## НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**

Навчально-науковий інститут атомної та теплової енергетики Кафедра інженерії програмного забезпечення в енергетиці

"На правах рукопису" УДК 004.4

«До захисту допущено» В.о. зав. кафедри

Олександр КОВАЛЬ

“ ” 2022 р.

## Магістерська дисертація

За освітньою програмою **«**Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій»

Спеціальності 121 Інженерія програмного забезпечення

на тему: «Програмний застосунок реалізації сценарію аналізу рівня міжнародного співробітництва наукової організації в науково-технічній сфері на основі веб-сервісів»

Виконав: студент 2 курсу, групи ТВ-12мп

Лелет Євгеній Олегович

(прізвище, ім’я, по батькові) (підпис)

Науковий керівник д.т.н., доцент Коваль О. В.

(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент к.т.н.Сенченко В'ячеслав Родіонович

(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Київ - 2022

## Національний технічний університет України

**«Київський політехнічний інститут ім. Ігоря Сікорського»**

Навчально-науковий інститут атомної та теплової енергетики Кафедра інженерії програмного забезпечення в енергетиці Рівень вищої освіти другий, магістерський

За освітньою програмою «Інженерія програмного інтелектуальних кібер- фізичних систем і веб-технологій»

Спеціальності 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

в.о. зав. кафедри

Олександр КОВАЛЬ

(підпис)

« » 202\_р.

# З А В Д А Н Н Я

**НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ**

Лелету Євгенію Олеговичу

(прізвище, ім’я, по батькові)

1. Тема дисертації: «Програмний застосунок реалізації сценарію аналізу рівня міжнародного співробітництва наукової організації в науково-технічній сфері на основі веб-сервісів»

Науковий керівник д.т.н., доцент Коваль О. В.

( прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “07”листопада 2022 року №4067-с

1. Строк подання студентом дисертації 9 грудня 2022 року
2. Вхідні дані до роботи: інформаційна система реалізації аналізу рівня міжнародного наукового співробітництва в науково-технічній сфері на основі веб-сервісів.
3. Перелік питань, які потрібно розробити: розглянути та проаналізувати існуючі системи; дослідити методи побудови систем з використанням мікросервісної архітектури; розробити систему що буде задовольняти всі потреби та покривати всі поставлені задачі.
4. Орієнтований перелік ілюстративного матеріалу: огляд існуючих систем, схеми архітектур використані в роботі, моделі баз даних, тестування системи.
5. Орієнтований перелік публікацій: відсутні.
6. Дата видачі завдання «01» жовтня 2022 р.

## КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів виконання магістерської дисертації | Строки виконання етапів магістерської дисертації | Примітка |
| 1 | Отримання завдання | 01.10.2021 | виконано |
| 2 | Дослідження предметної області | 01.10.2021 - 01.11.2021 | виконано |
| 3 | Постановка вимог до проєктування системи | 01.11.2021 - 01.15.2021 | виконано |
| 4 | Дослідження існуючих рішень | 15.11.2021 - 01.12.2021 | виконано |
| 5 | Підготовка публікацій | 01.12.2021 - 09.12.2021 | виконано |
| 6 | Розробка програмного продукту | 05.03.2022 - 07.10.2022 | виконано |
| 7 | Тестування | 07.10.2022 - 11.11.2021 | виконано |
| 8 | Захист програмного продукту | 17.10.2022 - 21.10.2022 | виконано |
| 9 | Підготовка магістерської дисертації | 22.10.2022 - 20.11.2022 | виконано |
| 10 | Передзахист | 21.11.2022 - 25.11.2022 | виконано |
| 11 | Захист | 19.12.2022 - 23.12.2022 | виконано |
|  |  |  |  |

Студент Лелет Є. О.

( підпис ) (прізвище та ініціали)

Науковий керівник Коваль О. В.

( підпис ) (прізвище та ініціали)

# РЕФЕРАТ

**Актуальність**. Всі завдання розвитку інформаційних технологій в науці з’являються від проблем збору та організації доступу до масивів даних, труднощів розвитку та роботи з комплексними моделями даних, необхідністю обчислень (що здійснюються за допомогою комп'ютерних систем високої продуктивності або за рахунок розподіленої мережі обчислень) і розуміння того, що рішення багатьох наукових проблем знаходиться на стику наукових дисциплін. Наприклад, одним з таких завдань є знаходження нових фактів та інформації (оптимальних рішень) на основі існуючих даних. Вирішити дану проблему може покращення та застосування технології пошуку на основі програмного застосунку реалізації сценарію аналізу рівня міжнародного співробітництва наукової організації в науково-технічній сфері на основі веб-сервісів.

**Об'єктом досліджень** є процес аналізу рівня міжнародного співробітництва наукової організації в науково-технічній сфері.

**Предметом дослідження** є програмний застосунок реалізації аналізу рівня міжнародного співробітництва наукової організації в науково-технічній сфері на основі веб-сервісів.

**Методи дослідження**: що застосовуються як на теоретичному, так і на емпіричному рівнях (абстрагування, аналіз, та синтез).

**Метою роботи** є розробка програмного застосунку реалізації аналізу рівня міжнародного співробітництва наукової організації в науково-технічній сфері на основі веб-сервісів.

Для виконання поставленої мети, з огляду на об’єкт та предмет досліджень виокремлено такі **завдання**:

* Дослідження сутності міжнародного співробітництва наукової організації в науково-технічній сфері
* Аналіз існуючих систем та виділення їх сильних та слабких сторін
* Реалізація системи аналізу рівня міжнародного співробітництва наукової організації в науково-технічній сфері
* Перевірка працездатності створеної системи

**Практичне значення** даного дослідження полягає в підвищенні ефективності взаємодії міжнародних наукових організацій в науково-технічній сфері.

**Структура і обсяг дипломної роботи**: Магістерська дисертація складається зі вступу, п’яти розділів, висновків та 1 додатку. В роботі містяться посилання на 14 джерел та 13 ілюстрацій. Повний обсяг роботи 118 с.

**Ключові слова:** веб-сервіс, C#, .NET., AWS, WEB

# ABSTRACT

**Topicality.** All the tasks of information technology development in science arise from the problems of collecting and organizing access to data sets, difficulties in developing and working with complex data models, the need for calculations (performed using high-performance computer systems or distributed network computing) and understanding that the solution of many scientific problems is at the intersection of scientific disciplines. For example, one such task is to find new facts and information (optimal solutions) based on existing data. This problem can be solved by the improvement and application of search technology based on the software application of the scenario of the analysis of the level of international cooperation of a scientific organization in the scientific and technical sphere based on web services.

**The object of research** is the process of analyzing the level of international cooperation of a scientific organization in the scientific and technical sphere.

**The subject of research** is a software application for the implementation of the analysis of the level of international cooperation of a scientific organization in the scientific and technical sphere based on web services.

**Research methods:** analysis (semantic search tasks), abstraction and synthesis (selection of the best technologies for a given task), formalization (construction of a mathematical conceptual model), experiment (includes modeling of the search engine), comparison (currently developed system with existing ones).

**The aim of the work** is the creation of a search engine using semantic search in a given subject area.

To achieve this goal, given the object and subject of research, the following tasks are identified:

* Study of the essence of international cooperation of a scientific organization in the scientific and technical sphere
* Analysis of existing systems and identification of their strengths and weaknesses
* Implementation of the system of analysis of the level of international cooperation of a scientific organization in the scientific and technical sphere
* Checking the functionality of the created system

**The practical significance** of this study is to increase the efficiency of interaction of international scientific organizations in the scientific and technical sphere. **Structure and volume of work:** The master's dissertation consists of an introduction, five chapters, conclusions and 1 appendix. The paper contains references

to 14 sources and 13 illustrations. The full volume of work is 118 p..

**Keywords:** web service, C#, .NET., AWS, WEB

# ЗМІСТ

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 9](#_TOC_250024)

[ВСТУП 10](#_TOC_250023)

* 1. [ВЕБ-СЕРВІСИ ТА ПРИНЦИПИ ЇХ ВИКОРИСТАННЯ 12](#_TOC_250022)
     1. [Базові поняття веб-сервісів 12](#_TOC_250021)
     2. [Практичне використання веб-сервісної архітектури 13](#_TOC_250020)

[Висновки до розділу 1 14](#_TOC_250019)

* 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ПОСТАВЛЕНОЇ ЗАДАЧІ 16
     1. [Аналіз аналогічних застосунків 16](#_TOC_250018)
        1. [Аналіз EC2 Instances AWS 18](#_TOC_250017)
        2. [Аналіз VPC AWS 26](#_TOC_250016)
        3. [Аналіз S3Bucket AWS 27](#_TOC_250015)
        4. [Аналіз DynamoDB AWS 28](#_TOC_250014)
        5. [Аналіз Elastic Beanstalk AWS 31](#_TOC_250013)

[Висновки до розділу 2 33](#_TOC_250012)

* 1. [ОПИС ПОБУДОВАНОЇ СИСТЕМИ 34](#_TOC_250011)
     1. [Загальний опис системи 34](#_TOC_250010)
     2. [Архітектура системи 34](#_TOC_250009)
     3. [Особливості реалізації системи 35](#_TOC_250008)
     4. [Схема взаємодії клієнтської та серверної частини 36](#_TOC_250007)

[Висновки до розділу 3 37](#_TOC_250006)

* 1. ОПИС ПРОГРАМНИХ ЗАСОБІВ РОЗВ’ЯЗАННЯ ЗАДАЧІ 38
     1. Обґрунтування вибору мови програмування 38
     2. Обґрунтування вибору бази даних 43
     3. Обґрунтування вибору фреймворку 48

[Висновки до розділу 4 51](#_TOC_250005)

* 1. РОЗРОБКА СТАРТАПУ ПРОЄКТА 52
     1. Описання ідеї проєкту 52
     2. Технологічний аудит ідеї проєкту 53
     3. Аналіз ринкових можливостей запуску стартап проєкт 54
     4. Рекомендації на рахунок подальших удосконалень 55

[Висновки до розділу 5 58](#_TOC_250004)

[ВИСНОВКИ 59](#_TOC_250003)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 61](#_TOC_250002)

[ДОДАТОК А 63](#_TOC_250001)

[ДОДАТОК Б 105](#_TOC_250000)

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

|  |  |
| --- | --- |
| ПЗ | програмне забезпечення |
| ПК | персональний комп’ютер |
| IDE | Integrated Drive Electronics |
| АРІ | Application Programming Interface |
| HTTP | HyperText Transfer Protocol |
| JSON | JavaScript Object Notation |
| DOM | Document Object Model |
| UI | User Interface |
| DB | Database |
| AWS | Amazon Web Services |

# ВСТУП

Оптимізація процесів взаємодії міжнародних наукових організацій в науково-технічній сфері є актуальною проблемою в науковому світі. Конкуренція в розробці динамічних інструментів є основою щоб шукати нові високотехнологічні рішення.

На сьогоднішній день технології дають можливість розробляти та успішно використовувати дуже якісні інструменти. Але також на сьогоднішній день немає необхідності в зберіганні інформації на твердих носіях, усе що необхідно у будь-який момент та у будь якій точці світу можно знайти в мережі інтернет. Великі технологічні корпорації мають можливість швидко та якісно прийняти до використання нову алгоритмічну систему лише за допомогою сучасного, створеного за всіма правилами сучасної розробки програмного забезпечення. В такому випадку процес навчання співробітників використовувати даний функціонал буде достатньо швидким.

Слід зазначити, що ми входимо в поле нової комунікації. Необхідно окреслити нову теоретичну територію, а також уточнити її методології, будувати підходи на основі знань, які враховують трансформації в цифрова комунікаційна екосистема. Для раптово, щоб почати транзит теорії функціоналістична комунікація до такої, яка рефлексує та пояснює контексти комунікації опосередкований комп'ютером. Аналізувати та розмірковувати про інтерактивне цифрове спілкування, в лінія, яка відновлює концепцію активної аудиторії, функціоналізовану досі впливами засобів масової інформації. Ми в новому комунікаційними середовищі з пристроями, платформами і програми, які зустрічаються в Інтернеті, також викликають потребу в синтезі, який описує, саме ці нові сценарії.

Основною метою дипломного проекту є створення програмного застосунку реалізації сценарію аналізу рівня міжнародного співробітництва наукової організації в науково-технічній сфері на основі веб-сервісів.

# ВЕБ-СЕРВІСИ ТА ПРИНЦИПИ ЇХ ВИКОРИСТАННЯ

## Базові поняття веб-сервісів

Веб-сервіси — це модульні автономні програми, які можна описувати, публікувати, розміщувати та викликати через мережу.

Сервер програм підтримує веб-служби, розроблені та реалізовані відповідно до специфікації веб-служб для Java™ EE (платформа Java, Enterprise Edition). Сервер програм підтримує моделі програмування JAX-WS (API Java для веб-служб XML) та JAX-RPC (API Java для RPC на основі XML). JAX-WS - це стратегічна модель програмування, яка спрощує розробку програм за рахунок підтримки стандартної моделі на основі анотацій для розробки програм та клієнтів веб-сервісів.

Загалом сценарії веб-служб бізнес-програма запитує службу в іншої програми. Запит обробляється через задану веб-адресу з використанням повідомлень SOAP через HTTP, транспорт JMS (служба повідомлень Java) або безпосередньо викликається як EJB (Enterprise JavaBeans). Служба отримує запит, обробляє його та повертає відповідь. Звіти про погоду та котирування акцій є прикладами найпростіших веб-сервісів. Виклик методу є синхронним, тобто метод очікує, доки не буде доступний результат. Бронювання авіаквитків або заповнення онлайн-замовлень на покупку є прикладами послуг, що підходять для пропозицій, операцій «бізнес-бізнес» (B2B) або «бізнес-клієнт» (B2C).

Веб-сервіси можуть включати саму службу або клієнта, який отримує доступ до служби.

## Практичне використання веб-сервісної архітектури

Веб-сервіси — це веб-програми, які дозволяють підвищити гнучкість ваших бізнес-процесів, інтегруючи їх із програмами, які інакше не спілкувалися б. Внутрішня бібліотечна програма видачі у вашій місцевій бібліотеці є гарним прикладом концепції веб-служб і того, як вона розвивалася. Поняття веб-сервісів існувало ще до появи самого терміна. Ця концепція закріпилася з народженням Інтернету. До появи Інтернету потрібно було йти в бібліотеку, шукати колекції, а потім брати книги. Якщо ви не могли знайти потрібну книгу, бібліотекар шукав книгу за допомогою комп’ютера чи телефону в сусідній бібліотеці. Бібліотекар замовив би книгу для вас, і ви б забрали її після доставки до вашої місцевої бібліотеки. Завдяки застосуванню веб-сервісів ви можете спростити відвідування бібліотеки.

Тепер ви можете шукати в локальній бібліотеці, а також в інших локальних бібліотеках одночасно. Коли інші бібліотеки пропонують вашій бібліотеці веб-службу для пошуку їхньої колекції (послуга, можливо, надається через Universal Description Discovery and Integration (UDDI)), ваші результати враховують їхні ресурси. Ви можете скористатися іншою програмою веб-служб і надіслати книгу додому. Використання програм веб-служб економить ваш час. Це також звільняє бібліотекаря від певних завдань, що дозволяє йому присвятити себе іншим завданням, притаманним його обов'язкам.

Веб-сервіси є реалізацією підходу до програмування так званої

«сервісно-орієнтованої архітектури» (SOA). Ідея полягає в тому, щоб створювати або генерувати програми шляхом виявлення та впровадження послуг, доступних у мережі, або шляхом виклику програм, які вже доступні для виконання завдання. Веб-сервіси забезпечують взаємодію, наприклад, програми веб-сервісів надають компоненти, створені різними мовами програмування, для взаємодії так, ніби вони

були створені за допомогою однієї мови. Щоб викликати свою реалізацію, веб-сервіси покладаються на існуючі транспортні технології, такі як HTTP, і стандартні методи кодування даних, такі як Extensible Markup Language (XML).

Можна сказати, що це стандартизоване середовище для розподілу зв'язку між клієнтськими та серверними програмами у всесвітньому павутинні . Його можна редагувати та є пакети оновлень.

Послуги AWS надаються клієнтам через мережу серверних ферм AWS, розташованих по всьому світу. Комісія базується на комбінації використання, апаратного забезпечення, операційної системи, програмного забезпечення або мережевих функцій, вибраних абонентом, необхідної доступності, резервування, безпеки та параметрів обслуговування. Абоненти можуть заплатити за один віртуальний комп’ютер AWS, виділений фізичний комп’ютер або їх кластери. Amazon надає певні частини безпеки для передплатників, тоді як інші аспекти безпеки є відповідальністю передплатника . AWS працює в багатьох глобальних географічних регіонах, включаючи сім у Північній Америці.

Крім того, веб-сервіс можна визначити як програмний модуль, призначений для виконання певного набору завдань. Стовпи програми можна знайти в Інтернеті, і вони також можуть бути названі відповідно. Під час дзвінка веб-служба може надавати функціональні можливості клієнту, який запитує цей портал. Все це відбувається за кілька секунд.

## Висновки до розділу 1

В даному розділі були розглянуті базові поняття веб-сервісів, які є вкритично важливими в розробці програмного продукту для магістерської роботи. Завдання, яке прагне вирішити програмний продукт, полягає в тому, щоб

покращити якість аналізу міжнародного наукового співробітництва в науково-технічній сфері.

# ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ПОСТАВЛЕНОЇ

**ЗАДАЧІ**

## Аналіз аналогічних застосунків

На сьогоднішній день не було виявлено існуючих зразків рішень поставленої задачі, а саме програмних застосунків реалізації сценарію аналізу рівня міжнародного співробітництва наукової організації в науково-технічній сфері, тому як зразок було обрано Amazon Web Services.

Amazon Web Services є постачальником хмарних послуг, він дозволяє нам мати сховище, обчислювальні ресурси, мобільні програми, бази даних та багато іншого в режимі хмарних обчислень.

Якщо вам потрібні будь-які з цих ресурсів, ви можете укласти контракт з Amazon Web Services, щоб отримати ці ресурси без необхідності вкладати кошти в інфраструктуру.

Іншими словами, щомісячна передплата на AWS оплачуватиметься за підключення до сервера/бази даних/сервісу, де оброблятиметься необхідна вам інформація, замість вкладати кошти в потужні та дорогі комп'ютери.

Ця перевага як для малого та середнього бізнесу, так і для підприємств, які перебувають у ранньому віці і не в змозі зробити великі інвестиції в інфраструктуру, а також для тих, хто, навіть маючи можливість зробити інвестиції, вирішує відкласти їх або віддає перевагу платному режиму. per use , аналогічний перевагам між покупкою автомобіля та орендою компанії. Таким чином, вони можуть перетворити їх на змінні витрати, які збільшуватимуться лише в тому випадку, якщо їм потрібно більше потужностей, тобто у разі зростання бізнесу.

Крім того, у разі наявності власної інфраструктури, AWS може стати вашим захистом, дозволяючи у разі збою вашого основного ЦПД за лічені хвилини переключити всі програми та користувачів на свою хмарну інфраструктуру, зберігши безперервність бізнесу.

AWS є у 190 країнах світу і має понад 10 000 асоційованих партнерів, включаючи Hiberus Sistemas. Його центри обробки даних розташовані в США, Японії, Австралії, Бразилії, Європі та Сінгапурі.

Хоча можливість обробки даних у зовнішній інфраструктурі може здатися дуже цікавою та вигідною з погляду витрат, проблема безпеки може викликати занепокоєння. AWS має сертифікати та аудити, такі як: PCI DSS level 1, FISMA Moderate, HIPAA AND SOC 1, ISO 27001 та аудит SOC 2, що робить його надійним на 100%.

Для сучасного бізнесу дуже важливо мати швидкий та надійний доступ до даних, для цього їм необхідно їх зберігати та керувати ними. AWS забезпечує доступ до таких баз даних, як MySQL, Oracle, Aurora, PostgreSQL, SQL Server, MongoDB.

Завдяки підключенню до хмари підприємствам не потрібно буде вкладати великі кошти в інфраструктуру, тому їх витрати будуть нижчими. Крім того, AWS вдалося стати все більш ефективною, скориставшись ефектом масштабу та зумівши скоротити свої витрати та перенести цю економію на витрати своїх клієнтів.

Одним з найбільш важливих аспектів AWS є те, наскільки легко отримати доступ до нього і знайти сервіси, здатні впоратися з необхідним робочим навантаженням. Крім того, AWS швидко адаптується до потреб ринку, тому AWS постійно з'являються нові сервіси.

Хоча AWS є постачальником інфраструктури, компанія, як і раніше, може і легко проводить аудит, контроль та управління, щоб забезпечити належне

управління своїми даними. Це дозволяє партнерам швидко отримувати доступ до інформації та, отже, діяти швидше.

AWS CPD призначені для продовження роботи в найнепередбачуваніших обставинах. Інфраструктури можуть бути спроектовані з урахуванням часу безвідмовної роботи 99,99999% (час простою менше 4 секунд на рік). Логотип Amazon Web Services зображено на рисунку 2.1.



Рисунок 2.1 - Логотип Amazon Web Services (AWS)

### Аналіз EC2 Instances AWS

Amazon Elastic Compute Cloud, більш відомий як Amazon EC2, безсумнівно, є одним із флагманських сервісів AWS. Це веб-сервіс, який надає компаніям обчислювальну потужність, необхідну їм для безпечного та гнучкого створення додатків у хмарі та управління ними.

Публічну бета-версію Amazon EC2 було запущено понад 15 років тому, а саме 25 серпня 2006 року. З того часу сервіс не переставав розвиватися, додаючи нові можливості для всіх типів клієнтів, від невеликих компаній з розробки або адміністрування систем до багатонаціональних гігантів, таких як Netflix, Airbnb або Volkswagen.

На сьогоднішній день Amazon EC2, безсумнівно, є одним із найбільших сервісів на ринку хмарних обчислень, оскільки понад 326 000 компаній у всьому світі використовують його сервіси віртуальних машин у хмарі. Число, яке не

перестає зростати, оскільки впровадження хмарних обчислень зростає у всіх секторах галузі.

Amazon EC2 пропонує можливість створювати гнучкі обчислювальні середовища з використанням віртуальних машин у хмарі. Користувачі підключаються до хмарних серверів Amazon через веб-інтерфейс, який дозволяє створювати та керувати екземплярами образів машин, званих AMI (Amazon Machine Image).

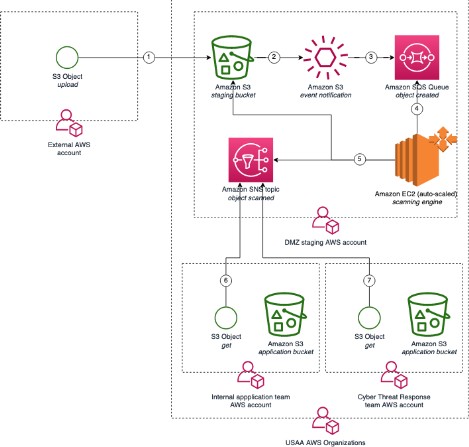
У кожному AMI можна запустити ту операційну систему, яку хоче користувач, а також встановити необхідне ПЗ та залежності. Віртуалізація серверів Amazon відома як інстанси, і користувачі можуть настроювати різні інстанси, від невеликих віртуальних машин з базовою продуктивністю до величезних машин з високою продуктивністю. Архітектурну діаграму EC2 AWS зображено на рисунку 2.2.

Рисунок 2.2 - Архітектурна діаграма EC2 Instances AWS

Таким чином, користувачі Amazon EC2 можуть вільно створювати, змінювати або видаляти екземпляри, не турбуючись про фізичну інфраструктуру або її адміністрування. В обмін користувач вносить плату за свої активні сервери, постійно контролюючи вартість і маючи можливість змінювати екземпляри без

обмежень, як вгору, так і вниз, звідси термін «еластичний», що дав назву Amazon EC2.

Amazon EC2 пропонує попередньо налаштовані шаблони інстансів, які спрощують користувачам налаштування та запуск своїх віртуальних машин. Кожен тип інстансу має різні характеристики: ЦП, пам'ять, сховище, швидкість мережі тощо. Ось деякі з найвідоміших:

* загальне призначення - екземпляри загального призначення, такі як T2, T3 або T4g, пропонують збалансоване поєднання обчислювальних ресурсів. Зазвичай вони запускаються в службах, які використовують ресурси в рівних частках, таких як веб-сервери або репозиторії коду.
* обчислення: екземпляри, такі як C4, C5 або C6g, оптимізовані для додатків, що потребують високопродуктивних процесорів. Вони використовуються для веб-серверів, які повинні підтримувати високий трафік, мультимедійних або ігрових серверів, наукового моделювання та інших програм з інтенсивним використанням.
* Пам'ять: іншими екземплярами, що найчастіше використовуються, є екземпляри R4, R5 або R6g. Вони характеризуються дуже високою швидкістю завантаження пам'яті, і навіть гарною швидкістю мережі. Вони зазвичай використовуються в аналітиці великих даних та баз даних у реальному часі.
* сховище: екземпляри D2, D3 або D3en оптимізовані для робочих навантажень, яким необхідно виконувати високорівневі операції читання та запису. Вони використовуються для великих баз даних або зберігання даних в оперативній пам'яті.
* прискорені обчислення - обчислювальні інстанси з прискоренням P2, P3 або P4 забезпечують високу продуктивність виконання великих операцій з використанням апаратних прискорювачів і співпроцесорів.

Вони використовуються, наприклад, у додатках машинного навчання, розпізнавання голосу, автономних транспортних засобів або сейсмічного аналізу серед багатьох інших додатків.

Amazon EC2 пропонує безліч переваг компаніям, які вирішили його використовувати. Ось деякі з найвидатніших переваг:

* Гнучкість. Як випливає із назви, Amazon Elastic Compute Cloud пропонує можливість змінювати екземпляри у будь-який час. Ви можете запускати одну, сотні або тисячі одночасно і без обмежень, як збільшувати, так і зменшувати пільги відповідно до ваших потреб.
* Контроль: користувач завжди контролює свої екземпляри. Користувач зможе керувати всіма своїми віртуальними машинами зі своєї панелі керування, маючи доступ у будь-який час із будь-якого місця завдяки своєму веб-інтерфейсу.
* Pay As You Go: Amazon пропонує плату за використання за хвилини. Користувач завжди знає свої щомісячні витрати у режимі реального часу. Таким чином компанії можуть легко контролювати свої витрати на хмарні обчислення.
* Економія: якщо ми порівнюємо ціну Amazon EC2 із традиційними обчисленнями, ми побачимо, наскільки значною є економія. Немає необхідності купувати сервери чи керувати ними. Не дбайте про його адміністрування або обслуговування.
* Зручність. Дивовижна особливість хмарних обчислень полягає в тому, що ви можете отримати доступ до нескінченної кількості обчислювальних ресурсів без їх управління та адміністрування. Це дозволяє компаніям долати бар'єри та підвищувати свою ефективність.
* Безпека. Amazon EC2 забезпечує надійну та безпечну мережу, тому ваші найважливіші програми та дані завжди будуть у безпеці у хмарі.

Крім того, ви можете захистити конфіденційність екземплярів, контролювати доступ до мережі або створювати групи безпеки.

* Довіра. Amazon EC2 забезпечує безвідмовну роботу на рівні 99,95% і забезпечує повністю надійне, швидке та передбачуване середовище віртуалізації.
* Масштабованість: використання хмари Amazon EC2 адаптується до потреб компаній, які можуть збільшувати або зменшувати кількість використовуваних ресурсів відповідно до своїх цілей та потреб у будь-який момент часу.
* Простота використання: користувачам Amazon EC2 не потрібно мати високі технічні знання з адміністрування серверів. Інстансами легко керувати завдяки інтуїтивно зрозумілому інтерфейсу.
* Міграція. Крім того, пропонуються різні інструменти міграції в хмару, які дозволяють ефективно та практично переносити програми, дані та служби.

Логотип AWS EC2 зображено на рисунку 2.3.



Рисунок 2.3 - Логотип Amazon EC2

### Аналіз VPC AWS

VPC (Virtual Private Cloud) або віртуальна приватна хмара відноситься до набору обчислювальних ресурсів, які можна налаштувати на запит і які розташовані в середовищі загальнодоступної хмари.

Віртуальна приватна хмара також пропонує можливість використовувати переваги віртуалізованої мережі, водночас використовуючи переваги загальнодоступних хмарних функцій, таких як самообслуговування, масштабованість та еластичність.

Віртуальна приватна хмара характеризується гарною службою безпеки, що гарантує захист ваших підмереж, мережевих шлюзів, вашого віртуального мережевого середовища і навіть ваших IP-адрес. Крім того, він дозволяє шифрувати відправляється та одержувану інформацію.

Крім того, воно відрізняється тим, що є середовищем, що адаптується до ваших потреб і вимог, що дозволяє вам проектувати і налаштовувати архітектуру вашої хмари в залежності від того, чого ви хочете з її допомогою досягти.

Ще одна характеристика віртуальної приватної хмари полягає в тому, що вона не контактує з іншими хмарами на мережному рівні, тому користувач має більший контроль над своїми даними і може запобігти їх змішування з даними загальнодоступної хмари або доступ третьої сторони. може отримати доступ до них.

Користувач також може використовувати ресурси, що пропонуються VPC, для визначення пріоритетів мережевого трафіку певних додатків, уникаючи навантаження і допомагаючи підвищити їхню продуктивність. Приклад роботи Amazon VPC наведено на рисунку 2.4.

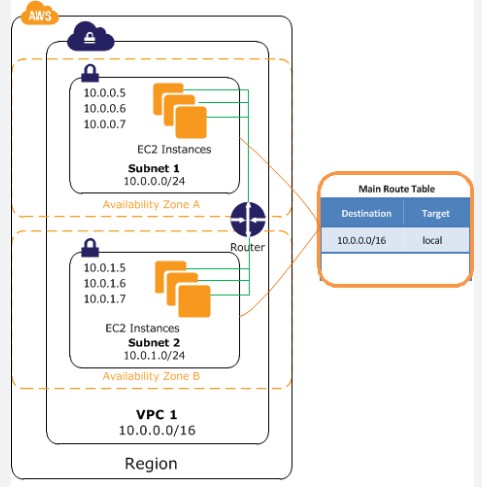


Рисунок 2.4 - Приклад роботи Amazon VPC

Вартість VPC в порівнянні з корпоративною приватною хмарою набагато нижча, тому що в більшості пропонованих послуг або програм вам доведеться платити тільки за певні покращення, що робить цей тип хмари хорошим варіантом для невеликих компаній.

Ця приватна віртуальна хмара дозволяє користувачеві створювати віртуальні приватні мережі та керувати такими елементами, як діапазони адрес, підмережі, параметри безпеки та інші. Для його використання необхідно спочатку налаштувати VPC на консолі серверів AWS. Потім можна додати такі інструменти, як Amazon Elastic Compute Cloud (EC2) або Amazon Relational Database Service (RDS). Нарешті, ви повинні визначити спосіб, яким VPC будуть взаємодіяти один з одним між обліковими записами і так далі.

VPC ізолює обчислювальні ресурси від інших обчислювальних ресурсів, доступних у публічній хмарі. Основними технологіями для ізоляції VPC від решти публічної хмари є:

Підмережі: підмережа – це діапазон IP-адрес у мережі, які зарезервовані, тому вони не доступні для всіх у мережі, фактично розділяючи частину мережі для приватного використання. У VPC це приватні IP-адреси, які недоступні через

загальнодоступний Інтернет, на відміну від типових IP-адрес, які є загальнодоступними.

VLAN: локальна мережа — це локальна мережа або група обчислювальних пристроїв, які підключені один до одного без використання Інтернету. VLAN — це віртуальна локальна мережа. Як і підмережа, VLAN є способом поділу мережі, але поділ відбувається на іншому рівні в моделі OSI (рівень 2 замість рівня 3).

VPN: віртуальна приватна мережа (VPN) використовує шифрування для створення приватної мережі поверх публічної мережі. Трафік VPN проходить через загальнодоступну інфраструктуру Інтернету – маршрутизатори, комутатори тощо, – але трафік зашифрований і невидимий нікому.

VPC матиме виділену підмережу та VLAN, які доступні лише клієнту VPC. Це запобігає доступу будь-кого іншого в загальнодоступній хмарі до обчислювальних ресурсів у межах VPC – фактично розміщуючи на столі знак

«Зарезервовано». Клієнт VPC підключається через VPN до свого VPC, тому дані, що надходять у VPC і з нього, не бачать інші користувачі публічної хмари.

Переваги використання VPC замість приватної хмари:

* Масштабованість: оскільки VPC розміщується постачальником загальнодоступної хмари, клієнти можуть додавати більше обчислювальних ресурсів за вимогою.
* Просте розгортання гібридної хмари: відносно просто підключити VPC до загальнодоступної хмари або до локальної інфраструктури через VPN. (Дізнайтеся про гібридні хмари та їхні переваги.)
* Краща продуктивність: веб-сайти та програми, розміщені в хмарі, зазвичай працюють краще, ніж ті, що розміщені на локальних серверах.
* Краща безпека: постачальники публічних хмар, які пропонують VPC, часто мають більше ресурсів для оновлення та підтримки інфраструктури, особливо для малого та середнього бізнесу. Для великих підприємств або

будь-яких компаній, які стикаються з надзвичайно жорсткими правилами безпеки даних, це менша перевага.

Логотип AWS VPC зображено на рисунку 2.5.



Рисунок 2.5 - Логотип Amazon VPC

### Аналіз S3Bucket AWS

Amazon S3Bucket — це загальнодоступний хмарний ресурс зберігання, доступний у Amazon Web Services (AWS) Simple Storage Service (S3), пропозиції для зберігання об’єктів. Відра Amazon S3, які схожі на папки файлів, зберігають об’єкти, які складаються з даних і їх описових метаданих.

Клієнт S3 спочатку створює сегмент у регіоні AWS за своїм вибором і дає йому глобально унікальне ім’я. AWS рекомендує клієнтам вибирати географічно близькі регіони, щоб зменшити затримку та витрати.

Після створення сегмента користувач вибирає рівень для даних, причому різні рівні S3 мають різні рівні резервування, ціни та доступності. Одне відро може зберігати об’єкти з різних рівнів зберігання S3.

Потім користувач може вказати привілеї доступу до об’єктів, що зберігаються у сегменті, за допомогою таких механізмів, як служба AWS Identity and Access Management, політики контейнерів і списки контролю доступу.

Клієнт AWS може взаємодіяти з сегментом Amazon S3 через консоль керування AWS, інтерфейс командного рядка AWS або інтерфейси прикладного

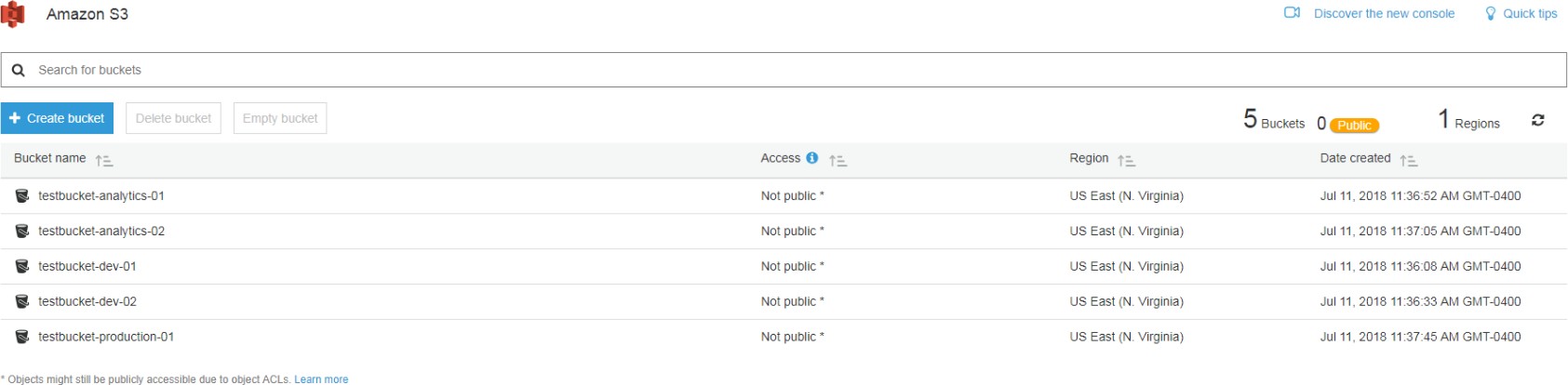
програмування (API). Вигляд консолі керування AWS S3Bucket зображено на рисунку 2.6.

Рисунок 2.6 - Amazon S3 buckets на консолі керування AWS

AWS пропонує кілька функцій для сегментів Amazon S3. ІТ-фахівець може увімкнути керування версіями для сегментів S3, щоб зберегти кожну версію об’єкта, коли над ним виконується операція, наприклад копіювання чи видалення. Це допомагає ІТ-команді запобігти випадковому видаленню об’єкта. Так само, після створення сегмента, користувач може налаштувати журнали доступу до сервера, журнали API на рівні об’єктів, теги та шифрування.

Крім того, S3 Transfer Acceleration допомагає виконувати швидкі та безпечні передачі від клієнта до сегмента S3 через межі AWS.

Немає обмежень щодо кількості об’єктів, які ІТ-фахівець може зберігати у відрі, хоча відра не можуть існувати всередині інших відер.

Продуктивність S3 залишається незмінною незалежно від того, скільки відер створює окрема особа. Кожен обліковий запис AWS може створити 100 сегментів, але можна отримати більше, якщо надіслати запит на збільшення ліміту обслуговування. Обліковий запис AWS, який створює сегмент, є його власником, і право власності не можна передати. Клієнт S3 може видалити сегмент, але інший користувач AWS може отримати це глобально унікальне ім’я.

AWS стягує з клієнтів плату за зберігання об’єктів у bucket та за переміщення об’єктів у відро та з нього. Ціна на bucket залежить від регіону. Логотип Amazon S3Bucket зображено на рисунку 2.7.

Рисунок 2.7 - Amazon S3 Bucket логотип

### Аналіз DynamoDB AWS

DynamoDB - це база даних NoSQL (не тільки SQL) типу "ключ-значення", доступна як сервіс на AWS.

Він оптимізований для забезпечення гарної продуктивності та горизонтальної масштабованості за рахунок додавання додаткових вузлів у кластер. Якщо розмір даних зросте, це не вплине на продуктивність бази даних.

Крім того, він має високу доступність (SLA 99,999% часу безвідмовної роботи, це менше 5 хвилин падінь на рік) і довговічність даних. Це досягається за рахунок зберігання кількох копій даних на різних вузлах. Якщо один із цих вузлів виходить з ладу, інший може зайняти його місце як основний вузол.

Традиційно, коли ми мали базу даних, ми повинні були керувати всім її обладнанням, безпекою, оновленнями та враховувати її обслуговування. З DynamoDB все це не потрібно, це керований сервіс, і хоча за ним стоять сервери та системи зберігання, він абсолютно прозорий, і вам нема про що турбуватися.

DynamoDB розподіляє дані по розділах, які можуть бути різними машинами, як фізичними, так і віртуальними, для підвищення надмірності.

У нього дуже розумна вартість, а модель оплати ґрунтується на використанні, тому це не фіксована вартість. З одного боку, ви платите за сховище, а з іншого — за попит (залежно від запитів) або за надання ресурсів для певної ємності (Read Capacity Units або Write Capacity Units).

Нарешті він дуже добре інтегрується з іншими сервісами AWS. Це одна із найсильніших сторін цієї бази даних. Наприклад, він дуже добре інтегрується з s3 для експорту таблиць, ми також можемо легко надсилати дані в Kinesis для виконання деякої потокової обробки, такої як тимчасові запити в реальному часі. Логотип Amazon Web Services DynamoDB зображено на рисунку 2.8.



Рисунок 2.8 - Amazon DynamoDB логотип

DynamoDB таблиці - це набори елементів, а елементи - це набори атрибутів або пар ключ-значення. Первинний ключ таблиці складається із ключа розділу та ключа сортування.

Крім того, можна використовувати глобальний вторинний індекс або GSI, який дозволяє виконувати фільтрацію запитів по стовпцях, відмінних від розділу або ключа сортування.

DynamoDB ідеально підходить для програм з добре відомими шаблонами доступу. Перш ніж використовувати цю послугу, бажано знати, як ви збираєтесь

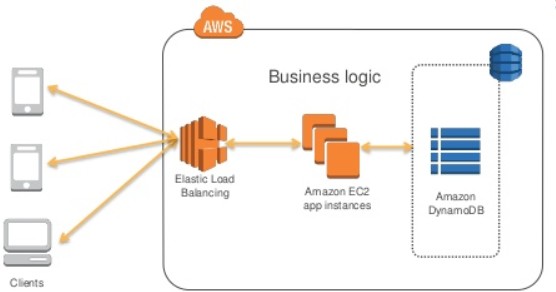
отримувати доступ до даних, щоб з першого моменту спроектувати правильну структуру таблиці. Схему роботи Amazon Web Services DynamoDB зображено на рисунку 2.9.

Рисунок 2.9 - Amazon DynamoDB схема роботи

При роботі з API DynamoDB, яка є базою даних типу «ключ-значення», ключ зазвичай надається для отримання збереженого зв'язаного значення. Зазвичай використовуються ORM (об'єктно-реляційне зіставлення), які реалізують API, що лежать в основі, залежно від мови, яку ви хочете використовувати. Для Java ми маємо DynamoDBMapper і для Python Boto3.

Щодо авторизації, вона базується на використанні служби IAM, тому не потрібно буде передавати облікові дані, такі як ім'я користувача та пароль, а потрібно буде передавати користувача з достатніми дозволами.

Також можна налаштувати DynamoDB Streams, що дозволить вам фіксувати зміни записів таблиці у режимі реального часу та запускати лямбда-функції. Ця функціональність дуже цікава для створення інформаційних панелей або систем оповіщення.

Ця база даних також не підходить, коли потрібно аналізувати історичні дані в режимі реального часу, для чого є кращі аналітичні варіанти, такі як сховища даних .

### Аналіз Elastic Beanstalk AWS

Beanstalk — це найшвидший спосіб запустити програму на AWS. Розробники можуть просто надіслати код, і служба автоматично оброблятиме такі деталі, як надання, балансування навантаження, автоматичне масштабування та моніторинг.

Програма — це логічний набір компонентів Beanstalk, які включають середовища, версії та конфігурації середовища. У Beanstalk програма концептуально схожа на каталог. Логотип Amazon Elastic Beanstalk зображено на рисунку 2.10.

Рисунок 2.10 - Amazon Elastic Beanstalk логотип

Версія програми стосується версії з назвою розгортання для веб-програми. Ця версія вказує на об’єкт S3, який містить код для розгортання. Програми можуть мати багато версій, і кожна версія програми унікальна. У середовищі виконання організації можуть розгорнути будь-яку версію програми, яку вони вже завантажили, або вони можуть негайно розгорнути нову версію. Організації також можуть завантажувати кілька версій програм.

Середовище – це версія програми, яка розгортається на ресурсах AWS. У кожному середовищі працює лише одна версія програми; однак одна й та сама версія або різні версії можуть запускати стільки середовищ, скільки потрібно.

Конфігурація середовища — це набір параметрів і параметрів, які визначають поведінку середовища та пов’язаних із ним ресурсів. Коли конфігурація середовища оновлюється, Beanstalk автоматично застосовує зміни до існуючих ресурсів або видаляє та розгортає нові ресурси залежно від типу зміни.

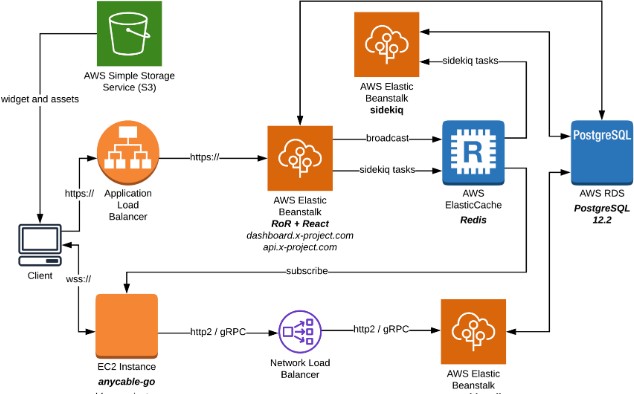
Під час запуску середовища вказується рівень середовища, платформа та тип середовища. Вибраний рівень середовища визначає, чи надає Beanstalk ресурси для підтримки веб-програми, яка обробляє запити HTTPS, або програм, які обробляють фонову обробку завдань. Рівень середовища, в якому програма обробляє веб-запити, відомий як рівень веб-сервера. Рівень середовища, на якому програма виконує завдання у фоновому режимі, називається робочим рівнем. Приклад робочої схеми Amazon Elastic Beanstalk зображено на рисунку 2.11.

Рисунок 2.11 - Amazon Elastic Beanstalk приклад робочої схеми Beanstalk забезпечує підтримку програмних платформ Java, Node.js, PHP,

Python, Ruby та Go з підтримкою веб-контейнерів Tomcat, Passenger, Puma та Docker.

Beanstalk надає кілька функцій керування, які спрощують розгортання та керування програмами на AWS. Організації мають доступ до створення метрик моніторингу в CloudWatch, таких як середнє використання ЦП, кількість запитів і середня затримка. Отримуйте сповіщення електронною поштою через SNS, коли програма змінюється або сервери додаються чи видаляються. Доступ до журналів сервера додатків можна отримати, не вимагаючи входу.

Крім того, розробники мають повний контроль над ресурсами AWS, розширюючи можливості своєї програми, і можуть виконувати різноманітні функції, просто налаштовуючи такі функції конфігурації, як:

* Тип екземпляра EC2
* Тип бази даних
* Увімкнення доступу до примірників EC2
* Підвищення безпеки програми шляхом увімкнення HTTPS в ELB
* Налаштування параметрів сервера та змінних середовища
* Налаштування функцій автоматичного масштабування для керування показниками та пороговими значеннями, які використовуються для додавання або видалення екземплярів

## Висновки до розділу 2

В даному розділі були розглянуті веб-сервіси на основі яких був побудований програмний модуль, а саме програмний застосунок реалізації сценарію аналізу рівня міжнародного співробітництва наукової організації в науково-технічній сфері на основі веб-сервісів.

# ОПИС ПОБУДОВАНОЇ СИСТЕМИ

## Загальний опис системи

Користувацький інтерфейс застосунку повинен мати сучасний та інтуїтивно зрозумілий інтерфейс, створений з урахуванням усіх сучасних правил та норм. Клієнт сервісу який має необхідні права повинен мати можливість використання усіх базових операцій з інформацією.

Застосунок повинен мати такі розділи:

* + - інформація про всі уже виконані аналізи;
    - про наукову взаємодію в науково-технічній сфері;
    - публікації результатів проведеного аналізу;
    - публікації аналізів що ставляться під сумнів;
    - інформація про наукові заходи та конференції що проводяться;
    - інформацію про взаємодію із зарубіжними партнерами;
    - інформацію про матеріально-технічне забезпечення;

## Архітектура системи

Для створення даного застосунку були прийнято рішення використати веб-сервісний тип архітектури. Веб-сервіси ґрунтуються на слабозв'язаній архітектурі. Це означає, що інтерфейси веб-сервісів динамічні за своєю природою (змінюються протягом заданого періоду часу). Але логіка клієнта не обов'язково змінюється при взаємодії із сервісом. Це полегшує інтеграцію кількох програм ефективнішим способом. Якби це була тісно пов'язана архітектура, то при кожній зміні інтерфейсу логіка клієнта мала б змінюватися, щоб синхронізувати її зі службою.

Схему веб-сервісної архітектури зображено на рисунку 3.1.

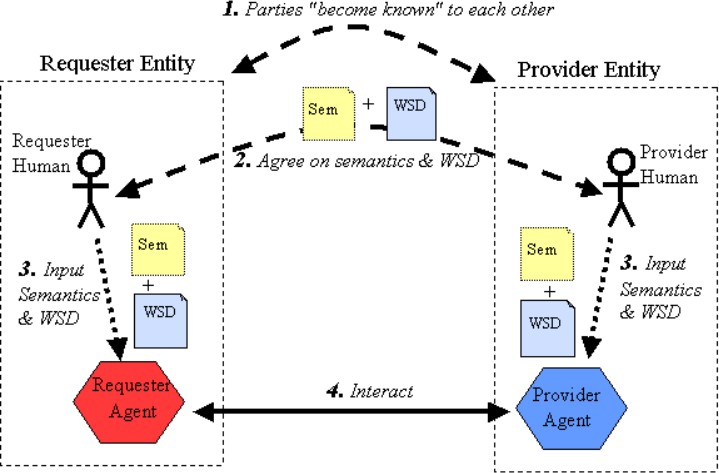


Рисунок 3.1 - Веб-сервісна архітектура

## Особливості реалізації системи

Формальним ядром архітектури є перерахування концепцій та взаємозв'язків, які грають центральну роль функціональної сумісності Web-сервисов.

Архітектура описується за допомогою кількох простих елементів: концепцій, взаємозв'язків та моделей. Концепти часто схожі на іменники в тому сенсі, що вони визначають речі або властивості, які ми очікуємо побачити в реалізаціях архітектури, так само відносини зазвичай є лінгвістично дієсловами.

Як і у разі будь-якого великомасштабного проекту, часто необхідно структурувати саму архітектуру. Ми робимо це за допомогою більш масштабної моделі. Модель – це невід'ємна частина архітектури, що фокусується на певній темі чи аспекті архітектури.

Очікується, що концепція матиме певну відповідність до будь-яких реалізацій архітектури. Наприклад, концепція повідомлення ідентифікує клас об'єкта (не плутати з об'єктами та класами, які можна знайти в

об'єктно-орієнтованих мовах програмування), який, як ми очікуємо, зможе ідентифікувати у будь-якому контексті веб-сервісів. Точна форма повідомлення може різнитися у різних реалізаціях, але концепція повідомлення каже нам, що шукати у цій конкретній системі, а чи не наказує її точну форму.

Не всі концепції будуть реалізовані в термінах об'єктів даних або структур, що зустрічаються у комп'ютерах чи комунікаційних пристроях; наприклад, людина або організація належить до людей та людських організацій. Інші концепції ще абстрактні; наприклад, надійність повідомлення означає властивість служби передачі повідомлень — властивість, яку не можна змінити, але яка важлива для веб-служб.

Кожне поняття представлене у звичайному, стилізованому вигляді, що складається з короткого визначення, перерахування відносин з іншими поняттями та трохи довшого пояснювального опису. Наприклад, поняття агента включає як пов'язані поняття той факт, що агент є обчислювальним ресурсом, має ідентифікатор і власника. Частина опису агента пояснює детальніше, чому агенти важливі для архітектури.

Основна функція перерахування Архітектури з погляду моделей, концепцій та взаємозв'язків полягає в тому, щоб дати рекомендації щодо відповідності архітектурі. Наприклад, архітектура зазначає, що повідомлення має відправник ; будь-яка реалізація цієї архітектури, яка дозволяє асоціювати повідомлення з його відправником, відповідає архітектурі. Наприклад, SMTP можна використовувати для надсилання повідомлень. Однак, оскільки SMTP (в даний час) допускає підробку ідентифікатора відправника, одного SMTP недостатньо для виконання цього обов'язку.

## Схема взаємодії клієнтської та серверної частини

API.

Клієнтська та серверна частини взаємодіють за допомогою викликів REST

Схему роботи REST API зображено на рисунку 3.2.

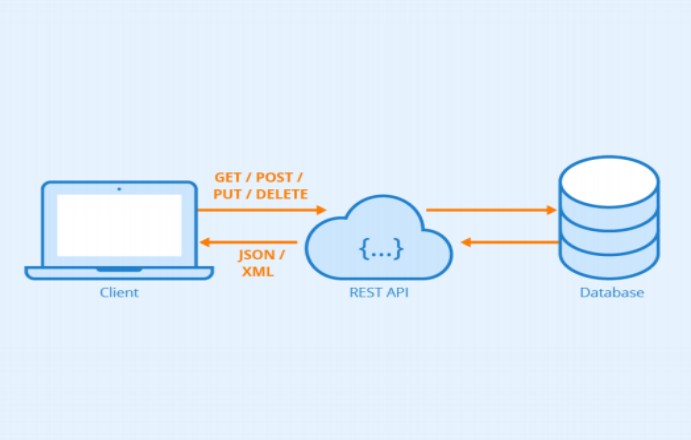


Рисунок 3.2 - Робота REST API

## Висновки до розділу 3

В цій частині роботи виконано опис застосунку реалізації сценарію аналізу рівня міжнародного співробітництва наукової організації в науково-технічній сфері на основі веб-сервісів Також було описано роботу з базою даних та наведено схему її реалізації, детальний. В цьому розділі описано як відбувається взаємодія.

# ОПИС ПРОГРАМНИХ ЗАСОБІВ РОЗВ’ЯЗАННЯ ЗАДАЧІ

## Обґрунтування вибору мови програмування

Так як розроблений програмний продукт має бути створений у вигляді веб сервісу, необхідно було обрати саме мову програмування для розробки даного типу програмного забезпечення.

Мова програмування має відповідати таким вимогам:

* + - швидкість та продуктивність;
    - підтримка сучасних технологій;
    - простота і раціональність застосування;
    - підтримка усіх необхідних парадигм програмування;
    - велика кількість фреймворків та бібліотек для розробки веб сервісів;

### Мова програмування C#

C# (читай C Sharp) являє собою еволюцію, яку Microsoft зробила з цієї мови, взявши найкраще з мов C і C++, і продовжила додавати функціональні можливості, взявши з інших мов, таких як Java, деякі з його розвиненого синтаксису. Він зробив його об'єктно-орієнтованим для всієї своєї платформи. . без втрати початкової потужності C. У версії .NET Core його компілятор був повністю перебудований, що робить програми на 600% швидше.

До переваг C# можна віднести:

* + - * C – подібний знайомий синтаксис;
      * наявність потужної IDE;
      * підтримка переважної більшості продуктів Microsoft;
      * низький поріг входження;
      * безкоштовність ряду інструментів для невеликих компаній;
      * велика кількість синтаксичного цукру;
      * типи даних мають фіксований розмір;
      * автоматичний збір сміття;

Як недолік можно відзначити те що для належної роботи необхідно виконати деякі мінімальні системні вимоги, як-от наявність Windows NT 4 або новішої версії, близько 4 Гб вільного місця для інсталяції тощо.

### Мова програмування PHP

PHP — це мова програмування для розробки програм і створення веб-сайтів, яка з кожним днем завойовує все більше прихильників. Простий у використанні та постійно вдосконалюваний, це безпечний варіант для тих, хто хоче працювати над якісними проектами без ускладнень.

Серед переваг PHP можна виділити:

* + - * простий синтаксис;
      * із PHP можна працювати у дуже великій кількості різноманітних редакторів коду
      * використовує дуже мало ресурсів
      * доволі ефективний у використанні
      * висока гнучкість, ємність та функціональність;
      * бюджетність, економічність;
      * багатозадачність та широкі можливості;

Як недолік відзначають погану систему безпеки й сумнівні моменти у синтаксисі.

### Мова програмування Java

Java - це об'єктно-орієнтована мова програмування, яка широко використовується для розробки додатків. Щоб знати, як програмувати в комп’ютерних інструментах, щоб створювати додатки та процеси на різних пристроях, необхідно розуміти та опанувати широко поширену мову, яка стає все більш важливою для спільноти. Тому сьогодні ми навчимо вас, що таке Java. Це тип мови програмування та обчислювальної платформи, створений і проданий компанією Sun Microsystems у 1995 році.

Він створений як об’єктно-орієнтована мова, його мета полягає в тому, щоб дозволити розробникам додатків написати програму один раз і запустити її на будь-якому пристрої.

Java має такі переваги:

* + - * незалежність від апаратної частини і ОС;
      * динамічність та адаптивність;
      * об’єктно орієнтований підхід до програмування;
      * простота – перша технічна перевага Java;
      * зручні та ефективні мережеві можливості;
      * безпека – важливий критерій враховуючи використання Java в розробці мережевих середовищ;
      * практично у будь-яких умовах вона працює доволі стабільнольно відносно інших мов
      * продуктивність – нові версії динамічних компіляторів;

Серед недоліків можно відзначити доволі велику кількість синтаксису й відносно слабку продуктивність.

### Мова програмування JavaScript

JavaScript — це потужна мова програмування, створена для браузера Netscape у 1995 році. Відтоді всі сучасні браузери перейняли її для додавання функціональності веб-сайтам, а нещодавно й веб-додаткам.

За минулі роки, починаючи з моменту створення, JavaScript став гігантом: він використовується не лише в Інтернеті, його можна знайти практично скрізь, навіть у космосі.

Головними відмінностями є:

* + - * зручність для веб розробки;
      * продуктивність;
      * потужні фреймворки;
      * проста у вивченні;
      * зручність у використанні;
      * простота засвоєння;
      * велика кількість готових інструментів;
      * велика спільнота розробників;

Серед недоліків можна відмітити те що JavaScript не має статичної типізації Але цей недолік багато спеціалістів вважать навпаки перевагою, що робить JavaScript зручніше.

З створенням нових фреймворків стає можливим використовувати Javascript практично для будь-яких цілей.

### Мова програмування Python

Python — це мова програмування високого рівня, яка використовується для розробки всіх видів програм. На відміну від інших мов, таких як Java або .NET, це

інтерпретована мова, її не потрібно компілювати для запуску програм, написаних на Python, і вони виконуються безпосередньо на комп’ютері за допомогою програми, званої інтерпретатором. , щоб не потрібно було «переводити» його на машинну мову.

Можна визначити такі позитивні якості Python:

* + - * зрозумілий та лаконічний синтаксис;
      * кросплатформність;
      * підтримка об’єктно-орієнтованого програмування;
      * величезна кількість модулів;
      * динамічна типізація;
      * інтеграція із C/C++;
      * якісна підтримка модульності;
      * вбудована підтримка Unicode;

Але у Python є й недоліки, ймовірно головний з них - це його низька швидкість виконання.

## Висновок щодо обраної мови

Проаналізувавши всі перераховані мови програмування , у висновку можна відмітити як їх головні переваги так й суттєві недоліки, й зробити висновки, яку ж мову використати:

* через те що Python має низьку швидкість виконання, він однозначно не є кращим вибором для реалізації проєкту, й тому було вирішено його відкинути
* серед недоліків мови програмування Java було відмічено малу ефективність та занадто ускладнений синтаксис, тому вирішено й

вона не є хорошим засобом для розробки даного наразі нашого застосунку

* PHP ж було відкинуто через його суттєві недоліки у безпеці а також, деякі протиріччя у синтаксисі, хоча він і має чимало переваг, він не підходить для реалізації даного застосунку

## Обґрунтування вибору бази даних

### Огляд видів баз даних

Серед усіх баз даних що існують на сьогодні виділяють два типи: реляційні, також відомі як SQL, та нереляційні, також відомі як NoSQL. Відмінність між ними у тому що вони представляють різні моделі: реляційну та не реляційну. В не реляційних базах даних дані представляють залежно від типу самої бази. В той же час реляційні бази даних представляють дані у вигляді таблиць.

Типи не реляційних баз даних:

* + - * колоночне сховище;
      * графове сховище;
      * документно-орієнтоване сховище;
      * ключ-значення (Key-value);

Серед найбільш використовуваних баз даних на даний момент є реляційна база даних. Цей тип бази даних є інструментом, який використовують усі типи компаній, незалежно від їх розміру чи сфери діяльності. У цій публікації ми збираємося пояснити, що таке реляційна база даних, її основні характеристики та принципи роботи.«Реляційна база даних — це тип бази даних, яка відповідає реляційній моделі». Таким чином, згідно з цим визначенням реляційної бази даних, це база даних, яка зберігає та надає доступ до точок даних, які пов’язані

одна з одною. Реляційна модель — це інтуїтивно зрозумілий і прямий спосіб представлення даних без необхідності їх ієрархізації.

Реляційні бази даних є найбільш широко використовуваною моделлю сьогодні (вперше висунута в 1970 році Едгаром Франком Коддом).

Реляційна база даних, по суті, є набором таблиць (або зв’язків), що складається з рядків (записів) і стовпців (полів); Таким чином, кожен запис (кожен рядок) має унікальний ідентифікатор, який називається ключем, а стовпці таблиці містять атрибути даних. Кожен запис зазвичай має одне значення для кожного атрибута, що полегшує створення зв’язків між точками даних.

Таким чином, однією з основних характеристик реляційної бази даних є те, що вона уникає дублювання записів і, у свою чергу, гарантує посилальну цілісність, тобто якщо один із записів буде видалено, цілісність інших записів не буде порушена. постраждали. Крім того, завдяки ключам інформація може бути легко доступна та відновлена в будь-який час.

Так само не може існувати дві таблиці з однаковими іменами, а зв’язок між батьківською та дочірньою таблицями здійснюється за допомогою первинних ключів (вони є основним ключем запису в таблиці) та зовнішніх ключів (вони розміщуються в дочірньої таблиці та містить те саме значення, що й первинний ключ батьківського запису).

Щоб мати можливість зберігати, керувати, консультувати та отримувати дані, що зберігаються в реляційній базі даних, необхідно використовувати спеціальне програмне забезпечення, яке називається системою керування реляційною базою даних (RDBMS). Це програмне забезпечення забезпечує інтерфейс між користувачами та/або програмами та базою даних, а також адміністративні функції для керування доступом, зберіганням і продуктивністю.

Переваги реляційної системи бази даних:

* + - * простота й доступність для розуміння користувача;
      * єдиною використовуваною інформаційною структурою є таблиця;
      * строгі правила проектування;
      * базування на математичному апараті;
      * повна незалежність даних;
      * зміни в програмному забезпеченні при зміні реляційної бази даних мінімальні;
      * для організації запитів і написання прикладного програмного забезпечення немає необхідності знати конкретну організацію бази даних у зовнішній пам’яті;
      * наявність потужних інструментів обробки даних;
      * велика кількість фахівців та документації у зв’язку з широкою популярністю;
      * підтримка стандарті SQL;
      * мінімальні затрати пам’яті;
      * для задач що допускають реалізацію за допомогою будь-якої з трьох моделей даних ієрархічна модель дозволяє отримати уявлення з мінімально необхідною пам’яттю;

Недоліки реляційних баз даних:

* + - * далеко не завжди предметна область може бути представлена у вигляді таблиць;
      * в результаті логічного проєктування з’являється безліч таблиць;
      * при великих об’ємах труднощі з розумінням структури даних;
      * займає відносно багато зовнішньої пам’яті;
      * відносно низька швидкість доступу до даних;
      * трудомісткість розробки;
      * некоректне уявлення сутностей реального світу;
      * семантичне перенавантаження;
      * слабка підтримка обмежень цілісності і корпоративних обмежень;
      * однорідна структура даних;
      * обмежений набір операцій;
      * складності при обробці рекурсивних запитів;
      * проблема неузгодженості типів даних;
      * інші проблеми пов’язані з паралельним використанням, змінами схеми і не розвиненими засобами доступу;

Бази даних NoSQL (так звані «не тільки SQL») є нетабличними базами даних і зберігають дані інакше, ніж реляційні таблиці. Бази даних NoSQL бувають різних типів залежно від моделі даних. Основними типами є документ, ключ-значення, широкий стовпець і графік. Вони забезпечують гнучкі схеми та легко масштабуються з великими обсягами даних і високим навантаженням користувачів. Коли люди використовують термін «база даних NoSQL», вони зазвичай використовують його для позначення будь-якої нереляційної бази даних. Деякі кажуть, що термін «NoSQL» означає «не SQL», а інші кажуть, що він означає «не тільки SQL». У будь-якому випадку більшість погоджується, що бази даних NoSQL — це бази даних, які зберігають дані у форматі, відмінному від реляційних таблиць.

Бази даних NoSQL з’явилися наприкінці 2000-х років, коли вартість зберігання різко знизилася.

Переваги NoSQL:

* + - * лінійна масштабованість – додавання нових вузлів в кластер збільшує загальну продуктивність системи;
      * гнучкість, що дозволяє оперувати напівструктурованими даними, реалізуючи в тому числі повнотекстовий пошук по базі;
      * можливість працювати з різними уявленнями інформації, в тому числі без завдання схеми даних;
      * висока доступність за рахунок реплікації даних і інших механізмів відмовостійкості, зокрема, шаринга – автоматичного розподілу даних за різними вузлами мережі, коли кожен сервер кластера відповідає тільки за певний набір інформації, обробляючи запити на його читання і запис. Це збільшує швидкість обробки даних і пропускну здатність додатку;
      * продуктивність за рахунок оптимізації для конкретних видів моделей даних (документованої, графової, колонкової або ключ - значення) і шаблонів доступу;
      * широкі функціональні можливості – власні SQL – подібні мови запитів, RESTful - інтерфейси, API і складні типи даних, наприклад map, list i struct, що дозволяють обробляти величезну кількість значень;

Недоліки NoSQL:

* + - * обмежена ємність вбудованої мови запитів, для вирішення цієї проблеми використовуються сторонні засоби трансляції класичних SQL – виразів в виконуваний код для конкретної не реляційної бази даних;
      * складності в підтримці всіх ACID – вимог до транзакцій (атомарність, консистентність, ізоляція, довговічність) через те, що NoSQL – база даних замість CAP – моделі (узгодженість, доступність, стійкість до поділу)скоріше відповідають моделі BASE (базова доступність, гнучкий стан і підсумкова узгодженість);
      * сильна прив'язка додатку до конкретної СКБД через специфіку внутрішньої мови запитів і гнучкої моделі даних, орієнтованої на конкретний випадок;
      * недостаток фахівців з NoSQL – баз в порівнянні з реляційними аналогами;

Оскільки витрати на зберігання швидко зменшувалися, кількість даних, які додатки потребували для зберігання та запитів, зросла. Ці дані були будь-яких форм і розмірів — структуровані, напівструктуровані та поліморфні — і визначити схему наперед стало майже неможливим. Бази даних NoSQL дозволяють розробникам зберігати величезні обсяги неструктурованих даних, надаючи їм велику гнучкість.

Крім того, популярність Agile Manifesto зростала, а розробники програмного забезпечення переосмислювали спосіб розробки програмного забезпечення. Вони усвідомлювали необхідність швидкого пристосування до мінливих вимог. Їм потрібна була здатність швидко повторювати та вносити зміни в увесь стек програмного забезпечення — аж до бази даних. Бази даних NoSQL дали їм таку гнучкість.

## Обґрунтування вибору фреймворку

### Огляд видів фреймворків

MVC розшифровується як Model View Controller і є не що інше, як архітектурний шаблон, вперше описаний Трюгве Реєнскаугом в 1979 року. Основна мета цього шаблону – розділити концепції на три основні рівні:

* + - * Модель: відповідає за керування даними нашої програми. Тобто в ньому ми маємо наголошувати на бізнес-логіці, наполегливості тощо.
      * View(s) : Складіть інтерфейс користувача. Подання має справу з поданням даних, що не містять абсолютно жодної бізнес-логіки.

Перед досягненням цього рівня дані вже були оброблені, і дані, що відображаються, повинні розумітися як результат попереднього процесу (під попереднім процесом я маю на увазі те, що сталося в Моделі).

* + - * Контролер(и): це шар, який діє як посередник між поданням та моделлю.

Для роботи з ASP.NET MVC ми маємо ряд шаблонів, наданих Microsoft. Якщо ми використовуємо Visual Studio 2008, необхідно встановити ці шаблони через Microsoft Web Platform Installer .

З іншого боку, з 2010 вже доступна остаточна версія Visual Studio 2010, де у нас є все необхідне для старту. У цьому пості я збираюся показати його з Visual Studio 2010, але зображення 2008 зовсім не відрізняються від сьогоднішніх.

Якщо ми вирішимо створити новий проект і позиціонуємо себе на відданій мові в розділі Web , ми побачимо, що у нас є два шаблони для цього шаблону.

Веб-додаток ASP.NET MVC 2 - це шаблон, який містить саму структуру MVC, а також ряд прикладів, що дозволяють перевірити, як це працює, і розпочати роботу на основі. Коли ми ще недостатньо знання про те, як працює цей шаблон, в ідеалі почати працювати з цим шаблоном.

З іншого боку, порожній веб-додаток ASP.NET MVC з'явився у другій версії ASP.NET MVC через великий попит, який існував завдяки відгукам розробників. Це шаблон із структурою MVC, але, на відміну від попереднього, повністю чистий, щоб розпочати роботу з нуля без необхідності видалення будь-яких прикладних файлів.

Якщо ми створимо проект із будь-яким із двох шаблонів, перше, що нас запитає, — чи хочемо ми одночасно створити паралельний проект для створення модульних тестів.

AutoMapper — це, по суті, інструмент, який дозволяє вам відображати один об’єкт на інший, заощаджуючи нам багато коду, і який надає деякі утиліти для налаштування способу відображення між об’єктами.

Відображення між об’єктами в загальній функціональності в багатьох наших розробках, оскільки вони дозволяють налаштовувати вихід даних, уникаючи розкриття конфіденційної інформації та надаючи клієнту лише ту інформацію, яка йому дійсно потрібна, на основі цього твердження, давайте подумаємо про клас (модель) Клієнт, який має такі поля, як ідентифікатор, ім’я, прізвище, вік, електронна адреса, дата створення та пароль серед інших, і тепер у нас є деякі функції, які розкривають інформацію клієнта, у цьому випадку ми розкриваємо інформацію, яку ми не повинні надавати, наприклад дата створення та ключ, у цьому випадку ми можемо створити новий клас (відомий як ViewModel) лише з полями, які ми збираємось виставити; і саме тут AutoMapper відіграє важливу роль, оскільки він допоможе нам перейти від нашої клієнтської моделі/класу до нашої ViewModel ClientViewModel досить простим і надійним способом.

Працює AutoMapper, з веб-програми ASP.NET MVC, однак її можна використовувати в будь-якому іншому типі програми.

AutoMapper у C# — це бібліотека, яка використовується для відображення даних з одного об’єкта на інший. Він діє як картограф між двома об’єктами та перетворює один тип об’єкта в інший.

Він перетворює вхідний об’єкт одного типу на вихідний об’єкт іншого типу, доки останній тип не відповідатиме або підтримуватиме конвенції AutoMapper.

Іноді в проектах реального часу рівні об’єктів мають справу зі зв’язком із службами або DataLayers. Щоб відобразити дані в додатку, нам потрібно розділити клас на ViewModel і Model. Однак шари інтерфейсу користувача можуть

не синхронізуватися з сутностями. Тому, щоб відобразити сутності в model або viewModel, нам потрібен AutoMapper.

Навіть у проектах реального часу інколи нам потрібно зіставити об’єкти між інтерфейсом користувача/доменом і службами/доменом. Однак відображення за допомогою традиційного методу, як обговорювалося раніше, може стати неспокійним.

Його назва вже інтуїтивно зрозуміла, це невелика бібліотека, яка допомагає нам зняти навантаження з наших плечей, ту частину коду, яку ми всі ненавидимо, як-от зв’язування полів одного об’єкта з іншим. Так само, як коли ми пов’язуємо сутність із бази даних із моделлю з нашого погляду.

## Висновки до розділу 4

Було виконано порівняння , розглянуто основні відмінності обох систем управління. Хоча існує безліч відмінностей між системами керування реляційними базами даних (RDBMS) і базами даних NoSQL, одна з ключових відмінностей полягає в тому, як дані моделюються в базі даних. У цьому розділі ми розглянемо приклад моделювання тих самих даних у реляційній базі даних і базі даних NoSQL. Потім ми висвітлимо деякі інші ключові відмінності між реляційними базами даних і базами даних NoSQL.

# РОЗРОБКА СТАРТАПУ ПРОЄКТА

## Описання ідеї проєкту

Спроєктовано та розроблено застосунок, який використовує сучасні технології та враховує тенденції до розробки програмного забезпечення. Головною функцією застосунку є можливість реалізації аналізу міжнародного співробітництва.

Враховуючи поставлене технічне завдання було реалізовано та протестовано наступні рішення для розробки цільового застосунку:

* + - була реалізована система додавання, редагування та видалення даних застосунку;
    - розроблений застосунок має високу швидкодію;
    - також бело реалізовано взаємодію клєнтської та серверної частини застосунку;
    - до застосунку підключено базу даних;
    - контент застосунку надійно захищений за допомогою захищеного протоколу передачі даних;
    - було реалізовано збагачене зведення застосунку;

Було реалізовано функцію аутентифікації користувачів та публікації даних з відповідними правами. Авторизованим користувачам надана можливість:

* + - робити нові аналізи;
    - редагувати вже існуючі дані;
    - видалити дані;
    - користуватися всіма функціональними можливостями даного застосунку;

Клієнти з повними правами доступу також мають можливість змінювати інформацію, редагувати, додавати та видаляти інших користувачі.

Взаємодія даного застосунку із БД реалізована класично, з використанням SQL – запитів.

Типи запитів до бази даних:

* + - UPDATE – команда для оновлення наявних даних у базі даних;
    - INSERT – команда для додавання даних у базу даних;
    - SELECT – команда для зчитування даних з бази даних;
    - DELETE – команда для видалення наявних даних з бази даних;

## Технологічний аудит ідеї проєкту

Тестування даного застосунку – це процес дослідження програмного забезпечення з метою виявлення недоліків і визначення подібності між реальною та очікуваної поведінки програмного забезпечення який здійснюється на основі набору тестів, обраних певним чином. У більш широкому сенсі, тестування програмного забезпечення – це техніка контролю якості програмного продукту, що включає в себе проєктування тестів, виконання тестування і аналіз отриманих результатів.

На сьогоднішній день програмні продукти розробляються в дуже малі терміни і при обмежених бюджетах проектів. Програмування сьогодні перейшло з розряду мистецтва в розряд ремесел для багатьох мільйонів фахівців. На сьогоднішній день можно констатувати що в такому поспіху розробники часто ігнорують необхідність забезпечення захищеності своїх продуктів, піддаючи тим самим користувачів на невиправданий ризик. Контроль якості є дуже важливим в процесі розробки програмного забезпечення, тому що забезпечує зручність та надійність створюваного застосунку. В даний час існує безліч підходів і методів

для вирішення завдання тестування програмного забезпечення, але ефективне тестування складних програмних систем – процес творчий, що не зводиться до проходження суворим і чітким правилам.

Перевірка роботи клієнтської частини була проведена шляхом використання всіх можливостей користувача, в тому числі: опублікування, редагування та видалення публікацій. Опублікування інформації різних об’ємів, типів інформації, та в усіх випадках веб сервіс показав правильну роботу.

Для перевірки коректної роботи серверної частини застосунку були використані автоматичні тести, завдяки яким у майбутньому тестувати систему буде значно швидше та легше.

## Аналіз можливостей запуску стартапу

Було проведено аналіз застосунків кращих учбових закладів Європи і України. Їх було порівняно між собою там визначено їх недоліки й переваги. Вони були розроблені з урахуванням технічного завдання учбових закладів що вони представляють. Можливості даних застосунків відрізняються відповідно до потреб учбових закладів

Основними критеріями для оцінки даних застосунків було визначено:

* за допомогою опитування було виявлено що одним з основних показників є сучасність дизайну;
* за допомогою тестування було виявлено що швидкодія застосунку не гірше, або навіть краще ніж у його аналогів;
* також важливим показником для застосунку у наші дні є зручність інтерфейсу;
* також важливо відмітити використання захищеного протоколу передачі даних;
* за допомогою тестування різними шляхами була перевірена зручність даного застосунку;
* важливим показником також є крос-платформність програмного забезпечення;
* наявність збагаченого зведення сайту також є важливим для даного застосунку;

В результаті порівняння даного застосунку з його аналогами можна стверджувати що розроблений застосунку відповідає всім вимогам та може конкурувати з іншими аналогами. Опублікування інформації різних об’ємів, типів інформації, та в усіх випадках застосунку показав правильну роботу.

## Рекомендації на рахунок подальшого вдосконалення

Для вдосконалення даного застосунку було визначено такі шляхи:

* створення функції зворотного зв’язку що допоможе користувачам висловити свою думку по мікросервіс;
* створення сховища даних що допоможе при подальшій підтримці застосунку;
* покращення інтерфейсу;
* збільшення кількості можливостей;
* необхідно розробити автоматичне програмне
* забезпечення для того щоб застосунок зміг витримувати більші навантаження;
* основне дзеркало і єдиний формат URL – завдяки добре продуманою URL – адресою люди і пошукові системи можуть відразу зрозуміти про що цільова сторінка;
* перевірка файлу robots.txt i sitemap.xml – простий текстовий файл robots.txt може сильно вплинути на те як пошукові системи сканують ваш сайт. Цей текстовий файл не є обов’язковим, але містить інструкції для пошуку систем, про те як сканувати сайт і підтримується всіма основними пошуковими системами. Однак цей протокол носить чисто рекомендаційний характер і може бути проігнорований роботами скануючими застосунку якщо вони того бажають;
* перевірка битих посилань – непрацюючі посилання дратують і погано відзначаються на вашому професіоналізмі. Вони не тільки утримують відвідувачів від того, щоб вони залишалися а сайті і поверталися на нього, але також можуть негативно вплинути на загальний рейтинг в пошукових системах;
* Meta Title I Meta Description – теги заголовка відображаються на сторінках результатів пошукової системи у вигляді клікабельного заголовка для даного результату і важливі для зручності використання та соціального обміну. Тег заголовку застосунку призначений для точного і короткого опису вмісту сторінки. Google зазвичай відображає перші 50 – 60 символів тега заголовка. Якщо отримати заголовок
* завдовжки до 60 символів, дослідження показують, що можна очікувати що близько 90% заголовків будуть відображатися правильно;
* оптимізація заголовків – теги заголовків надають структуру і контекст для статті. Кожен заголовок повинен дати читачеві уявлення про інформацію, яку вони можуть почерпнути з тексту абзацу, який слідує нижче;
* перевірка редиректів на сайті – перенаправлення застосунку буде приймати адресу одного сайту і вказувати на інший. Коли хто-небудь вводить або натискає на цей оригінальний URL, він потрапляє на нову сторінку або сайт;
* перевірка мікророзмітки – це семантичний словник тегів який ви можете додати в свій HTML, щоб поліпшити спосіб, яким пошукові системи читають та представляють вашу сторінку в пошуковій видачі;
* зайвий код на сторінках сайту – це перше що сканують боти пошукової системи. Провести роботу над видаленням непотрібних рядків коду над основним текстом, який заважає змісту ключових слів;

## Висновки до розділу 5

У цьому розділі було проведено аналіз створеного застосунку що являє собою програмний застосунок реалізації сценарію аналізу рівня міжнародного співробітництва наукової організації в науково-технічній сфері на основі

веб-сервісів.

Було порівняно розроблений програмний продукт з існуючими аналогами та зроблено висновок що даний застосунок відповідає всім вимогам а також нічим не поступається своїм аналогам. Також було проведено тестування розробленого застосунку різними шляхами, як ручне так і автоматичне, з описом інструментів що були застосовані.

# ВИСНОВКИ

Була виконана магістерська дисертація, в рамках якої було створено програмний застосунок реалізації сценарію аналізу рівня міжнародного співробітництва наукової організації в науково-технічній сфері на основі веб-сервісів.

З метою створення якісного програмного продукту попередньо було проведено детальний аналіз інших подібних застосунків, з переліку найкращих і найпопулярніших у світі, з метою визначення технічного завдання для розробленого застосунку. Після детального аналізу усього вище перерахованого було визначене технічне завдання для розробленого програмного продукту. Також у ході виконання даного проекту було виконано детальний аналіз програмних інструментів з метою визначення який саме з них найкраще підходить для реалізації даного застосунку. Було обрано такі інструменти розробки як мова програмування С#, фреймворк для розробки серверної частини - ASP.NET MVC, фреймворк для розробки клієнтської частини, та система керування базами даних. У даному проекті було створено архітектуру клієнтської та серверної частини застосунку. Шляхом впровадження системи авторизації було реалізовано розділення функціональних можливостей користувачів, а також захист даних від

сторонніх користувачів.

В ході розробки даного застосунку було виконано: а) досліджено літературу за темою роботи;

б) досліджено можливості мови програмування С#; в) здійснено проектування макетів застосунку;

г) розроблено технічну специфікацію;

Таким чином, під час виконання дипломної роботи було розроблено готовий застосунок, отримано нові навички й покращено старі в розробці веб сервісів, проектування дизайну, побудові архітектури та роботі з API.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jagiellonian University – Homepage [Електронний ресурс]. — Режим доступу до ресурсу: https://en.uj.edu.pl.
2. Technical University of Munich – Homepage [Електронний ресурс]. — Режим доступу до ресурсу: https://[www.tum.de.](http://www.tum.de/)
3. НМУ ім. О.О. Богомольця – Головна [Електронний ресурс]. — Режим доступу до ресурсу: https://nmuofficial.com.
4. Eric Matthes. Python crash course. No starch press, 2016.
5. Mark J.Price. C# 8.0 and .NET Core 3.0.
6. Cay S. Horstmann. Core Java Volume I—Fundamentals, 2018.
7. Eloquent JavaScript: A Modern Introduction to programming
8. Advantages And Disadvantages of RDBMS [Електронний ресурс]. — Режим доступу до ресурсу: [http://tiny.cc/4mo37y.](http://tiny.cc/4mo37y)
9. An Overview of SQL and NoSQL with it’s Pros and Cons [Електронний ресурс]. — Режим доступу до ресурсу: [http://tiny.cc/4mo37y.](http://tiny.cc/4mo37y)
10. Andrew Comeau. MySQL Explained: Your Step By Step Guide to Database Design. 2nd Edition, 2015
11. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems [Електронний ресурс]. — Режим доступу до ресурсу: [http://tiny.cc/ptp37y.](http://tiny.cc/ptp37y)
12. Di Battista, Giuseppe, Eades, Peter; Tamassia, Roberto, Tollis, Ioannis G. (1994), "Algorithms for Drawing Graphs: an Annotated Bibliography", Computational Geometry: Theory and Applications.
13. Thomas M. J. Fruchterman, Edward M. Reingold (1991). *Graph drawing by forcedirected placement*.
14. Client Server Architecture [Електронний ресурс]. ¾ Режим доступу до ресурсу: https://cio-wiki.org/wiki/Client\_Server\_Architecture.
15. Chris Northwood. The Full Stack Developer: Your Essential Guide to the Everyday Skills Expected of a Modern Full Stack Web Developer, 2018.
16. Hans-Jurgen Schonig. Mastering PostgreSQL 12: Advanced Techniques to Build and Administer Scalable and Reliable PostgreSQL Database Applications, 3rd Edition, 2019.
17. Ethan Brown. Web Development with Node and Express: Leveraging the JavaScript Stack, 2014.
18. Ben Hammersley. Developing feeds with RSS and Atom, 2005.

# ДОДАТОК А

Лістинг програми

Програмний застосунок реалізації сценарію аналізу рівня міжнародного співробітництва наукової організації в науково-технічній сфері на основі веб-сервісів.

УКР.НТУУ«КПІ»\_НН ІАТЕ\_ІПЗЕ\_ТВ-12мп

Аркушів 41

2022

MetroSetBadgeActionList.cs using MetroSet\_UI.Components; using MetroSet\_UI.Controls; using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

public class MetroSetBadgeActionList : DesignerActionList

{

private readonly MetroSetBadge \_metroSetBadge;

public MetroSetBadgeActionList(IComponent component) : base(component)

{

\_metroSetBadge = (MetroSetBadge)component;

}

public Style Style

{

get => \_metroSetBadge.Style;

set => \_metroSetBadge.Style = value;

}

public string ThemeAuthor => \_metroSetBadge.ThemeAuthor;

public string ThemeName => \_metroSetBadge.ThemeName; public StyleManager StyleManager

{

get => \_metroSetBadge.StyleManager;

set => \_metroSetBadge.StyleManager = value;

}

public string Text

{

get => \_metroSetBadge.Text;

set => \_metroSetBadge.Text = value;

}

public Font Font

{

get => \_metroSetBadge.Font;

set => \_metroSetBadge.Font = value;

}

public BadgeAlign BadgeAlignment

{

get => \_metroSetBadge.BadgeAlignment;

set => \_metroSetBadge.BadgeAlignment = value;

}

public string BadgeText

{

get => \_metroSetBadge.BadgeText;

set => \_metroSetBadge.BadgeText = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

var items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Text", "Text", "Appearance", "Gets or sets the The text associated with the control."),

new DesignerActionPropertyItem("Font", "Font", "Appearance", "Gets or sets the The font associated with the control."),

new DesignerActionHeaderItem("Badge"),

new DesignerActionPropertyItem("BadgeText", "BadgeText", "Badge", "Gets or sets the badge text associated with the control."),

new DesignerActionPropertyItem("BadgeAlignment", "BadgeAlignment", "Badge", "Gets or sets the badge alignment associated with the control.")

};

return items;

}

}

}

MetroSetButtonActionList.cs using MetroSet\_UI.Components; using MetroSet\_UI.Controls;

using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

public class MetroSetButtonActionList : DesignerActionList

{

private readonly MetroSetButton \_metroSetButton;

public MetroSetButtonActionList(IComponent component) : base(component)

{

\_metroSetButton = (MetroSetButton)component;

}

public Style Style

{

get => \_metroSetButton.Style;

set => \_metroSetButton.Style = value;

}

public string ThemeAuthor => \_metroSetButton.ThemeAuthor;

public string ThemeName => \_metroSetButton.ThemeName;

public StyleManager StyleManager

{

get => \_metroSetButton.StyleManager;

set => \_metroSetButton.StyleManager = value;

}

public string Text

{

get => \_metroSetButton.Text;

set => \_metroSetButton.Text = value;

}

public Font Font

{

get => \_metroSetButton.Font;

set => \_metroSetButton.Font = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

var items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Text", "Text", "Appearance", "Gets or sets the The text associated with the control."),

new DesignerActionPropertyItem("Font", "Font", "Appearance", "Gets or sets the The font associated with the control.")

};

return items;

}

}

}

MetroSetCheckBoxActionList.cs

using MetroSet\_UI.Components; using MetroSet\_UI.Controls; using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

internal class MetroSetRadioButtonActionList : DesignerActionList

{

private readonly MetroSetRadioButton \_metroSetRadioButton;

public MetroSetRadioButtonActionList(IComponent component) : base(component)

{

\_metroSetRadioButton = (MetroSetRadioButton)component;

}

public Style Style

{

get => \_metroSetRadioButton.Style;

set => \_metroSetRadioButton.Style = value;

}

public string ThemeAuthor => \_metroSetRadioButton.ThemeAuthor; public string ThemeName => \_metroSetRadioButton.ThemeName;

public StyleManager StyleManager

{

get => \_metroSetRadioButton.StyleManager;

set => \_metroSetRadioButton.StyleManager = value;

}

public string Text

{

get => \_metroSetRadioButton.Text;

set => \_metroSetRadioButton.Text = value;

}

public bool Checked

{

get => \_metroSetRadioButton.Checked;

set => \_metroSetRadioButton.Checked = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

var items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Text", "Text", "Appearance", "Gets or sets the The text associated with the control."),

new DesignerActionPropertyItem("Checked", "Checked", "Appearance", "Gets or sets a value indicating whether the control is checked."),

};

return items;

}

}

}

MetroSetControlBoxActionList.cs

using System.ComponentModel;

using System.ComponentModel.Design; using MetroSet\_UI.Components;

using MetroSet\_UI.Controls; using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

internal class MetroSetControlBoxActionList : DesignerActionList

{

private readonly MetroSetControlBox \_metroSetControBox;

public MetroSetControlBoxActionList(IComponent component) : base(component)

{

\_metroSetControBox = (MetroSetControlBox)component;

}

public Style Style

{

get => \_metroSetControBox.Style;

set => \_metroSetControBox.Style = value;

}

public string ThemeAuthor => \_metroSetControBox.ThemeAuthor;

public string ThemeName => \_metroSetControBox.ThemeName; public StyleManager StyleManager

{

get => \_metroSetControBox.StyleManager;

set => \_metroSetControBox.StyleManager = value;

}

public bool MaximizeBox

{

get => \_metroSetControBox.MaximizeBox;

set => \_metroSetControBox.MaximizeBox = value;

}

public bool MinimizeBox

{

get => \_metroSetControBox.MinimizeBox;

set => \_metroSetControBox.MinimizeBox = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

DesignerActionItemCollection items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("MaximizeBox", "Enable MaximizeBox", "Appearance", "Gets or sets a value indicating whether the Maximize button is Enabled in the caption bar of the form."),

new DesignerActionPropertyItem("MinimizeBox", "Enable MinimizeBox", "Appearance", "Gets or sets a value indicating whether the Minimize button is Enabled in the caption bar of the form.")

};

return items;

}

}

}

MetroSetDefaultButtonActionList.cs

using MetroSet\_UI.Components; using MetroSet\_UI.Controls; using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

public class MetroSetDefaultButtonActionList : DesignerActionList

{

private readonly MetroSetDefaultButton \_metroSetButton;

public MetroSetDefaultButtonActionList(IComponent component) : base(component)

{

\_metroSetButton = (MetroSetDefaultButton)component;

}

public Style Style

{

get => \_metroSetButton.Style;

set => \_metroSetButton.Style = value;

}

public string ThemeAuthor => \_metroSetButton.ThemeAuthor; public string ThemeName => \_metroSetButton.ThemeName;

public StyleManager StyleManager

{

get => \_metroSetButton.StyleManager;

set => \_metroSetButton.StyleManager = value;

}

public string Text

{

get => \_metroSetButton.Text;

set => \_metroSetButton.Text = value;

}

public Font Font

{

get => \_metroSetButton.Font;

set => \_metroSetButton.Font = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

DesignerActionItemCollection items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Text", "Text", "Appearance", "Gets or sets the The text associated with the control."),

new DesignerActionPropertyItem("Font", "Font", "Appearance", "Gets or sets the The font associated with the control.")

};

return items;

}

}

}

MetroSetDefaultDividerActionList.cs

using MetroSet\_UI.Components; using MetroSet\_UI.Controls;

using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

internal class MetroSetDividerActionList : DesignerActionList

{

private readonly MetroSetDivider \_metroSetDivider;

public MetroSetDividerActionList(IComponent component) : base(component)

{

\_metroSetDivider = (MetroSetDivider)component;

}

public Style Style

{

get => \_metroSetDivider.Style;

set => \_metroSetDivider.Style = value;

}

public string ThemeAuthor => \_metroSetDivider.ThemeAuthor;

public string ThemeName => \_metroSetDivider.ThemeName; public StyleManager StyleManager

{

get => \_metroSetDivider.StyleManager;

set => \_metroSetDivider.StyleManager = value;

}

public DividerStyle Orientation

{

get => \_metroSetDivider.Orientation;

set => \_metroSetDivider.Orientation = value;

}

public int Thickness

{

get => \_metroSetDivider.Thickness;

set => \_metroSetDivider.Thickness = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

DesignerActionItemCollection items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Orientation", "Orientation", "Appearance", "Gets or sets Orientation of the control."),

new DesignerActionPropertyItem("Thickness", "Thickness", "Appearance", "Gets or sets the

divider thickness."),

};

return items;

}

}

}

MetroSetEllipseActionList.cs using MetroSet\_UI.Components; using MetroSet\_UI.Controls;

using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

public class MetroSetEllipseActionList : DesignerActionList

{

private readonly MetroSetEllipse \_metroSetEllipse;

public MetroSetEllipseActionList(IComponent component) : base(component)

{

\_metroSetEllipse = (MetroSetEllipse)component;

}

public Style Style

{

get => \_metroSetEllipse.Style;

set => \_metroSetEllipse.Style = value;

}

public string ThemeAuthor => \_metroSetEllipse.ThemeAuthor;

public string ThemeName => \_metroSetEllipse.ThemeName; public StyleManager StyleManager

{

get => \_metroSetEllipse.StyleManager;

set => \_metroSetEllipse.StyleManager = value;

}

public string Text

{

get => \_metroSetEllipse.Text;

set => \_metroSetEllipse.Text = value;

}

public Font Font

{

get => \_metroSetEllipse.Font;

set => \_metroSetEllipse.Font = value;

}

public int BorderThickness

{

get => \_metroSetEllipse.BorderThickness;

set => \_metroSetEllipse.BorderThickness = value;

}

public Size ImageSize

{

get => \_metroSetEllipse.ImageSize;

set => \_metroSetEllipse.ImageSize = value;

}

public Image Image

{

get => \_metroSetEllipse.Image;

set => \_metroSetEllipse.Image = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

DesignerActionItemCollection items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Text", "Text", "Appearance", "Gets or sets the The text associated with the control."),

new DesignerActionPropertyItem("Font", "Font", "Appearance", "Gets or sets the The font associated with the control."),

new DesignerActionPropertyItem("BorderThickness", "BorderThickness", "Appearance", "Gets or sets the border thickness associated with the control."),

new DesignerActionPropertyItem("Image", "Image", "Appearance", "Gets or sets the image associated with the control."),

new DesignerActionPropertyItem("ImageSize", "ImageSize", "Appearance", "Gets or sets the image size associated with the control."),

};

return items;

}

}

}

MetroSetLableActionList.cs using MetroSet\_UI.Components; using MetroSet\_UI.Controls; using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

class MetroSetLabelActionList : DesignerActionList

{

private readonly MetroSetLabel \_metroSetLabel;

public MetroSetLabelActionList(IComponent component) : base(component)

{

\_metroSetLabel = (MetroSetLabel)component;

}

public Style Style

{

get => \_metroSetLabel.Style;

set => \_metroSetLabel.Style = value;

}

public string ThemeAuthor => \_metroSetLabel.ThemeAuthor; public string ThemeName => \_metroSetLabel.ThemeName;

public StyleManager StyleManager

{

get => \_metroSetLabel.StyleManager;

set => \_metroSetLabel.StyleManager = value;

}

public string Text

{

get => \_metroSetLabel.Text;

set => \_metroSetLabel.Text = value;

}

public Font Font

{

get => \_metroSetLabel.Font;

set => \_metroSetLabel.Font = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

DesignerActionItemCollection items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Text", "Text", "Appearance", "Gets or sets the The text associated with the control."),

new DesignerActionPropertyItem("Font", "Font", "Appearance", "Gets or sets the The font associated with the control.")

};

return items;

}

}

}

MetroSetLinkActionList.cs using MetroSet\_UI.Components; using MetroSet\_UI.Controls; using MetroSet\_UI.Enums;

using static System.Windows.Forms.LinkLabel;

namespace MetroSet\_UI.Actions

{

class MetroSetLinkActionList : DesignerActionList

{

private readonly MetroSetLink \_metroSetLink;

public MetroSetLinkActionList(IComponent component) : base(component)

{

\_metroSetLink = (MetroSetLink)component;

}

public Style Style

{

get => \_metroSetLink.Style;

set => \_metroSetLink.Style = value;

}

public string ThemeAuthor => \_metroSetLink.ThemeAuthor; public string ThemeName => \_metroSetLink.ThemeName;

public StyleManager StyleManager

{

get => \_metroSetLink.StyleManager;

set => \_metroSetLink.StyleManager = value;

}

public string Text

{

get => \_metroSetLink.Text;

set => \_metroSetLink.Text = value;

}

public Font Font

{

get => \_metroSetLink.Font;

set => \_metroSetLink.Font = value;

}

public LinkBehavior LinkBehaviour

{

get => \_metroSetLink.LinkBehavior;

set => \_metroSetLink.LinkBehavior = value;

}

public Color LinkColor

{

get => \_metroSetLink.LinkColor;

set => \_metroSetLink.LinkColor = value;

}

public Color ActiveLinkColor

{

get => \_metroSetLink.ActiveLinkColor;

set => \_metroSetLink.ActiveLinkColor = value;

}

public Color VisitedLinkColor

{

get => \_metroSetLink.VisitedLinkColor;

set => \_metroSetLink.VisitedLinkColor = value;

}

public bool LinkVisited

{

get => \_metroSetLink.LinkVisited;

set => \_metroSetLink.LinkVisited = value;

}

public LinkCollection Links => \_metroSetLink.Links;

public override DesignerActionItemCollection GetSortedActionItems()

{

DesignerActionItemCollection items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Text", "Text", "Appearance", "Gets or sets the The text associated with the control."),

new DesignerActionPropertyItem("Font", "Font", "Appearance", "Gets or sets the The font associated with the control."),

new DesignerActionPropertyItem("LinkVisited", "LinkVisited", "Appearance", "Gets or sets a value indicating whether a link should be displayed as though it were visited."),

new DesignerActionPropertyItem("LinkColor", "LinkColor", "Appearance", "Gets or sets the color used when displaying a normal link."),

new DesignerActionPropertyItem("ActiveLinkColor", "ActiveLinkColor", "Appearance", "Gets or sets the color used to display an active link."),

new DesignerActionPropertyItem("VisitedLinkColor", "VisitedLinkColor", "Appearance", "Gets or sets the color used when displaying a link that that has been previously visited."),

new DesignerActionHeaderItem("Behaviour"),

new DesignerActionPropertyItem("LinkBehaviour", "LinkBehaviour", "Behaviour", "Gets or sets a value that represents the behavior of a link."),

new DesignerActionPropertyItem("Links", "Links", "Behaviour", "Gets the collection of links contained within the LinkLabel.")

};

return items;

}

}

}

MetroSetListBoxActionList.cs

using MetroSet\_UI.Child;

using MetroSet\_UI.Components; using MetroSet\_UI.Controls; using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

internal class MetroSetListBoxActionList : DesignerActionList

{

private readonly MetroSetListBox \_metroSetListBox;

public MetroSetListBoxActionList(IComponent component) : base(component)

{

\_metroSetListBox = (MetroSetListBox)component;

}

public Style Style

{

get => \_metroSetListBox.Style;

set => \_metroSetListBox.Style = value;

}

public string ThemeAuthor => \_metroSetListBox.ThemeAuthor; public string ThemeName => \_metroSetListBox.ThemeName;

public StyleManager StyleManager

{

get => \_metroSetListBox.StyleManager;

set => \_metroSetListBox.StyleManager = value;

}

[TypeConverter(typeof(CollectionConverter))] [DesignerSerializationVisibility(DesignerSerializationVisibility.Content)]

[Editor("System.Windows.Forms.Design.StringCollectionEditor, System.Design", "System.Drawing.Design.UITypeEditor, System.Drawing")]

public MetroSetItemCollection Items => \_metroSetListBox.Items;

public int ItemHeight

{

get => \_metroSetListBox.ItemHeight;

set => \_metroSetListBox.ItemHeight = value;

}

public bool MultiSelect

{

get => \_metroSetListBox.MultiSelect;

set => \_metroSetListBox.MultiSelect = value;

}

public bool ShowScrollBar

{

get => \_metroSetListBox.ShowScrollBar;

set => \_metroSetListBox.ShowScrollBar = value;

}

public bool ShowBorder

{

get => \_metroSetListBox.ShowBorder;

set => \_metroSetListBox.ShowBorder = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

DesignerActionItemCollection items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework", "Gets or sets the

stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Items", "Items", "Appearance", "Gets the items of the ListBox."),

new DesignerActionPropertyItem("ItemHeight", "ItemHeight", "Appearance", "Gets or sets the height of an item in the ListBox."),

new DesignerActionPropertyItem("MultiSelect", "MultiSelect", "Appearance", "Gets or sets a value indicating whether the ListBox supports multiple rows."),

new DesignerActionPropertyItem("ShowScrollBar", "ShowScrollBar", "Appearance", "Gets or sets a value indicating whether the vertical scroll bar is shown or not."),

new DesignerActionPropertyItem("ShowBorder", "ShowBorder", "Appearance", "Gets or sets a value indicating whether the border shown or not."),

};

return items;

}

}

}

MetroSetNumericActionList.cs using MetroSet\_UI.Components; using MetroSet\_UI.Controls;

using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

internal class MetroSetNumericActionList : DesignerActionList

{

private readonly MetroSetNumeric \_metroSetNumeric;

public MetroSetNumericActionList(IComponent component) : base(component)

{

\_metroSetNumeric = (MetroSetNumeric)component;

}

public Style Style

{

get => \_metroSetNumeric.Style;

set => \_metroSetNumeric.Style = value;

}

public string ThemeAuthor => \_metroSetNumeric.ThemeAuthor; public string ThemeName => \_metroSetNumeric.ThemeName;

public StyleManager StyleManager

{

get => \_metroSetNumeric.StyleManager;

set => \_metroSetNumeric.StyleManager = value;

}

public int Maximum

{

get => \_metroSetNumeric.Maximum;

set => \_metroSetNumeric.Maximum = value;

}

public int Minimum

{

get => \_metroSetNumeric.Minimum;

set => \_metroSetNumeric.Minimum = value;

}

public int Value

{

get => \_metroSetNumeric.Value;

set => \_metroSetNumeric.Value = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

DesignerActionItemCollection items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

number of the Numeric."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Value", "Value", "Appearance", "Gets or sets the current

new DesignerActionPropertyItem("Minimum", "Minimum", "Appearance", "Gets or sets the

minimum number of the Numeric."),

new DesignerActionPropertyItem("Maximum", "Maximum", "Appearance", "Gets or sets the maximum number of the Numeric."),

};

return items;

}

}

}

MetroSetProgressBarActionList.cs

using MetroSet\_UI.Components; using MetroSet\_UI.Controls; using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

internal class MetroSetProgressBarActionList : DesignerActionList

{

private readonly MetroSetProgressBar \_metroSetProgressBar;

public MetroSetProgressBarActionList(IComponent component) : base(component)

{

\_metroSetProgressBar = (MetroSetProgressBar)component;

}

public Style Style

{

get => \_metroSetProgressBar.Style;

set => \_metroSetProgressBar.Style = value;

}

public string ThemeAuthor => \_metroSetProgressBar.ThemeAuthor; public string ThemeName => \_metroSetProgressBar.ThemeName;

public StyleManager StyleManager

{

get => \_metroSetProgressBar.StyleManager;

set => \_metroSetProgressBar.StyleManager = value;

}

public int Value

{

get => \_metroSetProgressBar.Value;

set => \_metroSetProgressBar.Value = value;

}

public int Maximum

{

get => \_metroSetProgressBar.Maximum;

set => \_metroSetProgressBar.Maximum = value;

}

public int Minimum

{

get => \_metroSetProgressBar.Minimum;

set => \_metroSetProgressBar.Minimum = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

DesignerActionItemCollection items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Minimum", "Minimum", "Appearance", "Gets or sets the minimum value of the progressbar."),

new DesignerActionPropertyItem("Maximum", "Maximum", "Appearance", "Gets or sets the maximum value of the progressbar."),

new DesignerActionPropertyItem("Value", "Value", "Appearance", "Gets or sets the current position of the progressbar."),

};

return items;

}

}

}

MetroSetRadiobuttonActionList.cs

using MetroSet\_UI.Components; using MetroSet\_UI.Controls;

using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

internal class MetroSetCheckBoxActionList : DesignerActionList

{

private readonly MetroSetCheckBox \_metroSetCheckBox;

public MetroSetCheckBoxActionList(IComponent component) : base(component)

{

\_metroSetCheckBox = (MetroSetCheckBox)component;

}

public Style Style

{

get => \_metroSetCheckBox.Style;

set => \_metroSetCheckBox.Style = value;

}

public string ThemeAuthor => \_metroSetCheckBox.ThemeAuthor; public string ThemeName => \_metroSetCheckBox.ThemeName;

public StyleManager StyleManager

{

get => \_metroSetCheckBox.StyleManager;

set => \_metroSetCheckBox.StyleManager = value;

}

public string Text

{

get => \_metroSetCheckBox.Text;

set => \_metroSetCheckBox.Text = value;

}

public bool Checked

{

get => \_metroSetCheckBox.Checked;

set => \_metroSetCheckBox.Checked = value;

}

public SignStyle SignStyle

{

get => \_metroSetCheckBox.SignStyle;

set => \_metroSetCheckBox.SignStyle = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

DesignerActionItemCollection items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Text", "Text", "Appearance", "Gets or sets the The text associated with the control."),

new DesignerActionPropertyItem("Checked", "Checked", "Appearance", "Gets or sets a value indicating whether the control is checked."),

new DesignerActionPropertyItem("SignStyle", "SignStyle", "Appearance", "Gets or sets the the

sign style of check.")

};

return items;

}

}

}

MetroSetRichTextBoxActionList.cs

using MetroSet\_UI.Components; using MetroSet\_UI.Controls; using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

internal class MetroSetRichTextBoxActionList : DesignerActionList

{

private readonly MetroSetRichTextBox \_metroSetRichTextBox;

public MetroSetRichTextBoxActionList(IComponent component) : base(component)

{

\_metroSetRichTextBox = (MetroSetRichTextBox)component;

}

public Style Style

{

get => \_metroSetRichTextBox.Style;

set => \_metroSetRichTextBox.Style = value;

}

public string ThemeAuthor => \_metroSetRichTextBox.ThemeAuthor;

public string ThemeName => \_metroSetRichTextBox.ThemeName; public StyleManager StyleManager

{

get => \_metroSetRichTextBox.StyleManager;

set => \_metroSetRichTextBox.StyleManager = value;

}

public string Text

{

get => \_metroSetRichTextBox.Text;

set => \_metroSetRichTextBox.Text = value;

}

public Font Font

{

get => \_metroSetRichTextBox.Font;

set => \_metroSetRichTextBox.Font = value;

}

public bool ReadOnly

{

get => \_metroSetRichTextBox.ReadOnly;

set => \_metroSetRichTextBox.ReadOnly = value;

}

public ContextMenuStrip ContextMenuStrip

{

get => \_metroSetRichTextBox.ContextMenuStrip;

set => \_metroSetRichTextBox.ContextMenuStrip = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

DesignerActionItemCollection items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Text", "Text", "Appearance", "Gets or sets the The text associated with the control."),

new DesignerActionPropertyItem("Font", "Font", "Appearance", "Gets or sets the The font associated with the control."),

new DesignerActionPropertyItem("ReadOnly", "ReadOnly", "Appearance", "Gets or sets a value indicating whether text in the rich text box is read-only."),

new DesignerActionPropertyItem("ContextMenuStrip", "ContextMenuStrip", "Appearance", "Gets or sets the ContextMenuStrip associated with this control."),

};

return items;

}

}

}

MetroSetScrollBarActionList.cs using MetroSet\_UI.Components; using MetroSet\_UI.Controls;

using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

class MetroSetScrollBarActionList : DesignerActionList

{

private readonly MetroSetScrollBar \_metroSetScrollBar;

public MetroSetScrollBarActionList(IComponent component) : base(component)

{

\_metroSetScrollBar = (MetroSetScrollBar)component;

}

public Style Style

{

get => \_metroSetScrollBar.Style;

set => \_metroSetScrollBar.Style = value;

}

public string ThemeAuthor => \_metroSetScrollBar.ThemeAuthor; public string ThemeName => \_metroSetScrollBar.ThemeName;

public StyleManager StyleManager

{

get => \_metroSetScrollBar.StyleManager;

set => \_metroSetScrollBar.StyleManager = value;

}

public int Maximum

{

get => \_metroSetScrollBar.Maximum;

set => \_metroSetScrollBar.Maximum = value;

}

public int Minimum

{

get => \_metroSetScrollBar.Minimum;

set => \_metroSetScrollBar.Minimum = value;

}

public int Value

{

get => \_metroSetScrollBar.Value;

set => \_metroSetScrollBar.Value = value;

}

public int SmallChange

{

get => \_metroSetScrollBar.SmallChange;

set => \_metroSetScrollBar.SmallChange = value;

}

public int LargeChange

{

get => \_metroSetScrollBar.LargeChange;

set => \_metroSetScrollBar.LargeChange = value;

}

public ScrollOrientate Orientation

{

get => \_metroSetScrollBar.Orientation;

set => \_metroSetScrollBar.Orientation = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

DesignerActionItemCollection items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

scroll bar orientation."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Orientation", "Orientation", "Appearance", "Gets or sets the

new DesignerActionHeaderItem("Behavior"),

new DesignerActionPropertyItem("Maximum", "Maximum", "Behavior", "Gets or sets the upper limit of the scrollable range."),

new DesignerActionPropertyItem("Minimum", "Minimum", "Behavior", "Gets or sets the lower limit of the scrollable range."),

new DesignerActionPropertyItem("Value", "Value", "Behavior", "Gets or sets a numeric value that represents the current position of the scroll bar box."),

new DesignerActionPropertyItem("LargeChange", "LargeChange", "Behavior", "Gets or sets the distance to move a scroll bar in response to a large scroll command."),

new DesignerActionPropertyItem("SmallChange", "SmallChange", "Behavior", "Gets or sets the distance to move a scroll bar in response to a small scroll command."),

};

return items;

}

}

}

MetroSetSwitchActionList.cs using MetroSet\_UI.Components; using MetroSet\_UI.Controls;

using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

internal class MetroSetSwitchActionList : DesignerActionList

{

private readonly MetroSetSwitch \_metroSetSwitch;

public MetroSetSwitchActionList(IComponent component) : base(component)

{

\_metroSetSwitch = (MetroSetSwitch)component;

}

public Style Style

{

get => \_metroSetSwitch.Style;

set => \_metroSetSwitch.Style = value;

}

public string ThemeAuthor => \_metroSetSwitch.ThemeAuthor;

public string ThemeName => \_metroSetSwitch.ThemeName; public StyleManager StyleManager

{

get => \_metroSetSwitch.StyleManager;

set => \_metroSetSwitch.StyleManager = value;

}

public string Text

{

get => \_metroSetSwitch.Text;

set => \_metroSetSwitch.Text = value;

}

public bool Switched

{

get => \_metroSetSwitch.Switched;

set => \_metroSetSwitch.Switched = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

DesignerActionItemCollection items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Text", "Text", "Appearance", "Gets or sets the The text associated with the control."),

new DesignerActionPropertyItem("Switched", "Switched", "Appearance", "Gets or sets a value indicating whether the control is switched."),

};

return items;

}

}

}

MetroSetTileActionList.cs using MetroSet\_UI.Components; using MetroSet\_UI.Controls; using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

public class MetroSetTileActionList : DesignerActionList

{

private readonly MetroSetTile \_metroSetTile;

public MetroSetTileActionList(IComponent component) : base(component)

{

\_metroSetTile = (MetroSetTile)component;

}

public Style Style

{

get => \_metroSetTile.Style;

set => \_metroSetTile.Style = value;

}

public string ThemeAuthor => \_metroSetTile.ThemeAuthor; public string ThemeName => \_metroSetTile.ThemeName;

public StyleManager StyleManager

{

get => \_metroSetTile.StyleManager;

set => \_metroSetTile.StyleManager = value;

}

public string Text

{

get => \_metroSetTile.Text;

set => \_metroSetTile.Text = value;

}

public Font Font

{

get => \_metroSetTile.Font;

set => \_metroSetTile.Font = value;

}

public Image BackgroundImage

{

get => \_metroSetTile.BackgroundImage;

set => \_metroSetTile.BackgroundImage = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

DesignerActionItemCollection items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Appearance"),

new DesignerActionPropertyItem("Text", "Text", "Appearance", "Gets or sets the The text associated with the control."),

new DesignerActionPropertyItem("Font", "Font", "Appearance", "Gets or sets the The font associated with the control."),

new DesignerActionPropertyItem("BackgroundImage", "BackgroundImage", "Appearance", "Gets or sets the BackgroundImage associated with the control."),

};

return items;

}

}

}

MetroSetToolTipActionList.cs

using System.ComponentModel;

using System.ComponentModel.Design; using MetroSet\_UI.Components;

using MetroSet\_UI.Enums;

namespace MetroSet\_UI.Actions

{

internal class MetroSetToolTipActionList : DesignerActionList

{

private readonly MetroSetSetToolTip \_metroSetSetToolTip;

public MetroSetToolTipActionList(IComponent component) : base(component)

{

\_metroSetSetToolTip = (MetroSetSetToolTip)component;

}

public Style Style

{

get => \_metroSetSetToolTip.Style;

set => \_metroSetSetToolTip.Style = value;

}

public string ThemeAuthor => \_metroSetSetToolTip.ThemeAuthor; public string ThemeName => \_metroSetSetToolTip.ThemeName;

public StyleManager StyleManager

{

get => \_metroSetSetToolTip.StyleManager;

set => \_metroSetSetToolTip.StyleManager = value;

}

public bool Active

{

get => \_metroSetSetToolTip.Active;

set => \_metroSetSetToolTip.Active = value;

}

public int AutomaticDelay

{

get => \_metroSetSetToolTip.AutomaticDelay;

set => \_metroSetSetToolTip.AutomaticDelay = value;

}

public int AutoPopDelay

{

get => \_metroSetSetToolTip.AutoPopDelay;

set => \_metroSetSetToolTip.AutoPopDelay = value;

}

public int InitialDelay

{

get => \_metroSetSetToolTip.InitialDelay;

set => \_metroSetSetToolTip.InitialDelay = value;

}

public bool StripAmpersands

{

get => \_metroSetSetToolTip.StripAmpersands;

set => \_metroSetSetToolTip.StripAmpersands = value;

}

public bool UseAnimation

{

get => \_metroSetSetToolTip.UseAnimation;

set => \_metroSetSetToolTip.UseAnimation = value;

}

public bool UseFading

{

get => \_metroSetSetToolTip.UseFading;

set => \_metroSetSetToolTip.UseFading = value;

}

public override DesignerActionItemCollection GetSortedActionItems()

{

DesignerActionItemCollection items = new DesignerActionItemCollection

{

new DesignerActionHeaderItem("MetroSet Framework"),

new DesignerActionPropertyItem("StyleManager", "StyleManager", "MetroSet Framework",

"Gets or sets the stylemanager for the control."),

new DesignerActionPropertyItem("Style", "Style", "MetroSet Framework", "Gets or sets the

style."),

new DesignerActionHeaderItem("Informations"),

new DesignerActionPropertyItem("ThemeName", "ThemeName", "Informations", "Gets or sets the The Theme name associated with the theme."),

new DesignerActionPropertyItem("ThemeAuthor", "ThemeAuthor", "Informations", "Gets or sets the The Author name associated with the theme."),

new DesignerActionHeaderItem("Misc"),

new DesignerActionPropertyItem("Active", "Active", "Misc", "Gets or sets a value indicating whether the ToolTip is currently active."),

new DesignerActionPropertyItem("AutomaticDelay", "AutomaticDelay", "Misc", "Gets or sets the automatic delay for the ToolTip."),

new DesignerActionPropertyItem("AutoPopDelay", "AutoPopDelay", "Misc", "Gets or sets the period of time the ToolTip remains visible if the pointer is stationary on a control with specified ToolTip text."),

new DesignerActionPropertyItem("InitialDelay", "InitialDelay", "Misc", "Gets or sets the time that passes before the ToolTip appears."),

new DesignerActionPropertyItem("StripAmpersands", "StripAmpersands", "Misc", "Gets or sets a value that determines how ampersand (&) characters are treated."),

new DesignerActionPropertyItem("UseAnimation", "UseAnimation", "Misc", "Gets or sets a value determining whether an animation effect should be used when displaying the ToolTip."),

new DesignerActionPropertyItem("UseFading", "UseFading", "Appearance", "Gets or sets a value determining whether a fade effect should be used when displaying the ToolTip."),

};

return items;

}

}

}

# ДОДАТОК Б

Презентація продутку

Програмний застосунок реалізації сценарію аналізу рівня міжнародного співробітництва наукової організації в науково-технічній сфері на основі веб-сервісів.

УКР.НТУУ«КПІ»\_НН ІАТЕ\_ІПЗЕ\_ТВ-12мп

Аркушів 13

2022

