Державний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЄКТ

на тему:

«Програмна компонента інтелектуальної системи формування запитів стейкхолдерів ІТ-галузі»

Студента 4 курсу, 6 групи,
спеціальності 121 «Інженерія програмного забезпечення»
освітньої програми «Інженерія програмного забезпечення» _____ Андрусенко Валерія Павловича
підпис студента

Науковий керівник
PhD, доцент кафедри інженерії програмного забезпечення та кібербезпеки _____ Десятко Альона Миколаївна
підпис керівника

Гарант освітньої програми
кандидат технічних наук,
доцент кафедри інженерії програмного забезпечення та кібербезпеки _____ Рзаєва Світлана Леонідівна
підпис гаранта

КИЇВ – 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь бакалавр
Спеціальність 121 «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
«14» листопада 2022 р.

Завдання
на випускний кваліфікаційний проєкт студентові

Андрусенко Валерію Павловичу
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проєкту «Програмна компонента інтелектуальної системи формування запитів стейкхолдерів ІТ-галузі»

Затверджена наказом ректора від «6» грудня 2022 р. № 3288

2. Строк здачі студентом закінченого проєкту 5 червня 2023

3. Цільова установка та вихідні дані до проєкту

Мета проєкту: проектування компоненти інтелектуальної системи.

Об'єкт дослідження: процес формування запитів в інформаційно-інтелектуальній системі

Предмет дослідження: розробка програмного забезпечення формування запитів в інтелектуальній системі

4. Консультанти проєкту із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускного кваліфікаційного проєкту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. МЕТОДИ ОБРОБКИ BIG DATA ТА ПЕРСПЕКТИВИ РОЗВИТКУ

1.1. Розвиток та трансформація поняття Big Data

1.2. Концепції роботи з обробкою Big Data

1.3. Перспективи розвитку великих даних

1.4. Висновки до розділу 1

РОЗДІЛ 2. ХАРАКТЕРИСТИКА ВЕБ-СЕРВІСУ DISCOVERY DATA

2.1. Дослідження засобів розробки програмного модуля в DiscoveryData

2.2. Опис структури веб-сервісу DiscoveryData

2.3. Модель автоматизованої системи DiscoveryData

2.4. Висновки до розділу 2

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО МОДУЛЮ

3.1. Інтерфейс програмного модулю веб-сервісу DiscoveryData

3.2. Опис структури програмного модулю

3.3. Висновки до розділу 3

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ТЕХНІЧНЕ ЗАВДАННЯ

МЕТОДИКА ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЯ

ДОДАТКИ

6. Календарний план виконання проєкту

№ пор.	Назва етапів випускного кваліфікаційного проєкту	Строк виконання етапів проєкту	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускного кваліфікаційного проєкту*	*21.09.2022*	*21.09.2022*
2.	*Розробка та затвердження завдання на проєкт*	*14.11.2022*	*14.11.2022*
3.	*Вступ та перелік літературних джерел*	*23.12.2022*	*23.12.2022*
4.	*Розділ 1. Аналіз вимог до програмного забезпечення*	*27.01.2023*	*27.01.2023*
5.	*Розділ 2. Проектування програмного забезпечення*	*03.03.2023*	*03.03.2023*
6.	*Розділ 3. Реалізація програмного додатку*	*14.04.2023*	*14.04.2023*
7.	*Висновки*	*28.04.2023*	*28.04.2023*
8.	*Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)*	*17.05.2023*	*17.05.2023*
9.	*Підготовка автореферату та презентації доповіді*	*26.05.2023*	*26.05.2023*
10.	*Попередній захист випускного кваліфікаційного проєкту*	*29.05.2023 – 02.06.2023*	
11.	*Зовнішнє рецензування випускного кваліфікаційного проєкту*	*05.06.2023*	*05.06.2023*
12.	*Здача прошитого випускного кваліфікаційного проєкту на кафедру*	*05.06.2023*	*05.06.2023*
13.	*Публічний захист випускного кваліфікаційного проєкту*		

7. Дата видачі завдання «14» листопада 2022 р.

8. Науковий керівник випускного кваліфікаційного проєкту Десятко А.М.
(прізвище, ініціали, підпис)

9. Гарант освітньої програми Рзаєва С.Л.
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Андрусенко В.П.
(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проєкту

Науковий керівник випускного кваліфікаційного проєкту

_____

(підпис, дата)

Відмітка про попередній захист _____

(ПІБ, підпис, дата)

12. Висновок про випускний кваліфікаційний проєкт

Випускний кваліфікаційний проєкт студента _____Андрусенко В.П.___

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми _____Рзаєва С.Л._____

(прізвище, ініціали, підпис)

Завідувач кафедри _____Криворучко О. В.

(підпис, прізвище, ініціали)

«_____»_____20 ___ р.

АНОТАЦІЯ

Відповідно до мети дослідження робота присвячена проектуванню компоненти інтелектуальної системи, яка допомагає підібрати рекомендованих кандидатів за критеріями роботодавців.

У результаті аналізу можливих сфер застосування запропонованої системи, її розробки є більш ніж доцільною. Також, аналіз даних про стан ІТ-галузі в Україні, кількість спеціалістів інформаційних технологій та випускників вищих закладів освіти вказує на те, що така система була б корисною не тільки роботодавцям, але й самим фахівцям.

Обидва компоненти (клієнтська та серверна частина) веб-додатку розроблені у середовищі Visual Studio Code. Сторона клієнта реалізована з використанням фреймворку React, сторона сервера – з використанням Node.js та фреймворку Express. Також, до системи підключена реляційна база даних MySQL.

Готове програмне забезпечення відповідає розробленому макету в Figma, функціональним та нефункціональним вимогам технічного завдання.

Ключові слова: JavaScript, HTML, CSS, фреймворк, React, Node.js, API, інтелектуальна система, веб-додаток, сайт, алгоритм, база даних, архітектура, запит, відповідь, кандидат, рекомендації, роботодавець.

ABSTRACT

According to the purpose of the research, the work is devoted to the design of a component of an intelligent system that helps to search for recommended candidates based on the criterias of employers.

As a result of the analysis of possible areas of application of the proposed system, its development is more than appropriate. Also, the analysis of data on the state of the IT industry in Ukraine, the number of information technology specialists

and graduates of higher education institutions indicates that such a system would be useful not only to employers, but also to the specialists themselves.

Both components (front-end and back-end) of the web application are developed in the Visual Studio Code environment. The client side is implemented using the React framework, the server side is implemented using Node.js and the Express framework. Also, a MySQL relational database is connected to the system.

The finished software meets the designed mock-up in Figma, the functional and non-functional requirements of the specification.

Keywords: JavaScript, HTML, CSS, framework, React, Node.js, API, intellectual system, web application, site, algorithm, database, architecture, request, response, candidate, recommendations, employer.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Big Data - це термін, що позначає величезний масив різної структурованої та неструктурованої інформації, а також методи її обробки й аналізу.

SQL – (Structed Query Language) – структурована мова запитів.

NoSQL – база даних, яка забезпечує механізм зберігання та видобування даних відмінний від підходу таблиць-відношень в реляційніх базах даних.

СУБД – система управління базами даних.

БД – база даних.

Xpath - (XML Path) – мова виразів, для визначення частини XML документа, або для обчислення величин (наприклад, рядкових, числових або булевих) на основі вмісту XML документа.

IDC – міжнародна корпорація даних, яка є провідним світовим постачальником ринкової інформації, консультаційних послуг та заходів для ринків інформаційних технологій, телекомунікацій та споживчих технологій.

CSS локатори – в CSS є патерни, згідно з якими стилі, створювані розробником, застосовуються до елементів сторінки (DOM). Ці патерни називаються локатори (selectors). Selenium WebDriver використовує той же принцип для знаходження елементів. І він набагато швидше, ніж пошук елементів на основі XPath.

ASP.NET – технологія створення веб-застосунків і веб-сервісів від компанії Майкрософт

HTML – (HyperText Markup Language) - це мова тегів, якою пишуться гіпертекстові документи для мережі Інтернет

ЗМІСТ

					ДТЕУ 121 06-01.БР		
					Програмна компонента інтелектуальної системи формування запитів стейкхолдерів ІТ-галузі	*Стадія*	*Аркуш*	*Аркушів*
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*		*Зміст*	3	40
Зав. каф.		Криворучко О.В.		14.04.23				
Керівник		Десятко А.М.		14.04.23				
Гарант		Рзаєва С.Л.		14.04.23	*Зміст*	*Факультет інформаційних технологій 4 курс, 6 група*		
Розробив		Андрусенко В.П.		14.04.23				

ВСТУП

Актуальність. В умовах розвитку інформаційних технологій та активного їх використання відбувається накопичення великої кількості інформації і кількість таких даних постійно зростає. Еволюційний процес розвитку людства, його хід і результати знаходять своє відображення в різноманітних фактах, відомостях, даних, інформації тощо. У різні історичні епохи ті відомості, що визнавалися доцільними для подальшого збереження, у доступний спосіб фіксувалися, завдяки чому могли бути використані не лише сучасниками, але й наступними поколіннями.

Таким чином, у найбільш узагальненому сенсі людство постійно продукує, накопичує і використовує певний «інформаційний продукт». В зв'язку з цим, розпочалося існування такого поняття як Big Data. На перший погляд, судячи з назви, йдеться просто про значні за розміром інформаційні масиви.

Однак великий обсяг – лише одна з особливостей феномену Big Data, у якому, з одного боку, знайшли втілення комп'ютерно-інформаційні тренди останніх десятиріч; з іншого – він сам здатен впливати й реально трансформує існуючі уявлення та напрацьовані впродовж тривалого часу практики й моделі поведінки як окремих індивідів, так і складних організаційних структур.

З появу даного феномену, зі стрімким розвитком технологій та появою великого обсягу статистичних даних з'явились складнощі з їх опрацюванням, з'явились проблеми стосовно обробки такої кількості великих даних, їх зберігання, а надалі й аналізу.

Big Data – «великі дані», наразі привертають до себе все більше й більше уваги і стають предметом вивчення дедалі ширшого кола дослідників – від аналітиків

даних до економістів, соціологів, маркетологів, медиків і т.д. Світ стоїть на порозі епохи великих даних, і люди стикаються з ними щодня.

Мета дослідження полягає в дослідженні методів і технік обробки масивів великих даних, способів їхнього зберігання та створенні програмного модулю в фінансовому проекті DiscoveryData, що обробляє масив великих даних та видає результат по заданому фільтру, а також створенні тестового покриття для розробленого програмного модулю.

Об'єктом дослідження – автоматизована система обробки даних DiscoveryData.

Предметом дослідження – методології автоматизації процесу обробки даних для отримання результату за заданим фільтром в DiscoveryData.

Для досягнення поставленої мети необхідно вирішити наступні *задачі дослідження:*

- проаналізувати перспективи великих даних у майбутньому;
- розглянути необхідні програмні інструменти та технології в DiscoveryData;
- дослідити архітектуру веб-сервісу DiscoveryData;
- розробити модель автоматизованої системи;
- розглянути та проаналізувати основні підходи та методи роботи з великими об'ємами даних;
- здійснити програмну реалізацію модуля в DiscoveryData;
- здійснити програмну реалізацію модулю обробки великих даних;
- дослідити види тестуваннь програмного продукту та проаналізувати тестові фреймворки;
- розробити тестове покриття програмного модулю.

Методи дослідження. Теоретичне дослідження, проблематика предметної області здійснюється на основі загально-наукового аналітичного методу.

Інформаційну базу дослідження склали праці провідних вчених, навчальні посібники, книги, статті та інформаційні ресурси глобальної мережі Інтернет у сферах функціонування автоматизованих систем обробки даних.

РОЗДІЛ 1.
МЕТОДИ ОБРОБКИ BIG DATA ТА ПЕРСПЕКТИВИ РОЗВИТКУ

1.1. Розвиток та трансформація поняття Big Data

Нині поняття Big Data не має зовсім чіткого трактування, оскільки теоретичне дослідження показало, що його розвиток доцільно розглядати в історичному контексті. Раніше вважалося, що критерієм віднесення до такого поняття як Big Data, перш за все, є потік даних, більший за 100 Гбайт на день.

Однак на сьогодні, виявилося, що цього недостатньо. Пізніше , як такий критерій, було запропоновано використовувати якісні ознаки – «певну кількість V»:Volume – значне збільшення обсягу даних у корпоративних системах; Velocity – швидкість отримання і обробки даних для задоволення запиту тощо. Однак такий підхід не розкриває це поняття повною мірою, а торкається окремих його аспектів.

В інших дослідженнях можна побачити визначення цього поняття як неможливості обробки даних традиційними способами. Однак таке визначення можна вважати вже застарілим, адже технології оброки великих обсягів даних експерти вже не відносять до новітніх.

Деякі фахівці пропонують взагалі відмовитися від цього поняття. Однак, зважаючи на його значне поширення, це не видається можливим. Розвиток інформаційних технологій змінив наше ставлення до даних та інформації, а це, у свою чергу, вплинуло на сутність поняття Big Data. Нині під цим скоріше розуміється особливий підхід – ідеологія обробки великих масивів «сирих» даних. Однак не виключено, що в майбутньому це поняття може зникнути як застаріле та неактуальне, адже великі обсяг даних і сучасні підходи до їх обробки стануть

звичайними засобами, і не буде необхідності робити акцент на їх інноваційній складовій. [13]

Історія Big Data починається набагато раніше. За версією одного з авторів Forbes, відправною точкою можна вважати 1944 р., коли американський бібліотекар Фремонт Райдер опублікував роботу The Scholar and the Future of the Research Library. Там він зазначив, що фонди університетських бібліотек в Америці збільшуються у два рази кожні 16 років і до 2040 р. бібліотека Єльського університету буде містити близько 200 млн. книг, для зберігання яких знадобиться майже 10 км полиць.

Пізніше поняття Big Data з'явилось 4 вересня 2008 року, саме тоді, коли вийшов спеціальний номер британського журналу «Nature», присвячений проблематиці бурхливого зростання глобальних даних та їхньої ролі у науці й суспільстві, що зображений на рисунку 1.1. [1, с. 14]

Рис. 1.1. Британський журнал «Nature» виданий 2008 року

І саме тут уперше звернено увагу на зміну відношення до даних. Дані розглядаються як ресурс на рівні природних копалин.

Аналітична компанія Gartner вводить визначення поняття Big Data через три «V»:[2, с. 86]

- Volume (об’єм) – це збільшення обсягів даних у корпоративних системах за рахунок збільшення обсягів трансакцій та використання традиційних і нових типів даних. Працівники даної компанії Gartner наголошували на тому, що дуже великий обсяг даних породжуватиме проблему їхнього зберігання, проте це також породжуватиме проблему аналізу даних.
- Variety (різноманіття та неструктурованість даних) – полягає у різноманітному форматі наявних даних, це можуть бути табличні дані (бази даних), ієрархічні дані, документи, дані вимірювання, дані електронної пошти, дані фінансових операцій тощо. Це породжує проблему переведення великих обсягів транзакційної інформації в рішення.
- Velocity (швидкість обробки) – означає як швидкість отримання даних, так і швидкість їх обробки для задоволення запиту.

На рисунку 1.2 зображено діаграму співвідношення цих трьох ознак Big Data. [3, с. 5]

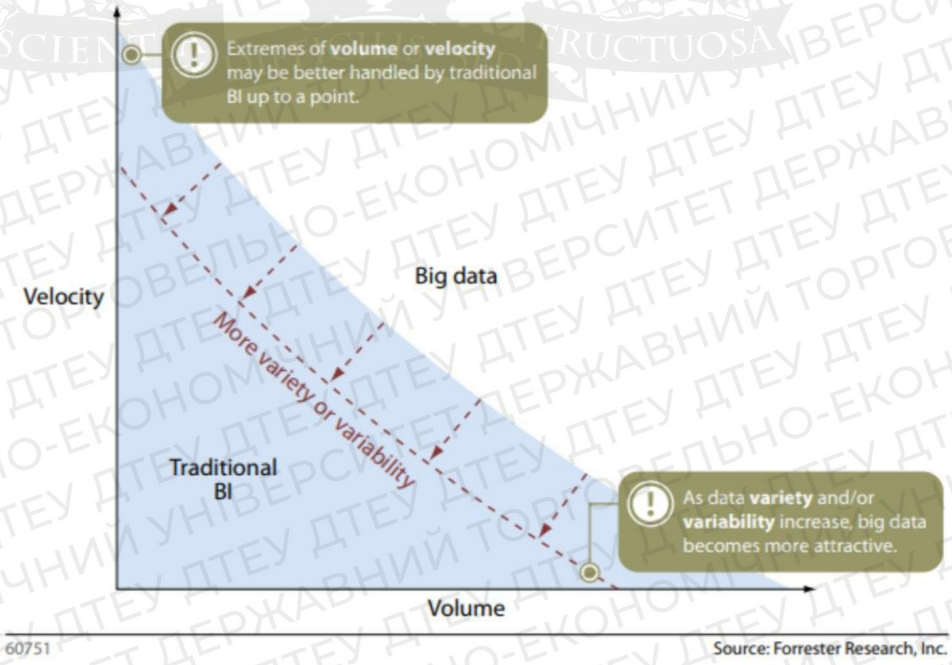


Рис. 1.2. Діаграма трьох ознак Big Data

З розвитком технологічного укладу людства стало можливим фіксувати та зберігати гігантські обсяги структурованих і неструктурованих даних, що отримали назву Big Data. Проте більш важливим є не набуття фізичної можливості володіти цими обсягами даних, а розуміння того, яким чином і з якою метою можна їх ефективно використати. Наразі як ніколи актуальним є перетворення даних в інформацію, а інформації – в знання, і саме з такого погляду слід розуміти відомий вислів «Хто володіє інформацією, той володіє світом».

1.2. Концепції роботи з обробкою Big Data

Усі можливі методи обробки даних поділяються на природні й технічні та програмні. Природні – це методи, засновані на органах чуттів людини (зір, дотик, нюх, слух, смак, спостереження, читання, логічне мислення, уява, порівняння, аналіз тощо); технічні методи включають апаратні (обробка здійснюється за допомогою спеціальних пристроїв – телефонів, магнітофонів, рентгенівських апаратів, мікроскопів тощо) та програмні (за допомогою програмних ресурсів).

Для збору і обробки Big Data доцільно використовувати технології хмарних обчислень. Хмарні обчислення – це нова парадигма для розміщення кластерів даних і надання різних послуг локальною мережею або через Інтернет. Хостинг кластерів даних дає змогу клієнтам зберігати та обчислювати величезну кількість даних у хмарі.

Технології Big Data спираються на обчислювальні кластери з безлічі обчислювачів, забезпечених локальною підсистемою зберігання. Доступ до даних і їх обробка здійснюються спеціальним програмним забезпеченням. Найбільш відомим в області Big Data є Apache Hadoop, яка представляє собою програмний фреймворк, що дозволяє зберігати і обробляти дані за допомогою комп'ютерних кластерів, використовуючи парадигму MapReduce. MapReduce - це революційний інструмент, представлений компанією Google, розроблений компанією Yahoo, був призначений для автоматичної обробки та розподілу величезних наборів даних.

Платформа Hadoop дозволяє скоротити час на обробку і підготовку даних, розширює можливості по аналізу, дозволяє оперувати новою інформацією та неструктурованими даними.

Компанія Oracle розбиває життєвий цикл обробки інформації на три етапи і використовує для кожного з них власне рішення:

1. Збір, обробка та структурування даних.
2. Агрегація і аналіз даних.
3. Аналітика даних в реальному часі.

Для збору, обробки та структурування даних, в якості вирішення застосовується Oracle Big Data Appliance - це встановлений Hadoop-кла-стер, Oracle NoSQL Database і засоби інтеграції з іншими сховищами даних. Завдання Oracle Big Data Appliance полягає в зберіганні та первинній обробці неструктурованою або частково структурованої інформації, тобто як раз в тому, що у систем на базі Hadoop виходить найкраще. [4, с. 245]

Ще одним із методів обробки є застосування інструменту Apache Spark, який був розроблений як альтернатива Hadoop, здатний виконувати швидше розподілені обчислення за допомогою примітивів в пам'яті. Завдяки своїй здатності завантажувати дані в пам’ять і повторно використовувати їх повторно, цей інструмент долає проблему ітеративної та онлайн-обробки, представлену MapReduce. Головне питання стосовно методів обробки та зберігання даних Big Data на корпоративному рівні: обрати реляційну (SQL) чи нереляційну (NoSQL) базу даних?

Головною причиною відмови від SQL баз даних (БД) є неправильна робота з самою базою. Більшість компаній не можуть собі дозволити тримати спеціалістів для постійного налагодження баз даних, а для того, щоб розпочати використовувати NoSQL БД не потрібно додаткових розробок. При розробці NoSQL БД особлива увага приділяється забезпеченню високої масштабованості та гнучкості рішень.[5, с. 46]

БД – це, перш за все, швидкий доступ до даних, що зберігаються в оперативній пам’яті, гнучкість використання та можливість швидкого розподілення даних між вузлами. Однак можливі такі сценарії, коли дані згодом виходять з-під контроля або вже просто не вміщуються в оперативній пам’яті.

1.3. Перспективи розвитку великих даних

Експоненціальне зростання глобальних даних, яке розпочалося десять років тому, не скоро виявить ознаки уповільнення. Замість того, щоб сповільнюватись, воно ще більше збільшується за допомогою Інтернету, медіа-файлів, текстових повідомлень, соціальних мереж та онлайн-пошуку.

Більшість експертів з великих даних погоджуються, що кількість генерованих даних у майбутньому буде зростати в геометричній прогресії. У звіті "Вік даних 2025" для Seagate, IDC прогнозує, що глобальна сфера даних досягне 175 цетабайт до 2025 року. [14]

Отже, що призвело до такого передбачення щодо майбутнього великих даних?

Тут є два фактори:

- Зростання кількості користувачів, які займаються всіма справами в Інтернеті - від соціальних мереж, покупок до ділових комунікацій.
- Мільярди вбудованих систем та підключених пристроїв, які щодня обмінюються, збирають та створюють цінні прогнозні аналітичні дані IoT у всьому світі.

Складно працювати з такими масивними наборами даних з точки зору їх обробки та зберігання. Таким чином, щоб забезпечити більшу еластичність, підприємства користуються допомогою хмари для технологій великих даних.

Будучи одночасно лякаючим і захоплюючим, майбутнє аналітики Big Data обіцяє змінити спосіб діяльності підприємств у фінансах, охороні здоров’я, виробництві та інших галузях. Величезний розміри великих даних можуть

створити додаткові виклики в майбутньому, включаючи конфіденційність даних та ризики безпеки, дефіцит спеціалістів з даних та труднощі при зберіганні та обробці даних.[6, с. 118]

Однак більшість експертів сходяться на думці, що Big Data означатимуть велику цінність. Це породить нові категорії робочих місць і навіть цілі відділи, відповідальні за управління даними у великих організаціях. З'являються нові регуляторні структури та стандарти поведінки, оскільки компанії продовжують використовувати особисті дані споживачів.

1.4. Висновки до розділу 1

Отже, Big Data – це термін, що позначає величезний масив різної структурованої та неструктурованої інформації, а також методи її обробки й аналізу.

В даному розділі також розглянуті найважливі методи обробки та зберігання Big Data. Серед яких такі технології як Apache Hadoop, MapReduce, NoSQL, SQL.

Можна підсумувати, що кількість підприємств, які використовують Big Data, безперервно зростає. Практика останніх років продемонструвала, що застосування результатів аналізу великих масивів даних може принести реальний ефект. Але, окрім переваг існує велика кількість проблем, вирішення яких вимагає застосування досить значних ресурсів. Очевидні потреби компаній в отриманні додаткової інформації про ринки, споживачів, конкурентів, працівниках, кон'юнктурі призводять до пошуку нових джерел інформації. Велика аналітика повинна бути забезпечена серйозними і зручними інструментами, як програмними, так і безпосередньо аналітичними.

РОЗДІЛ 2.

ХАРАКТЕРИСТИКА ВЕБ-СЕРВІСУ DISCOVERY DATA

2.1. Дослідження засобів розробки програмного модуля в DiscoveryData

Для реалізації проекту обрано програмну платформу «.Net Framework» та технологію ASP.NET. Код додатку буде реалізовано на мові програмування C#.

«.NET Framework» - це структура розробки програмного забезпечення від Microsoft. Це забезпечує кероване середовище програмування, де програмне забезпечення може бути розроблено, встановлено та виконано на операційних системах на базі Windows[17].

«.NET Framework» призначений для виконання наступних завдань:

1. для забезпечення узгодженого об’єктно-орієнтованого середовища програмування де код зберігається та виконується локально, який поширюється через Інтернет або виконується віддалено;
2. для забезпечення середовища виконання коду, який мінімізує конфлікти під час розгортання програмного забезпечення та керування версій;
3. для забезпечення середовища виконання коду, що сприяє безпечному виконанню коду, включаючи код, створений невідомим стороннім розробником;
4. для забезпечення середовища виконання коду, що усуває проблеми з продуктивністю середовищ виконання сценарію або коду, що інтерпретується;
5. забезпечення єдиних принципів розробки різних варіантів програм, таких як програми на базі Windows та веб-додатки;

6. взаємодія на основі галузевих стандартів, щоб забезпечити інтеграцію коду на основі «.NET Framework» з будь-яким іншим кодом.

Веб-сервіс DiscoveryData будується на основі ASP.NET.

ASP.NET - це платформа веб-розробки, яка забезпечує модель програмування, комплексну програмну інфраструктуру та різні служби, необхідні для створення надійних веб-програм для ПК, а також для мобільних пристроїв. Вона дозволяє створювати динамічні сторінки HTML. ASP.NET працює над протоколом HTTP і використовує команди та правила HTTP для встановлення двостороннього спілкування та співпраці між браузером і сервером.

Можна виділити ряд переваг технологій ASP.NET:

1. має перевагу в швидкості, якщо порівнювати з іншими технологіями, які засновані на скриптах;
2. завдяки користувальницьким елементам управління (controls) існує можливість виділяти шаблони, які використовуються частіше за інші;
3. можливість кешування всієї сторінки або її частини для збільшення продуктивності;
4. можливість кешування даних, що використовуються на сторінці;
5. наявність master-page, який задає шаблони оформлення сторінок;
6. простота у розгортанні.

Веб-сервіс DiscoveryData використовує об’єктно-орієнтовану мову програмування C#. Це мова зі строгим типом, що дозволяє розробникам створювати різні безпечні та надійні додатки, що працюють на платформі «.NET Framework».

C# можна використовувати для створення клієнтських додатків Windows, XML-веб-служб, розподілених компонентів, прикладних програм клієнт-сервера, прикладних баз даних тощо.

Visual C# надає:

- розвинений код редактора,

- зручні конструктори користувальницького інтерфейсу,
- інтегрований дебагер
- та багато інших засобів, які спрощують розробку додатків в C# для платформи «.NET Framework».

Мова C# була розроблена Андерсом Хейльсбергом та його командою під час розробки «.Net Framework».

Для реалізації системи автоматизації діяльності підприємства DiscoveryData було обрано програмні продукти: Microsoft SQL Server та Microsoft Visual Studio.

«Microsoft SQL Server» - комерційна система керування базами даних, що розповсюджується корпорацією Microsoft. Мова, що використовується для запитів - Transact-SQL, створена спільно Microsoft та Sybase. TransactSQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями.

«SQL Server Management Studio» – це утиліта, що входить до складу SQL Server для конфігурування, менеджменту та адміністрування всіх компонентів «Microsoft SQL Server». Утиліта включає скрипт-редактор і графічну програму, яка працює з об’єктами та налаштуваннями сервера.

Переваги «MS SQL Server»:

1. висока швидкість. SQL запити можна використовувати для отримання великої кількості записів з бази даних швидко та ефективно;
2. добре визначені існуючі стандарти. Бази даних SQL використовують давні стандарти, який приймаються ANSI та ISO;
3. не потрібно писати код. За допомогою стандартного SQL простіше керувати базою даних системи без необхідності писати великий обсяг коду;
4. простота адміністрування;
5. можливість підключення до Web.

«Visual Studio» – це програмний додаток, який базується на використанні компонентів та інших технологій для створення потужних реалізацій. Крім того,

середовище «Visual Studio» оптимізована для спільного проектування, розробки та розгортання корпоративних рішень. Даний продукт дозволяє розробляти як консольні додатки, так і додатки із графічним інтерфейсом.

2.2. Опис структури веб-сервісу DiscoveryData

DiscoveryData допомагає компаніям розвивати свій бізнес у сфері фінансових послуг та страхування завдяки наданим даним та аналітиці за допомогою хмарних додатків, інтеграції CRM та маркетингу, та аналізу ринку на основі даних. Понад 40 000 тис. зареєстрованих фірм, понад 730 000 тис. працівників, та понад 2 мільйонів зареєстрованих страхових агентів дозволяють розпорядникам активів, менеджарам по капіталу, страховим компаніям, фінансовим тенологічним провайдерам набирати кваліфікованих кандидатів, забезпечувати точні цільові та маркетингові роботи.

Програми розробки даних DiscoveryData розподілені по 6 екземпляів програм в одній зоні доступності. Якщо будь-який із екземплярів виходить з ладу або зона хостингу стає недоступною, будь-який екземпляр можна відновити за допомогою знімка обсягу до тієї ж зони або до альтернативної зони доступності.

Екземпляри – це окремі машини, на яких відбувається відтворення програм. Екземпляри програм розміщуються на екземплярах Amazon Web Services EC2.

Програми DiscoveryData моніторяться за допомогою декількох систем і програмою моніторингу подій. Якщо якась подія відбувається, менеджери організації отримують відповідного листа з описанням подій та рівень їхнього негативного впливу на систему.

Згідно рисунку 2.1 рішення (solution) DiscoveryData розроблено на основі шестикутної архітектури з двома портами: API на стороні даних та API на стороні споживача.

В даний час містить по одному адаптеру для кожного з портів .NET-компонентів на основі ADO.NET для API на стороні даних та служб REST для API на стороні споживача. Крім того, рішення містить компоненти, що забезпечують наскрізні функціональні можливості, не пов'язані з логікою API, такі як ведення журналу, автентифікація та компоненти, які відповідають за інтеграцію даних Discovery та SalesForce. Ця архітектура була обрана, оскільки дає можливість легкого розширення та адаптації логіки програми до будь-якого виду використання із зовнішніх систем шляхом створення адаптерів без зміни логіки програми. [12]

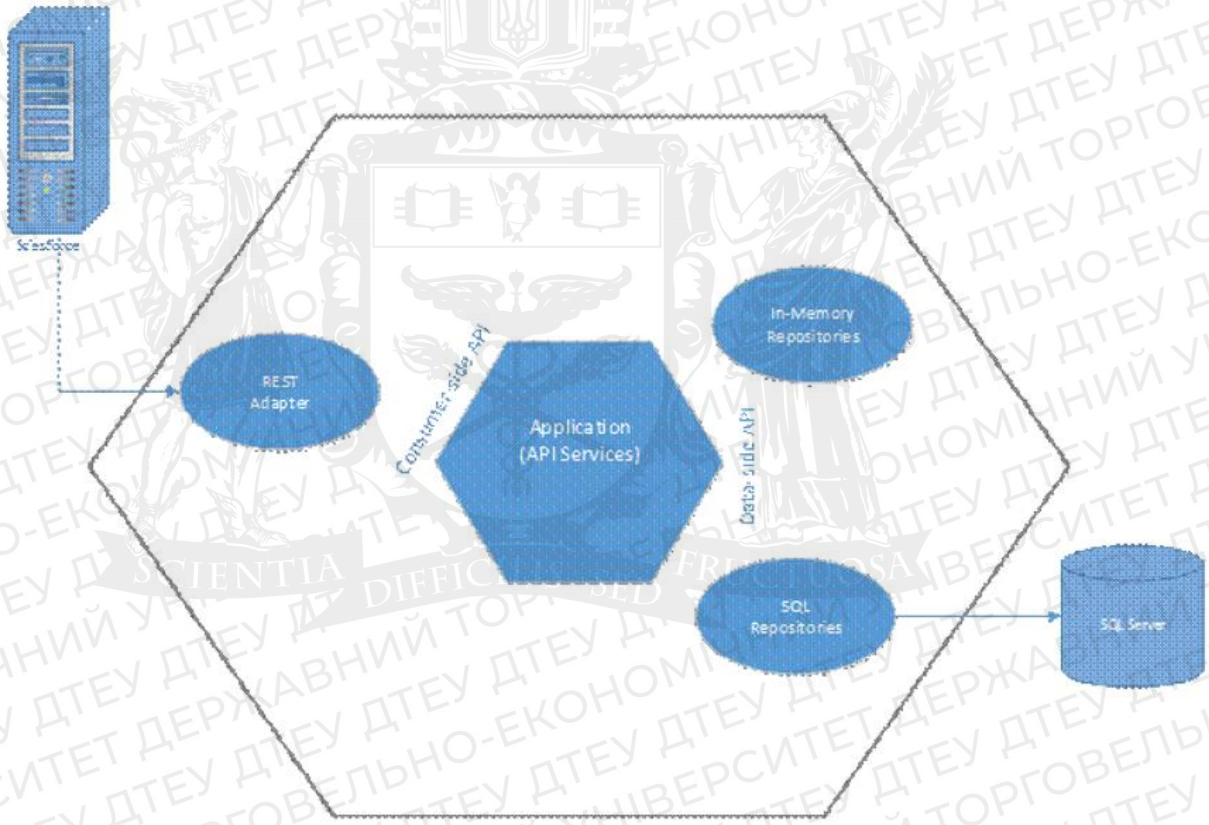


Рис. 2.1. Гексагональна архітектура (шестикутна архітектура)

Термін "Гексагональна архітектура" був введений (наскільки мені відомо) Алістером Коберном. Ця архітектура дозволяє взаємодіяти з додатком як користувачу, так і програмам, автоматичним тестів, скриптів пакетної обробки. Також дозволяє розробляти і тестувати програми без будь-яких додаткових пристроїв або баз даних.

Гексагональна архітектура – зветься архітектурою портів і адаптерів. Називають її так тому, що в рамках цієї архітектури є концепція різних портів, які можуть бути використані (адаптовані) для використання з іншими верствами.

База даних DiscoveryData оновлюється щотижня. Спочатку процес оновлення виконує на сервері Beta, а після схвалення – бета-оновлення копіюються на Prod-сервер. Плюси у застосованні гексагональної архітектури:

1. працювати з кодом класів і тестів стає значно простіше. Простіше знайти, простіше прочитати, простіше змінити;
2. максимальна гнучкість конфігурації сервісів. Якщо додати кешування для одного методу, то додаємо кешування тільки для нього. Якщо необхідно масштабувати тільки певні API, то масштабуємо тільки їх;
3. менше конфліктів при злитті гілок в системах версій вихідного коду, оскільки кожен розробник працює з окремим набором інтерфейсів і класів;

Згідно рисунку 2.2 розглянемо технологію серверних сторінок.

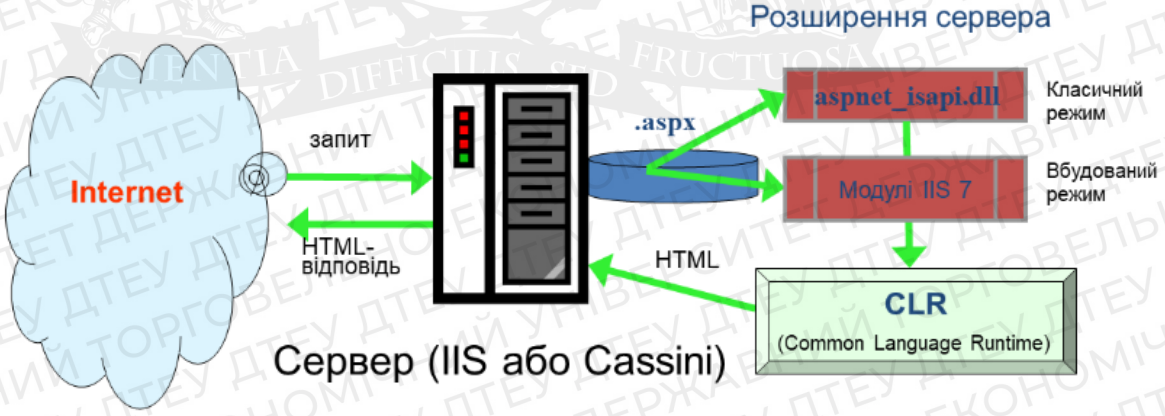


Рис. 2.2. Технологія серверних сторінок

Спочатку розглянемо схему роботи в мережі Internet, яку можна назвати «класичною», так як ця схема є історично першою.

Основними елементами класичної схеми є браузер і веб-сервер. При взаємодії браузер і веб-сервер проходять наступні етапи:

1. Браузер формує запит до сервера, використовуючи протокол HTTP. Як правило, браузер запитує HTML-сторінку, тобто текстовий файл, що містить HTML-код.
2. Сервер аналізує запит браузера і витягує з локального джерела даних необхідний файл.
3. Сервер формує HTTP-відповідь, що включає необхідну інформацію, і відсилає його браузеру по протоколу HTTP.
4. Браузер виконує відображення сторінки.

Класична схема проста, але має істотний недолік – сторінки статичні, і їх вміст не може змінюватися динамічно залежно від запитів клієнта. В даний час подібний підхід не відповідає більшості інформаційних послуг, наданих за допомогою мережі Internet. Класичним прикладом є типовий Інтернет-магазин. Якщо користувач на сторінці визначає якусь умову фільтрації, то браузер повинен відобразити список товарів, наявних в даний момент в магазині , які задовольняють фільтру

Тобто все більшого поширення набувають технології, при використанні яких сторінки (цілком або частково) генеруються на сервері безпосередньо перед відправкою клієнту. Ці технології зазвичай містять в своїй назві словосполучення Server Pages – «серверні сторінки» (ASP, ASP.NET).

Технологія «серверних сторінок» працює за схожими принципами:

1. Для подання інформації на сайті використовуються не сторінки з HTML-кодом, а серверні сторінки спеціального синтаксису (який часто є HTML-подібним).
2. При запиті серверної сторінки веб-сервер запускає окремий службовий процес, якому перенаправляється запит.
3. У службовому процесі сторінка аналізується, і на її основі генерується новий об'єкт (об’єкт класу), який відповідає класу сторінці.

4. Службовий процес виконує методи згенерованого об'єкта. Як правило, об'єкт має спеціальний метод, що генерує вихідний потік сторінки у вигляді HTML-коду.
5. Сервер перехоплює вихідний потік сторінки, формує HTTP- відповідь і відсилає його браузеру.
6. Браузер виконує відображення сторінки.

2.3. Модель автоматизованої системи DiscoveryData

Модель автоматизованої системи – це абстрактна модель автоматизованої системи, яка визначає структуру системи, механізми функціонування різних процесів та причинно-наслідкові зв'язки, властиві автоматизованій системі та істотні для досягнення мети моделювання. Головною метою створення моделі автоматизованої системи є опис зв'язків взаємозалежних процесів, що спрощує розуміння притаманних їх закономірностей.

Модель автоматизованої системи обробки даних DiscoveryData зображена на рисунку 2.3.

Наведемо пояснення до кожного елементу, що присутній на моделі. В даному веб-сервісі немає блоку реєстрації, оскільки всі компанії, працівники та страхові агенти проходять перевірку перш ніж отримати свій власний акаунт в даному сервісі. Після всіх перевірок, для них створюється акаунт з тимчасовим паролем. Сайт містить повну інформацію про фірм та її представників (телефони, адреси, ідентифікаційні коди, типи фірм, кількість працівників та інше).

Для представників компаній та страхових агентів, та безпосередньо компаній є спільні сторінки з попередньо збережними пошуками та основний пошук по загальним критеріям. Цей модуль виділено чевоним на рисунку 2.2 оскільки це є розроблений програмний модуль. Він є один із основних пошуків, тому що має базові фільтри для швидкого отримання інформації. Після введення інформації в

даному блоці, користувач може перейти на сторінку повного перегляду списку даних по заданим фільтрам.

Згідно рисунку 2.3 у кожної категорії, Люди чи Компанії, є свої підвиди. Категорія Люди поділяється на тих хто працює у фірмі, тих, хто працює у декількох фірмах та страхові агенти. Категорія Компанії поділяється на певні види фірм. Кожен підвид тої чи іншої категорії має специфічний набір фільтрів. Після здійснення фільтрації користувач може зробити такі завдання:

- переглядати детальну інформацію про учасників сайту (доступ до яких вам дозволено);
- створення списку для CallCenter;
- переглядати розміщення на карті;
- зберігати заданий фільтр;
- переглядати профіль;
- створювати нотатки;
- скачувати інформацію по заданим фільтрам чи її надрукувати.

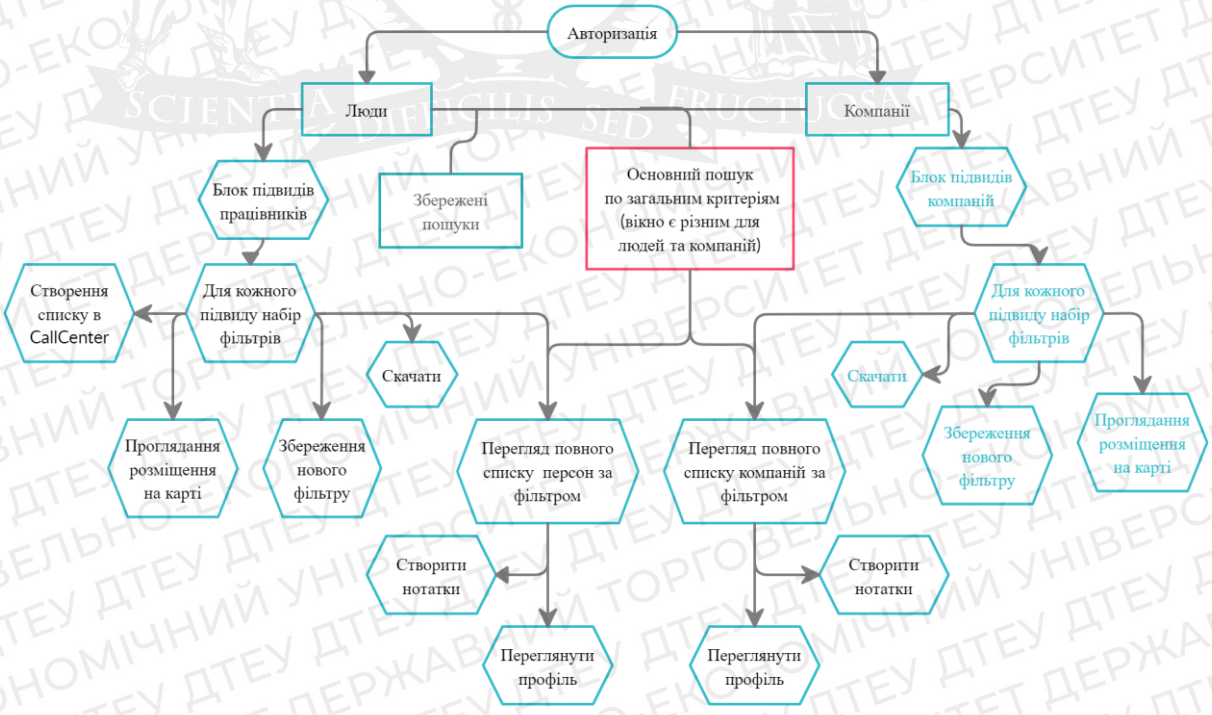


Рис. 2.3. Модель автоматизованої системи DiscoveryData

На рисунку 2.4 зображена одна з основних сторінок сайту. Тут знаходиться велика кількість фільтрів для отримання даних.

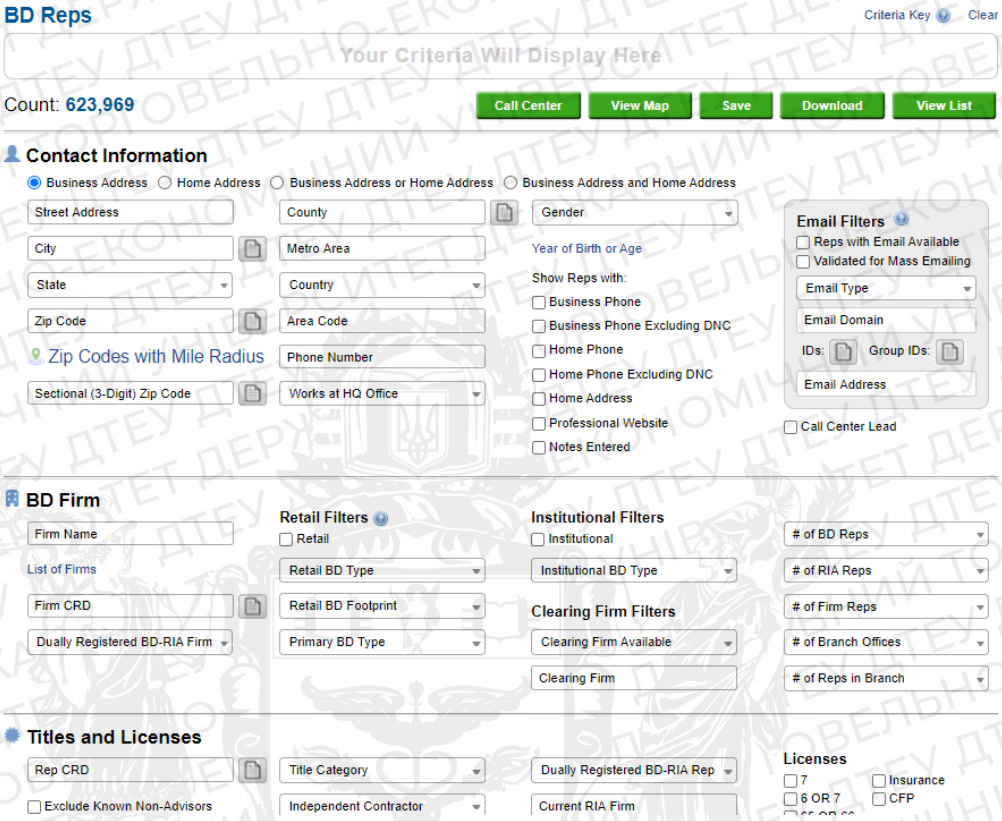


Рис. 2.4. Сторінка для фільтрування даних DiscoveryData

Користувач може обрати ті фільтри, які він рахує за потрібне, ці критерії будуть відображені у вікні «Your Criteria Will Display Here». Зелені кнопки допомагають надалі обробляти дані.

2.4. Висновки до розділу 2

Отже, у другому розділі розглянуті інструменти та технології, які були необхідні для створення програмного продукту та визначені їхні переваги.

Серед них: .NET Framework, ASP.NET, C#, Microsoft SQL Server, SQL Server Management Studio, Visual Studio. Розроблений програмний продукт написано об’єктно-орієнтованою мовою програмування C# з використанням платформи .NET Framework та ASP.NET. Також було досліджено, що автоматизована система

DiscoveryData розроблена на основі шестикутної архітектури, яка дає можливість легкого розширення та адаптації логіки програми до будь-якого виду використання із зовнішніх систем та розроблено модель веб-сервісу DiscoveryData, де здійснено опис найважливіших процесів та взамозв’язків цих процесів.

РОЗДІЛ 3.
РОЗРОБКА ПРОГРАМНОГО МОДУЛЮ

3.1. Інтерфейс програмного модулю веб-сервісу DiscoveryData

Щоб почати роботу з сайтом необхідно зареєструватись. Сайт DiscoveryData не має окремої сторінки для реєстрації. Кожних фірм, представників фірм, страхових агентів реєструють менеджемент DiscoveryData, адже кожний повинен пройти певну ідентифікацію.

Згідно рисунку 3.1 присутні два перемикача «Person» та «Company». Два перемикача застосовуються для того, щоб переключатись між сторінками, які передбачені лише для працівників та страхових агентів, а інші – для компаній.

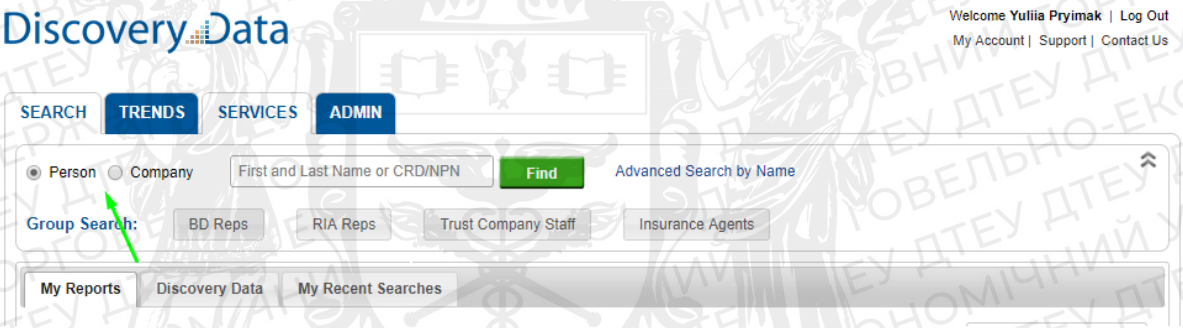


Рис. 3.1. Розміщення перемикачів Person та Company

Після вибору відповідного перемикача, користувач має змогу клікнути на розроблений модуль – Advanced Search by Name. Цей модуль використовується для фільтрації даних також, але він задає базову фільтрацію, яка є основною і спільною для всіх сторінок.

Згідно рисунку 3.2 можна помітити розміщення тексту, що є силкою для відкриття спливаючого вікна - Advanced Search by Name. Інформація щодо розміщення елементів завжди отримувалась від менеджменту проекту DiscoveryData.

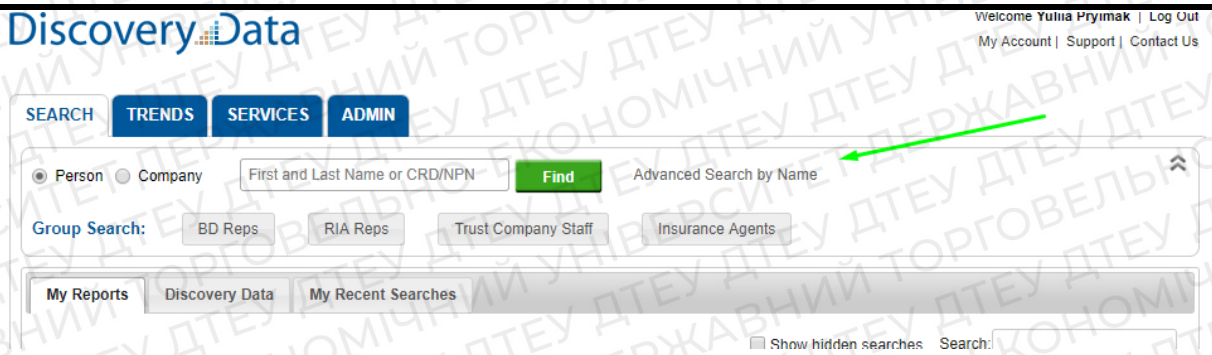


Рис. 3.2. Розміщення програмного модулю Advanced Search by Name

Після того як користувач нажимає на поле Advanced Search by Name, відкривається спливаюче вікно. Для Person (люди) та Company (фірми), це спливаюче вікно було розроблено різне.

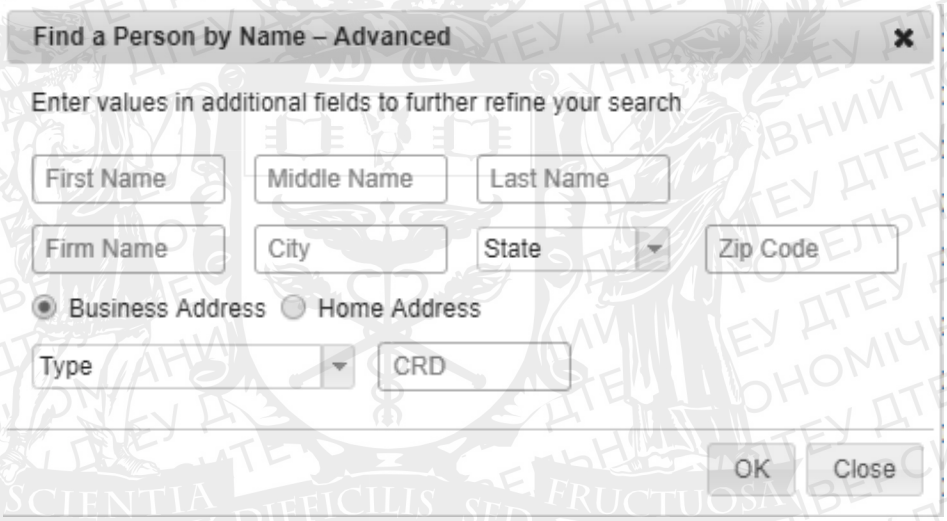


Рис. 3.3. Інтерфейс програмного модулю «Find a Person by Name – Advanced»

На рисунку 3.3. зображене спливаюче вікно для перемикача Person. Цей спливаюче вікно отримало назву – Find a Person by Name – Advanced. Тут зображено текст, що необхідно заповнити деякі поля для пошуку даних. У спливаючому вікні всі поля є необов'язкові, тому користувач може обрати ті поля, які знає для подальшого отримання даних.

Згідно рисунку 3.3 можна побачити такі поля: First Name (ім'я), Middle Name (друге ім'я), Last Name (прізвище), Firm Name (назва фірми), City (місто), State (штат), Zip Code (поштовий індекс), Business Address (адрес офісу), Home Address

(домашній адрес), Type (тип), CRD (це код є унікальним для кожної людини в системі).

Також користувач може знайти кнопку Ok – це підтверджує запит та відправляє його на сервер, та здійснює пошук в базі, після здійсненого пошуку, відкривається сторінка з даними в інтерфейсі, які відповідають вашому пошуку; та кнопку Cancel – відміняє дії спливаючого вікна та повертає вас на основну сторінку.

На рисунку 3.4 можна побачити приклад спливаючого вікна із заповненими даними для відображення дій реального користувача.

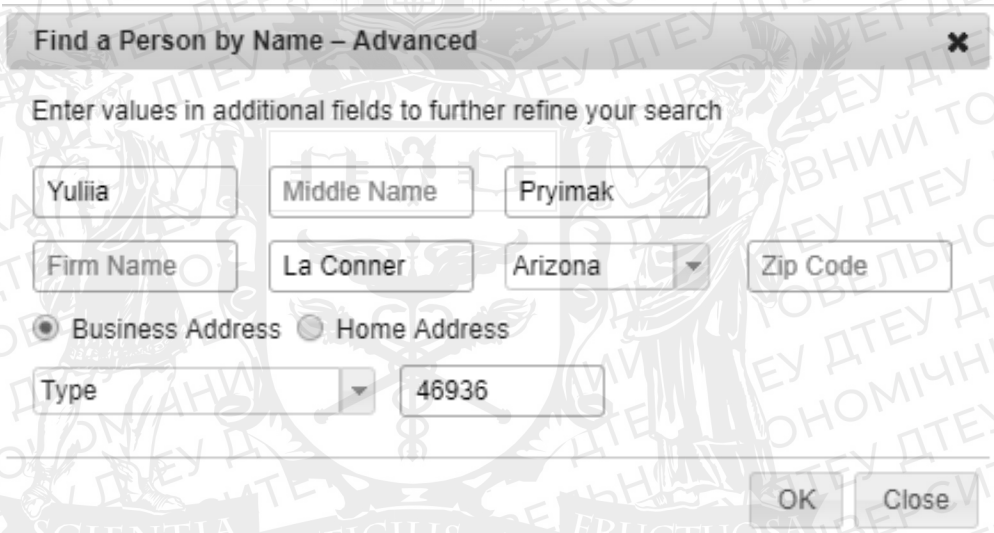


Рис. 3.4. Приклад заповнення спливаючого вікна «Find a Person by Name – Advanced»

На рисунку 3.5 продемонстровано сторінку результату пошуку, як бачить її реальний користувач після введення інформації у вікно Find a Company by Name – Advanced та нажаття кнопки ОК.

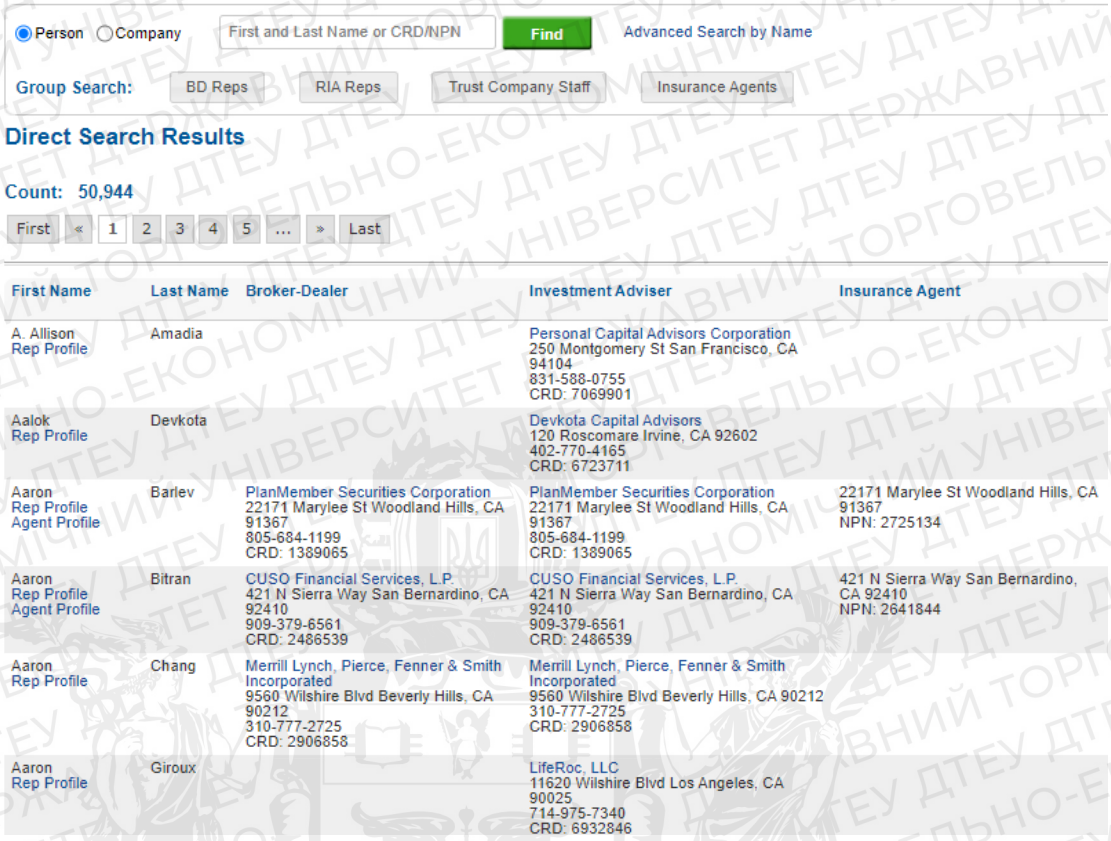


Рис. 3.5. Приклад сторінки резулату пошуку Find a Person by Name – Advanced»

На сторінці, що зображено на рисунку 3.5, користувач має змогу бачити загальну кількість записів, що відповідають заданому пошуку, імені та прізвища, силки для переходу на профіль цих персон, адреси в яких вони працюють та номери контактів.

На рисунку 3.6 зображене спливаюче вікно для перемикача Company. Цей попап отримав назву – Find a Company by Name – Advanced. Тут зображено текст, що необхідно заповнити деякі поля для пошуку даних. У спливаючому вікні поля є необов'язкові, тому користувач може обрати ті поля, які знає для подальшого отримання даних.

Згідно рисунку 3.6 можна помітити такі поля: Firm Name (назва фірми), City (місто), State (штат), Zip Code (поштовйи індекс), Type (тип), CRD (цей код є унікальним для кожної фірми в системі).

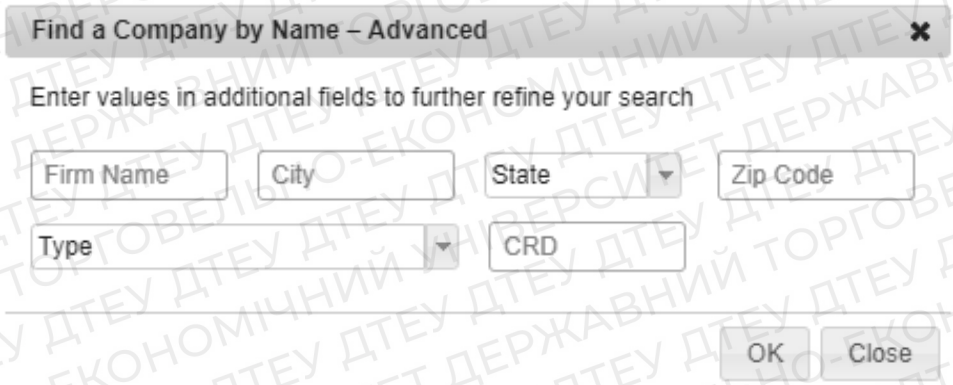


Рис. 3.6. Інтерфейс програмного модулю «Find a Company by Name – Advanced»

Також користувач має змогу знайти кнопки Ok та Cancel, які виконують аналогічні функції, як в попередньому вікні. Після нажаття на кнопку Ok клієнтська частина (браузер) відправляє заповнену форму з даними на веб-сервер, використовуючи проток HTTP. На веб-сервері дані обробляються за допомогою скриптів. Скрипти на основі отриманих даних формують SQL-запити і відправляють їх до сервера бази даних. Сервер бази даних отримує текст SQL-запитів, аналізуэ запит, обробляє їх, витягує з джерела даних необхідну відповідь, а пізніше вже сформовану відповідь - результат виконання цього, відправляє назад до веб-серверу. На основі отриманого результату веб-сервер формує HTML-сторінку, яку надсилає користувачу у веб-браузер. Браузер виконує відображення сторінки. За допомогою такої взаємодії клієнт не бачить код скриптів, а тільки результат, який вони повертають.

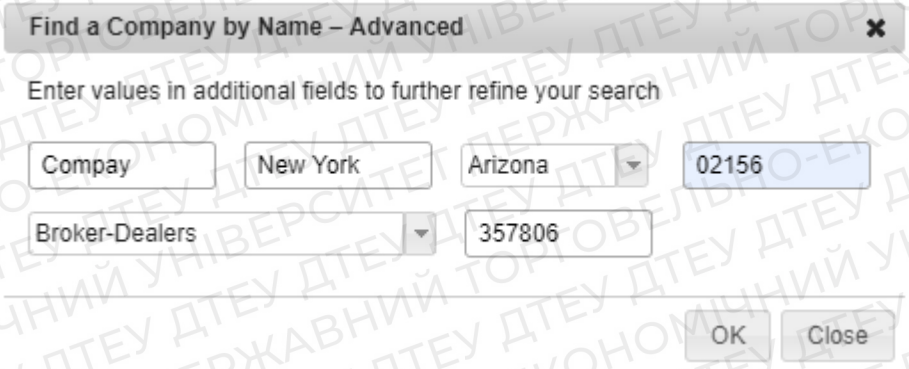


Рис. 3.7. Приклад №1 заповнення спливаючого вікна «Find a Company by Name – Advanced»

На рисунку 3.7, 3.8 можна побачити приклад спливаючого вікна із заповненими даними для відображення дій реального користувача.

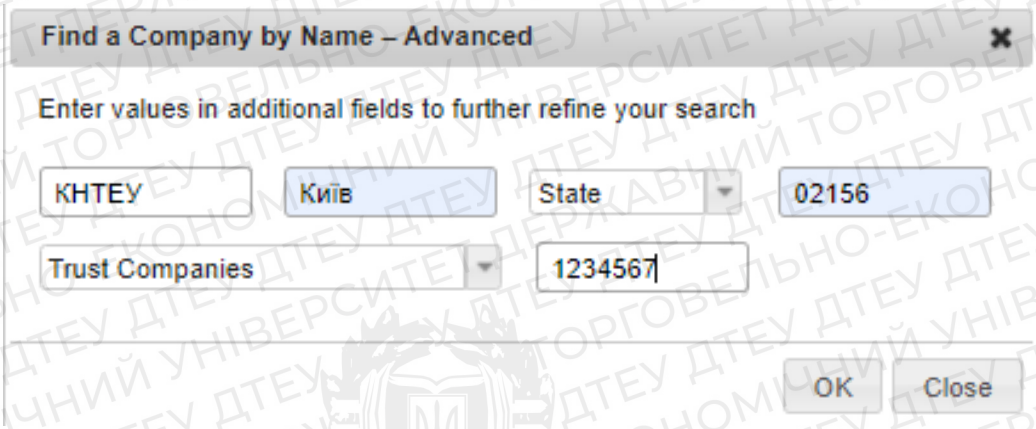


Рис. 3.8. Приклад №2 заповнення спливаючого вікна «Find a Company by Name - Advanced»

На рисунку 3.9 продемонстровано сторінку результату пошуку, як бачить її реальний користувач після введення інформації у вікно Find a Company by Name – Advanced та нажаття кнопки ОК.

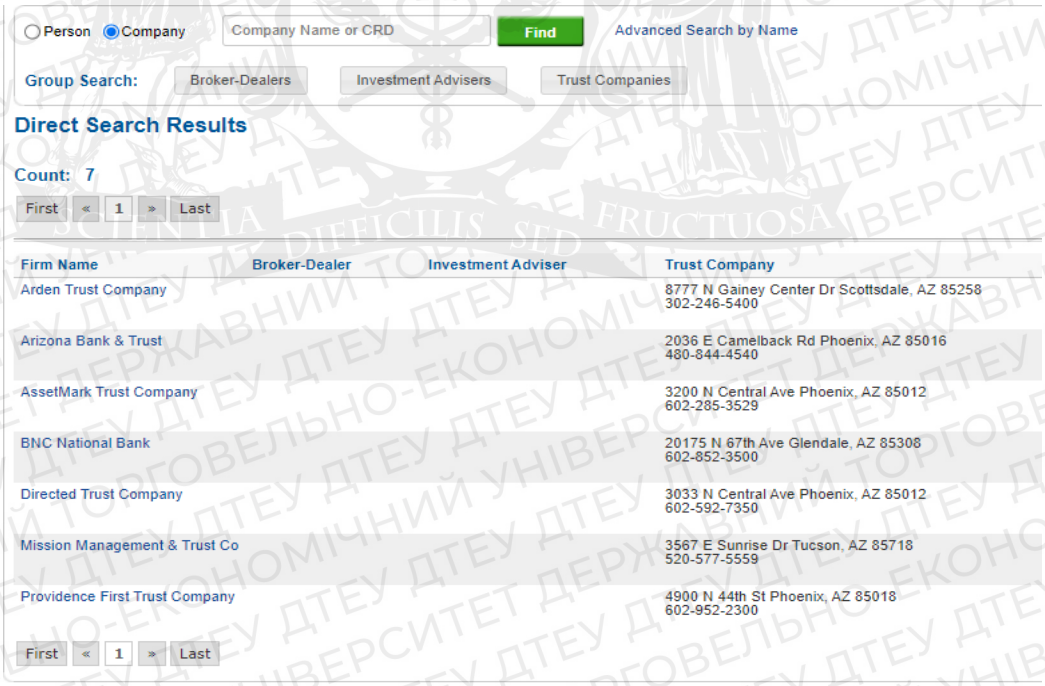


Рис. 3.9. Приклад сторінки результату пошуку «Find a Company by Name - Advanced»

На сторінці, що зображено на рисунку 3.9, користувач має змогу бачити загальну кількість записів, що відповідають заданому пошуку, назви фірм, що є

силками для переходу на профіль цих фірм, адресу, номери контактів та до якого типу фірм вони відносяться.

3.2. Опис структури програмного модулю

Структура розроблюваного програмного продукту спроектована таким чином, що модуль «Advanced Search by Name» буде динамічно підключатися до головної сторінки програми.

Це надасть змогу користувачеві швидко знайти даний модуль та застосувати його в своїх цілях.

Для того, щоб розробити даний модуль використовувалось середовище Microsoft Visual Studio. Середовище розробки – Microsoft Visual Studio — один з продуктів компанії Майкрософт, який представляє з себе інтегровану середу розробки програмного забезпечення і ряд інших інструментальних засобів.

Для розробки тестової системи використаний сервер Apache, на якому будуть розміщені усі модулі та класи системи. Для розробки модулю була використана технологія ASP.NET та мова програмування C#. У мові C# (створеному компанією Microsoft для підтримки середовища .NET Framework) перевірені часом засоби вдосконалені за допомогою найсучасніших технологій. C# надає дуже зручний і ефективний спосіб написання програм для сучасного безпечного середовища обчислювальної обробки даних, яка включає операційну систему Windows, Internet та інші компоненти.[15]

Згідно додатку А, визначені input для наших перемикачів Person та Company. В input визначаємо type = radio, для того, щоб це було дійсно перемикачі. Для опису нашого спливаючого вікна були використані такі теги, як:

- <p> - визначає текстовий абзац. Тег <p> є блоковим елементом, завжди починається з нового рядка, абзаци тексту йдуть один за одним розділяються між собою відбиттям. Величиною відбиття можна

керувати за допомогою стилів. Якщо закриває тега немає, вважається, що кінець абзацу збігається з початком наступного блокового елемента.

- <input> - тег <input> є одним з різнобічних елементів форми і дозволяє створювати різні елементи інтерфейсу і забезпечити взаємодію з користувачем. Головним чином <input> призначений для створення текстових полів, різних кнопок, перемикачів і прапорців.
- - елемент є блоковим елементом і призначений для виділення фрагмента документа з метою зміни виду вмісту. Як правило, вид блоку управляється за допомогою стилів. Щоб не описувати кожен раз стиль всередині тега, можна виділити стиль в зовнішню таблицю стилів, а для тега додати атрибут class або id з ім'ям селектора.
- <img> - тег <img> призначений для відображення на веб-сторінці зображень в графічному форматі GIF, JPEG або PNG. Адреса файлу з картинкою задається через атрибут src. Якщо необхідно, то малюнок можна зробити посиланням на інший файл, помістивши тег <img> в контейнер <a>. При цьому навколо зображення відображається рамка, яку можна прибрати, додавши атрибут border = "0" в тег <img>.
- <a> - тег <a> є одним з важливих елементів HTML і призначений для створення посилань. Залежно від присутності атрибутів name або href тег <a> встановлює посилання або якір. Якорем називається закладка всередині сторінки, яку можна вказати в якості мети посилання. При використанні посилання, яка вказує на якір, відбувається перехід до закладки всередині веб-сторінки.
-
 - тег
 встановлює новий рядок в тому місці, де цей тег знаходиться. На відміну від тега абзацу <p>, використання тега
 не додає порожній відступ перед рядком. Якщо текст, в якому використовується новий рядок, обтікає плаваючий елемент, то за

допомогою атрибута clear тега
 можна зробити так, щоб наступний рядок починалася нижче елемента.

- <table> - для позначення таблиці. Елемент <table> служить контейнером для елементів, що визначають вміст таблиці. Будь-яка таблиця складається з рядків і осередків, які задаються за допомогою тегів <tr> і <td>. Усередині <table> допустимо використовувати наступні елементи: <caption>, <col>, <colgroup>, <tbody>, <td>, <tfoot>, <th>, <thead> і <tr>.
- <tr> – тег <tr> служить контейнером для створення рядки таблиці. Кожна клітинка в межах такого рядка може задаватися за допомогою тега <th> або <td>.<td>- використовується для створенна одного елементу таблиці.
- Asp:TextBox – для отримання вводу користувача.
- asp: RadioButtonList – надає групу перемикачів з одним вибором, яка може бути динамічно створена за допомогою прив'язки даних.
- asp: RadioButton Class – представляє перемикачів управління.
- Select – створюэ елемент у вигляді випадаючого списку
- Option – визначає окремі пункти списку , що створюється за допомогою контейнера select.
- asp:Panel – представляэ елемент керування, який діє як контейнер для інших елементів керування.
- asp: Label – представляє елемент керування ярликом, який відображає текст на веб-сторінці.

Будь-яка серверна сторінка ASP.NET являє собою текстовий файл з розширенням .aspx.

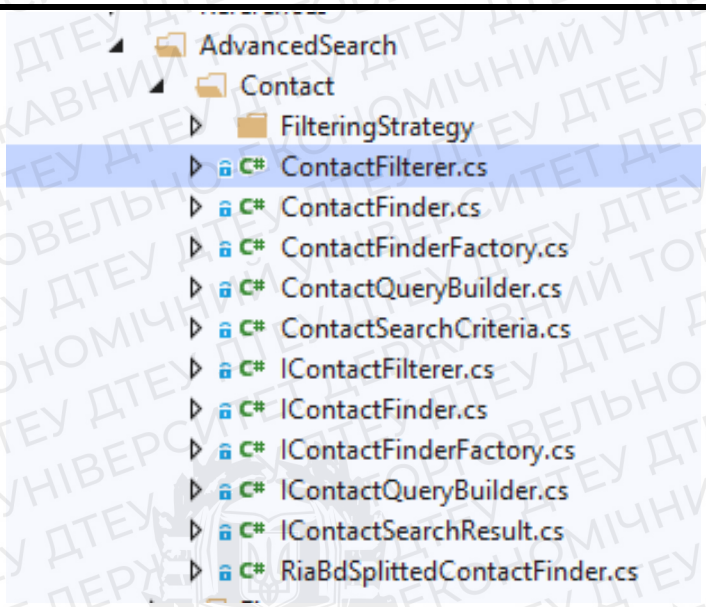


Рис. 3.7. Основні створені класи програмного модулю Advanced Search by Name

Розглянемо одні з вайжливіших класів. Згідно додатку Б розглядаємо клас ContactFilterer, що наслідується від інтерфейсу IContactFilterer. В даному класі є публічний метод ApplyFiltering, який отримує пошук – searchCriteria створений юзером та здійснює відповідну обробку, щоб вони відповідали заданому пошуку.

Присутня перевірка чи не є це поле пусте – null, а повернення методу є query – отриманий результат.

Згідно додатку В розглянемо клас ContactQueryBuilder, що наслідується від інтерфейсу IContactQueryBuilder. В даному класі написані публічні методи для пошуку контактів по CRD чи по імені, застосування сортування. Для реалізації цих методів було використано інтерфейс - Інтерфейс Iqueryable.

Інтерфейс Iqueryable - забезпечує функціональність для оцінки запитів щодо конкретного джерела даних, де тип даних не вказаний.

Інтерфейс IQueryable призначений для реалізації постачальниками запитів. Він повинен бути реалізований лише постачальниками, які також реалізують IQueryable <T>. Якщо постачальник також не реалізує IQueryable <T>, стандартні оператори запитів не можуть використовуватися у джерелі даних постачальника. Інтерфейс IQueryable успадковує інтерфейс IEnumerable, так що якщо він

представляє запит, результати цього запиту можуть бути перераховані. Перерахування викликає виконання дерева виразів, пов'язаного з об'єктом IQueryable. Визначення "виконання дерева виразів" є специфічним для постачальника запитів.

Наприклад, це може передбачати переклад дерева виразів на відповідну мову запитів для базового джерела даних. Запити, які не повертають незліченні результати, виконуються при виклику методу Execute.

```
public class ContactSearchCriteria
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string MiddleName { get; set; }
    public string FirmName { get; set; }
    public string City { get; set; }
    public string State { get; set; }
    public string Zip { get; set; }
    public string Crd { get; set; }
    public bool IsHomeAddress { get; set; }
    public SearchTypeEnum? SearchType { get; set; }
}
```

Рис. 3.8. Опис класу ContactSearchCriteria

Також був створений один з найважливіших класів ContactSearchCriteria, адже в даному класі були описані змінні, які використовувались у всіх методах при розробці даного модуля. Тут знайдемо і FirstName, і LastName, і City і так далі. Даний клас зображений на рисунку 3.8.[16]

3.3. Висновки до розділу 3

Отже, в даному розділі описано інтерфейс розробленого програмного модулю, як користувач може користуватись даним модулем та як формуються запити для відправки на веб-сервер, взаємодії з базою даних для подальшого отримання результату у браузері.

Також наведені пояснення до створеного лістингу коду та описи найважливіших методів, які використовуються. Розроблений програмний модуль дозволяє користувачеві ввести дані, надіслати поточний запит та переглядати необхідну інформацію із бази даних, яка відповідає заданому пошуку. Розробка проводилась з використанням середовища Microsoft Visual Studio. Для розробки модулю була використана технологія ASP.NET та мова програмування C#.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Отже, розроблено програмний модуль автоматизованої системи обробки даних DiscoveryData. Даний програмний модуль має поля для введення інформації, чекбокси та поля з випадаючим списком, які користувач може заповнити та після нажаття кнопки підтвердження, отримати необхідну йому інформацію за заданим фільтром.

Для досягнення цієї цілі було досіджено поняття Big Data, опрацьовано методи обробки та зберігання даних, аналіз інструментів програмних інструментів та технологій, які використовує веб-сервіс DiscoveryData, проаналізовано процеси організації тестування та обранно інструменти для написання тестового покриття розробленого програмного модулю.

Серед інструментів використовувались такі: програмна мова C#, технології ASP.NET, платформи .NET Framework, середовище Microsoft Visual Studio, база даних Microsoft SQL Server, тестові фреймворки NUnit, XUnit , тестовий паттерн PageObject, система неперервної інтеграції TeamCity.

Створений програмний модуль в автоматизованій системі DiscoveryData може вільно використовуватись користувачами веб-сервісу DiscoveryData, а саме працівниками компаній та страховими агентами. Оскільки модуль дозволяє вводити базову інформацію для пошуку, що застосовується для всіх видів фірм, для людей, що працюють в цих фірмах або ж страхових агентів, здійснює пошук в базі та повертає необхідний результат на інтерфейс, де користувач може переглядати у зручний йому спосіб та проводити надалі необхідне йому дослідження. Модуль дозволяє не заповнювати абсолютно всі поля, користувач може обрати деякі з них та здійснити пошук. Даний програмний модуль був протестований з

використанням Unit, UI тестів та беспосередньо використання Manual тестів, імітуючи дії реального користувача. Тести були пройдені успішно, тому розроблений програмний модуль в автоматизованій системі DiscoveryData повністю готовий до використання.

Також згідно здійснених доліджень обраної тематики, можна сказати, що як показує досвід європейських країн, найбільш ефективні рішення «великих даних» відкривають нові перспективи для розвитку компаній у різних сферах:

- фінансовій, в т.ч. банки та страхові компанії;
- транспортній, логістичній; паливно-енергетичній;
- аграрній;
- в органах державного управління тощо.

Організації, що почали використовувати технологію «великих даних» отримують значні конкурентні переваги, адже мають змогу, як швидко і безпечно обробляти та зберігати великі дані, так і приймати рішення на основі сучасної, актуальної, повної інформації. З моменту появи, концепція «великих даних» проникла в різні інформаційно-технологічні сфери і займає все більш важливу роль на практиці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Френкс Б. Революція в аналітиці. Як в епоху Big Data поліпшити ваш бізнес за допомогою операційної аналітики / Білл Френкс., 2018. – 316 с.
2. Вайгенд А. BIG DATA. Вся технологія в одній книзі / Андреас Вайгенд., 2018. – 348 с.
3. Ерл Т. Основи Big Data. Концепції, алгоритми і технології / Томас Ерл., 2018. – 320 с.
4. Furht B. Big Data Technologies and Applications / B. Furht, F. Villanustre., 2016. – 405 с.
5. Srinivasan S. Guide to Big Data Applications / Srinivasan., 2018. – 567 с.
6. Akhtar S. Big Data Architect's Handbook: A Guide to build proficiency in tools and systems used by leading Big Data experts / Akhtar., 2018. – 593 с.
7. Hunt A. Pragmatic Unit Testing in C# with NUnit / A. Hunt, D. Thomas., 2017. – 220 с.
8. Грегорі К. Гибкое тестирование. Практическое руководство для тестировщиков ПО и гибких команд / К. Грегорі, Л. Кріспін., 2010. – 464 с.
9. Блек Р. Ключевые процессы тестирования. Планирование, подготовка, проведение, совершенствование / Рекс Блек., 2016. – 537 с.
10. Kawalerowicz M. Continuous Integration in .NET / M. Kawalerowicz, C. Berntson., 2015. – 328 с.
11. Laster B. Continuous Integration vs. Continuous Delivery vs. Continuous Deployment: The Processes and Tools of Effective Continuous Delivery Pipelines / Brent Laster., 2020. – 438 с.

					ДТЕУ 121 06-01.БР
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*	
Зав. каф.		Криворучко О.В.		23.12.22	*Програмна компонента інтелектуальної системи формування запитів стейкхолдерів ІТ-галузі*
Керівник		Десятко А.М.		23.12.22	
Гарант		Рзаєва С.Л.		23.12.22	
Розробив		Андрусенко В.П.		23.12.22	*Список використаних джерел*

Стадія	*Аркуш*	*Аркушів*
СВД	39	40

Факультет інформаційних технологій
4 курс, 6 група

Інтернет-ресурси

12. Fidao C. Гексагональная архитектура [Електронний ресурс] / Chris Fidao. – 2015. – Режим доступу до ресурсу: https://habr.com/ru/post/267125/.
13. Кравчук С. Що таке Big Data? [Електронний ресурс] / С. Кравчук. – 1103. – Режим доступу до ресурсу: http://thefuture.news/bigdata.
14. Farah A. The Future of Big Data: Predictions from Way2Smile's experts for 2020-2025 [Електронний ресурс] / Azhar Farah. – 2020. – Режим доступу до ресурсу: https://www.way2smile.ae/blog/big-data-predictions-for-2020-2025.
15. ASP.NET Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/asp.net/index.htm.
16. C# Tutorials [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/.
17. .NET Framework (.NET) [Електронний ресурс] – Режим доступу до ресурсу: https://www.techopedia.com/definition/3734/net-framework-net.

ТЕХНІЧНЕ ЗАВДАННЯ

Розробити програмний модуль в автоматизованій системі обробки даних DiscoveryData з наступними характеристиками.

1. Загальні відомості

Назва проекту: програмний модуль в автоматизованій системі обробки даних DiscoveryData.

Планові строки розробки: жовтень 2019 – лютий 2020.

Потенційні користувачі: користувачі стаціонарних ПК з встановленою ОС Windows.

Область дії автоматизована система обробка даних DiscoveryData.

Призначення програмного рішення: можливість користувачу, який є працівником фірми, швидко знаходити необхідні дані за певним критерієм серед великого обсягу даних та мати змогу їх переглядати, управляти ними.

Мета створення програмного рішення: опанувати сучасні технології для розробки програмних рішень, застосувати їх на практиці.

Скорочене найменування програмного модулю: «Find a Person by Name – Advanced», «Find a Company by Name – Advanced».

2. Структура проекту

Front-End: спливаюче вікно, при нажатті на текст, що містить силку в системі DiscoveryData.

Back-End: створений запит на стороні користувача формує SQL-запити, відправляючи їх на сервер бази даних та повертає результат користувачу.

Інформація, з якою працюватимуть користувачі програмного продукту, зберігатиметься в базі даних.

					ДТЕУ 121 06-01.БР		
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*	*Програмна компонента інтелектуальної системи формування запитів стейкхолдерів ІТ-галузі*	*Стадія*	*Аркуш*	*Аркушів*
Зав. каф.		Криворучко О.В.		23.12.22		ТЗ	41	40
Керівник		Десятко А.М.		23.12.22				
Гарант		Рзаєва С.Л.		23.12.22	*Технічне завдання*	*Факультет інформаційних технологій 4 курс, 6*		
Розробив		Андрусенко В.П.		23.12.22				

3. Функціональні вимоги до програмного модулю:

- введення даних користувачем;
- відправлення заповненої форми з даними на веб-сервер;
- оброблення даних на веб-сервері;
- сформування SQL-запитів і відправлення їх до сервера бази даних;
- відправлення отриманого результату назад у веб-браузер (користувачу)
- відображення отриманого результату.

МЕТОДИКА ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЯ

Тестування програмного забезпечення – це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки.

Тестування може надавати об’єктивну, незалежну інформацію про якість, ризики відмови, як для користувачів так і для замовників. [8, с. 175]

Для тестування тестування розробленого програмного модулю були використані різні інструменти. Спочатку тести писались у Visual Studio, після чого за допомого розподіленої системи керування версіями файлів – GIT, вони відравлялись на сервер. Також в роботі було застосовно інструмент Team City, який інтегрований з системою контролю версій.

Team City – це система управління побудовою застосунків і неперервної інтеграції, для тестування програмного забезпечення та організації ефективної колективної роботи над кодом. Зайшовши в TeamCity я бачу перелік різних категорій тестів. Згідно додатку Е.1, можна помітити такі види тестів: Regression, Smoke, Performance tests та інше., які відносяться до одного проекту Discovery Company Automation.[10, с. 36]

Regression tests – це тести, які покривають більшість функціоналу, графіки інтерфейсу. Їх запускають при ситуації, коли знайдено баг і необхідно переконатись чи не допущено в іншому місці помилки після того як виправили даний баг. Також регресійні тести можуть запускатися після релізу якгось певного функціоналу. Smoke tests – це тести,які перевіряють основний функціонал без якого

існування сайту, програмного продукту неможливо. Performance tests – це тести перевіряють продуктивність сайту, з якою швидкістю вони запускають при вході 100 людей на сайт, або ж навіть 1 000 000 людей.

Зайшовши в Team City та обравши певний вид категорій, ми клікаємо на кнопку Run (запустити), обираємо певний вид браузера, вписуємо певні параметри, якщо це необхідно і нажимо кнопку OK. Надалі Team City шукає вільну машину – агента, щоб на ній запустити категорію тестів. У результаті, після повного проходження тестів, ми бачимо результати прогону. [11, с. 189]

Згідно додатку Е.2 ми бачимо пройдені тести з яких 46 пройшли успішно та 4 не пройшли, дату прогону, тривалість прогону, номер прогону, хто відповідальний за цей прогон. У додатку Е.2 також продемонстровано приклад тесту, який впав, його назва та опис помилки, що трапилась.

Для тестування розробленого програмного модулю було створено ряд тестового покриття.

Згідно Додатку Д продемонстровано приклад створених Unit tests. Серед яких є два тести:

- ApplyFiltering_schould_filter_by_firstName (перевірка функціоналу фільтрації по імені)
- ApplyFiltering_schould_filter_by_lastName (перевірка функціоналу фільтрації по прізвищу)

Для написання цих тестів було використано об’єктно-орієнтовану мову програмування C# з використанням фреймворку Unit – тестування NUnit та середи розробки VisualStudio.

Кожен тест має атрибут [Test], що виділяє їх серед інших. Також такі тести можуть мати такі атрибути як [TestFixture] – до цього атрибуту відносять як певну групу тестів, набір, який відноситься до одного модулю наприклад; [SetUp] – цей атрибут розміщується перед методом, який повинен виконуватись перед кожним тестом і є загальний для всіх, наприклад, відкрити браузер та ввести логін і пароль;

[TearDown] – цей атрибут розміщується перед методом, який повинен виконуватись після кожного тесту і є загальний для всіх, наприклад сюди можна віднести закриття браузера чи очищення якихось полів.[7, с. 65]

Згідно структури фреймворку NUnit тести будуються за таким принципом, що мають містити 3 компоненти:

- Arrange (передумови) – ініціалізація тестових даних, попередні установки.
- Act (дія) – власне те, що тестується.
- Assert (перевірка) – що має бути внаслідок виконання дії.

Розглянувши Додаток Д, тести побудовані саме таким чином.

Також для тестування розробленого програмного модулю було створено UI tests – тести інтерфейсу програмного модулю.

Згідно Додатку Г продемонстровано приклад створених UI tests. Серед яких є три тести:

- Advanced Person Search Business Address All Filters Included Person Is Found – тест по порядку вводить дані у розроблену форму для персон, обирає чек-бокс Business Address і в результаті тест знаходить таку персону за задним фільтром.
- Advanced Person Search Home Address All Filters Included Person Is Found - тест по порядку вводить дані у розроблену форму для персон, обирає чек-бокс Home Address і в результаті тест знаходить таку персону за задним фільтром.
- Advanced Company Search All Filters Included Person Is Found - тест по порядку вводить дані у розроблену форму для компаній і в результаті тест знаходить таку компанію за заданим фільтром.

Для написання цих тестів було використано об’єктно-орієнтовану мову програмування C# з використанням фреймворку XUnit та середи розробки VisualStudio. Кожен тест має атрибут [Fact], що виділяє їх серед іншихта в даному атрибуті можна вказати ім’я даного тесту. Також ці тести мають такі атрибути як

[Trait] – цей атрибут, який використовуються для вказування до якої категорії відноситься цей UI тест. Пізніше, ми ці категорії будемо використовувати у TeamCity, що облегшує нашу роботу у розподіленні тестів на групи.

Тести побудовані за використання паттерну Page Object. Page Object - один з найбільш корисних і використовуваних архітектурних рішень в автоматизації. Даний шаблон проектування допомагає інкапсулювати роботу з окремими елементами сторінки, що дозволяє зменшити кількість коду і спростити його підтримку. Якщо, наприклад, дизайн одній зі сторінок змінений, то нам потрібно буде переписати тільки відповідний клас, що описує цю сторінку.

Згідно Додатку Г, можна помітити, що спочатку були створені класи – MainPage, AdvancedPersonSearchDialog та інші. Кожен з цих сторінок спочатку описує елементи для роботи з цією сторінкою використовуючи XPath або CSS локатори для пошуку елементів веб-сторінки, а тоді вже методи для роботи з цими елементами – це можна проглянути у додатку Е.4.

Отже, в ході виконання випускної кваліфікаційної роботи було створено тестове покриття розробленого модулю DiscoveryData. При розробці тестового покриття були проаналізовані, розглянуті та використані сучасні інструменти та технології. Серед них об’єктно-орієнтована мова програмування C# з використанням тестових фреймворків NUnit, XUnit та із застосуванням тестового паттерну PageObject. Розробка проводилась з використанням середовища Microsoft Visual Studio. Для запуску тестів, отримання звітності про проходження тестів, аналізу тестів,які не пройшли - було використано систему TeamCity, яка є одним із найкращих систем неперервної інтеграції.

ДОДАТКИ

Додаток А

```
<dc:dcwidget id="SearchDCWidget" clientidmode="Static" runat="server">
    <Content>
        <a id="ExpandSearchPanel" class="hidden" href="#">Show Search Panel</a>
        <a ID="CollapsiblePanelControls" href="#" >
            <img id="ExpandIcon" src="/Images/icons/expand.png"/>
            <img id="CollapseIcon" src="/Images/icons/collapse.png"/>
        </a>
        <div ID="CollapsiblePanelBody">
        <div ID="EntitySearchPanel" do-not-clear="true">
            <input id="PersonRadioButton" type="radio" class="search-entity-radio-button" name="SearchEntities" value="Person" autocomplete="off"/><label for="PersonRadioButton">Person</label>
            <input id="CompanyRadioButton" type="radio" class="search-entity-radio-button" name="SearchEntities" value="Company" autocomplete="off"/><label for="CompanyRadioButton">Company</label>
            <asp:TextBox ID="EntityNameTextbox" class="search-entity-text-box" runat="server" ClientIDMode="Static" title="First and Last Name or CRD/NPN"></asp:TextBox>
            <asp:Button ID="FindEntityByNameButton" runat="server" OnClick="FindEntityByNameButton_Click" Text="Find" CssClass="button-standard-green-medium VAlignMiddle" />
            <a href='#' class="search-entity-advance-search">Advanced Search by Name</a>
            <br />
            <asp:Label ID="ErrorLabel" runat="server" Text="No match found" Style="color: Red;" Visible="false"></asp:Label>
        </div>
        <div class="group-panel-container">
            <label class="search-entity-label">Group Search:</label>
            <div id="PersonTypePanel" class="group-panel"></div>

            <div id="CompanyTypePanel" class="group-panel"></div>
            </div>
        </div>

    </Content>
</dc:dcwidget>
```

```
<div id="dialogAdvancedSearchContacts" class="hidden advanced-search-dialog" title="Find a Person by Name - Advanced">
    <asp:Panel TabIndex="-1" ID="Panel3" runat="server">
        <p>
            Enter values in additional fields to further refine your search
        </p>
        <br />
        <table width="100%" class="NoWrap table-td-padding-tl-10">
            <tr>
                <td>
                    <asp:TextBox ID="txtFirstName" runat="server" title="First Name" ClientIDMode="Static"></asp:TextBox>
                </td>
                <td>
                    <asp:TextBox ID="txtMiddleName" runat="server" title="Middle Name" ClientIDMode="Static"></asp:TextBox>
                </td>
                <td>
                    <asp:TextBox ID="txtLastName" runat="server" title="Last Name" ClientIDMode="Static"></asp:TextBox>
                </td>
                <td></td>
            </tr>
            <tr>
                <td>
                    <asp:TextBox ID="txtFirmName" runat="server" title="Firm Name" ClientIDMode="Static"></asp:TextBox>
                </td>
                <td>
                    <asp:TextBox ID="txtCity" runat="server" title="City" ClientIDMode="Static"></asp:TextBox>
                </td>
                <td>
                    <asp:DropDownList runat="server" ID="ddlStates" class="combobox" ClientIDMode="Static"></asp:DropDownList>
                </td>
                <td>
                    <asp:TextBox ID="txtZipCode" runat="server" title="Zip Code" ClientIDMode="Static"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td colspan="4">
                    <asp:RadioButtonList ID="contactAddressTypeRadiobuttonList" runat="server" ClientIDMode="Static" RepeatDirection="Horizontal">
                        <asp:ListItem Text="Business Address" Value="Business" Selected="True" />
                        <asp:ListItem Text="Home Address" Value="Home" />
                    </asp:RadioButtonList>
                </td>
            </tr>
            <tr>
                <td colspan="4">
                    <select id="ddlBDType" class="" runat="server" clientidmode="Static" style="display: inline;">
                        <option value="-1">Type</option>
                    </select>
                     
                        <asp:TextBox ID="txtCRDContact" runat="server" title="CRD" ClientIDMode="Static"></asp:TextBox>
                </td>
            </tr>
        </table>
    </asp:Panel>
</div>
```

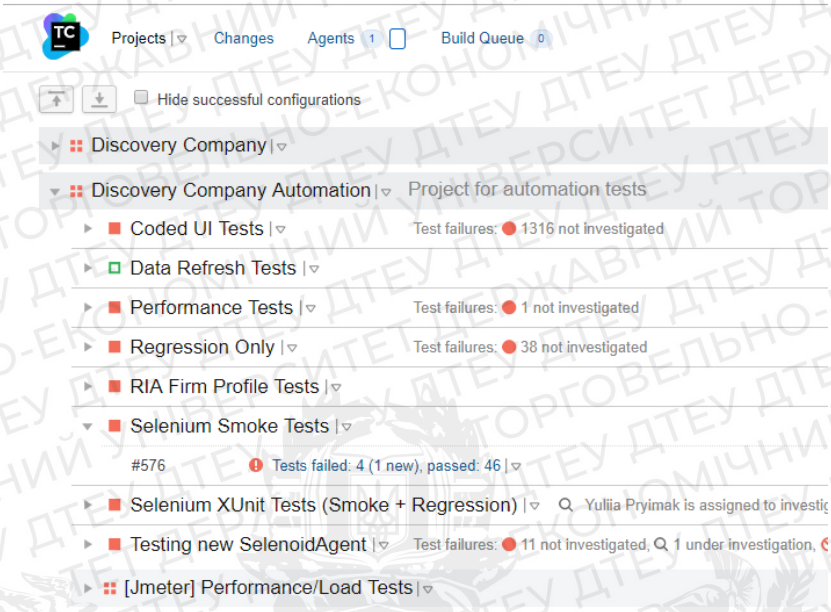
```
<div id="dialogAdvancedSearchCompanies" class="hidden advanced-search-dialog" title="Find a Company by Name - Advanced">
    <asp:Panel TabIndex="-1" ID="Panel4" runat="server">
        <p>
            Enter values in additional fields to further refine your search
        </p>
        <br />
        <table width="100%" class="NoWrap table-td-padding-tl-10">
            <tr>
                <td>
                    <asp:TextBox ID="txtAFCBNFirmName" runat="server" title="Firm Name" ClientIDMode="Static"></asp:TextBox>
                </td>
                <td>
                    <asp:TextBox ID="txtAFCBNCity" runat="server" title="City" ClientIDMode="Static"></asp:TextBox>
                </td>
                <td>
                    <asp:DropDownList runat="server" ID="ddlAFCBNStates" class="combobox" ClientIDMode="Static"></asp:DropDownList>
                </td>
                <td>
                    <asp:TextBox ID="txtAFCBNZipCode" runat="server" title="Zip Code" ClientIDMode="Static"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td colspan="4">
                    <select id="ddlAFCBNType" class="" runat="server" clientidmode="Static">
                        <option value="-1">Type</option>
                    </select>   
                        <asp:TextBox ID="txtAFCBNCRD" runat="server" title="CRD" ClientIDMode="Static"></asp:TextBox>
                </td>
            </tr>
        </table>
    </asp:Panel>
</div>

<script type="text/javascript">
    var searchPanelConfig = {
        preferredSearchEntity: "<%= PreferredSearchEntity%>",
        criteriaNonEmptyAndModified: function () { return $("#FilterContainer .filtergroup .emptyFilter").length === 0; },
        displayAreYouSureDialog: <%= DisplayAreYouSureDialog.ToString().ToLower() %>,
        findByNameButtonClientId: "<%= FindEntityByNameButton.UniqueID %>",
        txtCityClientID: '#<%=txtCity.ClientID %>',
        currentSearchType: "<%= CurrentSearchType %>",
        findEntityButtonUniqueId: "<%= FindEntityByNameButton.UniqueID %>"

};
</script>

<script src="/Scripts/source/features/search/legacy/SearchPanelControl.js?<% =DiscoveryCompany.Common.Utils.GetVersion() %>"></script>
```

```
public class ContactFilterer : IContactFilterer
{
    private readonly IContactFilteringStategyFactory _filteringStategyFactory;

    public ContactFilterer(IContactFilteringStategyFactory filteringStategyFactory)
    {
        _filteringStategyFactory = filteringStategyFactory;
    }

    public IQueryable<ContactInfo> ApplyFiltering(IQueryable<ContactInfo> query, ContactSearchCriteria searchCriteria)
    {
        if (!Utils.IsNullSearchParam(searchCriteria.FirstName))
        {
            query = query.Where(c => c.First.StartsWith(searchCriteria.FirstName.Trim()));
        }
        if (!Utils.IsNullSearchParam(searchCriteria.MiddleName))
        {
            query = query.Where(c => c.Middle.StartsWith(searchCriteria.MiddleName.Trim()));
        }
        if (!Utils.IsNullSearchParam(searchCriteria.LastName))
        {
            query = query.Where(c => c.Last.StartsWith(searchCriteria.LastName.Trim()));
        }

        var strategy = _filteringStategyFactory.CreateFilteringStrategy(searchCriteria.SearchType);

        query = strategy.ApplyTypeFiltering(query);

        if (!Utils.IsNullSearchParam(searchCriteria.Crd))
        {
            query = strategy.ApplyCrdFiltering(query, searchCriteria.Crd.Trim());
        }
        if (!Utils.IsNullSearchParam(searchCriteria.FirmName))
        {
            query = strategy.ApplyFirmNameFiltering(query, searchCriteria.FirmName.Trim());
        }

        if (searchCriteria.IsHomeAddress)
        {
            if (!Utils.IsNullSearchParam(searchCriteria.City))
            {
                query = strategy.ApplyHomeCityFiltering(query, searchCriteria.City.Trim());
            }
            if (!Utils.IsNullSearchParam(searchCriteria.State))
            {
                query = strategy.ApplyHomeStateFiltering(query, searchCriteria.State.Trim());
            }
            if (!Utils.IsNullSearchParam(searchCriteria.Zip))
            {
                query = strategy.ApplyHomeZipFiltering(query, searchCriteria.Zip.Trim());
            }
        }
        else
        {
            if (!Utils.IsNullSearchParam(searchCriteria.City))
            {
                query = strategy.ApplyBusinessCityFiltering(query, searchCriteria.City.Trim());
            }
            if (!Utils.IsNullSearchParam(searchCriteria.State))
            {
                query = strategy.ApplyBusinessStateFiltering(query, searchCriteria.State.Trim());
            }
            if (!Utils.IsNullSearchParam(searchCriteria.Zip))
            {
                query = strategy.ApplyBusinessZipFiltering(query, searchCriteria.Zip.Trim());
            }
        }

        return query;
    }
```

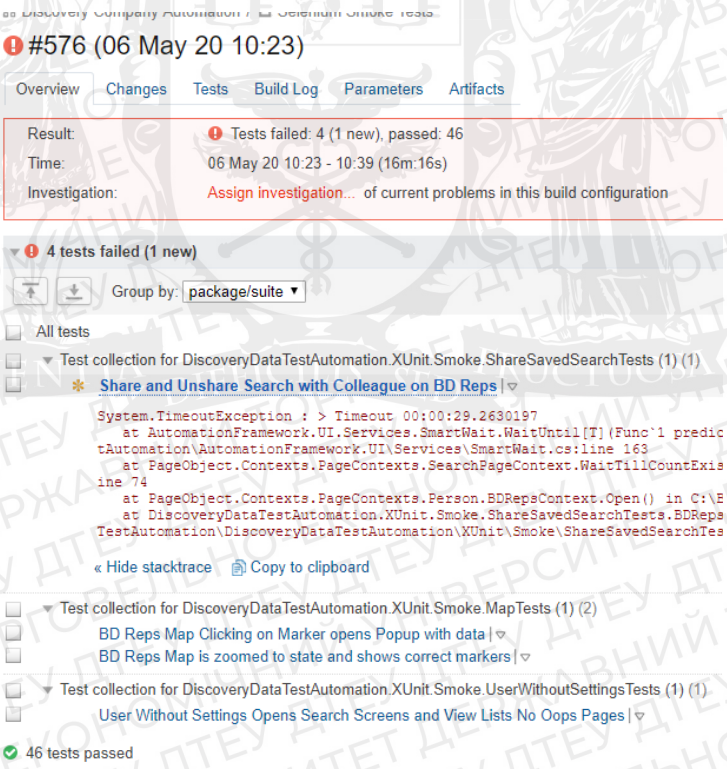
```
public class ContactQueryBuilder : IContactQueryBuilder
{
    private readonly Enterprise _context;
    private readonly IContactFilterer _contactFilterer;

    public ContactQueryBuilder(Enterprise context, IContactFilterer contactFilterer)
    {
        _context = context;
        _contactFilterer = contactFilterer;
    }

    public IQueryable<ContactInfo> FindContactsBySearchRequest(ContactSearchCriteria contactSearchCriteria)
    {
        var query = _context.ContactInfoes;
        return _contactFilterer.ApplyFiltering(query, contactSearchCriteria);
    }

    public IQueryable<ContactInfo> FindContactsByNameOrCrd(string val)
    {
        IQueryable<ContactInfo> query = _context.ContactInfoes;

        if (Utils.IsNullSearchParam(val)) return query;

        val = val.Trim();
        int crd;
        if (Int32.TryParse(val, out crd))
        {
            query = query.Where(c => c.CRD == crd || c.NPN == crd || c.TRFirmID == val);
        }
        else if (val.Any(Char.IsDigit))
        {
            query = query.Where(c => c.TRFirmID == val);
        }
        else
        {
            var parts = val.Split(' ').Where(s => !String.IsNullOrWhiteSpace(s)).ToArray();
            var first = parts[0];
            query = query.Where(c => c.First.StartsWith(first));
            if (parts.Count() > 1)
            {
                var last = parts[1];
                query = query.Where(c => c.Last.StartsWith(last));
            }
        }
        return query;
    }
```

```
public class AdvancedSearchTests : BaseXUnitTest
{
    public MainPageContext MainPage { get; }

    public DirectSearchResultsContext DirectSearchResults { get; }
    public ProfileContext Profile { get; }
    public AdvancedSearchTests()
    {
        MainPage = DiscoveryData.ContextService.GetPageContext<MainPageContext>();
        DirectSearchResults = DiscoveryData.ContextService.GetPageContext<DirectSearchResultsContext>();
        Profile = DiscoveryData.ContextService.GetPageContext<ProfileContext>();
    }

    [Fact(DisplayName = "Advanced Person Search Business Address All Filters Included Person Is Found")]
    [Trait(Category, TestCategory.Smoke)]
    [Trait(Component, TestComponent.Profile)]
    public void AdvancedPersonSearchBusinessAddress_AllFiltersIncluded_PersonIsFound()
    {
        var personName = new PersonName("Paul David", "Miller");
        MainPage.OpenAdvancedPersonSearchDialog();
        MainPage.AdvancedPersonSearchDialog.SelectBusinessAddress();
        MainPage.AdvancedPersonSearchDialog.EnterFirstName(personName.FirstName);
        MainPage.AdvancedPersonSearchDialog.EnterMiddleName(personName.MiddleName);
        MainPage.AdvancedPersonSearchDialog.EnterLastName(personName.LastName);
        MainPage.AdvancedPersonSearchDialog.EnterFirmName("CompanyExample");
        MainPage.AdvancedPersonSearchDialog.SelectCity("CityExample");
        MainPage.AdvancedPersonSearchDialog.SelectState(States.Illinois);
        MainPage.AdvancedPersonSearchDialog.EnterZipCode("60014");
        MainPage.AdvancedPersonSearchDialog.EnterType(AdvancedSearchDialogContainer.PersonType.BDReps);
        MainPage.AdvancedPersonSearchDialog.EnterCRD("5634721");
        MainPage.AdvancedPersonSearchDialog.ClickOk();
        DirectSearchResults.WaitUntilPageIsReady();
        DirectSearchResults.OpenFirstPersonProfile();
        Profile.EntityName.Should().Be(personName.ToString());
    }

    [Fact(DisplayName = "Advanced Person Search Home Address All Filters Included Person Is Found")]
    [Trait(Category, TestCategory.Smoke)]
    [Trait(Component, TestComponent.Profile)]
    public void AdvancedPersonSearchHomeAddress_AllFiltersIncluded_PersonIsFound()
    {
        var personName = new PersonName("Renu Jay", "Vora");
        MainPage.OpenAdvancedPersonSearchDialog();
        MainPage.AdvancedPersonSearchDialog.SelectHomeAddress();
        MainPage.AdvancedPersonSearchDialog.EnterFirstName(personName.FirstName);
        MainPage.AdvancedPersonSearchDialog.EnterMiddleName(personName.MiddleName);
        MainPage.AdvancedPersonSearchDialog.EnterLastName(personName.LastName);
        MainPage.AdvancedPersonSearchDialog.EnterFirmName("FirmNameExample");
        MainPage.AdvancedPersonSearchDialog.SelectCity("CityExample");
        MainPage.AdvancedPersonSearchDialog.SelectState(States.Texas);
        MainPage.AdvancedPersonSearchDialog.EnterZipCode("76034");
        MainPage.AdvancedPersonSearchDialog.EnterType(AdvancedSearchDialogContainer.PersonType.RIAReps);
        MainPage.AdvancedPersonSearchDialog.EnterCRD("5663446");
        MainPage.AdvancedPersonSearchDialog.ClickOk();
        DirectSearchResults.WaitUntilPageIsReady();
        DirectSearchResults.OpenFirstPersonProfile();
        Profile.EntityName.Should().Be(personName.ToString());
    }

    [Fact(DisplayName = "Advanced Company Search All Filters Included Person Is Found")]
    [Trait(Category, TestCategory.Smoke)]
    [Trait(Component, TestComponent.Profile)]
    public void AdvancedCompanySearch_AllFiltersIncluded_PersonIsFound()
    {
        const string companyName = "LEK Securities Corporation";
        MainPage.OpenAdvancedCompanySearchDialog();
        MainPage.AdvancedCompanySearchDialog.EnterFirmName(companyName);
        MainPage.AdvancedCompanySearchDialog.SelectCity("New York");
        MainPage.AdvancedCompanySearchDialog.SelectState("New York");
        MainPage.AdvancedCompanySearchDialog.EnterZipCode("10006");
        MainPage.AdvancedCompanySearchDialog.EnterType(AdvancedSearchDialogContainer.CompanyType.BrokerDealers);
        MainPage.AdvancedCompanySearchDialog.EnterCRD("234567");
        MainPage.AdvancedCompanySearchDialog.ClickOk();
        DirectSearchResults.WaitUntilPageIsReady();
        DirectSearchResults.OpenFirmProfile(companyName);
        Profile.EntityName.Should().Be(companyName);
    }
```

```
[Test]
public void ApplyFiltering_should_filter_by_firstName()
{
    //Arrange
    var searchCriteria = new ContactSearchCriteria
    {
        FirstName = "J"
    };

    _filteringStrategy.Setup(o => o.ApplyTypeFiltering(It.IsAny<IQueryable<ContactInfo>>()))
        .Returns((IQueryable<ContactInfo> o) => o);

    //Act
    var actual = _target.ApplyFiltering(_contacts.AsQueryable(), searchCriteria);

    //Assert
    actual.ShouldAllBeEquivalentTo(new[] {_contacts[0], _contacts[2], _contacts[3]});
    _filteringStrategy.Verify(o => o.ApplyTypeFiltering(It.IsAny<IQueryable<ContactInfo>>()));
}

[Test]
public void ApplyFiltering_should_filter_by_lastName()
{
    //Arrange
    var searchCriteria = new ContactSearchCriteria
    {
        LastName = "B"
    };

    _filteringStrategy.Setup(o => o.ApplyTypeFiltering(It.IsAny<IQueryable<ContactInfo>>()))
        .Returns((IQueryable<ContactInfo> o) => o);

    //Act
    var actual = _target.ApplyFiltering(_contacts.AsQueryable(), searchCriteria);

    //Assert
    actual.ShouldAllBeEquivalentTo(new[] { _contacts[1], _contacts[4]});
    _filteringStrategy.Verify(o => o.ApplyTypeFiltering(It.IsAny<IQueryable<ContactInfo>>()));
}
```



Е.1. Перший рисунок додатка Е



Е.2. Другий рисунок додатка Е

```
public class MainPage : BasePage
{
    public MainPage()
    {
        UrlPathName = "";
    }

    [FindBy(How.Id, "print-body")]
    public FrozenPanelContainer FrozenPanel { get; set; }

    [FindBy(How.Id, "tabs")]
    public FooterTabsContainer FooterTabs { get; set; }

    [FindBy(How.CssSelector, ".ui-dialog.ui-widget.ui-widget-content.ui-corner-all.ui-front.ui-dialog-button
    public OkCancelDialogContainer OkCancelDialog { get; set; }

    [FindBy(How.CssSelector, ".ui-dialog[aria-describedby='TemplateSelectDialog']")] [IsDisplayed]
    public SelectDownloadTemplateDialogContainer SelectDownloadTemplateDialog { get; set; }

    [FindBy(How.CssSelector, ".ui-dialog[aria-describedby='TemplatePreviewDialog']")] [IsDisplayed]
    public TemplatePreviewDialogContainer TemplatePreviewDialog { get; set; }

    [FindBy(How.CssSelector, ".ui-dialog[aria-describedby='TemplateEditDialog']")] [IsDisplayed]
    public TemplateEditDialogContainer TemplateEditDialog { get; set; }

    [FindBy(How.CssSelector, ".ui-dialog[aria-describedby='TemplateEditDialog']")]
    [IsDisplayed]
    public CreateNewTemplateDialogContainer CreateNewTemplateDialog { get; set; }

    [FindBy(How.CssSelector, ".ui-dialog[aria-describedby='dialogAdvancedSearchContacts']")] [IsDisplayed]
    public AdvancedPersonSearchDialogContainer AdvancedPersonSearchDialog { get; set; }
```

Е.3 . Третій рисунок додатка Е