Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій

(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки

(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)

(рівень вищої освіти)

(тема)

РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБРОБКИ ДАНИХ З РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

Виконав: студент 2 курсу, гр. АУТПм-19-1

Біленець Олексій Олександрович

(прізвище, ініціали)

Спеціальність

151 Автоматизація та комп'ютерно-інтегровані технології

освітньої програми Автоматизоване управління технологічними процесами

(код і повна назва напряму)

Тип програми освітньо-професійна

(повна назва освітньої програми)

Керівник Доцент Бабак І. М.

(посада, прізвище, ініціали)

Допускається до захисту

зав. кафедри _____ Невлюдов І.Ш.

(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет	Автоматики і комп'ютеризованих технологій
Кафедра	Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки
Рівень вищої освіти	другий (магістерський)
Спеціальність	Автоматизація та комп'ютерно-інтегровані технології
Тип програми	освітньо-професійна
Освітня програма	151 Автоматизоване управління технологічними процесами

(код і повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 2020 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Біленець Олексію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка автоматизованої системи обробки даних з розподілених інформаційних джерел
затверджена наказом по університету від 02.11. 2020 р. № 1510 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 16.12. 2020 р.
3. Вихідні дані до роботи JavaScript, HTML, CSS; наявність локального серверу Intel Pentium не менше 1,7ГГц; RAM – 2Гб; жорсткий диск – не менше 250 Гб; ОС Windows 10.

4. Перелік питань, що потрібно опрацювати в роботі:
4.1 Аналіз сучасного стану питання та обґрунтування технічного завдання
4.2 Розробка архітектури системи
4.3 Вибір шаблону проектування системи
4.4 Розробка автоматизованої системи
4.5 Розробка внутрішньої системи
4.6 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – 12 с. формату А4

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Керівник (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз сучасного стану питання та обґрунтування технічного завдання	20.09–26.09	виконано
2	Розробка архітектури системи	08.10–22.10	виконано
3	Розробка автоматизованої системи	01.11–26.11	виконано
4	Розробка програмного забезпечення системи	26.11–01.12	виконано
5	Оформлення пояснювальної записки	01.12–11.12	виконано
6	Подання у ЕК	11.12–16.12	виконано

Дата видачі завдання

Студент _____ Біленець О.О.

(підпис) (прізвище, ініціали)

Керівник роботи _____ _____

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи магістра містить 73 с.,16 рис., 3 табл., 28 джерел.

АВТОМАТИЗАЦІЯ, ОБРОБКА ДАНИХ, API, АВТОМАТИЗОВАНА СИСТЕМА, ІНФОРМАЦІЙНІ ДЖЕРЕЛА.

Об’єкт дослідження – процес автоматизації збору та обробки даних з розподілених джерел.

Методи дослідження – збір та обробка даних, парсинг даних, методи розробки веб-ресурсу.

Мета – скорочення часу отримання інформацій з розподілених інформаційних джерел за рахунок використання автоматизованої системи.

В ході виконання атестаційної роботи розроблено автоматизовану систему для збору та обробки даних з розподілених інформаційних джерел. Система представлена у вигляді веб-додатку, на якому користувачеві виводяться дані обрані для аналізу з веб-ресурсів, за параметрам, що вказую користувач. Проаналізовано аналоги даної системи, для програмної реалізації обрано мову програмування JavaScript, та фреймворк Vue.Js.

ABSTRACT

The explanatory note of the master's attestation work contains 73 pages, 16 figures, 3 tables, 28 sources.

AUTOMATION, DATA PROCESSING, API, AUTOMATED SYSTEM, INFORMATION SOURCES.

The purpose of the work is to develop an automated system that will collect data from various sources.

The object of research is the process of automating the collection and processing of data from distributed sources.

Research methods - data collection and processing.

The goal is to reduce the time of receiving information from distributed information sources through the use of an automated system.

During the certification work, an automated system for collecting and processing data from distributed information sources was developed. The web application shows an automated system on the example of collecting data from weather forecast sites.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface

JS – JavaScript

СОД – система обробки даних

ІС – інформаційна система

ВСТУП

Основні функції СОД (система обробки даних) полягають у зборі, видачі, накопиченні, збереженні та обробці великих об'ємів інформації. Збір інформації проводиться різного роду периферійними засобами, наприклад через канали зв'язку за допомогою модемів, локальні та глобальні комп'ютерні мережі, різного роду датчиків, що встроєні в технологічних виробничих лініях, а також з допомогою клавіатури та монітора. Накопичення та збереження інформації, забезпечується засобами збереження на жорстких магнітних, компакт лазерних і оптичних дисках. Обробка інформації виконується з допомогою центрального процесора та програмного забезпечення, що фактично керує роботою процесора для вирішення заданої проблеми [1].

Таким чином комплекс засобів збору та видачі інформації виконує зв’язок між, СОД і навколишнім середовищем.

Зазвичай, автоматизовані ІС (інформаційна система) включають автоматизовані робочі місця (АРМ) фахівців, засоби комунікації і обміну інформацією. Все це дозволяє ефективно автоматизувати роботу персоналу. Ефективність застосування ІС для управління економічними об'єктами (підприємствами, банками, торговими організаціями, державними установами і т.д.) залежить від здатності оперативно готувати управлінські рішення, адаптуватися до змін зовнішнього середовища та інформаційних потреб користувачів [2].

Світова практика переконливо свідчить, що корпоративні системи – потужний інструмент підвищення продуктивності праці і ефективності виробництва. Всесвітнім економічним форумом спільно з міжнародною школою бізнесу INSEAD, був відзначений тісний зв'язок між рівнем розвитку інформаційних технологій і економічним процвітанням країн на підставі того, що інформаційні технології відіграють провідну роль у розвитку інновацій, підвищенні продуктивності та конкурентоспроможності, диверсифікують

економіку і стимулюють ділову активність, тим самим сприяючи підвищенню рівня життя людей. У багатьох розвинених країнах галузь інформаційних технологій розвивається бурхливими темпами і стає запорукою сталого економічного розвитку.

Корпоративні інформаційні системи, призначені для автоматизації різних видів господарського обліку та управління корпорацією можна умовно поділити на три класи: локальні системи, середні інтегровані системи, великі інтегровані системи [3].

Локальні системи успішно справляються з вирішенням окремих задач обліку на підприємстві, але, як правило, не надають цілісної інформації для автоматизації управління. Перевагою цих систем є порівняно невисока ціна і відносна простота впровадження.

Прикладом середніх інтегрованих систем можуть бути системи «AVACO SOFT», «ABACUS Financial», широко використовувані у вітчизняній практиці «Галактика», «ПАРУС», «1С:Підприємство», «Регістри».

API – це прикладний програмний інтерфейс (англ. Application programming interface) спрощений спосіб підключити власну інфраструктуру за допомогою розробки власних хмарних додатків, але вони також дозволяють ділитися своїми даними із клієнтами та іншими зовнішніми користувачами. Загальнодоступні API представляють унікальну ділову цінність, оскільки вони можуть спростити та розширити спосіб зв’язку зі своїми партнерами, а також потенційно монетизувати дані (популярним прикладом є API Карт Google).

У великих організаціях та корпораціях є задача збору даних з інтеренет-ресурсів, завдяки автоматизованій системі можуть збирати та обробляти необхідні дані, та використовувати їй в своїх цілях, такими корпораціями е Facebook і Twitter, а також Google, ці корпорації користуються даними системами щодня, тому актуальність даного види автоматизованої системи зростає з кожним днем.

Таким чином метою магістерської випускної атестаційної роботи являється розробка автоматизованої системи обробки даних з розподілених інформаційних

джерел.

Об’єкт дослідження – процес автоматизації збору та обробки даних з розподілених джерел.

Предмет дослідження – розроблення автоматизованої системи збору даних з веб ресурсів.

Методи дослідження – збір та обробка даних, парсинг даних, методи розробки веб-ресурсу.

Мета – скорочення часу отримання інформацій з розподілених інформаційних джерел за рахунок використання автоматизованої системи.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати особливості проектування та розробки системи автоматизації;

– розробити алгоритми автоматизованої системи;

– розробити архітектуру системи обробки даних з розподілених інформаційних джерел;

– розробити веб-додаток для збору інформації з веб ресурсів та відображенні її для користувача;

– провести експериментальні дослідження створеного продукту;

– провести розрахунок основних виробничих факторів, які повинні задовольняти умовам роботи;

– оформити пояснювальну записку згідно з рекомендаціями, та вимогами ДСТУ 3008:2015 [2].

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ ТА ОБГРУНТУВАННЯ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Обґрунтування необхідності використання API

Web API – це прикладний програмний інтерфейс (англ. Application programming interface, API) для веб-сервера.

API визначає функціональність, яку надає програма (модуль, бібліотека), при цьому API дозволяє абстрагуватися від того, як саме ця функціональність реалізована.

API може розглядатися як абстракція над функціональністю і реалізацією компонентів. Іншими словами, API посилається на набір функцій, вбудованих в додатки, які можуть бути використані іншими додатками (або сам по собі), щоб взаємодіяти з додатком. API є відмінним способом надійно і безпечно розкрити функціональність програми для зовнішніх додатків. Всі функціональні можливості, доступні зовнішнім додаткам, обмежуються наданим API [5].

Необхідність та популярність використання API обумовлена такими факторами:

– у 2017 році до 75% інтернет трафіка припадає на мобільні пристрої. безліч мобільних додатків для різних сервісів працюють при використанні арі цих самих сервісів. це дозволяє стандартизувати та уніфікувати процес розробки додатків;

– тенденції розвитку open-source, тобто, відкритого програмного забезпечення якщо у вашого сервісу склалася певна аудиторія, яка користується ним, чому б не обернути це собі на користь і на користь аудиторії, звичайно ж, теж за допомогою арі користувачі при бажанні зможуть створити нові клієнти для вашого додатку, нові сервіси на його основі, розкрити нові його грані;

– максимальне розділення frontend (зовнішнє представлення, інтерфейс користувача) та backend (його програмна реалізація). що значно спрощує розробку додатків [5].

API-орієнтований веб-додаток – це веб-додаток, в якому весь або більшу частину функціоналу реалізується через виклики API. Наприклад, якщо користувач маєте увійти в систему, то він відправляєте свої дані через функцію API. Результатом виконання цієї функції буде або авторизація, або повідомлення про помилку. Одним з переваг створення API-орієнтованих веб-додатків, є те, що це допоможе побудувати функціональність, яка може бути використана будь-яким пристроєм, будь то браузер, мобільний телефон, планшет або навіть настільний додаток. Все, що потрібно зробити – це створити API таким чином, щоб всі ці пристрої могли взаємодіяти з ним. Таким чином можна створити централізований додаток, яке може приймати виклики з будь-якого пристрою, який є у користувача [6].

Принцип роботи API зображений на рисунку 1.1

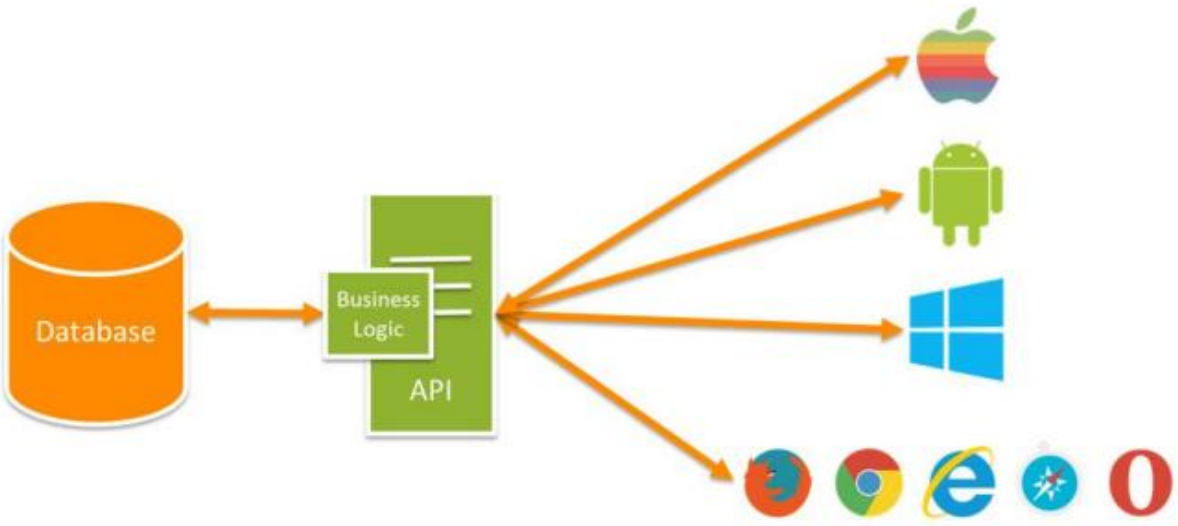

Рисунок 1.1 – Принцип роботи Web API [8]

Як бачимо з рисунку, користувач може користуватися продуктом з будь якої платформи та використовуючи будь-який браузер чи інший програмний продукт, не відчуваючи змін. API інкапсулює від кінцевого користувача

бізнеслогіку системи, надаючи йому зручний користувацький інтерфейс та реалізуючи взаємодію з базою даних «під капотом». Такий принцип роботи дозволяє повністю відділити програмну логіку від інтерфейсу користувача та забезпечити доступ до системи незалежно від того, з якого пристрою користувач намагається взаємодіяти. До того ж, така архітектура є дуже зручною для масштабування програмного забезпечення. При збільшенні кількості користувачів та створенні нових програм-клієнтів чи інтеграції у нові веб-сервіси бізнес-логіку системи не доведеться змінювати, оскільки API надає зручну оболонку для взаємодії для кожного клієнта [8].

Визначимо основні вимоги, які має задовольняти якісний та ефективний API:

– арі повинен мати стандартизований та зручний формат запитів, крім того, для підвищення швидкодії системи слід використовувати засоби кешування запитів, в свою чергу, відповіді сервера повинні мати явне чи неявне позначення як кешовані чи некешовані з метою попередження отримання клієнтами застарілих або невірних даних у відповідь на подальші запити;

– арі повинен мати стандартизований та зручний формат відповідей та підтримувати різні формати представлення даних, це дозволить забезпечити кросплатформенність системи, розширити сфери використання продукту, полегшить його популяризацію та інтеграцію в інші веб-сервіси;

– має бути передбачена можливість повернення помилок виконання запиту, причому помилки мають бути чітко описані, щоб не тільки користувач знав, що йому необхідно зробити, але й ви легко орієнтувалися, коли користувач надсилає вам запит для вирішення проблеми. але необхідно також уникати зайвої надлишковості, щоб не заплутувати користувачів;

– забезпечення захищеності та конфіденційності системи, арі має бути захищений від таких загроз як ddos-атаки, csrf-атаки;

– масштабованість. арі має бути структурований і спроектований так, щоб була можливість легко масштабувати систему або розширити її функціонал без погіршення швидкості та ефективності роботи [6].

API організований навколо REST (Representational State Transfer – передача стану уявлення). API має передбачувані URL-адреси, орієнтовані на ресурси, приймає закодовані у формі тіла, повертає відповіді, закодовані JSON, і використовує стандартні коди відповідей HTTP, автентифікацію.

Користувач можете використовувати Stripe API у тестовому режимі, який не впливає на дані в реальному часі та не взаємодіє з банківськими мережами. Ключ API, який використовується для автентифікації запиту, визначає, чи є запит у режимі реального часу або в режимі тестування [20].

JSON всюди в сучасних веб-додатках. Він легкий для читання, легкий і дуже добре працює з додатками, написаними на JavaScript. Але також порівняно легко отримати додатки, написані на інших мовах, щоб читати і генерувати їх, включаючи Java. Це означає, що до API, що повертає JSON, може отримати доступ додаток, написане на Java, Ruby, Python, JS, PHP і багатьох інших. Це робить API масштабується і незалежним від платформи [13].

API відрізняється від веб-додатку або сайту на базі бази даних (або статичних) тим, що він зазвичай не повинен містити інтерфейс – немає необхідності в HTML, CSS для відображення користувачеві через статичні сторінки або динамічно згенеровані шаблони, які об'єднують дані з багаторазовими макетами.

Запити на отримання або запис даних зазвичай виконуються без зовнішнього інтерфейсу, шляхом відправки HTTP-запиту на сервер.

API не повинен пояснювати, що відбувається всередині двигуна, коли водій натискаєте педаль газу. Ось чому, якщо людина навчилися водити автомобіль з двигуном внутрішнього згоряння, вона можете сісти за кермо електромобіля, не вивчаючи абсолютно новий набір навичок. API визначає можливу функціональність, яку програма в формі бібліотеки або модуля зможе здійснювати [20].

Слід мати на увазі, що назви деяких API-інтерфейсів часто використовуються для позначення як специфікації взаємодій, так і фактичного програмного компонента, з яким взаємодіє користувач.

Коли справа доходить до програмного забезпечення, API можна знайти буквально всюди. API йдуть рука об руку з однієї з найбільш фундаментальних концепцій інформатики: абстракцією. Абстракція – це просто спосіб упорядкувати складність системи, щоб можна було легко обробляти складні дії.

API для програмістів, це коли інші більш досвідчені програмісти беруть на себе величезну складність і визначають відносно простий набір взаємодій, який можете використовувати користувач замість того, щоб робити все це самостійно. У будь-якому проекті програми, ймовірно, безпосередньо використовуються десятки, якщо не сотні API-інтерфейсів, і кожен з цих API-інтерфейсів залежить від інших API-інтерфейсів і так далі [6].

API-інтерфейси – давня концепція в комп'ютерному програмуванні, і вони вже багато років є частиною інструментів розробників. Традиційно API-інтерфейси використовувалися для з'єднання компонентів коду, що працюють на одній машині. З ростом повсюдного поширення мереж стає доступним все більше і більше загальнодоступних API, іноді званих відкритими API. Загальнодоступні API-інтерфейси звернені назовні і доступні через Інтернет, що дозволяє писати код, який взаємодіє з кодом інших постачальників в Інтернеті. Цей процес відомий як інтеграція API.

Подібні гібридні програми коду дозволяють користувачам змішувати і зіставляти функціональні можливості різних постачальників в своїх власних системах [20].

API-інтерфейси є важливим компонентом розробки програмного забезпечення, і вони існують на кожному рівні програмного стека. Вони надають спосіб визначати абстракції і управляти ними, повідомляючи нам, що ми можемо робити з програмними компонентами і як ми можемо це робити. Добре спроектовані API-інтерфейси підтримують ефективне, плавне і легке впровадження і використання, в той час як погано спроектовані API-інтерфейси, як правило, викликають головний біль при кожному використанні [6].

1.2 Аналіз існуючих підходів вирішення технічної проблеми

На даний момент вже існує безліч різних типів API для додатків, вебсайтів і операційних систем.

Серед визначених класів є популярні JS API і інтерфейси, які дозволяють певним суб'єктам обмінюватися інформацією на мові програмування JavaScript.

Також є і Web API.

Найвідоміші типи API:

– віддалений виклик процедур (Remote Procedure Call – RPC);

– простий протокол доступу до об'єктів (Simple Object Access Protocol – SOAP);

– передача стану уявлення (Representational State Transfer – REST) [11].

На даний момент існують два види API: публічні та приватні.

Публічні API випускаються такими компаніями, як Slack і Shopify, в надії на те, що розробники будуть їх використовувати на своїх платформах. Компанії діляться набором вхідних параметрів, які розробники використовують, щоб досягти якогось результату.

Публічне API можна використовувати без проблеми доступ до документації можна отримати без проблем.

Приватний API використовуються всередині компанії. Якщо у компанії багато програмних продуктів, приватне API використовується, щоб програми спілкувались між собою. Компоненти API можуть змінюватися за бажанням компанії, тоді як зміни в публічному API може викликати великі неприємності [10].

Зараз доступні тисячі REST API практично на всіх можливих сайтах. Зазвичай для загальнодоступних даних, таких як погода або фондові ринки, користувач можете знайти десятки різних API, доступних для використання. Багато популярних веб-платформи, такі як Facebook і Twitter, також надають API для розробників. Деякі з пропрієтарних API мають обмеження на кількість звернень до них. Багато хто вимагає реєстрації і отримання закритого ключа.

Найбільш безпечні API вимагають настройки OAuth для безпечного входу користувачів.

Нижче приведен приклад публічних API, доступ до яких мають всі користувачі:

– poke api надає можливість отримати дані про 800+ покемонів;

– nasa api надає можливість отримати дані про астероїди, галактиках і багато іншого;

– open food facts величезна кількість даних про продукти харчування з усього світу;

– transloc openapi користувач отримає живі дані про громадський транспорт міст і кампусів;

– urban dictionary api документація міського словника;

– merriam-webster dictionary api для тих, кому потрібні визначення та синоніми реальних слів;

– numbers api цікаві факти і питання про числах;

– weatherbit api поточні та історичні дані про погоду;

– us government data api досить великий набір даних про сполучених штатах – сільське господарство, охорона здоров'я, громадська безпека і т.д;

– bible api пропонує найбільшу доступну колекцію біблії, найбільша історія [11].

1.2.1 Розробка автоматизованої системи за допомогою RPC

Віддалений виклик процедур, рідше виклик віддалених процедур (RPC– Remote Procedure Call) – клас технологій, що дозволяють комп'ютерним програмам викликати функції або процедури в іншому адресному просторі (на віддалених комп'ютерах, або в незалежній сторонньої системі на тому самому пристрої). Зазвичай реалізація RPC – технології включає в себе два компоненти: мережевий протокол для обміну в режимі клієнт-сервер і мова серіалізації об'єктів (або структур, для необ'єктних RPC) [16].

Різні реалізації RPC мають дуже відрізняється один від одного архітектуру і різняться в своїх можливостях: одні реалізують архітектуру SOA, інші – CORBA або DCOM.

На транспортному рівні RPC використовують в основному протоколи TCP і UDP, однак, деякі побудовані на основі HTTP (що порушує архітектуру ISO – OSI, так як HTTP – спочатку не транспортний протокол) [15].

Його принцип це – виклик віддалених процедур полягає в розширенні добре відомого і зрозумілого механізму передачі управління і даних усередині програми, що виконується на одній машині, на передачу управління і даних через мережу.

Засоби віддаленого виклику процедур призначені для полегшення організації розподілених обчислень і створення розподілених клієнт-серверних інформаційних систем. Найбільша ефективність використання RPC досягається в тих додатках, в яких існує інтерактивний зв'язок між віддаленими компонентами з невеликим часом відповідей і відносно малою кількістю переданих даних. Такі додатки називаються RPC-орієнтованими [16].

Характерними рисами виклику локальних процедур є:

– асиметричність, тобто одна з взаємодіючих сторін є ініціатором;

– синхронність, тобто виконання викликає процедури призупиняється з моменту видачі запиту і відновлюється тільки після повернення виклику процедури.

1.2.2 Розробка автоматизованої системи за допомогою SOAP API

SOAP – Simple Object Access Protocol – простий протокол доступу до об'єктів) – протокол обміну структурованими повідомленнями в розподіленої обчислювальної середовищі. Спочатку SOAP призначався в основному для реалізації віддаленого виклику процедур (RPC). Зараз протокол використовується для обміну довільними повідомленнями в форматі XML, а не тільки для виклику процедур. Офіційна специфікація останньої версії протоколу ніяк не розшифровує назву SOAP. SOAP є розширенням протоколу XML-RPC.

SOAP може використовуватися з будь-яким протоколом прикладного рівня: HTTP, HTTPS і ін. Однак його взаємодія з кожним із цих протоколів має свої особливості, які повинні бути визначені окремо. Найчастіше SOAP використовується поверх HTTP [15].

SOAP – це стандарт, який дозволяє вам описати такий віддалений виклик і вид, в якому буде повертатися результат. Таким чином користувачу потрібно розмістити необхідну функцію в додатку, доступному по мережі і отримувати виклики у вигляді SOAP пакетів. Після цього користувач валідіруе вхідні дані, запускає функцію і повертає результат в новому SOAP пакеті. Весь процес може працювати через HTTP, так що користувачу не доведеться відкривати купу портів в брандмауері. SOAP призначений для підтримки незалежного абстрактного протоколу зв'язку, що забезпечує комунікацію двох і більше додатків, сайтів, підприємств і т.п, реалізованих на різних технологіях і апаратних засобів. Стандартні запити і відповіді SOAP API відображаються у вигляді повідомлення в оболонці, що складається з чотирьох елементів з певними функціями для кожного з них. [14].

SOAP-повідомлення є XML-документ, повідомлення складається з трьох основних елементів: конверт (SOAP Envelope), заголовок (SOAP Header) і тіло (SOAP Body), розглянувши схему, зображену на рисунку 1.3.

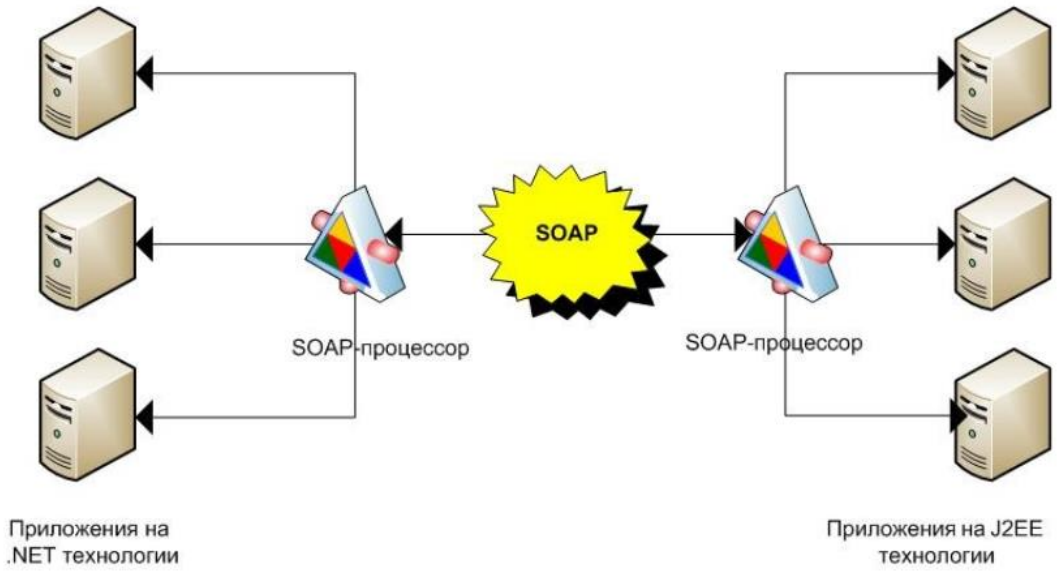

Рисунок 1.3– Архітектурна схема SOAP [14]

1.3 Обґрунтування вибору технологій розробки

Дана робота буда виконана на язику програмування JavaScript. Ця мова підходить для написання вебдодатків дуже добре, він зарекомендував себе як надійний, швидкій. Також у програмі використовується Node.Js. для налаштування віртуального серверу, для подальшого запуску веб-додатку, та puppeteer для автоматизованого отримання даних.

JavaScript являється однією з найпоширеніших мов програмування, причому її популярність продовжує зростати з кожним роком. І хоча з самого початку JavaScript позиціонувався як мова для клієнтської сторони додатку, зокрема для покращення інтерактивності сторінки, останнім часом його стали використовувати у якості серверної мови.

JavaScript – це мова сценаріїв, який використовується для створення та управління динамічним вмістом веб-сайту, т. Е Всім, що переміщається, оновлюється або іншим чином змінюється на вашому екрані без необхідності вручну перезавантажувати веб-сторінку. Такі функції, як:

– анімована графіка;
– слайд-шоу фотографій;
– пропозиції автозаповнення тексту;
– інтерактивні форми.

Ще кращий спосіб зрозуміти, що робить JavaScript треба подумати про певні веб-функції, які користувач використовуєте щодня і які, ймовірно, вважаються само собою зрозумілими, наприклад, коли тимчасова шкала Facebook автоматично оновлюється на екрані користувача або Google пропонує умови пошуку на основі декількох букв, коли користувач почав набір тексту. В обох випадках це JavaScript в дії. Згідно зі звітом платформи для розробки програмного забезпечення GitHub за 2020 рік, JavaScript впевнено посідає перше місце за популярністю серед учасників (рис. 1.4).

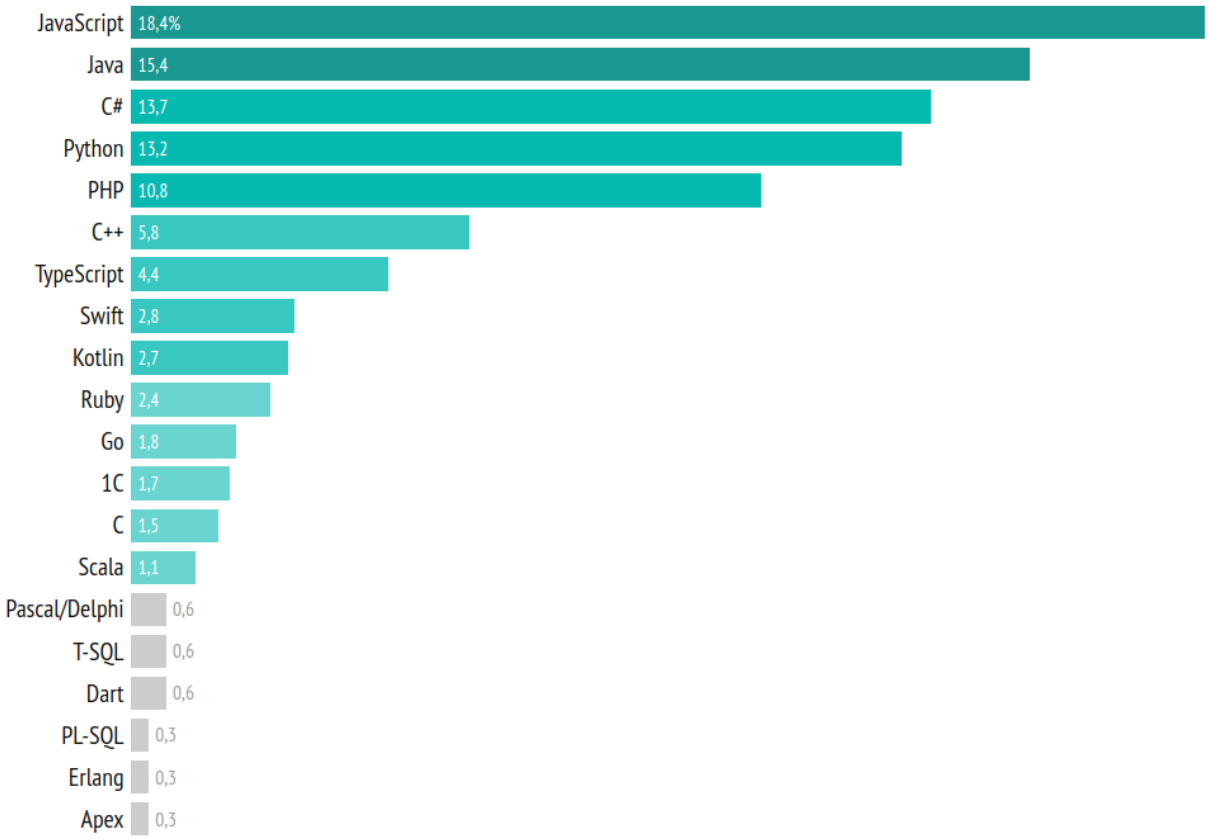

Рисунок 1.4 – Рейтинг мов програмування [4]

JavaScript неймовірно універсальний і доброзичливий до новачків. Маючи великий досвід, ви зможете створювати ігри, анімовану 2D і 3D графіку, повномасштабні програми з базами даних і багато іншого.

JavaScript сам по собі досить компактний, але дуже гнучкий. Розробниками написано велику кількість інструментів поверх основного мови JavaScript, які розблокують величезна кількість додаткових функцій з дуже невеликим зусиллям. До них відносяться:

– програмні інтерфейси додатка (API), вбудовані в браузери, що забезпечують різні функціональні можливості, такі як динамічне створення HTML і установку CSS стилів, захоплення і маніпуляція відеопотоком, робота з веб-камерою користувача або генерація 3D графіки і аудіо семплів;

– сторонні API дозволяють розробникам впроваджувати функціональність в свої сайти від інших розробників, таких як Twitter або Facebook;

– також ви можете застосувати до вашого HTML сторонні фреймворки і бібліотеки, що дозволить вам прискорити створення сайтів і додатків [24].

У кожної мови програмування є свої недоліки і слабкі місця. Однією і причин виникнення проблем є популярність мови. Коли мова програмування стає таким популярним як JavaScript, він стає об'єктом підвищеного інтересу для хакерів, шахраїв та інших шкідливих проявів третіх сторін, які намагаються знайти уразливості і слабкі місця в безпеці. Деякі слабкі місця:

– вразливий відношенню до експлойтів (шкідливий код, який використовує уразливості програмного продукту);

– може бути використаний для запуску шкідливого коду на комп'ютері користувача;

– не завжди підтримується деякими браузерами або пристроями;

– фрагменти JS коду можуть бути дуже великими;

– може по різному відображатися на різних пристроях, що призводить до відсутності цілісності.

JavaScript – це «мова сценаріїв». Мови сценаріїв – це мови програмування, які використовуються для автоматизації процесів, які користувачам в іншому випадку довелося б виконувати самостійно, крок за кроком. За винятком сценаріїв, будь-які зміни на відвідуваних вами веб-сторінках зажадають або перезавантаження сторінки вручну, або навігації по серії статичних меню, щоб перейти до потрібного вам контенту [24].

Мова сценаріїв, такий як JavaScript (JS, для тих, хто в курсі), виконує важку роботу, повідомляючи комп'ютерним програмам, таким як веб-сайти або веб-додатки, «щось робити». У випадку з JavaScript це означає вказівку тим динамічним функцій, описаних раніше, робити те, що вони роблять - наприклад, повідомляти зображень про анімації, фотографій циклічно переміщатися по

слайд-шоу або автозаповнення пропозицій у відповідь на запити. Це «сценарій» в JavaScript, який змушує все це відбуватиметься, здавалося б, самостійно.

Тим часом, оскільки JavaScript є невід'ємною частиною веб-функціональності, всі основні веб-браузери мають вбудовані механізми, які можуть відображати JavaScript. Це означає, що команди JS можна вводити безпосередньо в HTML- документ, і веб-браузери зможуть їх зрозуміти. Іншими словами, використання JavaScript не вимагає завантаження додаткових програм або компіляторів [24].

Використання JavaScript у якості серверної мови стало можливим за допомогою Node.js – програмної платформи, яка працює на двигуні V8, який транслює JavaScript в машинний код. Ця платформа перетворює JavaScript з вузькоспеціалізованої мови програмування у мову загального призначення. Оскільки Node.js має власний API пристроїв введення/виведення, це дозволяє JavaScript взаємодіяти з ними, підключати бібліотеки, написані на інших мовах програмування.

Node.js (або просто Node) – це серверна платформа для роботи з JavaScript через двигун V8. JavaScript виконує дію на стороні клієнта, а Node – на сервері. За допомогою Node можна писати повноцінні програми. Node вміє працювати з зовнішніми бібліотеками, викликати команди з коду на JavaScript і виконувати роль веб-сервера.

Використання JavaScript у якості серверної мови дає значний приріст у швидкості виконання коду. Більше того, так як рушій V8 перетворює код JavaScript прямо у машинний код, це робить Node.js чи на найшвидшою платформою. Так як Node.js працює на основі архітектури керування подіями, то він якнайкраще підходить для створення додатків реального часу. Оскільки і серверна, і клієнтська частина написані на JavaScript, процес синхронізації між ними є набагато швидшим, легшим та зручнішим [22].

Puppeteer – це «безголовий» Google Chrome на Node.js, який надає зрозумілий API для роботи. Це означає, що ви можете запустити Chrome з командного рядка, навіть не відкриваючи вікна браузера. Ви можете відкривати

сторінки, на яких буде рендери CSS, виконуватися JavaScript – все той же, що робить десктопний Chrome, але без інтерфейсу для користувача [23].

1.4 Використання фреймворку для розробки

Для полегшеної та зручної розробки було використано фреймворк Vue.js.

Vue.js – це JavaScript бібліотека для створення веб-інтерфейсів з використанням шаблону архітектури MVVM (Model-View-ViewModel).

Оскільки Vue працює тільки на «рівні уявлення» і не використовується для проміжного програмного забезпечення і бекенда, він може легко інтегруватися з іншими проектами і бібліотеками. Vue.js містить широку функціональність для рівня уявлень і може використовуватися для створення потужних односторінкових веб-додатків.

Функції Vue.js:

– реактивні інтерфейси;
– декларативний рендеринг;
– зв'язування даних;
– директиви (всі директиви мають префікс «v-». в директиву передається значення стану, а в якості аргументів використовуються html атрибути або Vue js події);
– логіка шаблонів;
– компоненти;
– обробка подій;
– властивості;
– переходи і анімація CSS;
– фільтри.

Vue підходить для невеликих проектів, яким необхідно додати трохи реактивності, додати форму за допомогою AJAX, відобразити значення при введенні даних користувачем, створити авторизацію або інші аналогічні

завдання. Vue легко масштабується і добре підходить для об'ємних проектів, тому його називають прогресивним фреймворком [21].

Vue також відмінно підходить для великих односторінкових додатків завдяки своїм основним компонентам, таким як Router і Vuex. З Vue можна як використовувати загальнодоступні API для створення додатків, так і реалізовувати виконуються сервером додатка. Але Vue найкраще підходить для розробки рішень, які використовують зовнішні API для обробки даних.

За допомогою Vue також можна створювати frontend блогу на популярних CMS. Vue.js відмінно підходить і для розробки динамічних інтерфейсів, які адаптуються під користувача.

Vue має кращу продуктивність і його набагато простіше оптимізувати, оскільки він не використовує брудну перевірку. AngularJS стає повільним, коли є багато спостерігачів, тому що кожен раз, коли що-небудь в області видимості змінюється, всі ці спостерігачі повинні бути повторно оцінені. Крім того, цикл дайджесту, можливо, доведеться запускати кілька разів для «стабілізації», якщо який-небудь спостерігач запускає інше оновлення. Користувачам AngularJS часто доводиться вдаватися до езотеричних методів, щоб обійти цикл дайджесту, а в деяких ситуаціях немає можливості оптимізувати область дії з великою кількістю спостерігачів.

Vue зовсім не страждає від цього, тому що він використовує прозору систему спостереження за залежностями з асинхронної постановкою в чергу - всі зміни запускаються незалежно, якщо у них немає явних відносин залежності [21].

1.5 Аналіз аналогів та недоліків автоматизованих систем

Інформація може існувати в різних формах у вигляді сукупностей деяких знаків (символів, сигналів тощо) на носіях різних типів. Інформація ж в Автоматизованій Системі (АС), визначається як сукупність усіх даних і програм, які використовуються в АС незалежно від засобу їх фізичного та логічного

представлення.

Залежно від мети функціонування та завдань, які покладені на АС на етапах збору та змістової обробки даних, розрізняють такі типи АС:

– інформаційно-пошукові;
– інформаційно-довідкові;
– інформаційно-управлінські;
– інтелектуальні інформаційні системи та системи підтримки прийняття рішень.

Інструменти web scraping (парсинг) розроблені для вилучення, збору будь-якої відкритої інформації з веб-сайтів. Ці ресурси потрібні тоді, коли необхідно швидко отримати і зберегти в структурованому вигляді будь-які дані з інтернету. Парсинг сайтів – це новий метод введення даних, який не вимагає повторного введення або копіпастінга [28].

Такого роду програмне забезпечення шукає інформацію під контролем користувача або автоматично, вибираючи нові або оновлені дані і зберігаючи їх в такому вигляді, щоб у користувача був до них швидкий доступ. Наприклад, використовуючи парсинг можна зібрати інформацію про продукти і їх вартості на сайті Amazon. Нижче розглянуто варіанти використання веб-інструментів вилучення даних і десятку кращих сервісів, які допоможуть зібрати інформацію, без необхідності написання спеціальних програмних кодів. Інструменти парсинга можуть застосовуватися з різними цілями і в різних сценаріях, нижче розглянуто найбільш поширені випадки використання.

Збір даних для дослідження ринку це – веб-сервіси вилучення даних допоможуть стежити за ситуацією в тому напрямку, куди буде прагнути компанія або галузь в наступні шість місяців, забезпечуючи потужний фундамент для дослідження ринку. Програмне забезпечення парсинга здатне отримувати дані від безлічі провайдерів, що спеціалізуються на аналітиці даних і у фірм з дослідження ринку, і потім зводити цю інформацію в одне місце для референції і аналізу.

Витяг контактної інформації це – інструменти парсинга можна використовувати, щоб збирати і систематизувати такі дані, як поштові адреси, контактну інформацію з різних сайтів і соціальних мереж. Це дозволяє складати зручні списки контактів і всієї супутньої інформації для бізнесу – дані про клієнтів, постачальників або виробників.

Рішення по завантаженню з StackOverflow з інструментами парсинга сайтів можна створювати рішення для офлайнового використання і зберігання, зібравши дані з великої кількості веб-ресурсів (включаючи StackOverflow). Таким чином можна уникнути залежності від активних інтернет з'єднань, так як дані будуть доступні незалежно від того, чи є можливість підключитися до інтернету.

Пошук роботи або співробітників для роботодавця, який активно шукає кандидатів для роботи в своїй компанії, або для здобувача, який шукає певну посаду, інструменти парсинга теж стануть незамінні, з їх допомогою можна налаштувати вибірку даних на основі різних доданих фільтрів і ефективно отримувати інформацію, без рутинного ручного пошуку [28].

Відстеження цін в різних магазинах, такі сервіси будуть корисні і для тих, хто активно користується послугами онлайн-шопінгу, відстежує ціни на продукти, шукає речі в декількох магазинах відразу.

Нижче наведено приклади автоматизованих систем обробки даних:

– import.io пропонує розробнику легко формувати власні пакети даних: потрібно тільки імпортувати інформацію з певної веб-сторінки і експортувати її в CSV. Можна отримувати тисячі веб-сторінок за лічені хвилини, не написавши жодного рядка коду, і створювати тисячі API відповідно до вимог користувача. Для збору величезних кількостей потрібної користувачеві інформації, сервіс використовує найновіші технології, причому за низькою ціною. Разом з веб-інструментом доступні безкоштовні програми для Windows, Mac OS X і Linux для створення екстракторів даних і пошукових роботів, які будуть забезпечувати завантаження даних і синхронізацію з онлайнової обліковим записом;

– webhose.io забезпечує прямий доступ в реальному часі до структурованих даних, отриманих в результаті парсинга тисяч онлайн джерел. Цей парсер здатний збирати веб-дані на більш ніж 240 мовами і зберігати результати в різних форматах, включаючи XML, JSON і RSS. Webhose.io – це веб-додаток для браузера, що використовує власну технологію парсинга даних, яка дозволяє обробляти величезні обсяги інформації з численних джерел з єдиним API. Webhose пропонує безкоштовний тарифний план за обробку тисячу запитів в місяць і 50 доларів за преміальний план, що покриває 5000 запитів в місяць;

– loudscrape здатний парсити інформацію з будь-якого веб-сайту і не вимагає завантаження додаткових додатків, як і Webhose. Редактор самостійно встановлює своїх пошукових роботів і витягує дані в режимі реального часу. Користувач може зберегти зібрані дані в хмарі, наприклад, Google Drive і Box.net, або експортувати дані в форматах CSV або JSON. CloudScrape також забезпечує анонімний доступ до даних, пропонуючи ряд проксі-серверів, які допомагають приховати ідентифікаційні дані користувача. CloudScrape зберігає дані на своїх серверах протягом 2 тижнів, потім їх архівує. Сервіс пропонує 20 годин роботи безкоштовно, після чого він буде коштувати 29 доларів на місяць;

– scrapinghub – це хмарний інструмент парсинга даних, який допомагає вибирати і збирати необхідні дані для будь-яких цілей. Scrapinghub використовує Crawlera, розумний проксі-ротатор, оснащений механізмами, здатними обходити захисту від ботів. Сервіс здатний справлятися з величезними за обсягом інформації і захищеними від роботів сайтами. Scrapinghub перетворює веб-сторінки в організований контент. Команда фахівців забезпечує індивідуальний підхід до клієнтів і обіцяє розробити рішення для будь-якого унікального випадку. Базовий безкоштовний пакет дає доступ до одного пошуковому роботу (обробка до 1 Гб даних, далі – 9 $ в місяць), преміальний пакет дає чотирьох паралельних пошукових роботів;

– parsehub може парсити один або багато сайтів з підтримкою JavaScript, AJAX, сеансів, cookie і редиректів. Додаток використовує технологію

самонавчання і здатне розпізнати найскладніші документи в мережі, потім генерує вихідний файл в тому форматі, який потрібен користувачу. ParseHub існує окремо від веб-додатки в якості програми робочого столу для Windows, Mac OS X і Linux. Програма дає безкоштовно п'ять пробних пошукових проектів. Тарифний план Преміум за 89 доларів передбачає 20 проектів і обробку 10 тисяч веб-сторінок за проект [28].

Основними недоліками автоматизації є загрози безпеці, уразливості: автоматизована система може мати обмеження рівень інтелекту, отже, більш сприйнятлива до скоєння помилок за межами свого безпосередньо набору знань (система як правило, не в змозі, застосовувати правила простої логіки загальних положень).

Для сучасних умов характерне застосування високоефективних внутрішньо фірмових систем інформації, що гуртуються на використанні найновіших інформаційних технологій, зокрема єдиної локальної комп’ютерної мережі. Управлінська внутрішня інформаційна система представляє собою сукупність інформаційних процесів для задоволення потреб в інформації на різних рівнях прийняття рішень. Інформаційна система включає компоненти обробки інформації, внутрішні і зовнішні канали передачі.

Інформація, особливо її автоматизована обробка, і тепер залишається важливим фактором підвищення ефективності діяльності будь-якої організації. Важливу роль у використанні інформації відіграють способи її реєстрації, обробки, нагромадження і передачі; систематизоване збереження інформації і її видача в потрібній формі; виробництво нової числової, графічної та іншої інформації.

В сучасних умовах у великих організаціях створені і ефективно діють інформаційні системи, які обслуговують процес підготовки і прийняття управлінських рішень і вирішують наступні задачі: обробку даних, обробку інформації, реалізацію інтелектуальної діяльності з метою створення інформації. Управлінські інформаційні системи послідовно реалізують принципи єдності виробничого процесу та інформаційного процесу супроводу через застосування

технічних засобів збору, нагромадження, обробки і передачі інформації в поєднанні з використанням аналітичних методів математичної статистики і моделей прогнозовано-аналітичних розрахунків та інших необхідних прикладних засобів [27].

Негативні особливості автоматизації:

– зростання числа безробітних, так як автоматизація вивільняє робочі руки;
– витрати на розробку засобів автоматизації зазвичай досить високі;
– завжди присутній загроза безпеці;
– початкова ціна зазвичай досить висока.

1.6 Висновок до першого розділу

У першому розділі розглянуто принципи роботи API, де і як цю технологію використовують, розглянуто необхідність використання цієї технології, її аналоги та де дана система використовується, які існують види розробки цієї системи, та переваги певного виду розробки.

Також обґрунтовано вибір технології розробки, чому саме JavaScript було обрано, опис характеристик Js, зручність у написанні даної системи обробки даних. Опис інших компонентів, що допомагали у розробці, таких як Vue.Js, nuxt.js, puppeteer.

Також проведено аналіз автоматизованої системи оброки даних, аналіз аналогічних систем, приведено декілька прикладів, де схожі системи використовуються, та їх недоліки.

2 РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ

2.1 Планування архітектури програмного забезпечення

Для даного проекту було обрано клієнт-серверну архітектуру, яка, як відомо, буває двох типів: з двома та трьома ланками. Для нашої системи найкраще підходить архітектура з трьома ланками, яка буде складатися з наступних компонентів [15]:

– клієнтський рівень – частина додатку, яка розроблена для взаємодії з користувачем (User Interface);

– сервер – частина програми, в якій міститься вся бізнес-логіка додатку. Клієнтський рівень може отримати доступ до функцій серверу через жорстко визначений інтерфейс;

– data layer – рівень роботи з базою даних. Ця частина системи надає серверу доступ до бази даних.

Вибір такого підходу гарантує наступні переваги:

– високий рівень захищеності системи;

– високий рівень масштабованості та гнучкості;

– висока продуктивність.

Загальна схема клієнт-серверної архітектури з трьома ланками зображена на рисунку 2.1.

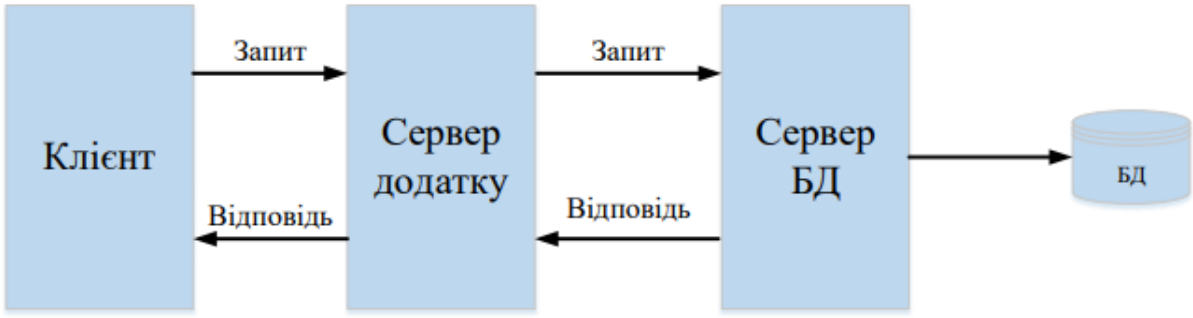

Рисунок 2.1 – Загальна схема трирівневої клієнт-серверної архітектури [11]

Приведення архітектури до моделі клієнт-сервер лежить в основі архітектурного стилю REST для розробки API. Завдяки цьому обмеженню система набуває таких бажаних характеристик як продуктивність, простота та гнучкість, масштабованість, адаптивність та здатність до змін, портативність та надійність [11].

Розмежування потреб є основним принципом, який лежить в основі даної архітектури. Відділення потреб інтерфейсу клієнта від потреб серверної частини, яка зберігає дані, підвищує портативність та переносимість коду клієнтської програми на інші платформи. Водночас спрощення серверної частини помітно покращує масштабованість всієї системи. Це дає можливість окремим частинам розвиватися незалежно одна від одної.

Трирівнева клієнт-серверна архітектура, яка складається з представлення даних (User Interface), прикладного компонента (проміжне програмне забезпечення, яке забезпечує прошарок між сервером додатку та рівнем бази даних) та рівня керування ресурсами (доступ до бази даних)

API REST використовують методи HTTP-запиту GET, POST, PUT і DELETE, визначені в RFC 2616. Отже, для взаємодії клієнта і сервера один з одним через REST-API угоди про протоколи не потрібні. За допомогою GET ресурси запитуються з RESTful API. POST використовується для поновлення або зміни стану ресурсу. За допомогою PUT можна створювати нові ресурси або замінювати вміст існуючих ресурсів. DELETE використовується для видалення ресурсів. Цих чотирьох методів HTTP зазвичай досить для більшості випадків використання [11].

Після надходження запиту GET на сервер, користувач отримує відповідь даних в форматі JSON рисунок 2.2.

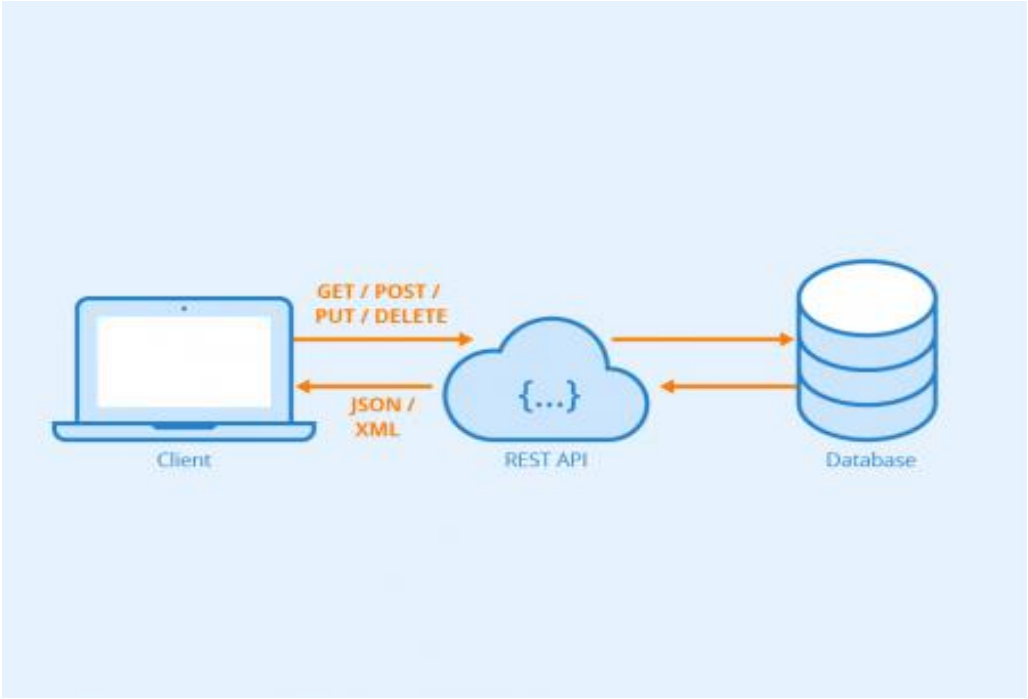

Рисунок 2.2 – Схема використання REST API [10]

JSON (JavaScript Object Notation) – це полегшений формат обміну даними. Людям легко читати і писати. Машини легко аналізують і генерують. Він заснований на підмножині стандарту мови програмування JavaScript.

JSON складається з двох структур:

– колекція пар ім'я / значення. На різних мовах це реалізовано як об'єкт, запис, структура, словник, хеш-таблиця, список з ключами або асоціативний масив;

– упорядкований список значень. У більшості мов це реалізовано як масив, вектор, список або послідовність.

Це універсальні структури даних. Практично всі сучасні мови програмування підтримують їх в тій чи іншій формі. Має сенс, щоб формат даних, взаємозамінний з мовами програмування, також був заснований на цих структурах. Дані JSON зберігаються в файлах, які мають розширення json. Відповідно до принципу вдало читання JSON, це просто текстові файли, які можна легко відкрити і вивчити [13].

2.2 Особливості використання REST архітектури для розробки API

Архітектурний стиль REST вже довгий час вважається найкращим та найбільш зручним підходом у розробці Web API. Його найближчий конкурент SOAP підходить лише для зовсім невеликих або дуже специфічних рішень.

Отже, REST – Representational State Transfer, «передача стану через представлення») – це підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів. Системи, які підтримують REST, називаються RESTful–системами [10].

У загальному випадку, REST є дуже простим інтерфейсом управління інформацією без використання якихось додаткових внутрішніх прошарків. Щоб краще зрозуміти принцип роботи REST архітектури, розглянемо схему, зображену на рисунку 2.3 [11].

Рисунок 2.3 – Архітектура REST [12]

2.3 Вибір шаблону проектування системи

Для проектування системи було обрано архітектурний шаблон MVC. MVC – це схема розподілення рівня даних додатку, інтерфейсу користувача та логіки

додатку на три окремих та незалежних компонента. Таким чином, модифікація кожного з трьох компонентів може відбуватися незалежно, це дає переваги у портативності та масштабованості системи [17].

Розглянемо компоненти MVC детальніше:

– модель (Model) – надає дані для обробки та відображення і реагує на команди контролера, змінюючи власний стан;

– представлення (View) – відповідає за відображення даних, які надає модель користувачу, реагуючи на зміни у моделі;

– контролер (Controller) – реагує на дії користувача та оповіщає модель про необхідність змін.

Основною ідеєю застосування цього підходу є розмежування бізнес-логіки (Model) від її візуального вигляду – представлення (View). За рахунок такого розмежування виконується принцип повторного використання коду. Крім того, це дуже корисно в ситуаціях, коли користувач повинен мати змогу бачити одні і ті ж дані одночасно, але у різних контекстах або з різних точок зору. Структура архітектурного шаблону MVC зображена на рисунку 2.4 [17].

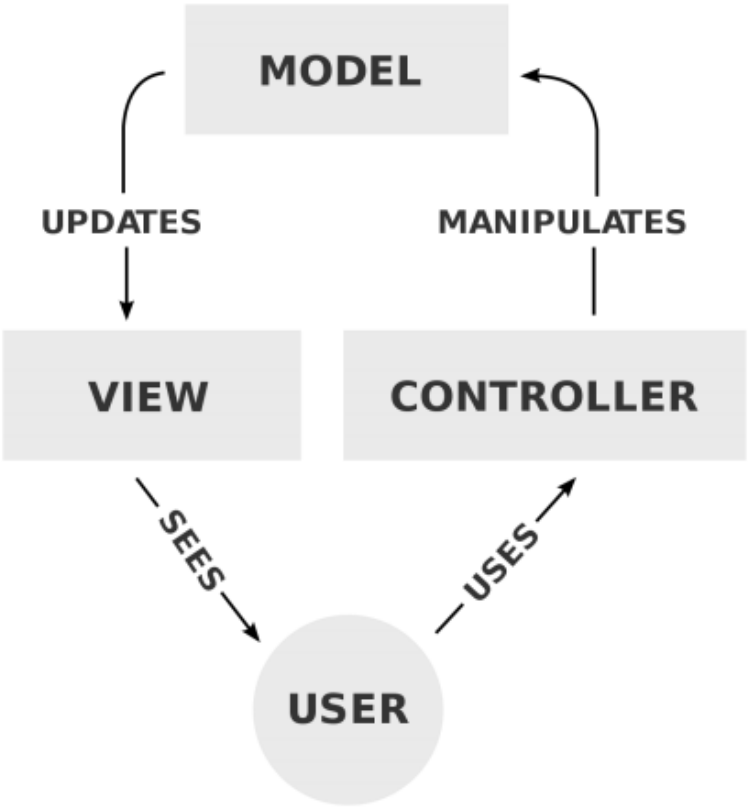

Рисунок 2.4 – Загальна схема MVC [18]

Модель надає дані та методи, які дозволяють працювати з ними, наприклад, запити до бази даних, перевірка на коректність отриманих даних. Модель є незалежною від представлення, вона зовсім не знає, як ці дані візуалізувати. Також вона повністю незалежна від контролера і не має спільних точок взаємодії з користувачем. Вона просто надає доступ до даних та керування ними [17].

Програма абсолютно абстрактна. Вона вміє робити корисні функції, і у користувача є потреби, які можна задовольнити за допомогою цієї програми. Тоді з'являються спеціалісти, які «знають», як, використовуючи цю програму, зробити те, що хоче користувач. Сама очевидна перевага, яку ми отримуємо від використання концепції MVC – це чіткий поділ логіки подання (інтерфейсу користувача) і логіки програми.

Коли модель публікує зміни, її не хвилює для кого, вона нічого не знає про View. Замість або разом зі View на тому кінці може бути інша підсистема [19].

2.4 Опис структури запитів

HTTP – це протокол, що дозволяє отримувати різні ресурси, наприклад HTML-документи. Протокол HTTP лежить в основі обміну даними в Інтернеті. HTTP є протоколом клієнт-серверного взаємодії, що означає ініціювання запитів до сервера самим одержувачем, зазвичай веб-браузером. Отриманий підсумковий документ буде складатися з різних під документів що є частиною підсумкового документа: наприклад, з окремо отриманого тексту, опису структури документа, зображень, відео-файлів, скриптів і багато чого іншого.

Клієнти і сервери взаємодіють, обмінюючись поодинокими повідомленнями (а не потоком даних). Повідомлення, відправлені клієнтом, зазвичай веб-браузером, називаються запитами, а повідомлення, відправлені сервером, називаються відповідями (рис 2.5).

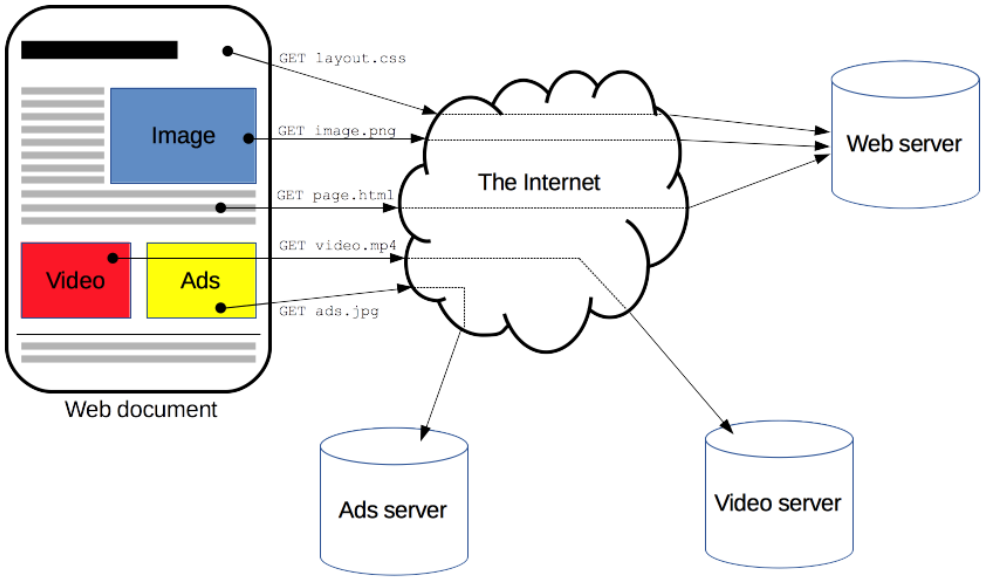

Рисунок 2.5 – Структура запитів та відповідей [25]

HTTP – це клієнт-серверний протокол, тобто запити відправляються якоїсь однієї стороною – учасником обміну (user-agent) (або проксі замість нього). Найчастіше в якості учасника виступає веб-браузер, але їм може бути хто завгодно, наприклад, робот, який мандрує по Мережі для поповнення та оновлення даних індексації веб-сторінок для пошукових систем [25].

Кожен запит (англ. Request) відправляється серверу, який обробляє його і повертає відповідь (англ. Response). Між цими запитами і відповідями як правило існують численні посередники, звані проксі, які виконують різні операції і працюють як шлюзи або кеш.

Щоб відобразити веб сторінку, браузер відправляє початковий запит для отримання HTML-документа цієї сторінки. Після цього браузер вивчає цей документ, і запитує додаткові файли, необхідні для отображенія змісту веб-сторінки (виконувані скрипти, деталі компонування сторінки – CSS таблиці стилів, додаткові ресурси у вигляді зображень і відео-файлів), які безпосередньо є частиною вихідного документа, але розташовані в інших місцях мережі. Далі браузер з'єднує всі ці ресурси для відображення їх користувачеві у вигляді єдиного документа – веб-сторінки. Скрипти, що виконуються самим браузером, можуть отримувати по мережі додаткові ресурси на наступних етапах обробки

веб-сторінки, і браузер відповідним чином оновлює відображення цієї сторінки для користувача.

Веб-сторінка є гіпертекстових документом. Це означає, що деякі частини тексту, що відображається є посиланнями, які можуть бути активовані (зазвичай натисканням кнопки миші) з метою отримання і відповідно відображення нової веб-сторінки (перехід по посиланню). Це дозволяє користувачеві переміщатися по сторінках мережі (Internet). Браузер перетворює ці гіперпосилання в HTTP-запити і в подальшому отримані HTTP-відповіді відображає в зрозумілому для користувача вигляді [25].

На іншій стороні комунікаційного каналу розташований сервер, який обслуговує користувача, надаючи йому документи на вимогу. З точки зору кінцевого користувача, сервер завжди є якоюсь однією віртуальною машиною, повністю або частково генеруючої документ, хоча фактично він може бути групою серверів, між якими балансується навантаження, тобто перерозподіляються запити різних користувачів, або складним програмним забезпеченням, опитують інші комп'ютери такі як кешуючий сервери, сервери баз даних, сервери додатків електронної комерції та інші.

Сервер не обов'язково розташований на одній машині, і навпаки - кілька серверів можуть бути розташовані (хоститься) на одній і тій же машині. Відповідно до версією HTTP / 1.1 і маючи Host заголовок, вони навіть можуть ділити той же самий IP-адрес [25].

Між веб-браузером і сервером знаходяться велика кількість мережевих вузлів передавальних HTTP повідомлення. Через шаруватої структури, більшість з них оперують також на транспортному мережевому або фізичному рівнях, стаючи прозорим на HTTP шарі і потенційно знижуючи продуктивність.

Структура рядка JSON практично нічим не відрізняється від запису JavaScript об'єкта.

Вона складається з набору пар ключ-значення. У цій парі ключ відділяється від значення за допомогою знаку двокрапки (:), а одна пара від іншого - за допомогою коми (,). При цьому ключ в JSON, на відміну від об'єкта обов'язково

повинен бути укладений в подвійні лапки. Це перша відмінність JSON від JavaScript об'єкта. В об'єкті ключ (ім'я властивості) не обов'язково слід укладати в подвійні лапки [26].

JSON існує як рядок, що необхідно при передачі даних по мережі, JSON повинен перетворитися в власний об'єкт JavaScript, якщо користувач хоче отримати доступ до даних. Це не велика проблема. JavaScript надає глобальний об'єкт JSON, який має методи для перетворення між ними [26].

JSON являє собою рядок, формат якої дуже схожий на буквений формат об'єкта JavaScript. Ви можете включати одні й ті ж базові типи даних всередині JSON, так само як і в стандартному об'єкті JavaScript – рядки, числа, масиви, булеві і інші об'єктні літерали. Це дозволяє побудувати ієрархію даних.

Робота з JSON в JavaScript зазвичай здійснюється в двох напрямках:

– переклад рядка JSON в об'єкт (даний процес ще називають парсинга);

– конвертація об'єкта в рядок JSON (іншими словами дію зворотне парсингу).

Парсинг JSON (переклад рядка JSON в об'єкт JavaScript) здійснюється за допомогою методу eval або parse.

Формат представлення даних JSON має такі переваги:

– зручні і швидкі в роботі методи, призначені для конвертації (парсинга) рядки JSON в об'єкт JavaScript і назад;

– зрозуміла і проста структура даних;

– дуже маленький розмір у порівнянні з іншими форматами даних (наприклад XML). Це пов'язано з тим, що формат JSON містить мінімальну можливу форматування, тобто при його написанні використовується всього кілька спеціальних знаків. Це дуже важлива перевага, тому що дані представлені в форматі JSON будуть швидше завантажуватися, ніж, якби вони були б представлені в інших форматах [26].

2.5 Висновок до другого розділу

У другому розділі описано архітектуру за якою проводилась розробка автоматизованої системи обробки даних, описаний принцип роботи архітектури, опис всіх рівнів взаємодії, та послідовність розробки автоматизованої системи.

Обраний шаблон mvc, котрий реалізує три рівні, перший це модель котра надає дані для обробки, другий це представлення котре відповідає за відображення даних, та третій контролер що реагує на дії користувача.

Описана розробка внутрішньої системи, весь її алгоритм, та як відбувається запит для збору даних з веб-ресурсів.

3 РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ

3.1 Розробка інтерфейсу системи

Розробку системи було поділено на основні етапи, які задовольняють вибраній архітектурі.

На першому етапі було розроблено API, завдяки якій збирається необхідна інформація з різних інформаційних джерел, а саме дані про погоду на десять діб з усіх великих міст України. Дані отримуються у форматі JSON та зберігаються на віртуальному сервері. На віртуальному сервері вони зберігаються у вигляді простої строки, яку в наступному етапі потрібно буде перетворити в дані, які будуть виводитись на веб-додатку.

На другому етапі відбувається перетворення зібраних даних у програмний код, який надалі буде виводитися на веб-додатку по обраним параметрам, які задаватиме користувач даної автоматизованої системи.

На третьому етапі було розроблено серверну частину системи, це віртуальній сервер, на який надходять зібрані дані з розподілених інформаційних джерел, також реалізовано розгортання серверу для відображення необхідних даних на веб-додатку.

На четвертому етапі було розроблено інтерфейс програми, створено сторінку, на якій будуть відображатися дані з розподілених інформаційних джерел про погоду на найближчі десять діб з усіх великих міст України. Також будуть відображатися перелік функцій, які може вибрати користувач даної системи, та вікно опцій, яке виконано в модальній формі випадаючого вікна.

Інтерфейс системи, було вирішено зробити максимально простим та доступним для кожного користувача. Потрібно було зробити відображення даних у максимально зручній формі, щоб кожен міг використовувати дану систему без витрати часу на ознайомлення із різного роду інструкціями, та не потребував допомоги спеціаліста. Також користувач повинен без зайвих

налаштувань легко потрапляти у систему і одразу починати роботу із нею.

Розробка інтерфейсу поділена на основні етапи, у випадку автоматизованої системи обробки даних з розподілених інформаційних джерел, було виділено такі етапи:

- відкриття головної сторінки;
- створення модального вікна;
- створення меню з вибором джерел;
- створення пункту з вибором дати;
- створення пункту з вибором міста;
- створення кнопки застосувати;
- створення кнопки опції.

Наступний етап, це стилістичні параметри, та позиціювання всіх елементів системи.

Останній етап створення автоматизованої системи обробки даних, є програмна реалізація із залученням обраного середовища розробки.

3.2 Приклад роботи з системою

Після відкриття системи у веб-додатку користувач потрапляє на головну сторінку, де відкривається модальне вікно, з усіма параметрами системи зображеному на рисунку 3.1.

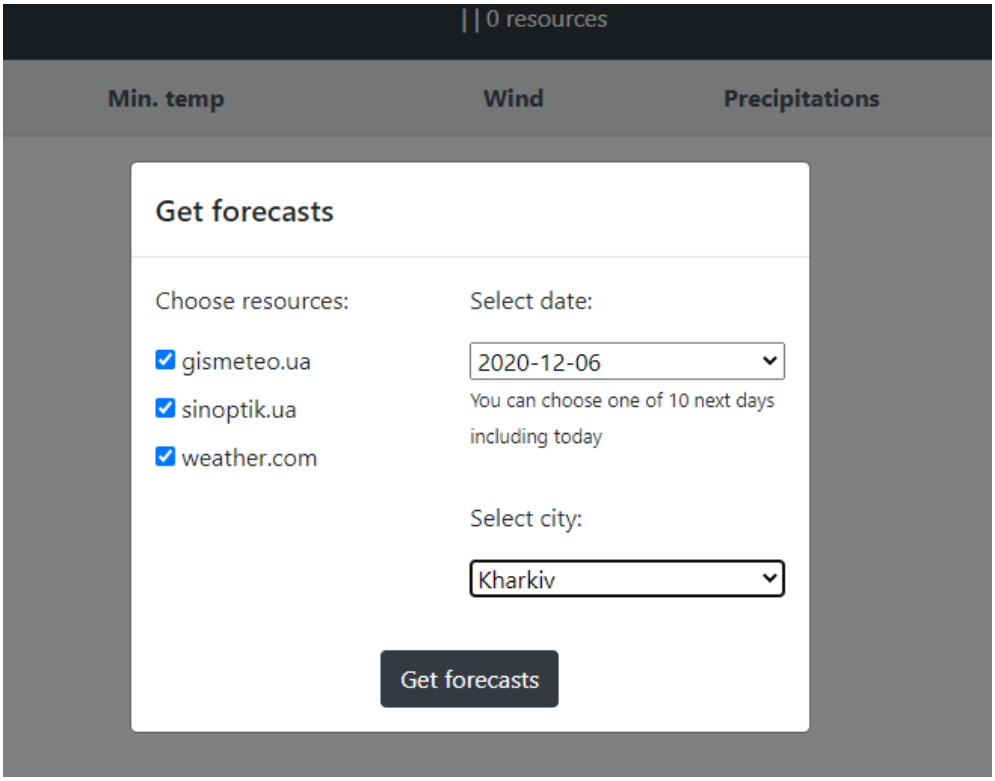

Рисунок 3.1 – Модальне вікно з необхідними параметрами

Користувач може вибрати необхідну кількість інформаційних джерел, з яких він хоче отримати дані, в даному випадку це – дані про погоду. Також користувач обирає необхідну дату зображену на рисунку 3.2, так місто зі списку, що випадає рисунок 3.3.

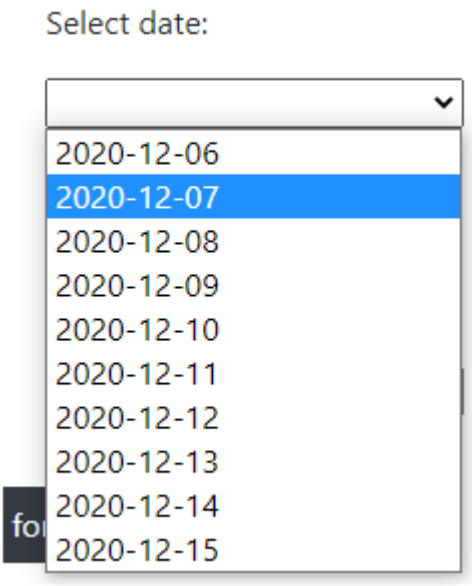

Рисунок 3.2 – Перелік діб

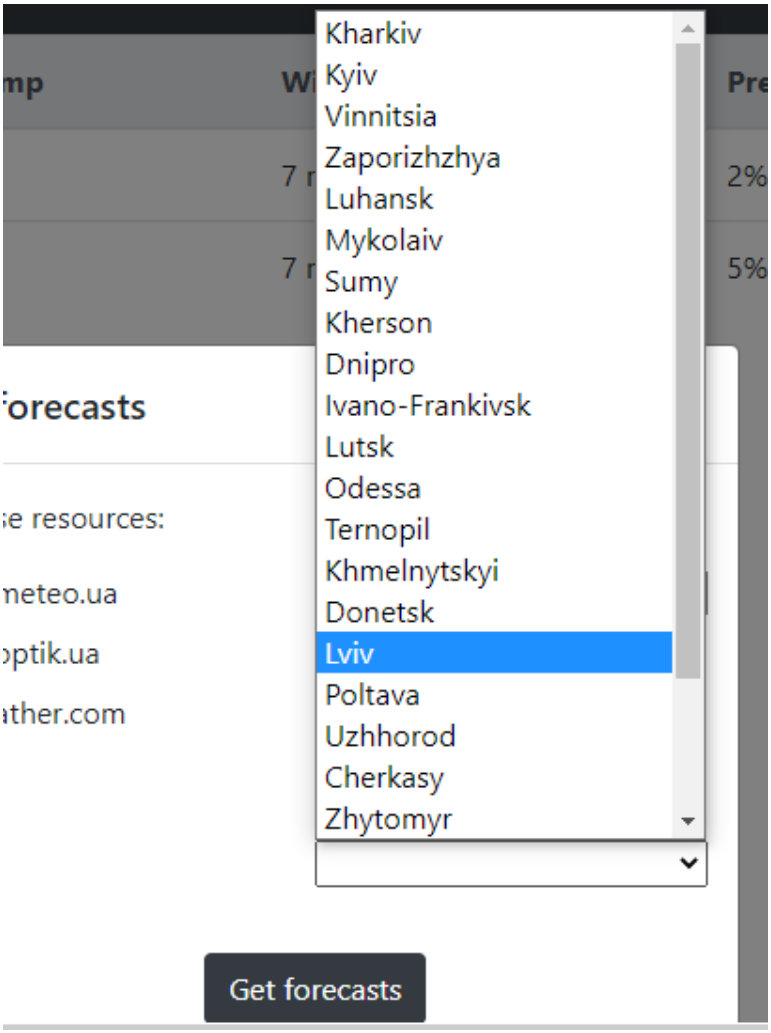

Рисунок 3.3 – Перелік міст України

Після того, як користувач обере необхідну кількість джерел, дату, та місто зі списку, то система обробить запит та на екрані з’являться необхідні дані зображені на рисунку 3.4.

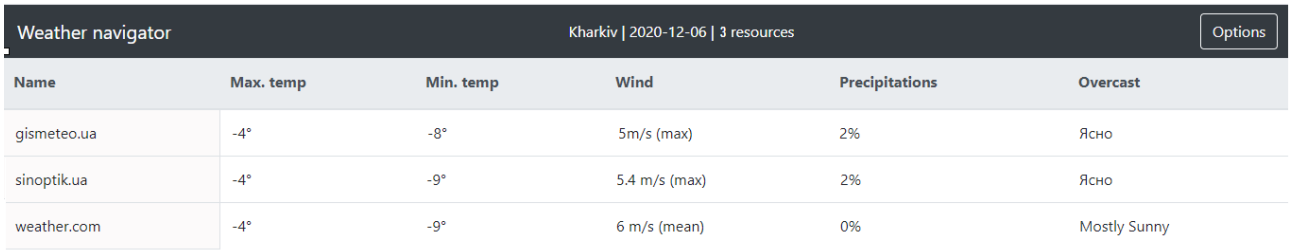

Рисунок 3.4 – Блок з вихідними даними

На рисунку зображені дані, які система зібрала з трьох інформаційних джерел. Система видає мінімальну, максимальну температуру, також швидкість вітру, вірогідну кількість осадків в процентному відношенні, та яка буде погода протягом доби.

У верхній частині веб-додатку для користувача відображається обране місто, добу, та кількість ресурсів, з яких збираються дані про погоду, а також кнопку опції зображену на рисунку 3.5, яка при натисканні відкриває модальне вікно, щоб користувач зміг змінити параметри пошуку зображеному на (рис 3.1).

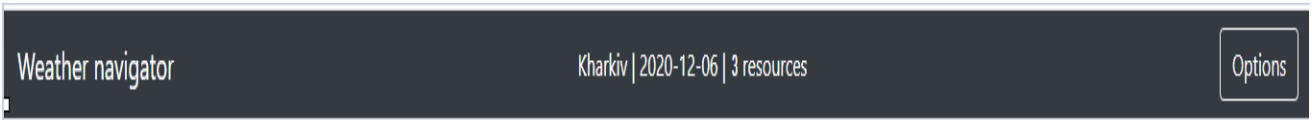

Рисунок 3.5 – Інформаційна панель

Після обраних параметрів та виведення даних користувачеві, він бачить всі необхідні дані які його цікавлять.

Дана система може бути модернізована, та перебудована під необхідну сферу використання. Даній системі е куди покращуватись та зростати.

На наступному етапі буде проводитись прогнозування витрат на розробку програмного забезпечення автоматизованої системи обробки даних.

3.3 Розробка програмного забезпечення системи

Щоб отримати необхідні дані з певного веб-ресурсу, відправляється GET запит до бази даних цього веб-ресурсу, після обробки даних, до користувача надходить відповідь з даними у форматі JSON. Ці дані зберігаються на віртуальному сервері, у вигляді строки. Після того, як дані надійшли до користувача на його віртуальний сервер завдяки парсингу ці дані обробляються та розбиваються на необхідні функції. Дані у форматі JSON приходять у такому форматі, які вони мають на веб-ресурсі через URL.

Далі використовується функція, яка відповідає за вибір дати та міста, якщо користувач обрав саме цей ресурс. Після вибору ресурсу, система обробляє яку дату яку вибрав користувач. Після того, як запит сформовано користувачем, через той самий URL з нашого віртуального серверу збираються дані про необхідне місто та день, та виводяться користувачеві на веб-додатку.

Дані будуть збиратись з таких сайтів як gismeteo.ua, weather.in.ua, sinoptik.ua. Після надходження запиту GET на ці сайти, будуть збиратись дані за допомогою певного URL:

– "GismeteoURL": "weather-kharkiv-5053/10-days/" – збирає дані про погоду у місті Харків за десять діб;

– "WeatherURL":"weather/tenday/l/a4611f0d16cabf84398cffee7963a6f7817abe09168b0c01cfdcbb209451e775" – збирає дані про погоду у місті Харків за десять діб;

– "SinoptikURL":"%D0%BF%D0%BE%D0%B3%D0%BE%D0%B4%D0%B0-%D1%85%D0%B0%D1%80%D1%8C%D0%BA%D0%BE%D0%B2"" – збирає дані про погоду у місті Харків за десять діб.

Ці URL знаходяться у відкритому доступі на офіційних веб-ресурсах.

Дані, які надходять з цих URL, зберігаються у форматі JSON на віртуальному сервері, вони мають однаковий вигляд для кожного з великих міст України.

Після того, як необхідні дані зібрані, починається обробка цих даних за допомогою парсингу.

Функція getGismeteo = async function () перевіяє запит на необхідний ресурс та перевіряє, чи обрав користувач дату та місто.

Функція userDate() перевіряє місяць, рік, та день, дана функція виводить наступні десять діб, включаючи дату, коли робиться запит.

Функція currentDate() перевіряє дату до години, хвилини, та секунди. Це зроблено для того, щоб користувач, коли буде користуватися системою о півночі, система правильно вела відлік десяти наступних діб.

Після обирання необхідних параметрів пошуку, створюється об’єкт cityObject.GismeteoURL, в який записуються дані про параметри, які вказав користувач.

В константу const temperatures записуються дані про температуру, дані про температуру,

Константа const maxTemp = await maxTempNode приймає максимальну температру.

Константа const minTemp = await minTempNode приймає мінімальну температуру.

Функція forecast.maxTemperature = maxTemp + '°' виводить дані про максимальну температуру в градусах.

Функція forecast.minTemperature = minTemp + '°' виводить дані про мінімальну температуру у градусах.

Константа const maxWindSpeed = await windNode приймає значення максимальної швидкості вітру.

Функція forecast.wind = maxWindSpeed + ' m/s (max)' виводить дані про швидкість вітру у метрах в секунду.

Константа const precipitations = await precsNode приймає дані про вірогідну кількість опадів.

Функція forecast.precips = precipitations + ' mm' виводить дані про вірогідну кількість опадів у відсотках.

Функція forecast.overcast () виводить погоду протягом доби.

Функція gismeteo.close() закриває збір даних з певного веб-ресурсу.

Нижче представлена блок-схема алгоритму роботи даної системи (рис 3.6).

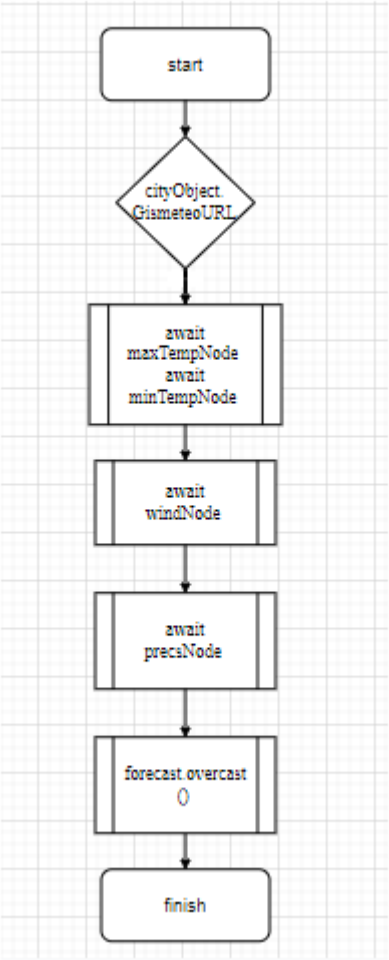

Рисунок 3.6 – Блок-схема алгоритму роботи автоматизованої системи

У процесі розробки такої діаграми простіше зрозуміти, які саме дані та дії можуть бути потрібні клієнту для взаємодії з нашим сервісом, а також, передачу значення яких станів слід очікувати від клієнта сервісу.

Дана система реалізована на архітектурному стилі Model-ViewController, тому файлова структура відповідає даній архітектурі, всі файли знаходяться у потрібних папках, з потрібною назвою, щоб не плутатися. На рисунку 3.7 зображена загальна структура папок та файлів, які було створено для розробки даної системи.

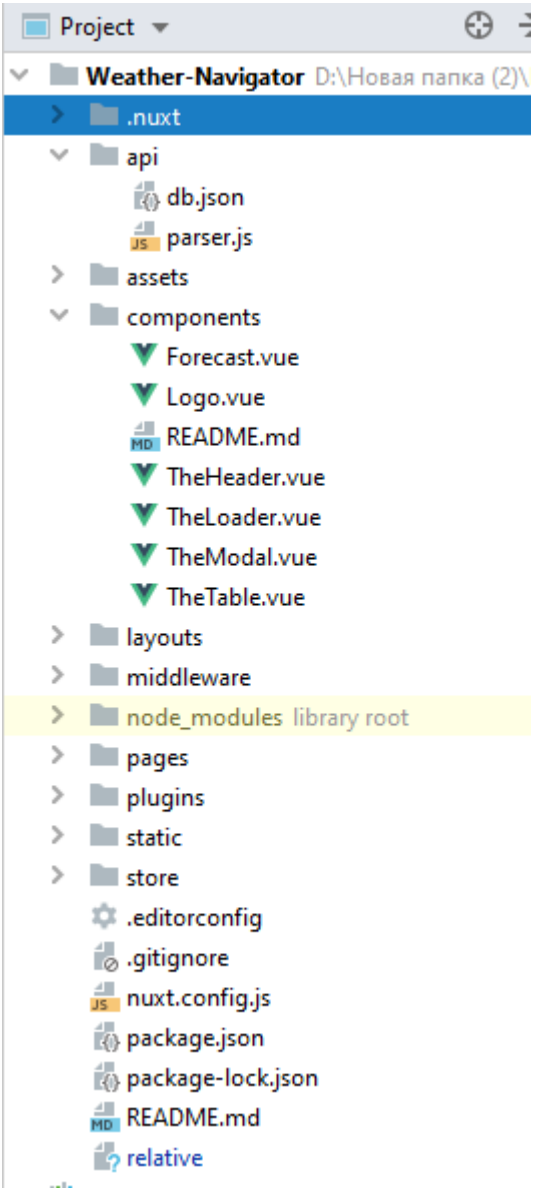

Рисунок 3.7 – Загальна структура папок та файлів проекту

У папці api знаходяться безпосередньо наші дані у форматі JSON, та файл parser.js, який оброблює ці дані та виводить їх користувачу.

У папці components знаходяться файлі, які відповідають за відображення даних на веб-додатку, таких як логотип, шапка з назвою міста та дата, модальне вікно, на якому користувачу показується перелік джерел, місто та дата.

Також у файлі forecast написана структура, за якою виводяться необхідні дані по необхідним параметрам, які користувач обрав. Цей файл відповідає за кінцевий результат системи.

3.4 Прогнозування витрат на виконання роботи та прогнозування комерційних ефектів

Спрогнозуємо витрати на виконання науково-дослідної роботи, та прогнозуймо комерційний ефект, для цього виконаємо такі етапи:

– розрахуємо витрати, що безпосередньо стосуються виконавців цього розділу роботи;

– розрахуємо загальні витрати на виконання цієї роботи;

– спрогнозуємо загальні витрати на виконання та впровадження результатів цієї роботи.

Основна заробітна плата кожного із розробників (дослідників) $З_о$ розраховується за формулою:

$$З_о = \frac{М}{Т_Р} \times t, \quad (3.1)$$

де М – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні;

t – число робочих днів роботи розробника (дослідника).

Для спеціаліста із місячним посадовим окладом у 10000 грн. і кількістю робочих днів у місяці – 21, заробітна плата складатиме:

$$З_о = \frac{10000}{21} \times 75 = 35714{,}28 \text{ (грн)}.$$

Результати обчислень місячного посадового окладу для спеціаліста, який розробляє автоматизовану систему обробки даних, зведемо до таблиці 3.1.

Таблиця 3.1 – Основна заробітна плата робітників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочій день	Число днів роботи	Витрати на заробітну плату, грн
Розробник	10000	476,19	75	35714,28
			∑	35714,28

Додаткова заробітна плата спеціалістів, що брали участь у розробці програмного засобу розраховується як 10% від основної заробітної плати спеціалістів та становить:

$$З_д = З_о \times 0,10, \quad (3.2)$$

$$З_д = 35714,28 \times 0,10 = 3571,42 \text{ (грн)}.$$

Нарахування на заробітну плату спеціалістів, що брали участь у розробці програмного засобу, розраховуються за формулою 3.3. Згідно чинного законодавства нарахування на заробітну платню становлять 22% від суми додаткової і основної заробітної плати всіх спеціалістів-робітників.

$$Н_{зп} = (З_о + З_д) \times \frac{22}{100\%}. \quad (3.3)$$

Обрахуємо нарахування на заробітну плату всіх спеціалістів, що працювали над програмним продуктом:

$$Н_{зп} = (35714,28 + 3571,42) \times \frac{22}{100\ \%} = 8642,85 \text{ (грн)}.$$

Амортизація обладнання, комп'ютерів і приміщень А, що використовувались для виконання цього етапу:

$$А = \frac{Ц \times Н_а}{100} \times \frac{Т}{12} \text{ (грн)}, \quad (3.4)$$

де Ц – вартість обладнання, грн.;

$Н_а$ – річна норма амортизаційних відрахувань;

Т – термін використання, місяці.

Розрахуємо норму амортизаційних відрахувань

$$Н_а = \frac{В_п - В_л}{В_п \times Т_{кв}} \times 100, \quad (3.4)$$

де Вп і Вл – відповідно первісна і ліквідаційна вартість об'єктів основних фондів;

Ткв – термін корисного використання, 2 роки.

Усі виконані розрахунки амортизаційних відрахувань занесемо до таблиці 3.2.

Витрати на матеріали, які були використані для розробки нового програмного засобу обраховуються за наступною формулою:

$$К = \sum_{s=1}^{n} Н_i \times Ц_i \times К_i , \quad (3.5)$$

де $Н_i$ – кількість комплектуючих і-го виду;

$Ц_i$ – ціна і-го комплектуючого;

$К_i$ – коефіцієнт транспортних затрат (1,0 … 1,15).

Таблиця 3.2 – Величина амортизаційних відрахувань

Найменування	Балансова вартість, грн.	Термін використання	Фактична тривалість використання, міс.	Величина амортизаційних відрахувань, грн
Комп'ютер	15000	24	3	937,5
Принтер	1000	12	3	62,5
			∑	999,5

Інформацію про матеріали, що використовуються для створення автоматизованої системи обробки даних з розподілених інформаційних джерел, наведено у наступній таблиці.

Таблиця 3.3 – Матеріали та комплектуючі необхідні для розробки

Найменування комплектуючих	Кількість, шт.	Ціна за одиницю, грн.	Сума грн.
Папір	1	80	80
Диск	4	10	40
Ручка	2	1	10
Флеш-накопичувак	1	240	240
Інтеренет	3	100	300
		∑	670

Визначимо витрати на силову електроенергію, опираючись на витрати на одиницю продукції та на тарифи на енергію:

$$В_е = В \times П \times Ф \times К_п, \quad (3.6)$$

де $В_е$ – вартість 1 кВт електроенергії (2,44);

П – установлена потужність обладнання, П = 80 Вт.;

Ф – кількість годин роботи обладнання, Ф = 850 год.;

КП – коефіцієнт потужності.

$$В_е = 2{,}44 \times 0{,}08 \times 850 \times 0{,}6 = 99{,}55 \text{ (грн)}.$$

Загальні витрати на розробку автоматизованої системи обробки даних:

$$В = 99{,}55 + 670 + 999{,}5 + 8642{,}85 + 3571{,}42 + 35714{,}28 = 49697{,}6 \text{ (грн)}.$$

Прогнозування загальних витрат на розробку та впровадження програмного засобу обчислюються за наступною формулою.

$$ЗВ = \frac{В_{заг}}{\beta}, \quad (3.7)$$

де β – коефіцієнт, що характеризує етап виконання роботи, $В_{заг} = В$ (у даному випадку β = 0,9).

Отже, загальні витрати на розробку та впровадження автоматизованої системи обробки даних становлять:

$$ЗВ = \frac{49697{,}6}{0{,}9} = 55219{,}55 \text{ (грн)}.$$

3.5 Перспектива розвитку розробленої системи

Наукова новизна розробленої системи полягає у розробці системи шляхом визначення недоліків систем аналогічного типу, які застосовуються на підприємствах. Даний вид автоматизованої системи застосовується на

підприємствах, які розроблять безпілотні літальні апарати, які необхідно тестувати під певними погодними умовами, безумовно даній системі є куди розвиватися та зростати.

В Україні не так багато підприємств, які розробляють безпілотники, тому такі системи є одиничними, але з кожним роком наша країна розвивається, можна побачити зростання військових безпілотників, тому ця система може стати прототипом наступної модернізованої автоматизованої системи для збору даних з розподілених інформаційних джерел.

Ця система може бути впровадженою не тільки для погодних даних, а для збору зовсім різних видів даних, система може бути налаштована на збір даних про кількість товарів на складі, для цього достатньо змінити декілька параметрів пошуку та інформаційні джерела.

3.6 Забезпечення безпечних умов правці при розробці додатку

Приміщення, в яких планується установка та подальша робота з комп’ютером, повинні відповідати проектній документації будинку, погодженій з уповноваженими державними органами. Крім того, роботодавець повинен враховувати санітарні нормативи освітлення, вимоги до параметрів мікроклімату (температура, відносна вологість), ступеня і сили вібрації, звукового шуму і вогнестійкості приміщення, а також характери-стики електромагнітного, ультрафіолетового та інфрачервоного полів.

Конкретні показники зазначених санітарних норм див. в Державних санітарних правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98, затверджених Постановою Головного державного санітарного лікаря України №7 від 10 грудня 1998 року. Правила поширюються на умови й організацію праці при роботі з візуальними дисплейними терміналами (ВДТ) усіх типів вітчизняного та зарубіжного виробництва на основі електронно-променевих трубок (ЕПТ), що використовуються в електронно-обчислювальних машинах (ЕОМ) колективного

використання та персональних ЕОМ (ПЕОМ). Так, наприклад, роботодавцю заборонено установлювати комп’ютери в приміщеннях, розташованих у підвалах будинків.

Для уникнення можливих аварій та замикань, поряд з приміщеннями, де вестиметься робота з комп’ютером (над чи під ними), також не дозволяється проведення робіт, що потребують здійснення надмірно вологих технологічних процесів.

Відповідне приміщення повинно бути укомплектоване системами центрального або індивідуального опалення, кондиціонування чи вентиляції повітря. Але при установці зазначених систем, необхідно переконатись, що батареї опалення, водопровідні труби, вентиляційні кабелі тощо, надійно сховані під захисними щитками, які перешкоджатимуть можливому потраплянню робітника під напругу.

У кожній кімнаті, де обладнуватимуться робочі місця співробітників, що працюватимуть на комп’ютері, повинні бути наявні елементи природного та штучного освітлення. При цьому, на вікнах слід встановити легко регульовані жалюзі чи штори, які дозволять працівникам коригувати рівень освітлення в приміщенні. Бажано розмістити комп’ютери в кімнаті таким чином, щоб світло потрапляло на екрани моніторів з півдня чи північного сходу.

З метою досягнення максимального рівня безпечності і охорони праці при роботі з комп’ютером, виробничі приміщення необхідно обладнати аптечками першої медичної допомоги, системами автоматичної пожежної сигналізації і вогнегасниками. В приміщенні, в якому разом працюють 5 або більше комп’ютерів, на видимому місці установлюється службовий вимикач, який у разі потреби дозволить повністю відключити електричне живлення кімнати.

Роботодавець, який використовує найману працю робітників, повинен забезпечити відповідність їхніх робочих місць комфортним та безпечним умовам. Розмір одного робочого місця має становити не менше 6 квадрат-них метрів. При необхідності, суміжні робочі місця співробітників, що працюють з комп’ютером, слід розділити перегородками висотою до 2 метрів. При

визначенні достатнього розміру приміщення і робочого місця на одну особу необхідно додатково враховувати шафи, сейфи, тумби або інші предмети меблів чи обладнання, які знаходяться в кімнаті. На столі працівника можливо розмістити допоміжні для роботи пристрої (принтери, колонки, сканери), а також місця для зберігання документів, за умови, що це не обмежуватиме видимість екрану і не заважатиме працівнику. У разі надмірного шуму чи вібрації технічного обладнання, роботодавець повинен забезпечити працівників антивібраційними килимками.

Робочий стілець співробітника має бути підйомно-поворотним, легко регульованим за висотою та забезпечувати належну підтримку та зручне положення спини і хребта особи. Щодня необхідно проводити вологе прибирання приміщення, та очищати робоче місце та безпосередньо монітор комп'ютера від запиленості.

На підприємстві забороняється: проводити ремонт та технічне обслуговування комп'ютера за робочим місцем працівника; самочинно ремонтувати або намагатись здійснити технічне налагодження комп'ютера без залучення компетентних спеціалістів; складувати на робочому місці зайві документи, деталі та предмети, що не потрібні для роботи; використовувати монітори з нечітким зображенням та монітори, у яких наявні поламки екрану; працювати з матричним принтером без антивібраційного покриття та зі знятою кришкою. Допускати до роботи осіб, які не пройшли затверджений на підприємстві курс охорони праці для роботи з комп'ютером, не дозволяється.

3.7 Висновок до третього розділу

У третьому розділі розроблена автоматизована система та її інтерфейс, описані всі функції даної системи що вміє дана система та додані скріншоти веб-додатка.

Повністю описано послідовність етапів роботи користувача з розробленою системою. Обґрунтована перспектива розвитку даної автоматизованої системи, як її можливо модернізувати та пристосувати для інших промислових цілей.

Описана структура запитів та відповідей, як генерується та відправляться запит, куди він надходить, та в якому форматі надходить відповідь, також описана структура JSON файлу, і як з цього формату перекладається інформація, що виводиться користувачеві.

Проведено розрахунок економічної частини проекту, а саме прогнозування витрат дослідно-конструкторської та конструкторсько-технологічної роботи. Завдяки цім розрахункам встановили, що дана система є економічно вигідною та рентабельною.

ВИСНОВКИ

У магістерській кваліфікаційній роботі розроблено автоматизовану систему обробки даних з розподілених інформаційних джерел. Дана система дозволяє збирати дані з веб-сайтів, доступ до яких забезпечується, та виводити необхідні дані на веб-додатку, а саме дані про погоду на найближчі десять діб у всіх великих містах України.

Для демонстрації системи розроблено веб-додаток, на якому користувач може обирати та змінювати всі необхідні параметри.

Для реалізації додатку обрано програмну платформу Node.js, мову програмування JavaScript, та фреймворк Vue.Js, та декілька ключових бібліотек, такі як bootstrap js, jQuery. Також розроблено архітектуру системи, спроектовано модулі програми та кінцеві точки API.

Після тестування, було встановлено, що даний вид автоматизованої системи може підлаштовуватись до збору інформаційних даних з відкритих джерел.

Виконано прогнозування витрат на виконання роботи та прогнозування комерційних ефектів від результатів розробки. Встановлено, що даний вид автоматизованої системи має економічну доцільність та економічність. Тому такі системи доцільно впроваджувати на підприємства де проводиться збір інформації з веб-ресурсів для подальшого аналізу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Невлюдов, І.Ш. дипломне проектування для студентів усіх форм навчання спеціальностей 151 «автоматизація та комп'ютерно-інтегровані технології» [Текст] / І.Ш. Невлюдов, А. О. Андрусевич, О. В. Токарєва, Г. В. Пономарьова – Київ-58, пр. Космонавта Комарова, 1, 2016 – 320с.

2. ДСТУ 3008-15. документація. звіти у сфері науки і техніки. структура і правила оформлення [Текст] – Введ. 2015-06-22. – К. Держстандарт України, 2017 – 29 с.

3. Методичні вказівки до магістерської атестаційної роботи для студентів спеціальності 8.05090203 «Інтелектуальні технології мікросистемної радіоелектронної техніки» / Упоряд: І.Ш. Невлюдов, В.А. Палагін, Є.А. Разумов Фризюк, І.В. Жарікова. - Харків: ХНУРЕ. - 2011. - 49 с.

4. Рейтинг використання інформаційних технологій в корпораціях [Електронний ресурс] – режим доступу до ресурсу https://ela.kpi.ua/bitstream/123456789/22390/1/EV2017_437-443.

5. Трояновська Т. І. Метод розробки API / Т. І. Трояновська, М. Ф. Маховик // «Інформаційні технології та взаємодії» : IV Міжнародна науково-практична конференція, 8-10 листопада 2017 р. м. Київ. – С. 131-132.

6. API інтерфейси прикладного програмування: [Електронний ресурс] – режим доступу до ресурсу https://www.infoworld.com/article/3269878/what-is-an-api-application-programming-interfaces-explained.html.

7. Трояновська Т. І. Інформаційна технологія доставки контенту у системах комп'ютеризованої підготовки спеціалістів. // Гороховський О. І., Трояновська Т. І., Азаров О. Д. Монографія. Вінниця : ВНТУ, 2016.–160 с.

8. Савицька Л. А.. Метод розробки API підвищеної швидкодії / Л. А. Савицька, Т. І. Трояновська, М. Ф. Маховик // Наукові підсумки 2017 року, XV Міжнародна науково-практична інтернет-конференція. – м. Вінниця, 15 грудня 2017 року. – Ч.6, С. 50-53с.

9. API в веб-приложениях : [Електронний ресурс] – режим доступу до ресурсу : https://mkdev.me/posts/chto-takoe-api-v-veb-prilozheniyah-i-zachem-on-nuzhen.

10. Working with Web API : [Електронний ресурс] – режим доступу до ресурсу : https://launchschool.com/books/working_with_apis.

11. RESTful API Server – Doing it the right way: [Електронний ресурс] – режим доступу до ресурсу : http://blog.mugunthkumar.com/articles/restful-api-serverdoing-it-the-right-way-part-1/.

12. Подходы к проектированию RESTful API: [Електронний ресурс] – режим доступу до ресурсу : https://dataart.ru/news/podhody-k-proektirovaniyu-restfulapi/.

13. JSON пояснення формату: [Електронний ресурс] – режим доступу до ресурсу https://www.json.org/json-en.html.

14. SOAP API : [Електронний ресурс] – режим доступу до ресурсу : https://evmhistory.ru/programming/soap.html.

15. Что Такое API и Как Это Использовать: [Електронний ресурс] – режим доступу до ресурсу : https://www.mobidea.com/academy/ru/shto-takoe-api.

16. Розробка RPC: [Електронний ресурс] – режим доступу до ресурсу : https://ru.wikipedia.org/wiki/%D0%A3%D0%B4%D0%B0%D0%BB%D1%91%D0

17. Розробка моделі проектування MVC: [Електронний ресурс] – режим доступу до ресурсу : https://ru.wikipedia.org/wiki/Model-View-Controller.

18. Структура MVC: [Електронний ресурс] – режим доступу до ресурсу : https://ru.wikipedia.org/wiki/Model-View-Controller.

19.Що таке MVC [Електронний ресурс] – режим доступу до ресурсу : https://tproger.ru/articles/mvc.

20.Довідник по API [Електронний ресурс] – режим доступу до ресурсу : https://stripe.com/docs/api.

21.Розбір фреймворку Vue.Js. [Електронний ресурс] – режим доступу до ресурсу : https://timeweb.com/ru/community/articles/obzor-vue-js-1

22.Що таке Node.Js [Електронний ресурс] – режим доступу до ресурсу:

https://netology.ru/blog/node.

23.Використання puppeteer [Електронний ресурс] – режим доступу до ресурсу https://habr.com/ru/company/ruvds/blog/341348.

24.Метод розробки API на JavaScript [Електронний ресурс] – режим доступу до ресурсу http://inmad.vntu.edu.ua/portal/static/4BEBB20A-1D45-4219-A13A-EEA58400D78B.pdf.

25.Структура запитів та відповідей [Електронний ресурс] – режим доступу до ресурсу https://developer.mozilla.org/ru/docs/Web/HTTP/Overview.

26.Структура JSON файлу [Електронний ресурс] – режим доступу до ресурсу https://itchief.ru/javascript/json#id-1.

27. Переваги і недоліки автоматизації виробництва [Електронний ресурс] – режим доступу до ресурсу http://ivbud.com/interesting/8135-perevahy-i-nedoliky-avtomatyzatsii-vyrobnytstva.

28. Десять інструментів, що дозволяють парсити інформацію з веб-сайтів [Електронний ресурс] – режим доступу до ресурсу https://habr.com/ru/post/340038/.