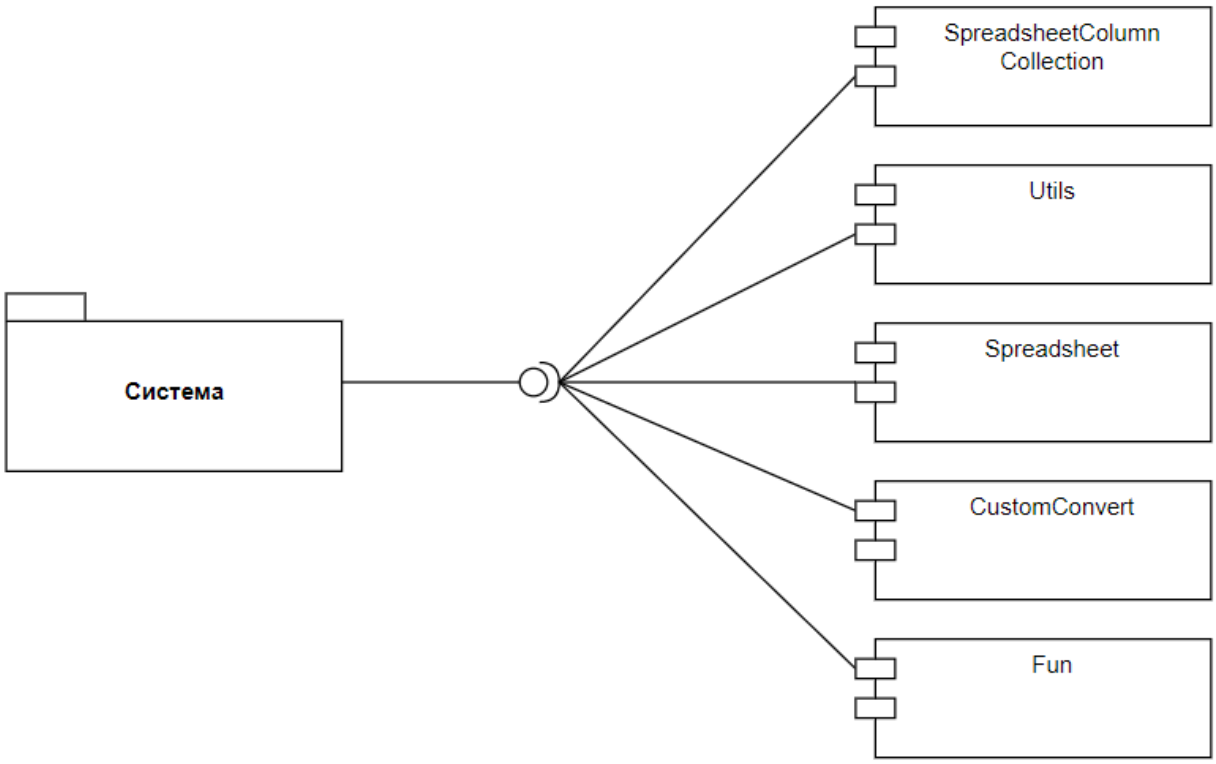


Рисунок 3.5 – Модулі розроблюваної системи

3.4 Експорт даних до звіту, збереження та друк

Після того, як було знайдено потрібні дані, проведено ранжування обо фільтрацію вони використовуються для формування відповідних документів. Для цього було створено три шаблони у програмі Microsoft Word, до яких було додано відповідні закладки, призначені для додавання інформації з програми.

Ініціалізація додатку Word у програмі реалізовано наступним чином:

Microsoft.Office.Interop.Word.Application wordapp = new Microsoft.Office.Interop.Word.Application();// ініціалізація додатку

wordapp.Visible = true;

Microsoft.Office.Interop.Word.Document doc = wordapp.Documents.Add("doc", Type.Missing, Type.Missing, Type.Missing);// створення нового документа із використанням шаблону із назвою Doc.

Для того, щоб відповідні значення зі списку було передано до створеного документа, використовується функція, що дозволяє заповнити створені закладками відповідними даними з діапазону комірок. Наприклад, для створення архівної довідки на підставі даних з таблиці використовується наступна частина програми:

```
doc.Bookmarks["NS_empl"].Range.Text = txtRes.Text;
string buv;
if(ObjWorkSheet.Cells[i, "M"] = 1) {buv = "встановлено";} else {buv = "не встановлено";};
doc.Bookmarks["BUV"].Range.Text = buv;
doc.Bookmarks["D1"].Range.Text = buv;
doc.Bookmarks["N1"].Range.Text = ObjWorkSheet.Cells[i, "H"];
doc.Bookmarks["PR"].Range.Text = comboBox2.Text;
doc.Bookmarks["D3"].Range.Text = ObjWorkSheet.Cells[i, "J"];
doc.Bookmarks["N2"].Range.Text = ObjWorkSheet.Cells[i, "K"];
doc.Bookmarks["G1"].Range.Text = ObjWorkSheet.Cells[i, "B"];
doc.Bookmarks["G2"].Range.Text = ObjWorkSheet.Cells[i, "C"];
doc.SaveAs2("dovid_shab_1","doc");
```

3.5 Висновки за розділом 3

Таким чином, даний розділ розглядає питання функціонування системи, що полягає у налаштуванні тристоронньої взаємодії Таблиця Excel – Інтерфейс користувача – документ Word. На підставі проведеного аналізу можливих програмних засобів реалізації такої взаємодії було обрано оптимальні засоби, які дозволили виконати поставлені задачі. Таким чином, було використано об’єктні моделі роботи із додатками Micorosft Office, а саме Microsoft.Office.Interop.Word, Microsoft.Office.Interop.Excel.

Запропоновано схему функціонування системи в цілому. Ця схема відповідає архітектурі взаємодії користувача із програмою за допомогою

інтерфейсу, а також дозволяє виконати усі функції, які покладені на систему, найголовніші з них – пошук студента, створення відповідних документів на підставі знайдених даних.

Крім того, розглянуто основні механізми функціонування програмного засобу.

Запропоновані засоби функціонування як системи в цілому, так і її частин, дозволяють забезпечити логічну та раціональну роботу програмного засобу для аналізу та обробки даних про студентів вищого навчального закладу.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Опис застосування

Програму реалізовано у вигляді окремого застосування, що дозволяє використовувати дані та засоби Microsoft Excel у разі необхідності, при цьому використовуючи спроєктоване рішення для представлення таблиць у вигляді баз даних.

Для функціонування програми визначаються вхідні дані, що задають файл, в якому зберігається таблиця заданої структури, та безпосередньо сама структура даних.

Для доступу до даних, що зберігаються в файлах Microoft Excel, використано засоби Microsoft Office API компонентів .NET Framework. Для реалізації відповідних функцій використано бібліотеку Microsoft.Office.Interop.Excel.

Використані класи:

– Excel.Application – об’єкт найвищого рівня об’єктної моделі Microsoft Excel;

– Excel.Worbook представляє окрему книгу в межах застосування Microsoft Excel;

– Excel.Worksheet є членом колекції аркушів кожного окремого об’єкту книги.

4.2 Умови виконання програми

Для нормального функціонування ПЗ необхідно виконання технічних умов.

– Вимоги до клієнтської частини:

– IBM-сумісний ПК під управлінням ОС з лінійки Windows не старше 7 версії;

- вільне місце на накопичувачі – від 1 Гб;
- оперативна пам’ять стандарту DDR3 об’ємом від 4 Гб;
- мережева карта.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для завантаження програми ReportDesigner треба вставити програмний носій зі створеним продуктом у відповідний пристрій зчитування і відкритити з нього файл ReportDesigner.exe будь-яким стандартним способом.

4.3.2 Вхідні й вихідні дані

Вхідними даними є файли Microsoft Excel та структура даних, які в даних файлах зберігаються.

Вихідними даними є результати пошуку за фільтрами, представлені у вигляді звітів.

4.4 Методика та результати тестування

4.4.1 Сценарій тестування

Для проведення експериментів використовувались різні вхідні дані, тобто книги Microsoft Excel, що мали різну структуру. Дослідження проводилось за всіма доступними критеріями пошуку для кожного варіанту вхідних даних. Результати виконання операцій було представлено у всіх можливих відображеннях.

4.4.2 Хід виконання тестування

Результати експериментів продемонстрували, що програма дозволяє зручно працювати з даними, представленими у вигляді таблиць Microsoft Excel.

Робота з програмою розпочинається з запуску програми на виконання за допомогою файлу ReportDesigner.exe. Після цього на екрані з’явиться форма, представлена на рисунку 4.1.

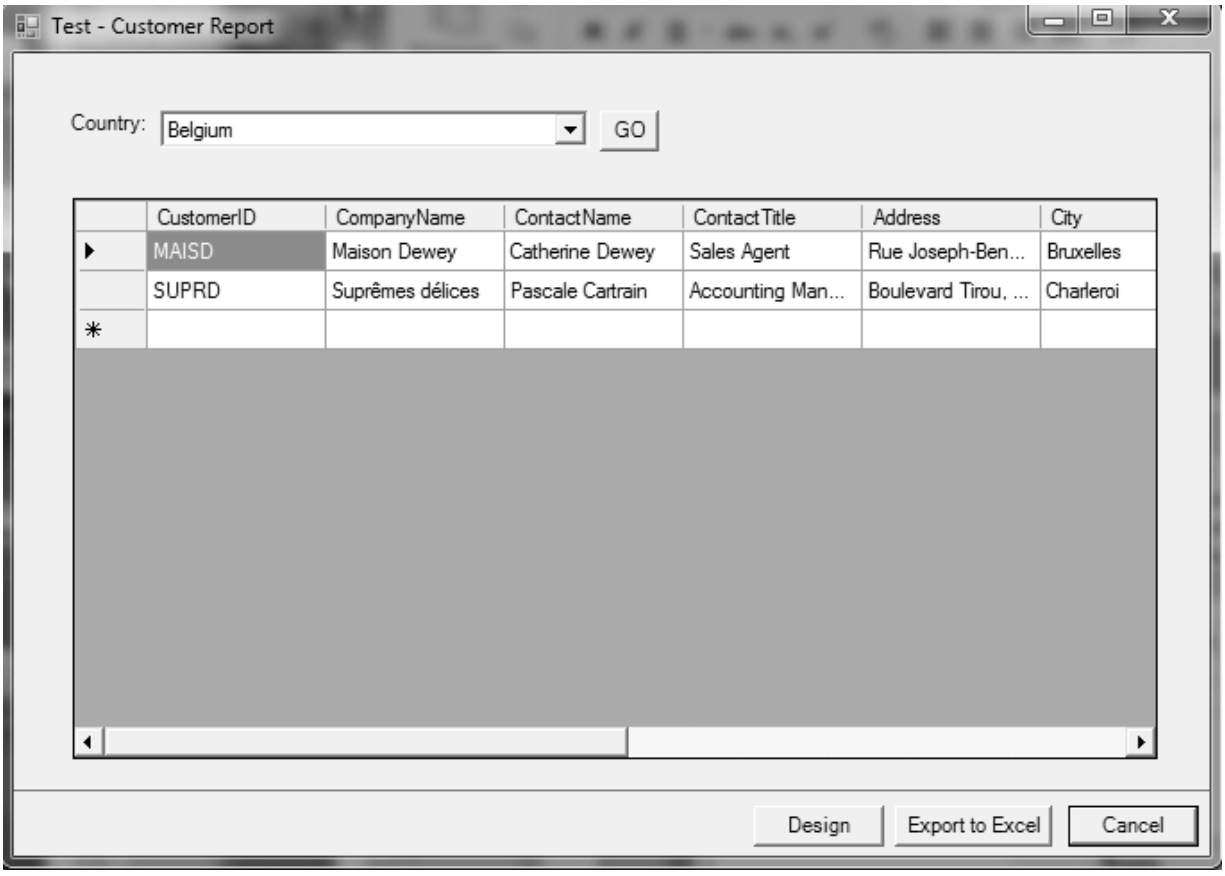


Рисунок 4.1 – Робота з програмою

На рис. 4.1 приведена форма, яка відображає дані з таблиці Microsoft Excel. Обираючи параметр для кожного звіту, можна виконати пошук даних.

Кнопка Export to Excel дозволяє представити результати пошуку безпосередньо у вигляді нового файлу застосування Microsoft Excel.

Кнопка Design дозволяє визначити структуру розглядаємого файлу, додаючи та вилучаючи колонки (рисунок 4.2).

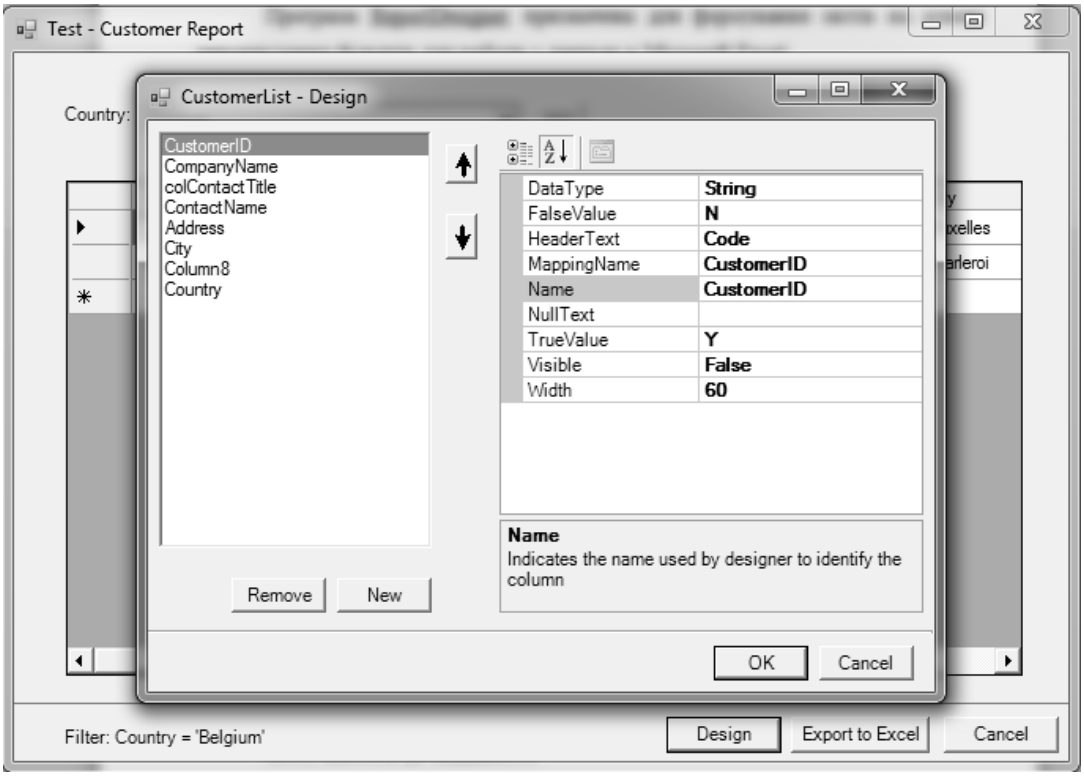


Рисунок 4.2 – Налаштування структури даних

Якщо отримані результати пошуку експортувати в Microsoft Excel, то на екрані з’явиться книга з відповідними даними (рисунок 4.3).

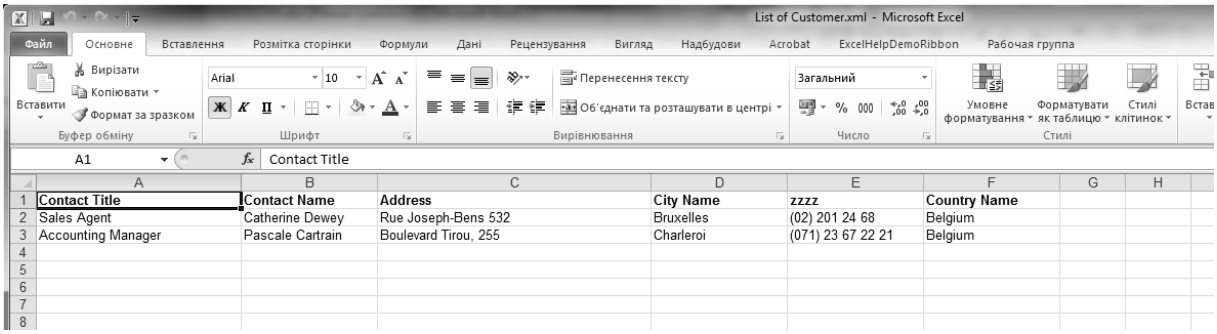


Рисунок 4.3 – Звіт

Для прикладу наведемо результати роботи з іншими таблицями, що характеризують інформацію за співробітниками.

На рисунку 4.4 приведено варіанти визначення фільтрів для пошуку даних.

На рисунку 4.5 продемонстровано, як відбувалось визначення структури даних за співробітниками.

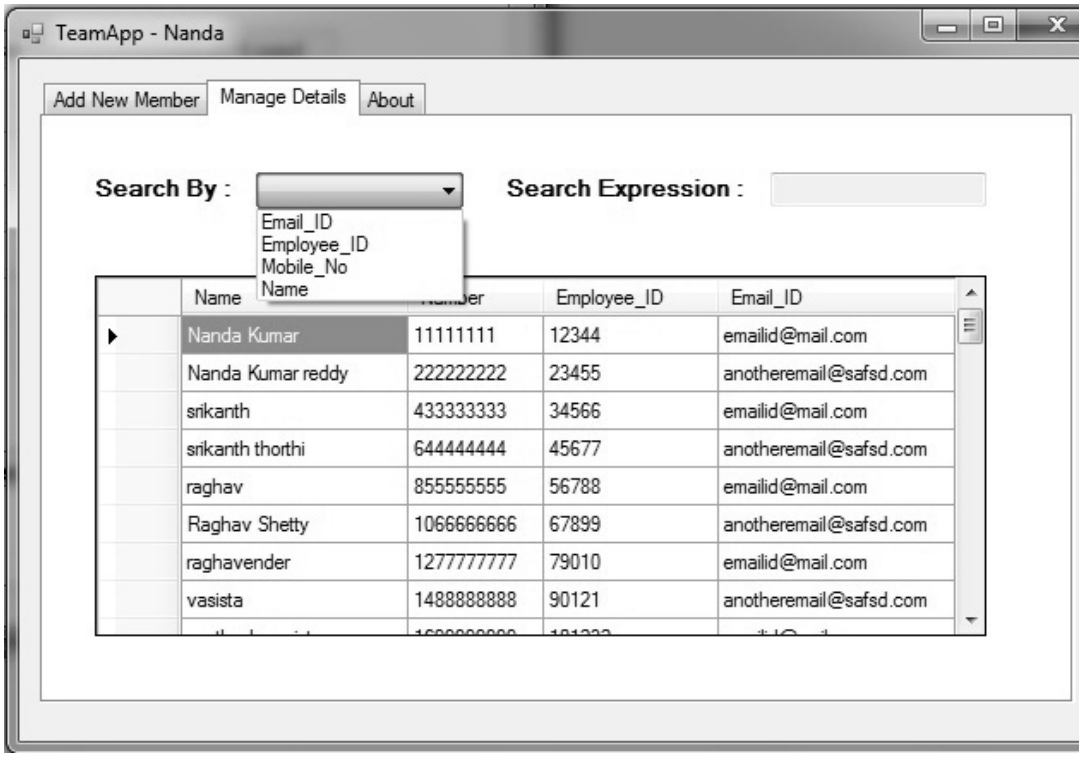


Рисунок 4.4 – Вибір інформації з таблиці

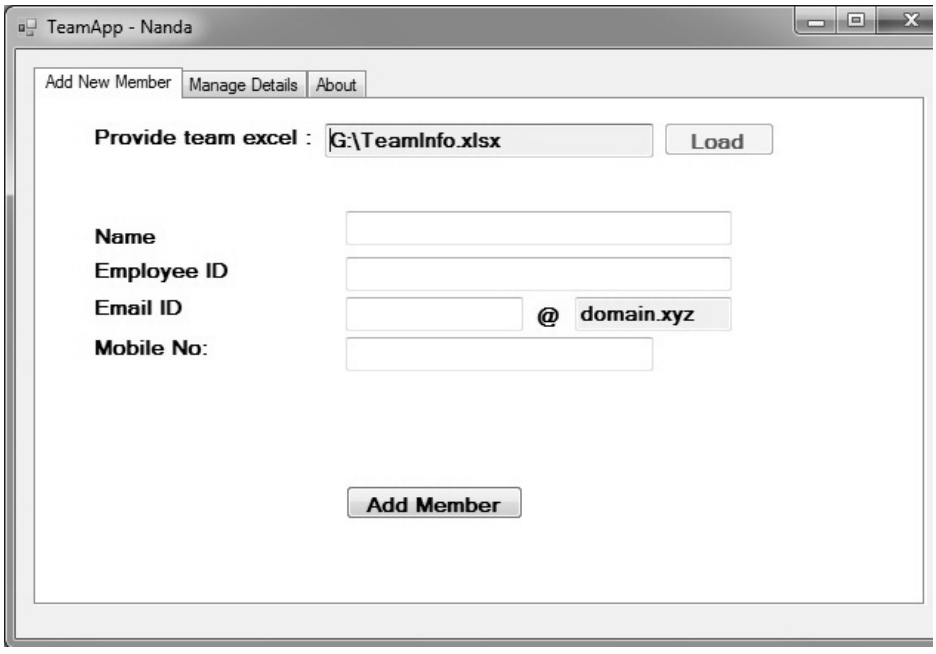


Рисунок 4.5 – Визначення структури даних

ВИСНОВКИ

В ході виконання даної дипломної кваліфікаційної роботи бакалавра було проведено огляд та аналіз програмних засобів для часткової автоматизації процесів електронного документообігу. Відібрані матеріали були систематизовані, проаналізовані, узагальнені та надані у пояснювальній записці.

Для цього було виконано наступні завдання:

- розглянуто проблемну область;
- проаналізовано сучасні найбільш популярні аналоги систем автоматизації електронного документообігу;
- виконано проєктування застосунку;
- розроблено застосунок;
- розроблено сценарій та проведено тестування.

Створено технічне завдання, в якому вказано мету розробки та її призначення, а також основні вимоги до програмного засобу, порядок її функціонування.

Розроблений програмний продукт має структуру та виконує функції, необхідні для виконання формування звітності за колекціями документів.

Для програмної реалізації ПЗ було проаналізовано сучасні засоби для проєктування та розробки застосунків під управлінням ОС Windows на основі виявлених особливостей та архітектури застосунку, що розробляється в рамках даної випускної роботи. Було виявлені необхідні засоби. Отже, для розробки практичної частини було обрано інструментарій: середовище – Visual Studio разом з засобами платформи .NET, мова програмування – C#.

В результаті розробки випускної роботи, було розроблено ПЗ з наступними можливостями:

- завантажувати колекції файлів Microsoft Excel;
- читати та записувати дані в файли Microsoft Excel;

– виконувати пошук вмісту, використовуючи фільтри та формувати на їх основі звіти.

В результаті виконання завдання було отримано програмний продукт, що відповідає всім вимогам технічного завдання і дозволяє виконувати пошук даних в таблицях за допомогою фільтрів з формуванням звітів як засобами розробленого застосування, так і засобами Microsoft Excel.

Отримані результати експериментів дозволяють відзначити ефективність виконання поставлених завдань.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Юхименко Ю. П. Проблеми інформатизації документообігу у вищих навчальних закладах України [Текст] / Ю. П. Юхименко // Бібліотекознавство. Документознавство. Інформологія. – 2008. – № 3. – С. 50-53.

2. Мартинюк Г. Ф. Вивчення електронного документообігу за дистанційною формою навчання з використанням інформаційно-комунікаційних технологій [Текст] / Г. Ф. Мартинюк. // Ефективна економіка. – 2017. – № 2.

3. Батурін Є. Л. Інформаційна технологія підсистеми ідентифікації на основі електронних ключів в системах електронного документообігу [Текст] / Є. Л. Батурін, В. Ю. Воловщиков, В. Ф. Шапо // Вісник Національного технічного університету "ХПІ". Серія : Системний аналіз, управління та інформаційні технології. – 2020. – № 1. – С. 89-96.

4. Ситник І. П. Системи електронного документообігу в електронному бізнесі [Текст] / І. П. Ситник, А. І. Мельниченко // Науковий вісник Ужгородського національного університету. Серія : Міжнародні економічні відносини та світове господарство. – 2015. – Вип. 4. – С. 174-178.

5. Назарова І. Я. Можливості та функції електронного документообігу [Текст] / І. Я. Назарова // Економічний простір. – 2020. – № 159. – С. 166-70.

6. Горупа І. В. Особливості впровадження електронного документообігу підприємства. Огляд [Текст] / І. В. Горупа // Міжнародний науковий журнал "Інтернаука" . – 2018. – № 7(2). – С. 17-20.

7. Харина Ю. А. Автоматизована система документообігу та проблеми її впровадження [Текст] / Ю. А. Харина // Сучасна спеціальна техніка. – 2017. – № 3. – С. 50–55.

8. Сервіс «Docsvision» / [Электрон. ресурс] – Режим доступа: https://docsvision.com/

9. Система «ВЧАСНО» / [Електрон. ресурс] – Режим доступу: https://vchasno.ua/

10. Сервіс «Document.Online» / [Електрон. ресурс] – Режим доступу: https://document.online/

11. Матвієнко О. Вимоги до функціональних можливостей програмних засобів (для бібліотечних процесів) [Текст] / О. Матвієнко // Бібліотечний вісник. – 1996. – № 4. – С. 8-9.

12. Kruglik V. Information system of software distribution / V. Kruglik // Інформ. технології в освіті : зб. наук. пр. – 2011. – Вип. 9. – С. 169-174.

13. C# documentation / [Електрон. ресурс] – Режим доступу: https://docs.microsoft.com/en–us/dotnet/csharp/

14. Мова C# і платформа. Net Core / [Электрон. ресурс] – Режим доступа: https://metanit.com/sharp/tutorial/1.1.php

15. Visual Studio / [Электрон. ресурс] – Режим доступа: https://visualstudio.microsoft.com/ru/

16. Пурський О. І. Метод побудови мережі вітрин інтернет-магазинів на основі архітектури MVC [Текст] / О. І. Пурський, Д. П. Мазоха // Бізнес Інформ. – 2017. – № 10. – С. 319-324.

17. Императивное программирование и объектно–ориентированное моделирование: Java, UML, OCL [Текст] : учеб. пособие для студентов высш. учеб. заведений / А. Ф. Верлань [и др.]. – Одесса : Экология, 2013. – 430 с.

18. Постіл С. Д. UML. Уніфікована мова моделювання інформаційних систем [Текст] : навч. посіб. / С. Д. Постіл ; Ун-т держ. фіск. служби України. – Ірпінь : Ун-т держ. фіск. служби України, 2019. – 321 с.

ДОДАТОК А

Технічне завдання

Вступ

Програмний засіб «Застосунок формування звітності за колекціями документів – ЗФЗКД» або ж «ReportDesigner» призначений для забезпечення ефективної роботи працівників структурних підрозділів підприємств та установ. Програма повинна дозволити працювати з книгами Microsoft Excel як з банками даних

А.1 Підстава для розробки

Підставою для розробки слугує завдання на випускну роботу на тему «Програмна реалізація застосунку для формування звітності за колекціями документів», затверджене наказом Національного університету «Запорізька політехніка» № 103 від 30 березня 2021 р.

А.2 Призначення розробки

Програма ReportDesigner призначена для формування звітів на основі використання фільтрів для роботи з даними в Microsoft Excel.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Програмний продукт повинен виконувати такі основні функції:

- автоматизація роботи працівників структурних підрозділів підприємств та установ;
- пошук інформації в колекції документів – електронних книг Microsoft Excel;
- виведення інформації за окремими фільтрами;
- додавання та редагування інформації;

– формування на основі інформації з програми документа – звіту із використанням шаблона – бланка установи.

– Користувацький інтерфейс повинен забезпечити просте та ефективне спілкування користувача із програмним продуктом.

А.3.2 Вимоги до надійності

Програма повинна коректно відображати робочі форми, своєчасно відповідати на натискання клавіш та рухи миші, не призводити до «зависання» комп'ютерів спеціалістів, точно розраховувати усі статистичні дані та коректно відображати документи, які необхідно підготувати.

А.3.3 Умови експлуатації

Програма може зберігатися та переноситися на будь-яких існуючих носіях інформації.

Експлуатація програми здійснюється згідно експлуатаційної документації на розроблену програму, яка відповідає стандартам і містить інформацію, необхідну для її освоєння та експлуатації.

А.3.4 Вимоги до складу та параметрів технічний засобів

Для нормального функціонування ПЗ необхідно виконання технічних умов.

Вимоги до клієнтської частини:

– IBM-сумісний ПК під управлінням ОС з лінійки Windows не старше 7 версії;

– вільне місце на накопичувачі – від 1 Гб;

– оперативна пам'ять стандарту DDR3 об'ємом від 4 Гб;

– мережева карта.

А.3.5 Вимоги до маркування і пакування

Програмний продукт повинен поставлятися на носії інформації з обов'язковим назвою програми на упаковці – «Застосунок формування звітності за колекціями документів – ЗФЗКД».

А.3.6 Вимоги до транспортування й зберігання

Вимоги до транспортування й зберігання аналогічні вимогам, пропонованим до носія, на якому записана програма.

А.3.7 Вимоги до програмної документації

Програмний продукт повинен поставлятися у наступному складі:

- технічне завдання;
- опис програми;
- текст програми.

А.4 Стадії та етапи розробки

Виділяють такі етапи в розробці програмного забезпечення:

а) збирання та аналіз вимог. Проводиться збір вимог до програмного забезпечення, їх систематизація, документування, аналіз, виявлення протиріч, неповноти, рішення конфліктів в процесі розробки програмного забезпечення;

б) розробка технічного завдання. На основі раніше зібраних вимог формується вихідний документ (технічне завдання) на проєктування програмного забезпечення. Описується всі вимоги до створюваного продукту;

в) створення логотипу та дизайн основних сторінок;

г) налаштування бази даних;

д) розробка структури. Дозволяє оцінити інформативність та компоновку елементів системи на початковому етапі розробки;

е) підключення адміністративного інтерфейсу. Має містити повне управління настройками системи, окремих модулів та контентом сайту;

ж) робота над програмним кодом. Розробка основного функціоналу системи;

з) наповнення та тестування. Професійні тестувальники знаходять «баги» у системі та фіксують їх. Заповнюється базовій «контент», щоб сайт мав привабливіший вигляд для перших користувачів;

и) розміщення та відкритий доступ, технічна підтримка. Соціальна мережа розміщується на сайті унтернет-магазину та стає доступною для всіх користувачів;

к) оформлення документації.

А.5 Порядок контролю та приймання системи

Приймання системи здійснюється відповідно до складу та змісту робіт, наведених у Технічному завданні, і відповідно до умов контракту.

Звіт про проведену роботу направляється Замовнику на паперовому носії в 2-х екземплярах, підписаних належним чином, а також на електронному носії.

Вимоги до оформлення проміжних звітів і підсумкового звіту узгоджуються з Виконавцем при підписанні контракту.

Одночасно з підсумковим звітом про виконання робіт Виконавець надає Замовнику підписаний зі сторони Виконавця Акт здачі-приймання виконаної роботи.

Додаток Б

Опис програми

Б.1 Загальні відомості

Програмний засіб «Застосунок формування звітності за колекціями документів – ЗФЗКД» або ж «ReportDesigner» призначений для забезпечення ефективної роботи працівників структурних підрозділів підприємств та установ. Програма повинна дозволити працювати з книгами Microsoft Excel як з банками даних

Б.2 Функціональне призначення

Програмний продукт повинен виконувати такі основні функції:

– автоматизація роботи працівників структурних підрозділів підприємств та установ;

– пошук інформації в колекції документів – електронних книг Microsoft Excel;

– виведення інформації за окремими фільтрами;

– додавання та редагування інформації;

– формування на основі інформації з програми документа – звіта із використанням шаблона – бланкаустанови.

Б.3 Опис логічної структури

Структуру роботи підсистеми можна представити у вигляді схеми зображеної на рисунках Б.1 та Б.2.

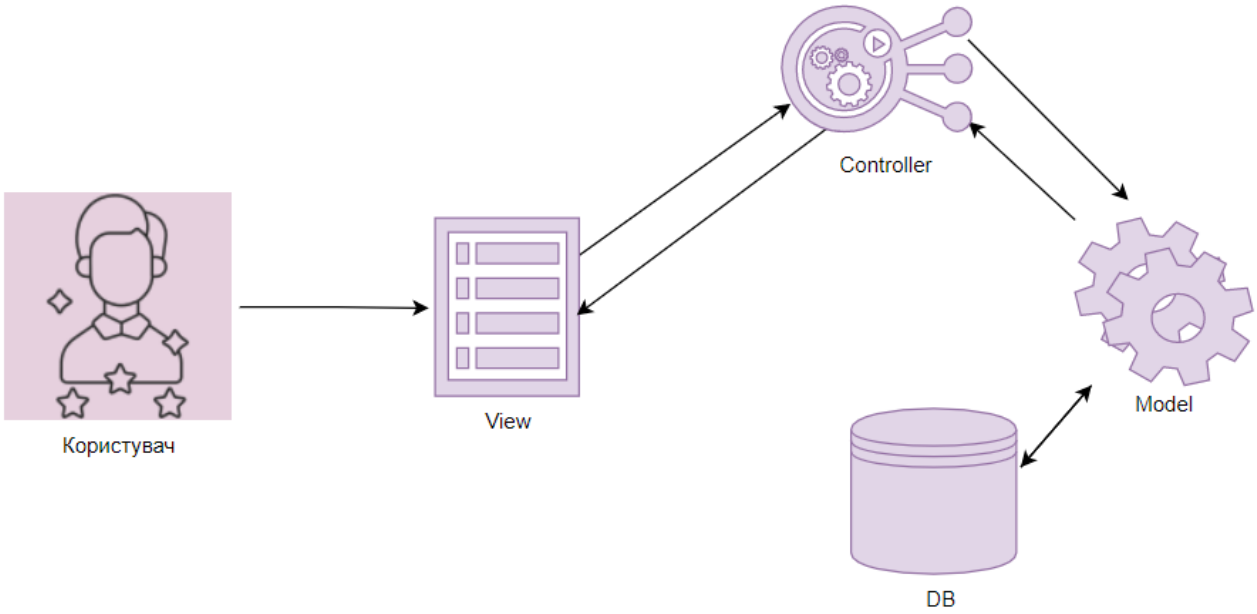


Рисунок Б.1 – Архітектура системи «ЗФЗКД» на основі предметної області

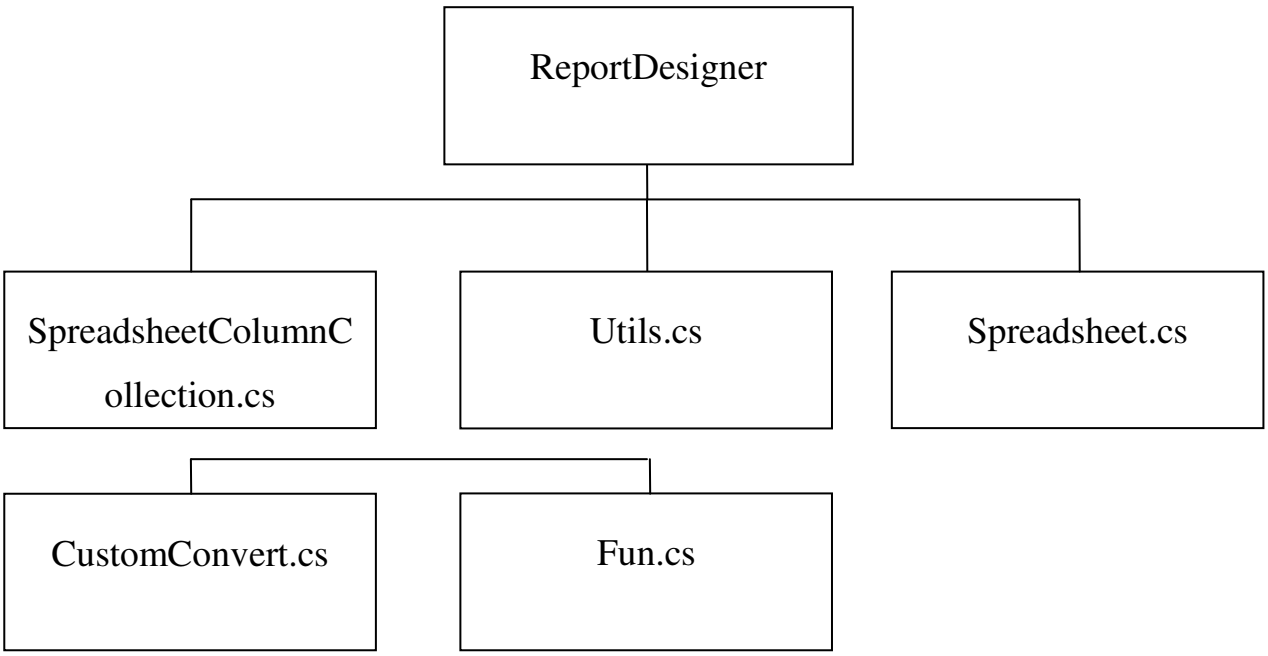


Рисунок Б.2 – Структурна схема програми

Порядок роботи з підсистемою можна описати наступним чином.

Робота з програмою розпочинається з запуску програми на виконання за допомогою файлу ReportDesigner.exe. Після цього на екрані з’явиться форма, представлена на рисунку Б.3.

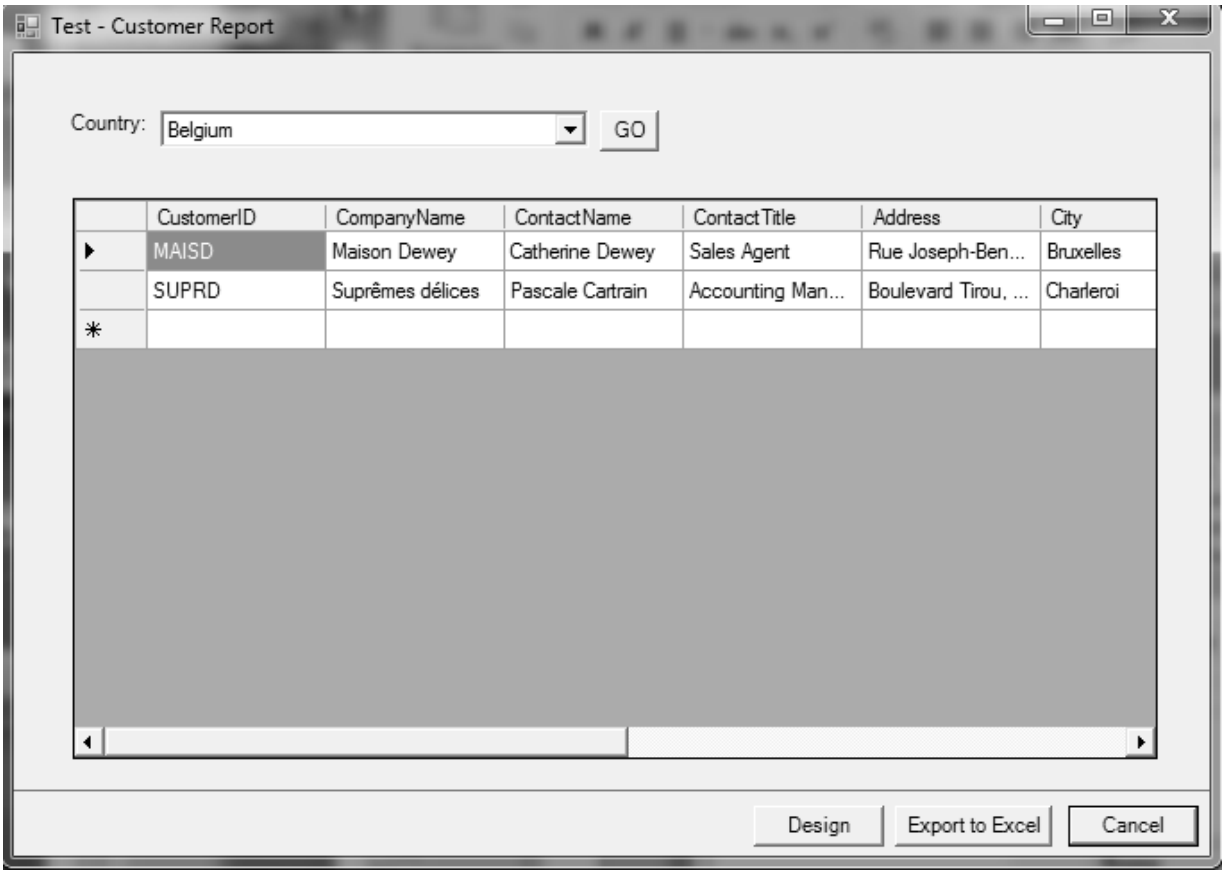


Рисунок Б.3 – Робота з програмою

На рисунку Б.3 приведена форма, яка відображає дані з таблиці Microsoft Excel. Обираючи параметр для кожного звіту, можна виконати пошук даних.

Кнопка Export to Excel дозволяє представити результати пошуку безпосередньо у вигляді нового файлу застосування Microsoft Excel.

Кнопка Design дозволяє визначити структуру розглядаємого файлу, додаючи та вилучаючи колонки (рисунок Б.4).

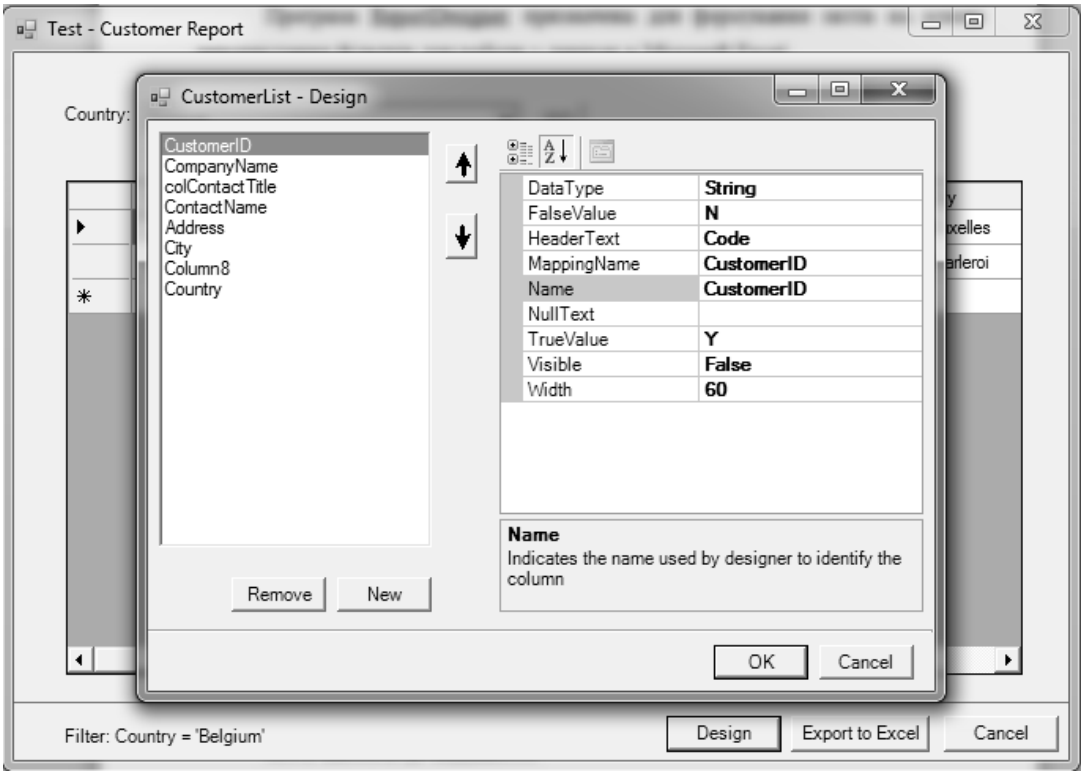


Рисунок Б.4 – Налаштування структури даних

Якщо отримані результати пошуку експортувати в Microsoft Excel, то на екрані з’явиться книга з відповідними даними (рисунок Б.5).

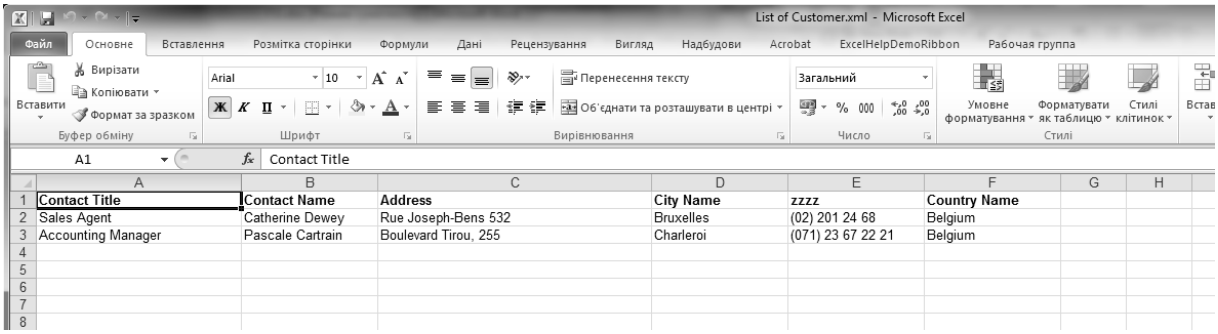


Рисунок Б.5 – Звіт

Б.4 Використані технічні засоби

Для нормального функціонування ПЗ необхідно виконання технічних умов.

Вимоги до клієнтської частини:

– IBM-сумісний ПК під управлінням ОС з лінійки Windows не старше 7 версії;

– вільне місце на накопичувачі – від 1 Гб;

– оперативна пам’ять стандарту DDR3 об’ємом від 4 Гб;

– мережева карта.

Б.5 Виклик і завантаження

Для завантаження програми ReportDesigner треба вставити програмний носій зі створеним продуктом у відповідний пристрій зчитування і відкритити з нього файл ReportDesigner.exe будь-яким стандартним способом.

В.6 Вхідні та вихідні дані

Вхідними даними є файли Microsoft Excel та структура даних, які в даних файлах зберігаються.

Вихідними даними є результати пошуку за фільтрами, представлені у вигляді звітів.

ДОДАТОК В

Текст програми

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
using System.Xml;
using System.Reflection;
using System.Data;
using System.ComponentModel;
using System.Runtime.InteropServices;


namespace Somsoft.ReportDesigner
{
    internal static class Utils
    {
        /// <summary>
        /// Get the excel style information from a external
file.
        /// </summary>
        public   static      string   GetExcelStyles(string
fileName)
        {
            if (!File.Exists(fileName))
                return GetExcelStyle();

            StreamReader sReader;
            string lineText;
            StringBuilder fileText = new StringBuilder();
            sReader = File.OpenText(fileName);
            lineText = sReader.ReadLine();
            fileText.Append(lineText);
            while (lineText != null)
            {
                lineText = sReader.ReadLine();

                fileText.Append(lineText);
                fileText.Append("\n");
            }
            sReader.Close();
            return fileText.ToString();
        }
        public static string GetExcelStyle()
        {
            StringBuilder style = new StringBuilder();
            string newLine = "\n";

            style.Append (" <Styles>" + newLine);
            style.Append   ("     <Style   ss:ID=\"Default\"
ss:Name=\"Normal\">" + newLine);
            style.Append      ("                 <Alignment
ss:Vertical=\"Bottom\"/>" + newLine);
```

```
                style.Append ("   <Borders/>" + newLine);
                style.Append ("   <Font/>" + newLine);
                style.Append ("   <Interior/>" + newLine);
                style.Append ("   <NumberFormat/>" + newLine);
                style.Append ("   <Protection/>" + newLine);
                style.Append ("  </Style>" + newLine);
                style.Append  ("    <Style  ss:ID=\"s22\">"  +
newLine);
                style.Append     ("              <NumberFormat
ss:Format=\"Short Date\"/>" + newLine);
                style.Append ("  </Style>" + newLine);
                style.Append   ("       <Style   ss:ID=\"s27\"
ss:Name=\"Hyperlink\">" + newLine);
                style.Append  ("    <Font  ss:Color=\"#0000FF\"
ss:Underline=\"Single\"/>" + newLine);
                style.Append ("  </Style>" + newLine);
                style.Append  ("    <Style  ss:ID=\"s24\">"  +
newLine);
                style.Append  ("      <Font  x:Family=\"Swiss\"
ss:Bold=\"1\"/>" + newLine);
                style.Append ("  </Style>" + newLine);
                style.Append  ("    <Style  ss:ID=\"s25\">"  +
newLine);
                style.Append  ("      <Font  x:Family=\"Swiss\"
ss:Italic=\"1\"/>" + newLine);
                style.Append ("  </Style>" + newLine);
                style.Append  ("    <Style  ss:ID=\"s26\">"  +
newLine);
                style.Append     ("                <Alignment
ss:Horizontal=\"Center\" ss:Vertical=\"Bottom\"/>" + newLine);
                style.Append ("  </Style>" + newLine);
                style.Append (" </Styles>" + newLine);

                return style.ToString();
            }
            public static string GetExcelWorkSheetOptions()
            {
                StringBuilder sb = new StringBuilder();
                sb.Append("\n<WorksheetOptions
xmlns=\"urn:schemas-microsoft-com:office:excel\">\n<Selected/>\n
</WorksheetOptions>\n");
                return sb.ToString();
            }
            public static void CreateSpreadsheetColumn(XmlNode
items, SpreadsheetColumnCollection columnStyles)
            {
                string MappingName = string.Empty;
                string HeaderText = string.Empty;
                int Width = 10;
                string NullText = "";
                string TrueValue = "Y";
```

```
                    string FalseValue = "N";
                    string Name = string.Empty;
                    bool Visible = true;
                    string DataType = string.Empty;

                    foreach (XmlNode item in items)
                    {
                        switch (item.Name)
                        {
                            case "MappingName":
                                MappingName = item.InnerText;
                                break;
                            case "HeaderText":
                                HeaderText = item.InnerText;
                                break;
                            case "Width":
                                Width                                   =
Convert.ToInt32(item.InnerText);
                                break;
                            case "NullText":
                                NullText = item.InnerText;
                                break;
                            case "TrueValue":
                                TrueValue = item.InnerText;
                                break;
                            case "FalseValue":
                                FalseValue = item.InnerText;
                                break;
                            case "Name":
                                Name = item.InnerText;
                                break;
                            case "Visible":
                                Visible                                 =
Convert.ToBoolean(item.InnerText);
                                break;
                            case "DataType":
                                DataType = item.InnerText;
                                break;
                        }
                    }
                    if (MappingName.Length == 0)
                        return;
                    if (Name.Length == 0)
                        Name = columnStyles.GetNewId();
                    SpreadsheetColumn       col       =       new
SpreadsheetColumn();
                    col.HeaderText = HeaderText;
                    col.MappingName = MappingName;
                    col.Width = Width;
                    col.NullText = NullText;
                    col.TrueValue = TrueValue;
```

```
                col.FalseValue = FalseValue;
                col.Name = Name;
                col.Visible = Visible;
                col.DataType = GetType(DataType);

                columnStyles.Add(col);
            }
            public static SpreadsheetColumnType GetType(string
typeName)
            {
                return
(SpreadsheetColumnType)CustomConvert.ToEnum(typeof(SpreadsheetCo
lumnType), typeName);
            }
            public static  string GetHtmlStyle()
            {
                string style = @"
                  <style>
                        .ColHeaderBorder
                        {
                            border-left:1px solid #999999 ;
                            border-top:1px solid #999999 ;
                            border-bottom:1px solid #999999 ;
                            font-family='verdana,arial';
                            font-size=12px;
                            font-color=black;
                            font-weight=bold;
                        }
                        .ColDataBorder
                        {
                            border-left:1px solid #999999 ;
                            border-bottom:1px solid #999999 ;
                            font-family='verdana,arial';
                            font-size=12px;
                            font-color=black;
                        }
                        .TableBorder
                        {
                            border-right:1px solid #999999 ;
                            font-family='verdana,arial';
                            font-size=12px;
                            font-color=black;
                        }
                  </style>";
                return style;
            }


            /// <summary>
            /// Get the excel header information
            /// </summary>
```

```
            public static string GetExcelHeader()
            {
                // Excel header
                System.Text.StringBuilder     sb     =     new
System.Text.StringBuilder();
                sb.Append("<?xml version=\"1.0\"?>\n");
                sb.Append("<?mso-application
progid=\"Excel.Sheet\"?>\n");
                sb.Append("<Workbook      xmlns=\"urn:schemas-
microsoft-com:office:spreadsheet\" ");
                sb.Append("xmlns:o=\"urn:schemas-microsoft-
com:office:office\" ");
                sb.Append("xmlns:x=\"urn:schemas-microsoft-
com:office:excel\" ");
                sb.Append("xmlns:ss=\"urn:schemas-microsoft-
com:office:spreadsheet\" ");

sb.Append("xmlns:html=\"http://www.w3.org/TR/REC-html40\">\n");
                sb.Append("<DocumentProperties
xmlns=\"urn:schemas-microsoft-com:office:office\">");
                sb.Append("<Author>Somnath               Mondal
(somnath.mondal@microsoft.com)</Author>");
                sb.Append("</DocumentProperties>");
                sb.Append("<ExcelWorkbook  xmlns=\"urn:schemas-
microsoft-com:office:excel\">\n");

sb.Append("<ProtectStructure>False</ProtectStructure>\n");

sb.Append("<ProtectWindows>False</ProtectWindows>\n");
                sb.Append("</ExcelWorkbook>\n");

                return sb.ToString();
            }
            public  static  string  GetColumnTag(object  data,
SpreadsheetColumn  mapColumn,  OutputType  outputType,  string
dateSeperator, bool exportDateAsString)
            {
                string colData = string.Empty;
                string                dataType                =
mapColumn.DataType.ToString();
                string styleId = string.Empty;

                if (!mapColumn.Visible)
                    return string.Empty;

                if (Convert.IsDBNull(data))
                    colData = mapColumn.NullText;
                else
                    colData = data.ToString();
                if (colData.Length > 0 && IsEmptyData(colData))
                    colData = string.Empty;
```

```
                switch (mapColumn.DataType)
                {
                    case SpreadsheetColumnType.Boolean:
                        colData                                 =
CustomConvert.ToBoolean(colData, mapColumn);
                        dataType                                =
SpreadsheetColumnType.String.ToString();
                        break;
                    case SpreadsheetColumnType.Date:
                        colData = CustomConvert.ToDate(colData,
mapColumn,dateSeperator,outputType );
                        if (exportDateAsString)
                        {
                            dataType                            =
SpreadsheetColumnType.String.ToString();
                        }
                        else
                        {
                            styleId = "ss:StyleID=\"s22\"";
                            dataType                            =
SpreadsheetColumnType.DateTime.ToString();
                        }
                        break;
                    case SpreadsheetColumnType.DateTime:
                        colData                                 =
CustomConvert.ToDateTime(colData, mapColumn, dateSeperator);
                        dataType                                =
SpreadsheetColumnType.String.ToString();
                        break;
                    case SpreadsheetColumnType.Number:
                        if (outputType == OutputType.Html)
                            styleId = "align='right'";
                        break;

                }
                if (colData.Length == 0)
                    dataType                                    =
SpreadsheetColumnType.String.ToString();

                string tag = string.Empty;
                switch (outputType)
                {
                    case OutputType.Html:
                        tag = "<td class='ColDataBorder' " +
styleId + ">" + colData + " </td>";
                        break;
                    case OutputType.Excel:
                        tag = "<Cell " + styleId + "><Data
ss:Type=\"" + dataType + "\">" + colData + "</Data></Cell>";
                        break;
                }
```

```
                return tag;

            }
            public static bool IsEmptyData(string data)
            {
                if (data == null)
                    return true;
                if (data.Length == 0)
                    return true;
                string newData = data.Replace("-", "");
                newData = newData.Replace(" ", "");
                if (newData.Length == 0)
                    return true;
               return false;
            }
            public                static                string
GetTableHeaderStyle(SpreadsheetColumnCollection     columnStyles,
OutputType      outputType,      DataView      dataView,      bool
generateAutoColumn)
            {
                System.Text.StringBuilder     sb     =     new
System.Text.StringBuilder();
                string tag;
                int index = 1;
                if (!generateAutoColumn)
                {
                    foreach    (SpreadsheetColumn    col    in
columnStyles)
                    {
                        if (col.Visible)
                        {
                            tag = "<Column ss:Index=\"" + index
+ "\" ss:Width=\"" + col.Width + "\"/>";
                            index++;
                            sb.Append(tag + "\n");
                        }
                    }
                    sb.Append("<tr>");
                    foreach    (SpreadsheetColumn    col    in
columnStyles)
                    {
                        if (col.Visible)
                        {
                            string colValue = string.Empty;
                            switch (outputType)
                            {
                                case OutputType.Html:
                                    colValue       =       "<td
class='ColHeaderBorder' style='width:" + col.Width + "pt' >" +
col.HeaderText + "</td>";
```

```
                                    break;
                                case OutputType.Excel:
                                    colValue      =      "<Cell
ss:StyleID=\"s24\"><Data ss:Type=\"String\">" + col.HeaderText +
"</Data></Cell>";
                                    break;
                            }
                            sb.Append(colValue);
                        }
                    }
                    sb.Append("</tr>");
                }
                else
                {
                    sb.Append("<tr>");
                    foreach       (DataColumn       dc       in
dataView.Table.Columns)
                    {
                        // Cell Tags
                        string colValue = string.Empty;
                        switch (outputType)
                        {
                            case OutputType.Html:
                                colValue         =         "<td
class='ColHeaderBorder'>" + dc.ColumnName + "</td>";
                                break;
                            case OutputType.Excel:
                                colValue        =        "<Cell
ss:StyleID=\"s24\"><Data  ss:Type=\"String\">"  +  dc.ColumnName  +
"</Data></Cell>";
                                break;
                        }
                        sb.Append(colValue);

                    }
                    sb.Append("</tr>");
                }
                return sb.ToString();
            }
            public static string GetSheetName()
            {
                string month = DateTime.Now.Month.ToString();
                string day = DateTime.Now.Day.ToString();
                if (day.Length == 1)
                    day = "0" + day;
                if (month.Length == 1)
                    month = "0" + month;
                string  sheet  =  day  +  "."  +  month  +  "."  +
DateTime.Now.Year   +   "   "   +   DateTime.Now.Hour   +   "h"   +
DateTime.Now.Minute;
                return sheet;
```

```
            }
        }

    }
    using System;
    using System.Collections;
    using System.ComponentModel;
    using System.Runtime.InteropServices;
    using System.Collections.Generic;

    namespace Somsoft.ReportDesigner
    {
        public     class     SpreadsheetColumnCollection     :
CollectionBase, IEnumerable<SpreadsheetColumn>
        {

             public SpreadsheetColumnCollection()
             {
             }

             public SpreadsheetColumn this [int index]
             {
                  get
                  {
                       return (SpreadsheetColumn) List[index];
                  }
                  set
                  {
                       List[index] = value;
                  }
             }
            public int Add(SpreadsheetColumn spreadsheetColumn)
             {
                spreadsheetColumn.Container = this;
                return List.Add(spreadsheetColumn);
             }
            public      void      AddRange(SpreadsheetColumn[]
spreadsheetColumns)
             {
                  int i=0;
                while (i <= spreadsheetColumns.Length)
                  {
                    this.Add(spreadsheetColumns[i]);
                       i++;
                  }
             }

            public   void   AddRange(SpreadsheetColumnCollection
spreadsheetColumns)
             {
                  int i=0;
```

```
                while (i < spreadsheetColumns.Count)
                  {
                    this.Add(spreadsheetColumns[i]);
                       i++;
                  }
               }

              public       bool       Contains(SpreadsheetColumn
spreadsheetColumn)
               {
                  return List.Contains(spreadsheetColumn);
               }

               public void CopyTo(SpreadsheetColumn[] array, int
index)
               {
                    List.CopyTo(array, index);
               }

               public        int        IndexOf(SpreadsheetColumn
spreadsheetColumn)
               {
                    return List.IndexOf(spreadsheetColumn);
               }
              public  void  Insert(int  index,  SpreadsheetColumn
spreadsheetColumn)
               {
                  List.Insert(index, spreadsheetColumn);
               }
              public        void        Remove(SpreadsheetColumn
spreadsheetColumn)
               {
                  List.Remove(spreadsheetColumn);
               }
               public bool IsDuplicateKey(string name)
               {
                    int count=0;
                    foreach(SpreadsheetColumn col in List)
                    {
                         if(col.Name.ToUpper() == name.ToUpper())
                              count++;
                    }
                    return count>0;
               }
              public string GetNewId()
               {
                    int count=List.Count;
                    string keyName = string.Empty ;
                    while(true)
                    {
                         count++;
```

```
                        keyName = "Column" + count;
                        bool found = false;
                        foreach(SpreadsheetColumn col in List)
                        {
                             if(col.Name.ToUpper()            ==
keyName.ToUpper())
                             {
                                  found=true;
                                  break;
                             }
                        }
                        if(!found)
                             break;
                   }
                   return keyName;
               }


              #region IEnumerable<int> Members

              public      new      IEnumerator<SpreadsheetColumn>
GetEnumerator()
              {
                  foreach (SpreadsheetColumn data in InnerList)
                  {
                      yield return data;
                  }
              }
              #endregion
          }
      }

      using System;
      using System.Data;
      using System.Collections;
      using System.Diagnostics;
      using System.ComponentModel;
      using System.Reflection;
      using System.Xml;
      using System.IO;
      using System.Text;
      using System.Security.Permissions;
      using System.IO.IsolatedStorage;

       [assembly:
System.Runtime.InteropServices.ComVisible(false)]

      [assembly: CLSCompliant(true)]

[assembly:IsolatedStorageFilePermission(SecurityAction.RequestMi
nimum, UserQuota=1048576)]
```

```
      [assembly:SecurityPermission(SecurityAction.RequestMinimum,
UnmanagedCode=true)]
      [assembly:FileIOPermission(SecurityAction.RequestMinimum,
Unrestricted=true)]


      namespace Somsoft.ReportDesigner
      {
          #region Enums
          public enum OutputType
          {
               Html,
               Excel
          }
          #endregion
          public class Spreadsheet
          {
      //          public delegate void StatusEventHandler(object
sender, StatusEventArgs e);
               #region Variables
               private SpreadsheetColumnCollection columnStyles =
new SpreadsheetColumnCollection();
               private       string       styleConfigFile       =
"SpreadsheetStyles.config";
               private string dateSeperator = "-";
               private string sheetName = "WorksSeet1";
               private string exportFile = "TestExcel";
               private OutputType outputType = OutputType.Excel;
               private bool generateAutoColumn;
               private bool exportDateAsString = true;
               #endregion
               #region constant
               private const int ERROR_FILE_NOT_FOUND =2;
               private const int ERROR_ACCESS_DENIED = 5;
               #endregion
               public bool ExportDateAsString
               {
                    get
                    {
                         return exportDateAsString;
                    }
                    set
                    {
                         exportDateAsString=value;
                    }
               }
               public bool GenerateAutoColumn
               {
                    get
                    {
                         return generateAutoColumn;
```

```
        }
        set
        {
            generateAutoColumn = value;
        }
    }
    public OutputType OutputType
    {
        get
        {
            return outputType;
        }
    }
    public string DateSeperator
    {
        get
        {
            return dateSeperator;
        }
        set
        {
            dateSeperator = value;
        }
    }
    public string StyleConfigFile
    {
        get
        {
            return styleConfigFile;
        }
        set
        {
            styleConfigFile = value;
        }
    public DataView DataSource
    {
        get
        {
            return dataView;
        }
       set
       {
           dataView = value;
       }
    }
    public SpreadsheetColumnCollection ColumnStyles
    {
        get
        {
            return columnStyles;
        }
```

```
                }
                private int GetTotalColumnWidth()
                {
                     int width = 0;
                     foreach(SpreadsheetColumn        col        in
columnStyles)
                     {
                          if(col.Visible)
                               width+=col.Width;
                     }
                     return width;
                }
                public void GenerateWorkSheet(OutputType type)
                {
                   StatusEventArgs arg;
                   generateAutoColumn = (columnStyles.Count == 0);
                     outputType = type;
                     StringBuilder           sbExcelXml    =    new
StringBuilder ();
                   if (OnProgress != null)
                   {
                       arg = new StatusEventArgs();
                       arg.Message   =   "Start   Generating   "   +
System.DateTime.Now.ToString();

                       OnProgress(this, arg);
                   }

                     //First Write the Excel Header
                     switch(outputType)
                     {
                          case OutputType.Excel:

     sbExcelXml.Append(Utils.GetExcelHeader());
                               break;
                     }
                     sbExcelXml.Append(Utils.GetExcelStyles
(styleConfigFile));
          sbExcelXml.Append(Utils.GetExcelWorkSheetOptions());

                     sbExcelXml.Append("<Worksheet  ss:Name=\""  +
Utils.GetSheetName() + "\">");

                     switch(outputType)
                     {
                          case OutputType.Html:

     sbExcelXml.Append(Utils.GetHtmlStyle());
                               sbExcelXml.Append("<Table
class='TableBorder'  border=0  cellspacing=0  style='width:"  +
GetTotalColumnWidth() + "pt'>" );
```

```
                              break;
                         case OutputType.Excel:
                              sbExcelXml.Append("<Table>");
                              break;
                    }
      sbExcelXml.Append(Utils.GetTableHeaderStyle(columnStyles,ou
tputType,dataView,generateAutoColumn));

                    foreach(DataRowView  dr in dataView)
                    {
                         sbExcelXml.Append("<tr>");

                         if(!generateAutoColumn)
                         {
                              foreach(SpreadsheetColumn   col   in
columnStyles)
                              {
                                   // Cell Tags
                                   colValue                      =
Utils.GetColumnTag(dr[col.MappingName],
col,outputType,dateSeperator,exportDateAsString );
                                   sbExcelXml.Append(colValue);
                              }
                         }
                         else // generate auto column header
                         {
                              foreach(DataColumn       dc       in
dataView.Table.Columns )
                              {
                                   colValue =  string.Empty;

     if(!Convert.IsDBNull(dr[dc.ColumnName]))
                                  colValue                       =
dr[dc.ColumnName].ToString();

                                   switch(outputType)
                                   {
                                        case OutputType.Html:
                                      colValue       =       "<td
class='ColDataBorder'>" + colValue + " </td>";
                                             break;
                                        case OutputType.Excel:
                                      colValue    =    "<td>"    +
colValue + "</td>";
                                             break;
                                   }
                                  sbExcelXml.Append(colValue);
                              }
                         }
                         sbExcelXml.Append("</tr>");
                    }
```

```
                    sbExcelXml.Append("</Table>");
                    sbExcelXml.Append("</Worksheet>");
                    sbExcelXml.Append("</Workbook>\n");
                if (OnProgress != null)
                {
                    arg = new StatusEventArgs();
                    arg.Message    =    "Finished        "    +
System.DateTime.Now.ToString();
                    OnProgress(this, arg);
                }

                  try
                  {
                       string fileName =string.Empty ;
                       fileName= GetOutputFileName();
                       // Delete file before creating new one
                       File.Delete(fileName);

                       StreamWriter       sw       =       new
StreamWriter(fileName,true,System.Text.Encoding.Unicode);
                       switch(outputType)
                       {
                            case OutputType.Html:

     sw.Write(sbExcelXml.ToString());
                                 break;
                            case OutputType.Excel:

     sw.Write(CustomConvert.ToExcelText(sbExcelXml.ToString()));
                                 break;
                       }
                       sw.Close ();
                  }
                  catch(Exception ex)
                  {
                    if (OnError != null)
                    {
                        arg = new StatusEventArgs();
                        arg.Message = ex.ToString();
                        OnError(this, arg);
                    }
                  }

                // Raise event
                if (OnFinish != null)
                {
                    arg = new StatusEventArgs();
                    arg.Message      =      "Data      Exported
Successfully.";
                    OnFinish(this, arg);
                }
```

```
                }

                public void Preview()
                {
                     Process myProcess = new Process();
                   StatusEventArgs arg;
                     try
                     {
                          string fileName = GetOutputFileName();
                          myProcess.StartInfo.FileName = fileName;
                          myProcess.StartInfo.Verb = "Open";
                          myProcess.Start();
                     }
                     catch (Win32Exception e)
                     {
                          if(e.NativeErrorCode                    ==
ERROR_FILE_NOT_FOUND)
                          {
                           if (OnError != null)
                           {
                                arg = new StatusEventArgs();
                                arg.Message = e.Message + ". Check
the path.";
                                OnError(this, arg);
                           }
                          }

                          else     if     (e.NativeErrorCode     ==
ERROR_ACCESS_DENIED)
                          {
                           if (OnError != null)
                           {
                                arg = new StatusEventArgs();
                                arg.Message = e.Message + ". You do
not have permission to print this file.";
                                OnError(this, arg);
                           }
                          }
                     }
                }
                public string SheetName
                {
                     get
                     {
                          return sheetName;
                     }
                     set
                     {
                          sheetName = value;
                     }
```

```
                }
                public string ExportFile
                {
                     get
                     {
                          return GetOutputFileName();
                     }
                     set
                     {
                          exportFile = value;
                     }
                }
        ()
                {
                     string fileName = exportFile;
                     switch(outputType)
                     {
                          case OutputType.Html:
                               fileName+=".htm";
                               break;
                          case OutputType.Excel:
                               fileName+=".xml";
                               break;

                     }
                     return fileName;
                }
                public     void     InitializeReportColumns(string
configFile, string reportId)
                {
                     columnStyles.Clear();


                     if(!File.Exists(configFile))
                          return;

                     XmlDocument xmldoc = new XmlDocument();
                     xmldoc.Load(configFile);
                     XmlNodeList              reports              =
xmldoc.GetElementsByTagName(reportId);

                     if(reports.Count == 0)
                          return;

                     foreach( XmlNode items in reports.Item(0))
                     {

     Utils.CreateSpreadsheetColumn(items,columnStyles );
                     }
                }
```

```
        }
    }
    using System;
    using System.Collections.Generic;
    using System.Text;
    using System.Reflection;

    namespace Somsoft.ReportDesigner
    {
        internal static class CustomConvert
        {
            public static object ToEnum(Type type, string
inString)
            {
                foreach (FieldInfo fi in type.GetFields())
                    if (fi.Name == inString)
                        return fi.GetValue(null);
                throw new Exception(string.Format("Can't
convert {0} to{1}", inString, type.ToString()));
            }
            public static string ToBoolean(object data,
SpreadsheetColumn mapColumn)
            {
                string tag = string.Empty;
                if (Convert.IsDBNull(data))
                {
                    tag = mapColumn.NullText;
                    return tag;
                }

                if (data.ToString().Length == 0)
                {
                    tag = mapColumn.NullText;
                    return tag;
                }
                bool result = false;
                try
                {
                    result = Convert.ToBoolean(data);
                }
                catch
                { // do nothing
                }

                tag = result ? mapColumn.TrueValue :
mapColumn.FalseValue;

                return tag;
            }
```

```
            public static string ToDate(object data,
SpreadsheetColumn mapColumn, string dateSeperator, OutputType
outputType)
            {
                string tag = string.Empty;
                if (Convert.IsDBNull(data))
                {
                    tag = mapColumn.NullText;
                    return tag;
                }

                if (data.ToString().Length == 0)
                {
                    tag = mapColumn.NullText;
                    return tag;
                }
                try
                {
                    DateTime                dt                =
DateTime.Parse(data.ToString());
                    string month = dt.Month.ToString();
                    string day = dt.Day.ToString();
                    if (day.Length == 1)
                        day = "0" + day;
                    if (month.Length == 1)
                        month = "0" + month;

                    switch (outputType)
                    {
                        case OutputType.Html:
                            tag = day + dateSeperator + month +
dateSeperator + dt.Year;
                            break;
                        case OutputType.Excel:
                            tag  =  dt.Year  +  dateSeperator  +
month + dateSeperator + day;
                            break;
                    }
                }
                catch
                { // do nothing
                }

                return tag;
            }
            public static string ToDateTime(object data,
SpreadsheetColumn mapColumn, string dateSeperator)
            {
                string tag = string.Empty;
                if (Convert.IsDBNull(data))
                {
```

```
                    tag = mapColumn.NullText;
                    return tag;
                }

                if (data.ToString().Length == 0)
                {
                    tag = mapColumn.NullText;
                    return tag;
                }
                try
                {
                    string dateTimeString = data.ToString();
                    DateTime                dt                =
DateTime.Parse(dateTimeString);
                    string month = dt.Month.ToString();
                    string day = dt.Day.ToString();
                    if (day.Length == 1)
                        day = "0" + day;
                    if (month.Length == 1)
                        month = "0" + month;

                    tag  =  day  +  dateSeperator  +  month  +
dateSeperator + dt.Year + " " + dt.Hour + "h" + dt.Minute;
                }
                catch
                { // do nothing
                }

                return tag;
            }
            public static string ToExcelText(string htmlText)
            {

                // Just to replace TR with Row
                htmlText   =   htmlText.Replace("<tr>",   "<Row
ss:AutoFitHeight=\"1\" >\n");
                htmlText      =      htmlText.Replace("</tr>",
"</Row>\n");

                //replace the cell tags
                htmlText       =       htmlText.Replace("<td>",
"<Cell><Data ss:Type=\"String\">");
                htmlText      =      htmlText.Replace("</td>",
"</Data></Cell>\n");

                return htmlText;
            }

        }
    }
    using System;
```

```
using System.Collections;
using System.Collections.Generic;
using System.IO;
using Excel = Microsoft.Office.Interop.Excel;
using System.Diagnostics;
using System.Windows.Forms;

namespace ReportDesigner
{
    static class Fun
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new ExcelTool());
        }
    }

    public class eshExcel
    {
        #region DATA MEMBERS
        private Excel.Application excelApplication = null;
        private Excel.Workbook excelWorkbook = null;
        private Excel.Sheets excelSheets = null;
        private Excel.Worksheet excelWorksheet = null;

        // to represent values that are missing/not
applicable while passing parameters
        private static object esh_missing =
System.Reflection.Missing.Value;

        private static object esh_visible = true;
        private static object esh_true = true;
        private static object esh_false = false;

        // to set the application visible or invisible
        private static object esh_app_visible = false;

        // used in the closingof the application
        private object esh_filename = null;

        #endregion

        #region OPEN WORKBOOK VARIABLES
```

```
            private object esh_update_links = 0;
            private object esh_read_only = esh_true;
            private object esh_format = 1;
            private object esh_password = esh_missing;
            private     object     esh_write_res_password     =
esh_missing;
            private  object  esh_ignore_read_only_recommend  =
esh_true;
            private object esh_origin = esh_missing;
            private object esh_delimiter = esh_missing;
            private object esh_editable = esh_false;
            private object esh_notify = esh_false;
            private object esh_converter = esh_missing;
            private object esh_add_to_mru = esh_false;
            private object esh_local = esh_false;
            private object esh_corrupt_load = esh_false;

            #endregion

            #region CLOSE WORKBOOK VARIABLES

            private object esh_save_changes = esh_false;
            private object esh_route_workbook = esh_false;

            #endregion

            #region EXCEL OBJECT CONSTRUCTORS

            /// <summary>
            /// public default constructor for Excel Object
used for reading the excel sheet.
            ///
            /// Eshwar Date : 31 January 2008
            /// </summary>
            public eshExcel()
            {
                this.InitExcel();
            }

            /// <summary>
            ///  Creating the excel object keeping the
application visible or invisible.
            /// </summary>
            /// <param name="visibility"></param>
            public eshExcel(bool visibility)
            {
                esh_app_visible = visibility;
                this.InitExcel();
            }

            #endregion
```

```
            #region destructor

            ~eshExcel()
            {
                this.StopExcel();
                MessageBox.Show("Shutting EXCEL.EXE Process");
            }

            #endregion
            #region INIT EXCEL APPL
            /// <summary>
            /// starting the excel application
            /// </summary>
            private void InitExcel()
            {
                if (excelApplication == null)
                {
                    excelApplication           =           new
Excel.ApplicationClass();
                }
                excelApplication.Visible                      =
(bool)esh_app_visible;
            }

            #endregion

            #region STOP EXCEL APPL

            private void StopExcel()
            {
                if (this.excelApplication != null)
                {
                    Process[]            pProcess            =
System.Diagnostics.Process.GetProcessesByName("Excel");
                    pProcess[0].Kill();
                }
            }

            #endregion


            #region OPEN EXCEL WORKBOOK
            public   void   OpenFile(string   filename,   string
password)
            {
                esh_filename = filename;
                if (password.Length > 0)
                {
                    esh_password = password;
                }
                try
```

```
                {
                    this.excelWorkbook                          =
this.excelApplication.Workbooks.Open(filename,
                                esh_update_links,    esh_read_only,
esh_format, esh_password, esh_write_res_password,
                                esh_ignore_read_only_recommend,
esh_origin, esh_delimiter, esh_editable, esh_notify,
                                esh_converter,      esh_add_to_mru,
esh_local, esh_corrupt_load);
                }
                catch (Exception ee)
                {
                    if   ((ee.Message).Contains("could   not   be
found"))
                    {
                        throw                                   (new
FileNotFoundException(ee.Message));
                    }
                    else
                    {
                        throw   (new   Exception("Unknown   error
while opening the file"));
                    }
                }

            }

            #endregion

            public void CloseFile()
            {
                this.excelWorkbook.Close(esh_save_changes,
esh_filename, esh_route_workbook);
            }

            #region GET EXCEL SHEETS FROM WORKBOOK

            public void GetExcelsheets()
            {
                if (this.excelWorkbook != null)
                {
                    this.excelSheets                            =
this.excelWorkbook.Worksheets;
                }
            }

            #endregion

            #region OPEN REQUIRED EXCEL SHEET
```

```
            public      bool      OpenReqExcelWorksheet(string
worksheetName)
            {
                bool sheet_found = false;

                if (this.excelSheets != null)
                {
                    for     (int     i     =     1;     i     <=
this.excelSheets.Count; ++i)
                    {
                        this.excelWorksheet                    =
(Excel.Worksheet)excelSheets.get_Item((object)i);
                        if     (this.excelWorksheet.Name     ==
worksheetName)
                        {

((Excel._Worksheet)excelWorksheet).Activate();
                            sheet_found = true;
                            return sheet_found;
                        }

                    }
                }
                return sheet_found;
            }

            #endregion

            #region GET RANGE
            public string[][] GetRange(string startRange)
            {
                try
                {
                    Excel.Range       currentRangeCells       =
this.excelWorksheet.get_Range(startRange,
System.Reflection.Missing.Value);
                    string                range                =
currentRangeCells.Cells.Value2 as string;
                    char[] splitter = { ':' };
                    string[] rangeArray = range.Split(splitter,
2);
                    string[][]           stringArray           =
GetRange(rangeArray[0], rangeArray[1]);
                    return stringArray;
                }
                catch (Exception e)
                {
                    if (e.Message.Contains("Exception:"))
                    {
                        throw (e);
                    }
```

```
                    else
                    {
                        throw (new Exception("Exception: Range
Extraction"));
                    }
                }
            }

            public string[][] GetRange(string startRange,
string endRange)
            {
                try
                {
                    Excel.Range currentRangeCells =
this.excelWorksheet.get_Range(startRange, endRange);
                    System.Array dataArray =
(System.Array)currentRangeCells.Cells.Value2;
                    string[][] stringArray =
this.ToStringArray(dataArray);
                    return stringArray;
                }
                catch (Exception e)
                {
                    if (e.Message.Contains("Exception:
Conversion to string array"))
                    {
                        throw (e);
                    }
                    else
                    {
                        throw (new Exception("Exception: Range
Extraction"));
                    }
                }

            }

            #endregion

            #region TO STRING ARRAY
            /// <summary>
            /// Converting the data values retrieved from the
rangecells to string values.
            /// </summary>
            /// <param name="data"></param>
            /// <returns></returns>
            private string[][] ToStringArray(System.Array data)
            {
                try
                {
```

```
                        string[][]        sArray        =        new
string[data.GetUpperBound(0)][];
                        for  (int  i  =  data.GetLowerBound(0);  i  <=
data.GetUpperBound(0); ++i)
                        {
                            sArray[i      -      1]      =      new
string[data.GetUpperBound(1)];
                            for  (int  j  =  data.GetLowerBound(1);  j
<= data.GetUpperBound(1); ++j)
                            {
                                if (data.GetValue(i, j) == null)
                                {
                                    sArray[i - 1][j - 1] = "-----";
                                }
                                else
                                {
                                    sArray[i   -   1][j   -   1]   =
data.GetValue(i, j).ToString();
                                }
                            }
                        }
                        return sArray;
                    }
                    catch
                    {
                        throw (new Exception("Exception: Conversion
to string array"));
                    }
                }
                #endregion

            }
        }
```

ДОДАТОК Д

Слайди презентації

Рисунок Д.1 – Слайд 1

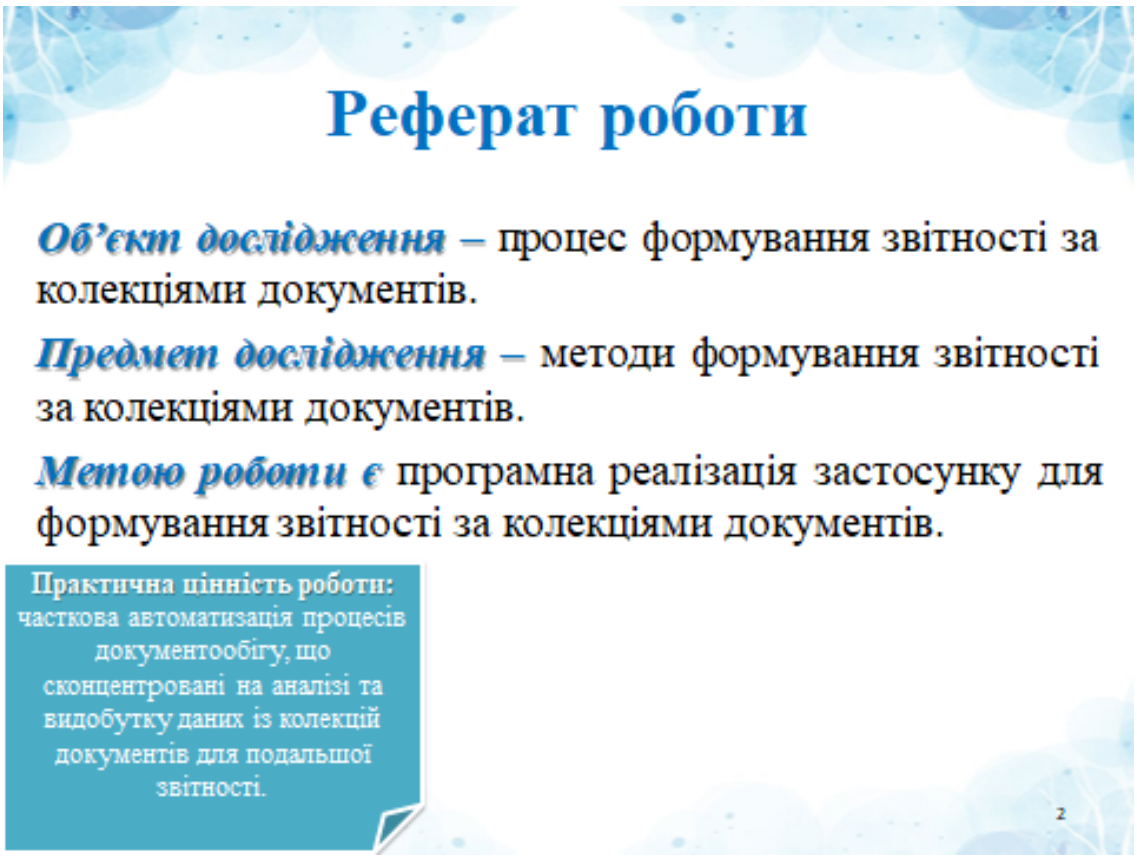


Рисунок Д.2 – Слайд 2

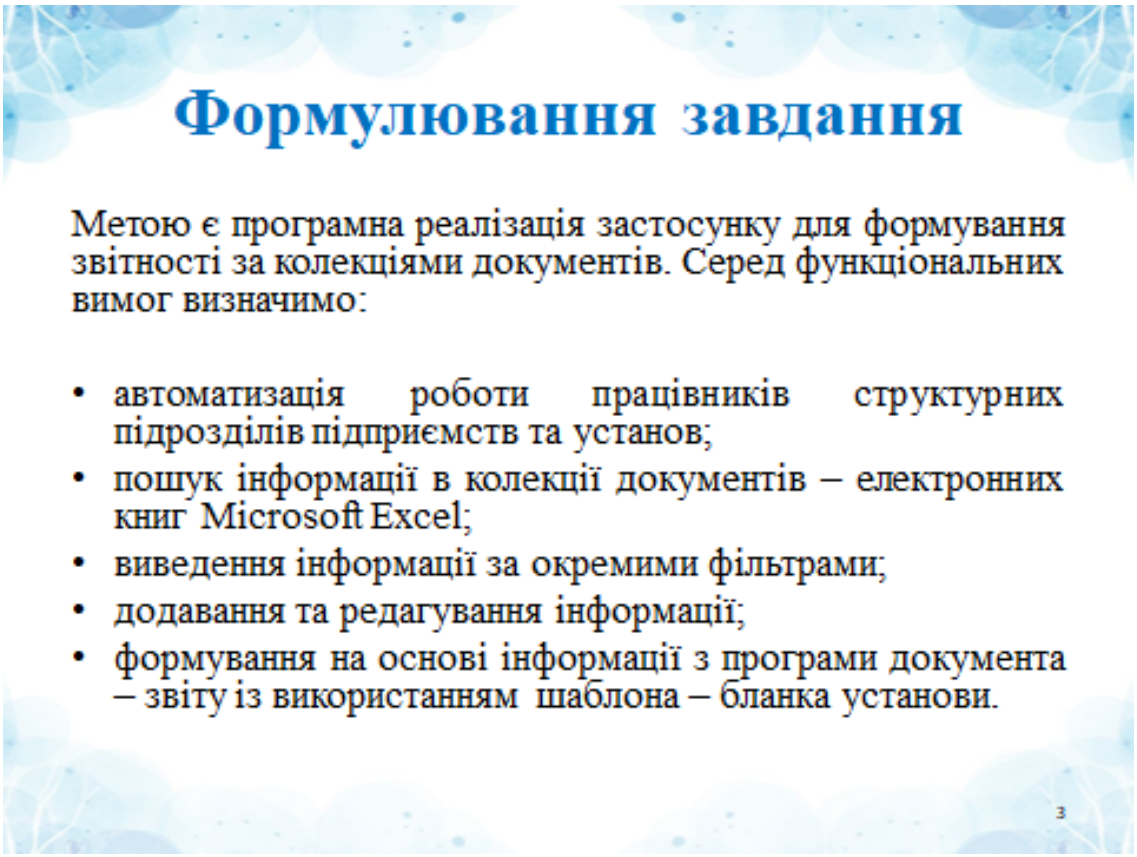


Рисунок Д.3 – Слайд 3

Рисунок Д.4 – Слайд 4

Порівняння розробки із існуючими рішеннями

Критерій порівняння	Docsvision	ВЧАСНО	Document. Online	ЗФЗКД
Вартість використання	Висока	Є безкоштовна версія	Висока	Безкоштовно
Підтримка роботи над колекціями документів	Відсутня	Відсутня	Відсутня	Підтримує
Внутрішнє ранжування документів	Відсутнє	Відсутнє	Відсутнє	Підтримує
Збереження архівної інформації	Підтримує	Тільки в платній версії	Підтримує	Підтримує

5

Рисунок Д.5 – Слайд 5

Вибір технічних засобів розробки

	C#	Python	Java
Наявність літератури та простота вивчення	+ +	+ +	+ +
Стандартні бібліотеки шаблонів	+	+	+
Кроссплатформність	±	+	+ +
Мультипарадигмність	+ +	+ +	±
Сумісність з пакетом Microsoft Office	+ +	±	±
Простота розроблення GUI	+ +	+	+

6

Рисунок Д.6 – Слайд 6

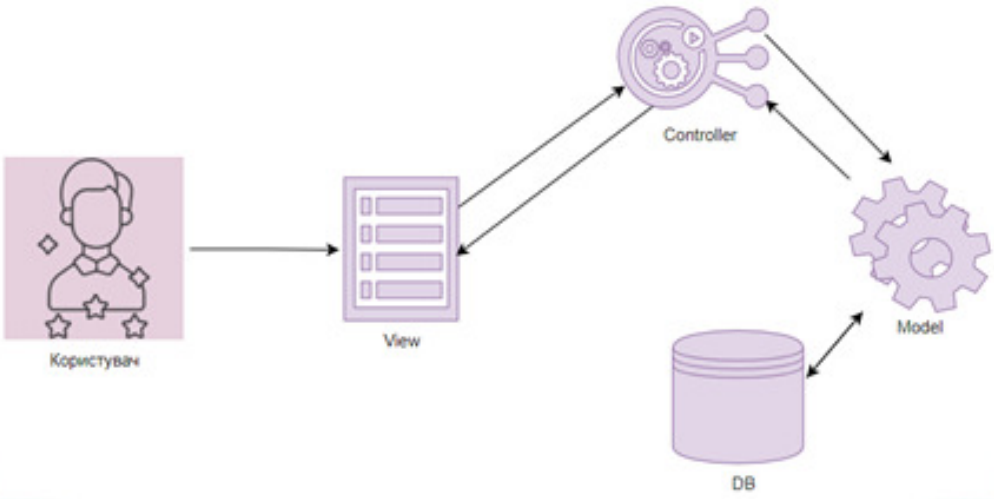
Вибір технічних засобів розробки

	Microsoft Visual Studio	Code::Blocks
Кроссплатформність	+	+
Підтримка та оновлення	+	±
Засоби для роботи із файлами пакету Microsoft Office	+	±
Підтримка репозиторіїв	+	±

7

Рисунок Д.7 – Слайд 7

Архітектура системи на основі предметної області



8

Рисунок Д.8 – Слайд 8

Рисунок Д.9 – Слайд 9

Рисунок Д.10 – Слайд 10

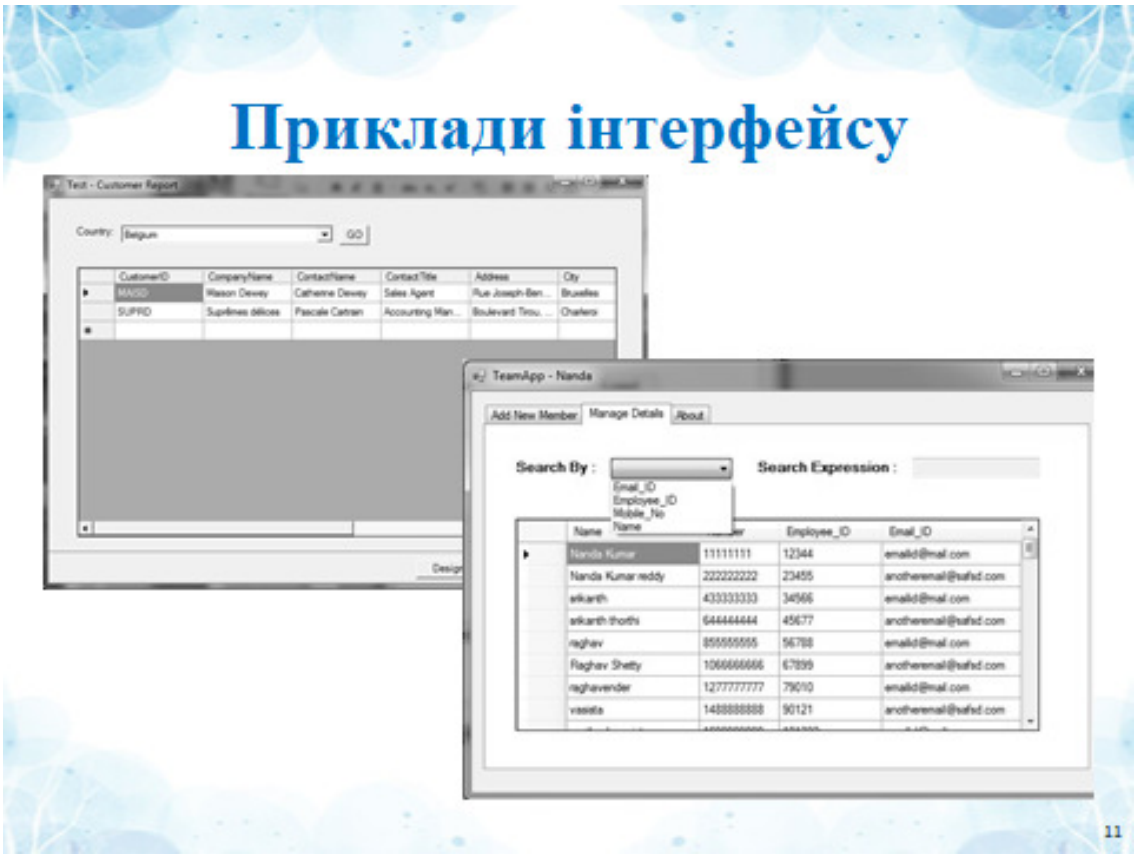


Рисунок Д.11 – Слайд 11

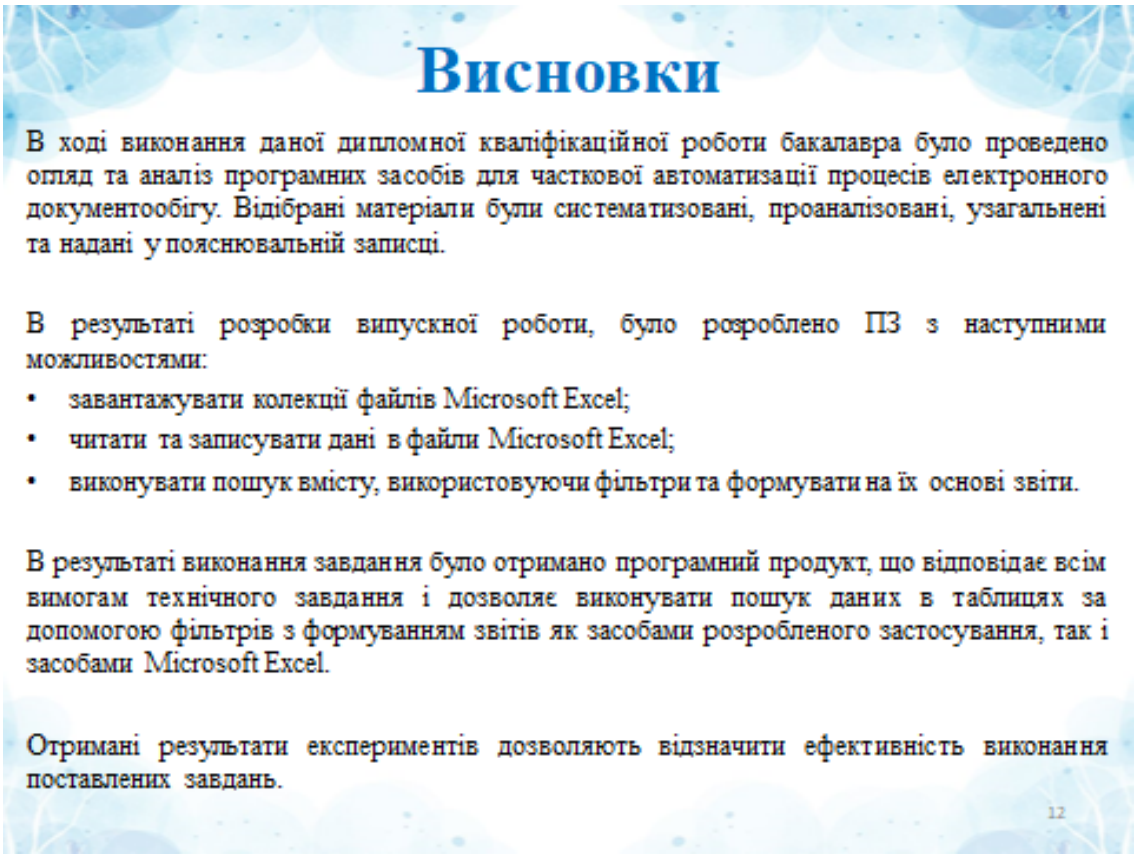


Рисунок Д.12 – Слайд 12