Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка автоматизованої системи для супроводу процесів реєстрації та життєвого циклу доменних імен

Виконав: студент VI курсу, групи СНнм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

(підпис) Мац О. І. (прізвище та ініціали)

Керівник (підпис) Млинко Б. Б. (прізвище та ініціали)

Нормоконтроль (підпис) Готович В. А. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Луцик Н. С. (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_» травня 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Мацу Олегу Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка автоматизованої системи для супроводу процесів реєстрації та життєвого циклу доменних імен

Керівник роботи Млинко Богдана Богданівна, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 24 » листопада 2023 року № 4/7-1100

2. Термін подання студентом завершеної роботи 27 травня 2024р.

3. Вихідні дані до роботи Літературні джерела, інтернет джерела, технічна документація, матеріали практики.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз системи доменних імен та існуючих проблем пов'язаних з життєвим циклом доменних імен. 2 Аналіз засобів для реалізації системи моніторингу та реєстрації доменних імен. 3 Розробка та тестування системи моніторингу та реєстрації доменних імен. 4 Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульна сторінка. 2 Актуальність. 3 Мета. 4 Задачі. 5 Об'єкт та предмет дослідження. 6 Аналіз відомих систем моніторингу та реєстрації. 7 Проблематика реєстрації та відновлення доменів. 8 Життєвий цикл домена. 9 Аналіз засобів для реалізації системи. 10 Платформа Anvil.works. 11 Розробка та тестування. 12 Порівняльний аналіз запропонованого рішення та відомих сервісів. 13 Висновки. 14 Завершальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В. С., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В. М., ст. викладач		

7. Дата видачі завдання 24 листопада 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	25.11.2023	*Виконано*
2.	Підбір наукових джерел про систему доменних імен	26.11.2023-28.11.2023	*Виконано*
3.	Опрацювання наукових публікацій та збір даних по	29.11.2023-1.12.2023	*Виконано*
	темі роботи		
4.	Виконання дослідження згідно мети кваліфікаційної	2.12.2023-4.12.2023	*Виконано*
	роботи		
5.	Оформлення розділу 1 «Аналіз системи доменних імен	5.12.2023-7.01.2024	*Виконано*
	та існуючих проблем пов’язаних з життєвим циклом		
	доменних імен»		
6.	Оформлення розділу 2 «Аналіз засобів для реалізації	8.01.2024-10.02.2024	*Виконано*
	системи моніторингу та реєстрації доменних імен»		
7.	Оформлення розділу 3 «Розробка та тестування	11.02.2024-13.04.2024	*Виконано*
	системи моніторингу та реєстрації доменних імен»		
8.	Виконання завдання до підрозділу «Охорона праці»	14.04.2024-30.04.2024	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	1.05.2024-17.05.2024	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	18.05.2024-19.05.2024	*Виконано*
11.	Нормоконтроль	20.05.2024-21.05.2024	*Виконано*
12.	Перевірка на плагіат	22.05.2024	*Виконано*
13.	Попередній захист кваліфікаційної роботи		*Виконано*
14.	Захист кваліфікаційної роботи		

Студент _____ Мац О. І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Млинко Б. Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка автоматизованої системи для супроводу процесів реєстрації та життєвого циклу доменних імен // Кваліфікаційна робота освітнього рівня «Магістр» // Мац Олег Ігорович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНнм-61 // Тернопіль, 2024 // С. 77, рис. – 40, табл. – 4, кресл. – 14, додат. – 4, бібліогр. – 60.

Ключові слова: домен, веб-сайт, веб-додаток, система, реєстрар, регістрі, api, dns

Кваліфікаційна робота присвячена розробці автоматизованої системи для супроводу процесів реєстрації та життєвого циклу доменних імен.

Робота складається із вступу, чотирьох розділів, висновку, списку посилань на використану літературу та додатків.

В першому розділі кваліфікаційної роботи описана система доменних імен. Висвітлено основні етапи життєвого циклу доменного імені. Розглянуто проблеми пов'язані з життєвим циклом доменів, з якими стикаються їх власники.

В другому розділі кваліфікаційної роботи досліджено доступні засоби для реалізації системи моніторингу та реєстрації доменних імен. Досліджено популярні мови програмування, а також розглянуто хостингові платформи для розміщення додатку. Подано та проаналізовано ряд сервісів які надають послуги з моніторингу доменних імен та підтримують API інтеграцію.

В третьому розділі кваліфікаційної роботи описано етапи розробки та тестування автоматизованої системи моніторингу та реєстрації доменних імен. Проаналізовано відомі системи моніторингу доменних імен. Проведено порівняльний аналіз запропонованого рішення з відомими системами моніторингу.

Четвертий розділ присвячений охороні праці та безпеці життєдіяльності.

ANNOTATION

Development of an automated system to accompany registration processes and domain names life cycle // The educational level "Master" qualification work // Mats Oleh // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, SNnm-61 group // Ternopil, 2024 // P. 77, fig. - 40, tables - 4, posters – 17, annexes - 4, ref. - 60.

Key words: domain, website, web application, system, registrar, registry, api, dns.

The qualification work is devoted to the development of an automated system to accompany registration processes and domain names life cycle.

The work consists of an introduction, four chapters, a conclusion, a list of references and appendices.

The first section of the qualification work contains a description of the domain name system. The main stages of the life cycle of a domain name are highlighted. The problems related to the life cycle of domains faced by their owners are considered.

In the second section of the qualification work, the available tools for the implementation of the domain name monitoring and registration system were investigated. Popular programming languages were reviewed, as well as hosting platforms for hosting the application were considered. A number of services that provide domain name monitoring services and support API integration are presented and analyzed.

The third section of the qualification work describes the stages of development and testing of an automated system. Well-known domain name monitoring systems were analyzed. A comparative analysis of the proposed solution with known monitoring systems was carried out.

The fourth chapter is devoted to the safety of work and health.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І ТЕРМІНІВ

Веб-сайт – сукупність веб-сторінок та залежного вмісту, доступних у мережі Інтернет, які об’єднані як за змістом, так і за навігацією під єдиним доменним ім’ям. Фізично сайт може розміщуватися як на одному, так і на кількох серверах.

Доменне ім’я (англ. Domain name), або Домен (англ. Domain) – частина простору ієрархічних імен мережі Інтернет, що обслуговується групою серверів системи доменних імен (DNS-серверів) та централізовано адмініструється.

Реєстр доменних імен або регістрі (англ. Registry) – це база даних усіх доменних імен певної доменної зони. Як правило, регістрі надають послуги лише акредитованим реєстраторам, не співпрацюючи з кінцевими користувачами напряму.

Реєстратор (реєстрар) доменних імен (англ. Registrar) – це юридична особа, яка має право на створення нового доменного імені. Також вона має право на подовження терміну вже наявних доменних імен у самому домені. Реєстратори співпрацюють з регістрі та надають послуги кінцевим користувачам доменних імен.

Система доменних імен (англ. Domain Name System, DNS) – ієрархічна розподілена система перетворення імені хоста (комп’ютера або іншого мережевого пристрою) в IP-адресу.

Фреймворк (англ. Framework) – інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою.

API (англ. Application Programming Interface) – Прикладний Програмний Інтерфейс.

IP-адреса (англ. Internet Protocol address) – унікальний числовий ідентифікатор мережевого рівня, що використовується для адресації комп’ютерів чи пристроїв у мережах, які побудовані з використанням стека протоколів TCP/IP (наприклад, Інтернет).

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному цифровому світі, де інтернет-простір стрімко розширюється, доменні імена відіграють ключову роль як фундаментальні елементи веб-присутності будь-якої організації чи індивідуального користувача. Вони не тільки слугують цифровою адресою для веб-сайтів, але й є важливою частиною корпоративної ідентичності та маркетингових стратегій. В цьому контексті, ефективне управління доменними іменами, що охоплює їх реєстрацію, відновлення, моніторинг та управління життєвим циклом, стає критично важливим завданням.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Магістр» є створення автоматизованої системи для супроводу процесів реєстрації та життєвого циклу доменних імен, яка б слугувала комплексним рішенням для спрощення існуючих процедур управління доменами та забезпечення власників доменів потужним інструментом для ефективного моніторингу та управління їхніми доменними портфелями. Для досягнення поставленої мети потрібно виконати ряд задач, зокрема:

- Дослідити відомі на даний час системи для автоматичної реєстрації доменів.
- Проаналізувати основні проблеми, з якими стикаються власники доменів та адміністратори веб-сайтів, включаючи проблематику втрати доменів через пропущені терміни відновлення, складнощі моніторингу статусу доменів, які можуть звільнитися, та потребу в автоматизації процесів реєстрації.
- Провести порівняльний аналіз відомих засобів для реалізації систем моніторингу.
- Проаналізувати методи які можна використати для створення власної спеціалізованої системи для моніторингу та реєстрації доменних імен.
- Розробити та протестувати систему для моніторингу та реєстрації доменних імен.

Об’єкт дослідження. Процеси моніторингу та реєстрації доменних імен.

Предмет дослідження. Методика створення автоматизованих систем для реєстрації доменних імен.

Наукова новизна одержаних результатів полягає в тому що вперше запропоновано економічно вигідну альтернативу для автоматизації моніторингу статусів доменних імен та реєстрації доменів.

Практичне значення одержаних результатів. Створено ефективну систему для автоматичного моніторингу та реєстрації доменних імен, яку можна вдосконалювати та масштабувати.

Апробація результатів магістерської роботи. Основні результати проведених досліджень обговорювались на VII міжнародній студентській науково-технічній конференції «Природничі та гуманітарні науки. Актуальні питання» Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 2024 р.).

Публікації. Основні результати кваліфікаційної роботи опубліковано у двох працях конференції (Див. додатки А та Б).

Структура й обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури з 60 найменувань та 4 додатків. Загальний обсяг кваліфікаційної роботи складає 77 сторінок, з них 61 сторінка основного тексту, який містить 40 рисуноків та 4 таблиці.

1 АНАЛІЗ СИСТЕМИ ДОМЕННИХ ІМЕН ТА ІСНУЮЧИХ ПРОБЛЕМ ПОВ'ЯЗАНИХ З ЖИТТЄВИМ ЦИКЛОМ ДОМЕННИХ ІМЕН

1.1 Огляд системи доменних імен (DNS)

Система доменних імен (DNS) є однією з основних технологій Інтернету, яка дозволяє користувачам легко орієнтуватися в мережі без необхідності запам'ятовувати складні числові адреси. DNS перетворює зручні для людини доменні імена (наприклад, example.com) на IP-адреси (наприклад, 192.0.2.1), які використовуються для ідентифікації кожного ресурсу в Інтернеті [36].

Головна функція DNS полягає у перекладі доменних імен на відповідні IP-адреси та навпаки. Цей процес, відомий як розв'язування імен, є критично важливим для функціонування Інтернету, оскільки людям легко запам'ятовувати імена, тоді як мережеве обладнання працює з числовими адресами [36].

1.1.1 Історія

Користування простішим, запам'ятовуваним іменем замість числової адреси хосту відноситься до епохи ARPANET. Інститут досліджень Стенфорда (тепер SRI International) підтримував текстовий файл HOSTS.TXT, який відображав імена хостів на числові адреси комп'ютерів в ARPANET. Елізабет Фейнлер розробила і підтримувала перший каталог ARPANET. Підтримка числових адрес відбувалася через Список Призначених Чисел із співпраці з Джоном Постелом в Інституті наук про інформацію університету Південної Каліфорнії (ISI), чий тим взаємодіяв з SRI [49].

Адреси присвоювалися вручну. Комп'ютери, включаючи їх імена хостів та адреси, додаються до основного файлу за допомогою звернення до Центру інформації мережі SRI (NIC), яким керувала Фейнлер, за допомогою телефону протягом робочих годин. Пізніше Фейнлер створила WHOIS-каталог на сервері в NIC для отримання інформації про ресурси, контакти та суб'єкти. Вона та її команда розробили концепцію доменів. Фейнлер запропонувала, що домени повинні

базуватися на місці розташування фізичної адреси комп'ютера. Комп'ютери в навчальних закладах мали б домен edu, наприклад. Вона та її команда керували Реєстром імен хостів з 1972 по 1989 рік [49].

До початку 1980-х років ведення єдиної, централізованої таблиці хостів стало повільним і незручним, і з'явилася потреба в автоматизованій системі іменування для вирішення технічних та кадрових питань. Постел направив завдання вирішення компромісу між п'ятьма конкуруючими пропозиціями рішень до Пола Мокапетріса. Мокапетріс замість цього створив систему доменних імен в 1983 році, будучи університеті Південної Каліфорнії [49].

Робоча група Інтернет-інженерії опублікувала оригінальні специфікації в RFC 882 та RFC 883 у листопаді 1983 року. Вони були оновлені у RFC 973 у січні 1986 року [49].

У 1984 році чотири студенти з UC Berkeley, Дуглас Террі, Марк Пейнтер, Девід Ріггл та Сонгніан Чжоу, написали першу реалізацію сервера імен Unix для Berkeley Internet Name Domain, який загалом називається BIND. У 1985 році Кевін Данлап з DEC значно переробив реалізацію DNS. Майк Карелс, Філ Альмквіст та Пол Віксі потім взяли на себе підтримку BIND. Консорціум Інтернет-систем був заснований у 1994 році Ріком Адамсом, Полом Віксі та Карлом Маламудом, в основному для забезпечення місця для розробки та підтримки BIND. Від версії 4.9.3 BIND розвивався та підтримувався ISC, з підтримкою від спонсорів ISC. Як співрозробники/програмісти, Боб Геллі та Пол Віксі випустили першу готову до виробництва версію BIND версії 8 у травні 1997 року. З 2000 року над BIND працювали понад 43 різних основних розробники [49].

У листопаді 1987 року RFC 1034 та RFC 1035 замінили специфікації DNS 1983 року. Кілька додаткових запитів на коментарі пропонували розширення основних протоколів DNS [49].

1.1.2 Ключові поняття DNS

Домен (англ. domain) – це частина простору ієрархічних імен мережі Інтернет, яка обслуговується групою серверів доменних імен (DNS-серверів) та

централізовано адмініструється. DNS-сервери зберігають інформацію про вузли, чиї імена належать домену, і виконують трансляцію цих імен у адреси. Кожен домен має унікальне ім’я, а кожен комп’ютер, підключений до Інтернету, зазвичай має доменне ім’я. Домени мають ієрархічні відносини. Два домени, розташовані на сусідніх рівнях ієрархії, називаються відповідно доменом вищого та нижчого рівнів. Домени найвищого (верхнього) рівня можуть бути сформовані за організаційним або географічним принципами. Географічні домени об’єднують вузли, що належать конкретній державі, в основному комп’ютери, що розташовані на території США [48].

Піддомен (англ. subdomain) – це підлеглий домен (наприклад, wikipedia.org є піддоменом домену org, а uk.wikipedia.org – піддоменом домену wikipedia.org). Теоретично такий розподіл може досягати глибини до 127 рівнів, і кожна мітка може містити до 63 символів, поки загальна довжина, включаючи крапки, не перевищить 254 символів. Проте на практиці реєстратори доменних імен застосовують більш суворі обмеження. Наприклад, якщо у вас є домен виду mydomain.ua, ви можете створювати для нього різні піддомени, такі як mysite1.mydomain.ua, mysite2.mydomain.ua тощо [36].

Зона (англ. zone) – це частина дерева доменних імен (включаючи ресурсні записи), яка розміщується як єдине ціле на деякому сервері доменних імен (DNS-сервері), а частіше – одночасно на декількох серверах. Метою виділення частини дерева в окрему зону є передача відповідальності за відповідний домен іншій особі або організації. Це називається делегуванням. Як зв’язкова частина дерева, зона всередині теж являє собою дерево. Якщо розглядати простір імен DNS як структуру із зон, а не окремих вузлів/імен, також утворюється дерево; можна говорити про батьківські та дочірні зони, про старші та підлеглі. На практиці більшість зон 0-го і 1-го рівня (ua, com тощо) складаються з єдиного вузла, якому безпосередньо підпорядковуються дочірні зони. У великих корпоративних доменах (2-го і більше рівнів) іноді створюються додаткові підпорядковані рівні без виділення їх у дочірні зони [36].

Делегування – це операція передачі відповідальності за частину дерева доменних імен іншій особі або організації. Завдяки делегуванню в DNS

забезпечується розподільність, адміністрування та зберігання даних. Технічно делегування виражається у виділенні цієї частини дерева в окрему зону та розміщенні цієї зони на DNS-сервері, керованому цією особою чи організацією. При цьому в батьківську зону включаються «склеюючі» ресурсні записи (NS і A), які містять вказівки на DNS-сервери дочірньої зони, а вся інша інформація, що належить до дочірньої зони, зберігається вже на DNS-серверах цієї дочірньої зони [35].

DNS-сервер – це програма, призначена для відповідей на DNS-запити за відповідним протоколом. Також DNS-сервером можуть називати хост, на якому запущено цю програму [36].

DNS-клієнт – це програма або модуль у програмі, що забезпечує з'єднання з DNS-сервером для визначення IP-адреси за його доменним іменем [36].

Авторитетність (англ. authoritative) – характеристика, що вказує на розміщення зони на DNS-сервері. Відповіді DNS-сервера можуть бути двох типів: авторитетні (коли сервер заявляє, що він відповідає за цю зону) і неавторитетні (англ. non-authoritative), коли сервер обробляє запит і повертає відповідь від інших серверів. У деяких випадках, замість передачі запиту далі, DNS-сервер може повернути вже відоме йому (отримане від попередніх запитів) значення в режимі кешування [48].

DNS-запит (англ. DNS query) – це запит від клієнта або сервера до DNS-сервера. Запити можуть бути рекурсивними або нерекурсивними (див. Рекурсія). Система DNS включає ієрархію DNS-серверів, яка відповідає ієрархії зон [36].

Кожна зона обслуговується як мінімум одним авторитетним DNS-сервером (англ. authoritative), який містить інформацію про домен [48].

1.1.3 DNS записи

Записи DNS, або ресурсні записи (англ. Resource Records або Host Records) – це одиниці зберігання і передачі інформації в системі DNS. Кожен ресурсний запис складається з таких полів:

- Ім’я (NAME) – доменне ім’я, до якого прив’язаний або якому «належить» цей ресурсний запис.
- TTL (Time To Live) – допустимий час зберігання цього ресурсного запису в кеші неавторитетного DNS-сервера (вимірюється в секундах).
- Тип (TYPE) ресурсного запису – визначає формат і призначення цього ресурсного запису.
- Клас (CLASS) ресурсного запису – теоретично передбачається, що DNS може використовуватися не тільки з TCP/IP, але й з іншими типами мереж, і код у полі класу визначає тип мережі.
- Довжина поля даних (RDLEN).
- Поле даних (RDATA), формат і зміст якого залежать від типу запису.

Найбільш важливі типи DNS-записів:

- Запис A (address record) або адресний запис зв’язує ім’я хоста з IP-адресою. Наприклад, запит A-запису для імені referrals.icann.org поверне його IP-адресу – 192.0.34.164.
- Запис AAAA (IPv6 address record) зв’язує ім’я хоста з адресою протоколу IPv6. Наприклад, запит AAAA-запису для імені K.ROOT-SERVERS.NET поверне його IPv6-адресу – 2001:7fd::1.
- Запис CNAME (canonical name record) або канонічний запис імені (псевдонім) використовується для перенаправлення на інше ім’я.
- Запис MX (mail exchange) або поштовий обмінник вказує сервер(и) обміну поштою для даного домену.
- Запис NS (name server) вказує на DNS-сервер для даного домену.
- Запис PTR (pointer) або покажчиковий запис зв’язує IP-адресу хоста з його канонічним ім’ям. Запит в домені in-addr.arpa на IP-адресу хоста у зворотній формі поверне повне доменне ім'я (FQDN) цього хоста (див. зворотний запит DNS). Наприклад, для IP-адреси 192.0.34.164: запит PTR-запису 164.34.0.192.in-addr.arpa поверне його канонічне ім’я referrals.icann.org. Для зменшення обсягу небажаної кореспонденції (спаму) багато серверів-одержувачів електронної пошти перевіряють наявність PTR-запису для хоста, з якого відбувається

відправлення. У цьому випадку PTR-запис для IP-адреси має відповідати імені відправляючого поштового сервера, яким він представляється в процесі SMTP-сесії.

- Запис SOA (Start of Authority) або початковий запис зони вказує, на якому сервері зберігається еталонна інформація про даний домен. Він містить контактну інформацію особи, відповідальної за дану зону, параметри часу кешування зонної інформації та таймінги для взаємодії DNS-серверів.
- Запис SRV (server selection) вказує на сервери для різних сервісів, використовується, зокрема, для Jabber і Active Directory [36].

Далі потрібно розглянути синтаксис доменних імен.

1.1.4 Синтаксис доменних імен

Доменне ім’я складається з одного або кількох елементів, технічно названих мітками, які традиційно з'єднуються і розділяються крапками, наприклад, example.com. Найправіша мітка вказує на домен верхнього рівня; наприклад, доменне ім’я www.example.com належить до домену верхнього рівня com [1].

Ієрархія доменів спускається від правої до лівої мітки в назві; кожна мітка ліворуч вказує на підрозділ або піддомен домену праворуч. Наприклад, мітка example вказує на вузол example.com як піддомен домену com, а www є міткою для створення www.example.com, піддомену example.com. Кожна мітка може містити від 1 до 63 октетів. Порожня мітка залишена для кореневого вузла. Повне доменне ім’я не може перевищувати загальну довжину 253 символів ASCII у своєму текстовому представленні [48].

Хостнейм – це доменне ім’я, яке має принаймні одну пов’язану IP-адресу. Наприклад, доменні імена www.example.com та example.com також є хостнеймами, тоді як домен com не є. Проте інші домени верхнього рівня, зокрема кодові домени верхнього рівня країн, можуть мати IP-адресу, і якщо це так, то вони також є хостнеймами [15]. Хостнейми накладають обмеження на символи, дозволені у відповідному доменному імені. Дійсний хостнейм також є дійсним доменним ім’ям, але дійсне доменне ім’я не обов’язково є дійсним хостнеймом [58].

1.1.5 Принцип роботи DNS

Доменне ім’я та IP-адреса не є тотожними поняттями – одна IP-адреса може бути пов’язана з багатьма доменними іменами, що дозволяє одному комп’ютеру обслуговувати декілька веб-сайтів (цей процес відомий як віртуальний хостинг). Також, одному доменному імені можна призначити кілька IP-адрес, що сприяє розподілу навантаження між серверами. Це забезпечує високу доступність системи завдяки використанню декількох серверів з ідентичним вмістом та механізмами синхронізації даних між ними [48].

DNS має ієрархічну структуру, на вершині якої знаходяться 13 кореневих серверів. Від них ідуть домени верхнього рівня (TLDs), такі як .com, .net, .org, країнові коди (наприклад .UA, .EU) та нові загальні домени верхнього рівня. Кожен домен верхнього рівня керується організацією-реєстром, яка відповідає за делегування доменів другого рівня. Домени другого рівня – це те, що ми звикли бачити в адресному рядку кожного веб-сайту, наприклад «google.com». Така структура допомагає ефективно масштабувати систему та підтримувати її роботу у випадку збоїв деяких серверів [36].

Коли користувач вводить доменне ім’я в браузері, запит спочатку надходить до рекурсивного сервера (зазвичай наданого провайдером інтернет-послуг). Якщо відповідь не знаходиться в кеші, рекурсивний сервер звертається до кореневих серверів. Кореневі сервери направляють на сервери імен для відповідного TLD, які, у свою чергу, вказують на авторитетні сервери імен для конкретного домену, де можна отримати кінцеву IP-адресу [24].

На рисунку 1.1 зображено приклад черги DNS яка опрацьовується при першому відвідуванні веб-сайту користувачем.

Таким чином, щоб відвідати певний веб-сайт, браузер має опрацювати цілий ланцюг послідовних запитів та відповідей від DNS-серверів, так звану DNS чергу (DNS Query). При всіх наступних відвідуваннях веб-сайту отримані дані будуть братися з локального DNS кешу пристрою користувача або DNS кешу інтернет-провайдера, доки ці кеші не будуть автоматично очищені [3].

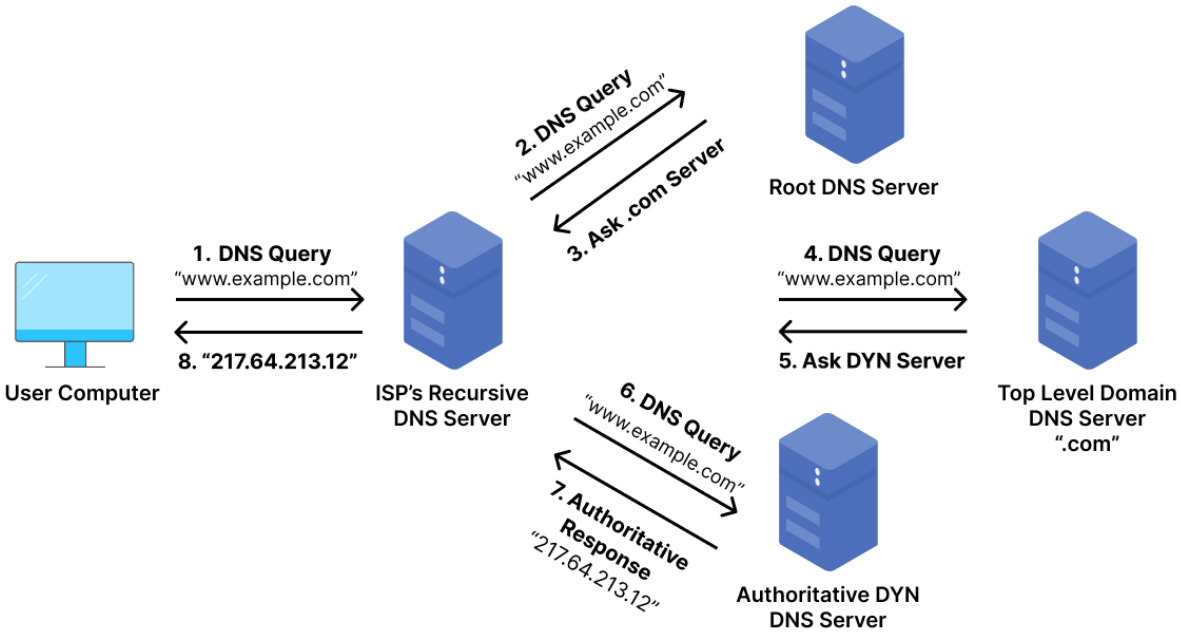

Рисунок 1.1 – Приклад DNS черги для першого відвідування веб-сайту www.example.com

DNS не лише спрощує доступ до ресурсів Інтернету, але й підтримує глобальну масштабованість мережі, що дозволяє з легікістю додавати нові сайти та керувати ними. Крім того, DNS сприяє розподілу навантаження, забезпеченню безпеки та можливості швидкої зміни місця розташування ресурсів в Інтернеті без зміни їхніх назв [34].

Система доменних імен стикається з багатьма викликами, особливо з питаннями безпеки, такими як DNS spoofing або cache poisoning, де зловмисники можуть перенаправляти користувачів на шкідливі сайти, підробляючи DNS-відповіді. Для боротьби з цими загрозами було розроблено розширення безпеки DNS (DNSSEC), яке додає цифрові підписи до DNS-даних, дозволяючи перевіряти їхню автентичність [38].

DNS є критично важливою технологією, яка забезпечує доступність інтернет-ресурсів по всьому світу. Розуміння принципів роботи та структури DNS є важливим для розробників та адміністраторів веб-сайтів, оскільки це сприяє ефективному управлінню доменними іменами та забезпеченню безпеки веб-простору [31].

1.1.6 Проблеми з безпекою

Спочатку безпека не була основною вимогою при проєктуванні програмного забезпечення DNS або будь-якого іншого програмного забезпечення для раннього Інтернету, оскільки мережа не була доступна для загальної громадськості. Однак у 1990-х роках, коли Інтернет почав активно використовуватися в комерційному секторі, вимоги до заходів безпеки для захисту цілісності даних та аутентифікації користувачів значно зросли [23].

Було виявлено кілька вразливостей, які зловмисники змогли використати. Однією з таких вразливостей є отруєння кеша DNS, коли дані передаються до кешуючих резолверів під виглядом даних від авторитетного сервера, тим самим забруднюючи сховище даних потенційно неправдивою інформацією та тривалими часами життя (time-to-live). У результаті, законні запити можуть бути перенаправлені на мережеві хости, що діють із зловмисними намірами [32].

Відповіді DNS традиційно не мають криптографічного підпису, що створює багато можливостей для атак. Розширення безпеки системи доменних імен (DNSSEC) змінюють DNS, додаючи підтримку криптографічно підписаних відповідей [20]. Як альтернатива DNSSEC, було запропоновано DNSCurve. Інші розширення, такі як TSIG, забезпечують криптографічну аутентифікацію між довіреними пірами і часто використовуються для авторизації операцій з передачі зони або динамічного оновлення [22].

Деякі доменні імена можуть використовуватися для обману користувачів. Наприклад, paypal.com і paypa1.com – різні імена, але користувачі можуть не розрізнити їх у графічному інтерфейсі залежно від вибраного шрифту. У багатьох шрифтах літера "l" та цифра "1" виглядають дуже схоже або навіть однаково. Ця проблема, відома як атака гомографів IDN, є актуальною в системах, які підтримують інтернаціоналізовані доменні імена, оскільки багато символів в ISO 10646 можуть виглядати однаково на звичайних комп'ютерних екранах. Цю вразливість часто використовують для фішингу [22].

Методи, такі як перевірка зворотного DNS шляхом здійснення прямого DNS-запиту, можуть бути використані для підтвердження результатів DNS [25].

Якщо не приділити належної уваги налаштуванню, DNS може "витікати" з безпечних або приватних з'єднань, і в деяких випадках зловмисники використовували DNS для обходу брандмауерів та викрадення даних, оскільки DNS часто вважається безпечним [40].

1.2 Реєстрація доменних імен

Право використання доменного імені делегується реєстраторами доменних імен, які мають акредитацію від Корпорації з присвоєння імен і номерів в Інтернеті (ICANN) або інших організацій, які контролюють системи імен і номерів в Інтернеті [33]. Крім ICANN, кожен домен верхнього рівня (TLD) технічно обслуговується адміністративною організацією, яка керує реєстром. Регістрі відповідає за роботу бази даних імен у своїй авторитетній зоні (TLD). Реєстрант – це особа або організація, яка подала заявку на реєстрацію домену. Реєстр отримує інформацію про реєстрацію від кожного реєстратора доменних імен, який має акредитацію на призначення імен у відповідній зоні, і публікує цю інформацію за допомогою протоколу WHOIS. З 2015 року розглядається використання RDAP [17].

ICANN публікує список TLD, реєстрів TLD та реєстраторів доменних імен. Інформація про реєстранта, пов'язана з доменними іменами, зберігається в онлайн-базі даних, доступній через сервіс WHOIS. Для більшості з понад 290 кодових доменів верхнього рівня країн (ccTLDs), реєстри зберігають WHOIS-інформацію, яка включає дані про реєстранта, сервери імен та дати закінчення реєстрації. Наприклад, за управління даними домену .de відповідає DENIC, німецький національний центр інформації про мережі. Починаючи з 2001 року, більшість реєстрів загальних доменів верхнього рівня (gTLD) перейшли на так званий "товстий" підхід, при якому WHOIS-дані зберігаються у центральних реєстрах, а не в базах даних реєстраторів [33].

Домени верхнього рівня, такі як COM і NET, працюють за принципом «тонкого» реєстру. В цій моделі реєстратори доменів (наприклад, GoDaddy, BigRock, PDR, VeriSign та інші) утримують основні WHOIS дані, такі як

інформація про реєстратора і сервери імен. У контрасті, домени ORG використовують виключно реєстр [30].

Деякі реєстри доменних імен, часто відомі як центри інформації мережі (NIC), функціонують також як реєстратори для кінцевих користувачів, пропонуючи доступ до WHOIS даних. Реєстри для доменів верхнього рівня, таких як COM, NET і ORG, застосовують модель реєстр-реєстратор, що включає багатьох реєстраторів доменних імен. У цій системі реєстр відповідає тільки за базу даних доменних імен і відносини з реєстраторами. Реєстранти (власники доменних імен) є клієнтами реєстратора, інколи через додаткові угоди з реселлерами [30].

1.3 Життєвий цикл доменних імен

Життєвий цикл доменного імені є ключовим аспектом управління доменами, який важливо розуміти власникам веб-сайтів, розробникам, а також фахівцям з IT-безпеки. Він описує послідовність станів, через які проходить доменне ім’я від моменту його реєстрації до видалення з реєстру. Розуміння цього циклу допомагає запобігти небажаній втраті домену та використовувати можливості, пов’язані з доменами, що стають доступними [35].

Різні доменні реєстратори та регістрі можуть по-різному називати певні етапи життєвого циклу доменного імені, встановлювати їм різні часові рамки або взагалі оминати деякі стани доменів. Втім, загальну послідовність станів доменного імені можна охарактеризувати наступними станами:

- Реєстрація. Життєвий цикл доменного імені починається з моменту його реєстрації. Власник вибирає доступне доменне ім’я та реєструє його через акредитованого реєстратора на певний термін, який зазвичай становить від 1 (мінімальний період) до 10 років (максимальний період).
- Активний стан. Після реєстрації домен перебуває в активному стані. В цей період власник може використовувати домен як адресу для веб-сайту, налаштувати електронну пошту та інші сервіси. Власник має право в будь-який час (під час цього стану) продовжити термін дії домену.

- Пільговий період (англ. Grace Period). Після закінчення терміну реєстрації, якщо домен не було продовжено, він входить в період благодійності. Цей період триває від 30 до 45 днів, протягом якого власник може продовжити реєстрацію домену за звичайною ціною (без додаткових штрафів). Хоч ніхто і не змушує доменних реєстраторів давати такий період своїм користувачам, його наявність стала стандартом в індустрії.

- Період утримання (англ. Redemption Period). Якщо доменне ім'я не було продовжено після завершення періоду благодійності, воно переходить у стан утримання. Цей період триває близько 30 днів. Відновлення домену в цей час все ще можливе, але за значно вищу плату (до стандартної ціни поновлення додається комісія яку з реєстратора стягує регістрі).

- Видалення. Після періоду утримання, якщо домен так і не був продовжений, він входить в 5-денний період видалення, під час якого домен неможливо поновити.

- Вільний стан. Після періоду видалення доменне ім'я знову стає доступним для реєстрації будь-ким. На відміну від періоду утримання, де право викупити домен (хоч і за значну ціну) мав лише власник домена, тут починає діяти правило «першим прибув, першим отримав» (англ. «first come, first served») і будь-хто може зареєструвати домен наново за звичайною ціною [35].

Це створює можливості для інвесторів у домени та нових власників, які шукають популярні або цінні імена.

Всі фази життєвого циклу доменного імені зображено на рисунку 1.2.

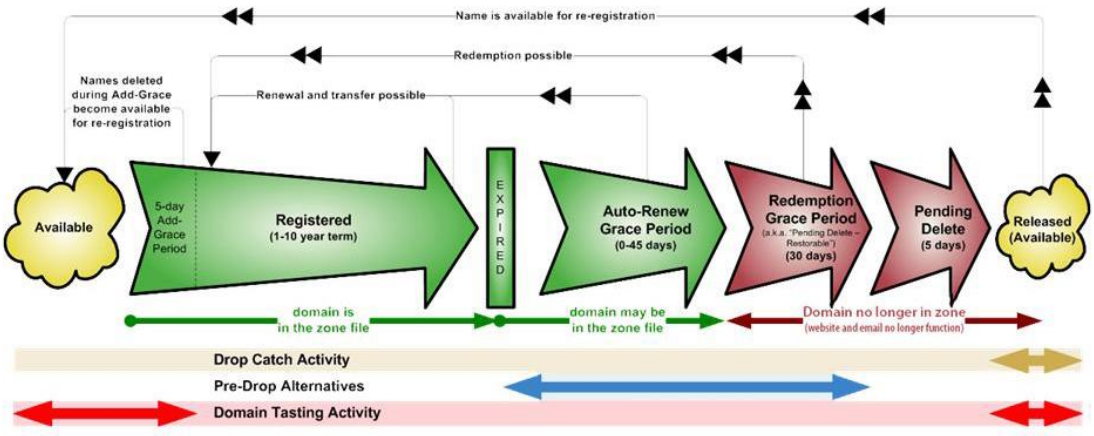

Рисунок 1.2 – Життєвий цикл доменного імені

Розуміння життєвого циклу доменних імен дозволяє власникам доменів уникнути втрати контролю над своїми доменами через їх несвоєчасне подовження. Також це знання корисне для тих, хто займається покупкою та продажем доменів, оскільки дає можливість ідентифікувати моменти, коли цінні домени можуть звільнитися [9].

1.4 Проблематика реєстрації та відновлення доменів

Процес реєстрації та відновлення доменних імен, хоча і здається простим на перший погляд, має ряд проблематичних аспектів, які можуть створювати труднощі для власників доменів та адміністраторів веб-сайтів. Ці проблеми включають, але не обмежуються, питаннями забезпечення безпеки, управління доменами, відновленням після закінчення терміну дії та суперечками щодо власності. Серед поширених проблем можна виділити наступні:

- Вибір доменного імені: Зі зростанням кількості веб-сайтів стає все складніше знайти бажане доменне ім’я, яке ще не зайняте. Це може призвести до вибору довших і менш запам’ятовуваних імен, що негативно впливає на бренд та маркетинг.
- Спекуляції на доменних іменах. Інвестори часто реєструють доменні імена з метою їх подальшого продажу за значно вищими цінами, що ускладнює доступ до привабливих імен для звичайних користувачів та компаній.
- Фішинг та шахрайство. Шахраї можуть реєструвати доменні імена, схожі на популярні бренди, для створення фішингових сайтів, що підриває довіру до доменної системи.
- Проблеми з відновленням доменів та висока вартість відновлення. Пропуск терміну відновлення: Власники доменів можуть пропустити сповіщення про закінчення терміну дії домену через фільтрацію спаму або зміну електронної пошти, що призводить до втрати домену. Після переходу домену в стан утримання (Redemption Period) вартість його відновлення може бути значно вищою, що створює фінансові труднощі для власників.

- Автоматичне поновлення. Налаштування автоматичного поновлення може призвести до несподіваного списання коштів з рахунку власника, особливо якщо вони більше не зацікавлені в домені.
- Суперечки щодо власності на доменні імена. Такі суперечки можуть виникати через незаконне використання торгових марок, кіберсквотинг або непорозуміння між сторонами, які вважають себе законними власниками домену. Вирішення таких суперечок часто вимагає звернення до юридичних процедур або арбітражу [43].

Проблематика реєстрації та відновлення доменів вимагає від власників доменів уважності, планування та розуміння правил роботи доменної системи. Важливо вчасно відновлювати доменні імена, уважно стежити за сповіщеннями від реєстраторів та бути обізнаними з потенційними ризиками та способами їх уникнення.

1.5 Аналіз систем автоматизації для автоматичної реєстрації доменів

Автоматична реєстрація доменів, зокрема через сервіси бекордерингу, є важливим інструментом для зацікавлених осіб у придбанні цінних доменних імен, які вже зареєстровані, але можуть звільнитися. Цей процес вимагає від систем автоматизації високої точності, швидкості та надійності, оскільки конкуренція за популярні домени часто буває дуже високою [2].

Системи автоматичної реєстрації доменів моніторять статус обраних доменних імен і автоматично намагаються зареєструвати домен відразу ж, як тільки він стає доступним.

Робота таких систем включає в себе: моніторинг статусу домену (систематична перевірка статусу обраного домену для визначення моменту, коли він може звільнитися) та автоматичну реєстрацію (швидке подання запиту на реєстрацію домену, як тільки він стає доступним, часто в мілісекунди після звільнення).

Популярні сервіси для відстеження та реєстрації доменів:

- GoDaddy Domain Backorders: Як найбільший з реєстраторів доменів, GoDaddy пропонує послугу моніторингу, яка дозволяє користувачам «замовити» домени, що скоро звільняться. GoDaddy використовує свої ресурси для того, щоб максимізувати шанси на успішну реєстрацію.
- NameJet: Відомий своїми аукціонами за доменні імена, NameJet також пропонує послуги бекордерингу. Сервіс спеціалізується на забезпеченні доступу до цінних доменів і має партнерства з рядом реєстраторів для підвищення шансів на успіх.
- SnapNames: Цей сервіс пропонує одну з найбільш потужних систем бекордерингу, дозволяючи користувачам реєструвати інтерес до доменів, які скоро можуть звільнитися. SnapNames використовує розширені технології для моніторингу та автоматичної реєстрації доменів.

Кожна така системи моніторингу стикається з рядом викликів та обмежень:

- Конкуренція: Оскільки багато користувачів можуть намагатися зареєструвати один і той же домен, навіть найкращі системи моніторингу не можуть гарантувати успіх.
- Вартість: Послуги бекордерингу часто вимагають додаткової плати, яка не по-вертається у випадку невдачі.
- Етика: Деякі критикують практику бекордерингу як спосіб спекуляції на доменних іменах, що може перешкоджати використанню доменів для легітимних цілей.

Моделі інформаційних процесів та систем розглядалися в джерелах [42, 52-55]. На основі проведених досліджень з’ясовано що схема роботи з системою моніторингу та автоматичної реєстрації доменів виглядає наступним чином:

1. Користувач повідомляє яке доменне ім’я він хоче відстежувати а в результаті потім зареєструвати.

2. Система підраховує ціну домена яка включає собівартість реєстрації та комісію сервісу.

3. Користувач оплачує послуги сервісу та очікує на реєстрацію домену.

4. Після отримання оплати система починає відстежувати статус домена та намагатися зареєструвати його.

5. Якщо домен було успішно зареєстровано користувач отримує його у своє користування.

6. Якщо ж з якоїсь причини домен не було зареєстровано, користувач отримує повне відшкодування коштів.

Окрім вищезгаданих кроків можна додати різноманітні «покращення життя» для користувача системи.

1.6 Висновок до першого розділу

В першому розділі наукової роботи описано структуру системи доменних імен та проблеми пов’язані з життєвим циклом доменів. Також, розглянуто існуючі системи автоматизації, призначені для автоматичної реєстрації доменів, їх переваги та недоліки, а також ефективність їхнього застосування у різних сценаріях.

Системи автоматичної реєстрації доменів відіграють важливу роль у сфері управління доменними іменами, надаючи інструменти для моніторингу та купівлі цінних доменів. Втім, успіх використання цих систем залежить від багатьох факторів, включаючи швидкість дії, точність моніторингу та, звісно, деяку долю везіння.

2 АНАЛІЗ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ МОНІТОРИНГУ ТА РЕЄСТРАЦІЇ ДОМЕННИХ ІМЕН

2.1 Огляд мов програмування для розробки мережевих застосунків

Розробка мережевих застосунків вимагає від розробників глибокого розуміння мов програмування, які найкраще підходять для створення надійних, масштабованих та ефективних систем. Вибір мови програмування може значно вплинути на продуктивність застосунку, легкість розробки, а також можливості для майбутнього розширення. Нижче представлено огляд ключових мов програмування, які широко використовуються для розробки мережевих застосунків.

Python є однією з найпопулярніших мов програмування завдяки своїй гнучкості, легкості вивчення та великій кількості бібліотек [18]. Фреймворки, такі як Django та Flask, роблять Python ідеальним вибором для розробки веб-застосунків, API та автоматизації задач. Python також широко використовується для розробки систем моніторингу та аналітики завдяки своїм потужним аналітичним бібліотекам [8].

JavaScript, зокрема у середовищі Node.js, дозволяє розробникам створювати повноцінні мережеві застосунки, використовуючи одну мову як на клієнтській, так і на серверній стороні. Це сприяє швидкій розробці та легкому обміну кодом між частинами застосунку [41]. Node.js відомий своєю високою продуктивністю при роботі з асинхронними операціями та подієво-орієнтованими застосунками [6].

Java залишається однією з найбільш надійних та перевірених часом мов для розробки мережевих застосунків, особливо у корпоративному секторі. Java має зручний та продуманий API, великий вибір інструментів та різноманітні фреймворки [16]. В мережі є багато тематичних форумів, порталів та інших веб-ресурсів, де обмінюються знаннями користувачі Java. Java використовується для створення великих розподілених систем, веб-сервісів та мікросервісних архітектур завдяки широкому спектру доступних фреймворків, таких як Spring та Hibernate [5].

Go, або Golang, розроблена в Google, швидко набула популярності для розробки мережевих застосунків завдяки своїй простоті, високій продуктивності та вбудованій підтримці конкурентності [19]. Go особливо підходить для створення високопродуктивних мережевих серверів, мікросервісів та інструментів для DevOps [7].

C# від Microsoft широко використовується для розробки мережевих застосунків у екосистемі .NET. Ця мова пропонує потужні можливості для створення веб-застосунків, веб-API та мікросервісів з використанням ASP.NET [37]. C# підтримує об’єктно-орієнтоване, функціональне та асинхронне програмування, що робить її гнучкою для різноманітних сценаріїв розробки [28].

2.2 Вибір хостингової платформи

Вибір хостингової платформи для розробки автоматизованої системи супроводу процесів реєстрації та життєвого циклу доменних імен є критичним етапом, який визначає не тільки швидкість та ефективність розробки, але й масштабованість, безпеку та легкість подальшого обслуговування системи [4]. В цьому контексті, важливо розглянути як традиційні, так і інноваційні рішення, серед яких платформа Anvil.works виступає як обнадійливий кандидат.

2.2.1 Amazon Web Services (AWS)

Amazon Web Services або AWS є дочірньою компанією Amazon.com, що надає платформу хмарних обчислень в оренду приватним особам, компаніям та урядам на основі платної підписки. Існує і безкоштовна підписка, яка доступна протягом перших 12 місяців. Технологія дозволяє абонентам мати у своєму розпорядженні повноцінний віртуальний кластер комп’ютерів, який завжди доступний через Інтернет [46].

Віртуальні комп'ютери AWS оснащені більшістю характеристик реальних комп'ютерів, включаючи апаратні компоненти як процесор, відеокарта, оперативна та локальна пам'ять, а також жорсткий диск чи SSD. Користувачі можуть

вибирати операційну систему, налаштовувати мережеві з'єднання та використовувати вже встановлені додатки, такі як веб-сервери, бази даних, CRM і т.д [10]. Кожна система на AWS також віртуалізує консольний ввід/вивід, такий як клавіатура, дисплей і миша, дозволяючи користувачам AWS підключатися до своєї системи через веб-браузер. Це перетворює браузер на інтерфейс до віртуального комп'ютера, даючи можливість користувачам увійти в систему, налаштувати її та користуватися нею як звичайним комп'ютером, налаштовуючи її для надання інтернет-сервісів та послуг своїм клієнтам [44].

На рисунку 2.1 зображено панель управління сервісу CloudWatch від AWS.

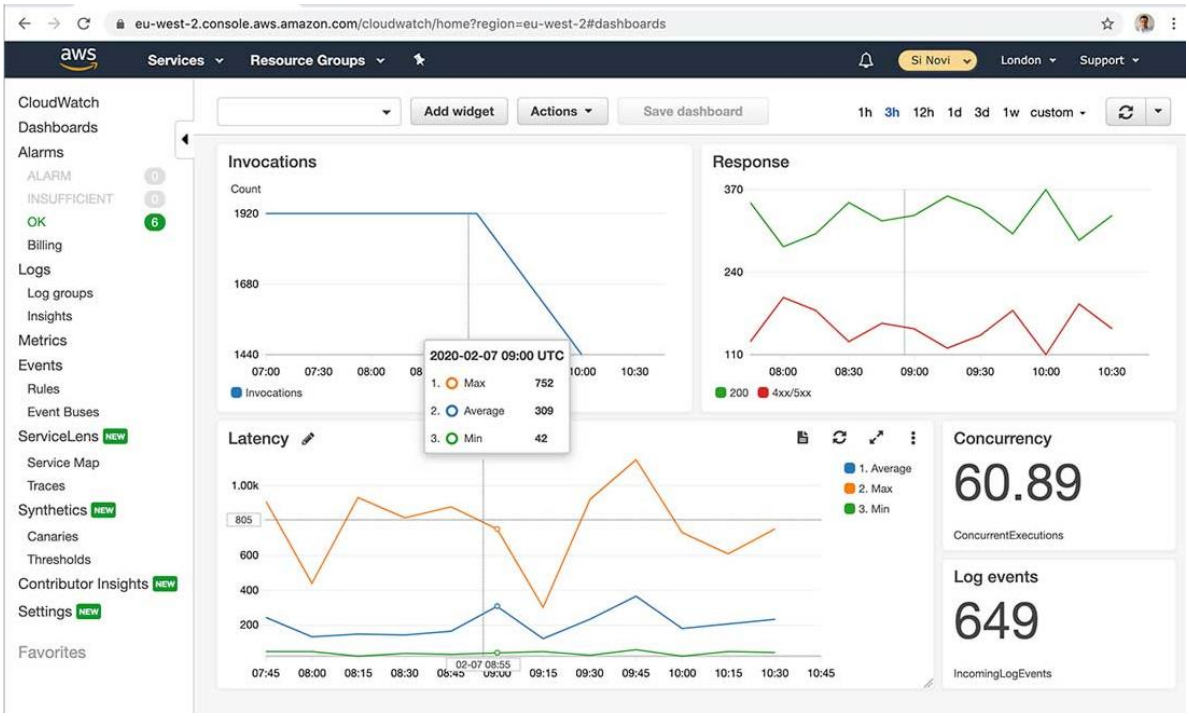

Рисунок 2.1 – Панель керування сервісом CloudWatch

Серед відомих обчислювальних сервісів від AWS варто виокремити наступні:

- Amazon Elastic Compute Cloud (EC2): Це IaaS-сервіс, який надає користувачам віртуальні сервери, керовані через API, побудовані на гіпервізорі Xen. Аналогічні хмарні сервіси включають Microsoft Azure, Google Compute Engine та Rackspace, а також рішення для локального розгортання, такі як OpenStack чи Eucalyptus.

- Amazon Elastic Beanstalk: Цей PaaS-сервіс дозволяє легко розміщувати та керувати веб-додатками. Схожі платформи включають Google App Engine, Heroku та OpenShift, які також підтримують локальне використання.
- Amazon Lambda (AWS Lambda): Платформа для безсерверних обчислень, яка автоматично запускає код у відповідь на події, такі як HTTP запити, забезпечуючи потрібні ресурси. Lambda тісно інтегрована з іншими сервісами AWS, проте аналогічні платформи, такі як Google Cloud Functions та OpenWhisk, також здобувають популярність [46].

Станом на березень 2024 року, AWS працює в 33 географічних регіонах, включаючи вісім у Північній Америці, один у Південній Америці, вісім в Європі, три на Близькому Сході, один в Африці та дванадцять у Азіатсько-Тихоокеанському регіоні [39].

Більшість регіонів AWS автоматично доступні для користувачів у їхніх облікових записах. Однак регіони, які були запущені після 20 березня 2019 року, вимагають від користувачів вручну активувати їх для використання у своєму обліковому записі. Для таких регіонів ресурси Identity and Access Management (IAM), включаючи користувачів та ролі, доступні лише в активованих регіонах [46].

Кожен регіон AWS знаходиться повністю в межах однієї країни, і усі дані та послуги, які він надає, обмежені цим регіоном. В кожному регіоні існує декілька "Зон доступності", кожна з яких містить один або кілька незалежних центрів обробки даних з власними резервними джерелами енергії, мережами та засобами зв'язку, які розміщені в окремих будівлях. Ці зони доступності спеціально ізольовані одна від одної, щоб запобігти поширенню збоїв між ними. Деякі сервіси, як-от S3 та DynamoDB, функціонують в окремих зонах доступності, тоді як інші можуть бути налаштовані на реплікацію між зонами для розподілу навантаження і зменшення ризику відмов [11].

Станом на грудень 2014 року, Amazon Web Services використовувала приблизно 1,4 мільйона серверів, розташованих у 11 регіонах і 28 зонах доступності. Глобальна мережа AWS Edge Locations включає понад 300 точок присутності по

всьому світу, включно з локаціями в Північній Америці, Європі, Азії, Австралії, Африці та Південній Америці [14].

Станом на березень 2024 року, AWS планує розширення, анонсувавши запуск шести нових регіонів у таких країнах як Малайзія, Мексика, Нова Зеландія, Таїланд, Саудівська Аравія та країнах Європейського Союзу. У березні 2023 року Amazon Web Services підписала угоду з урядом Нової Зеландії про відкриття великих центрів даних на території цієї країни, що стало значним кроком у розширенні інфраструктури компанії [21].

2.2.2 Google Cloud Platform

Google Cloud Platform, розроблена компанією Google, це комплекс хмарних служб, що працюють на тій же інфраструктурі, що використовується для споживчих продуктів компанії, як-от Google Search і YouTube. Платформа включає інструменти управління та набір модульних хмарних сервісів, зокрема обчислювальні потужності, зберігання даних, аналіз даних та машинне навчання. Для активації облікового запису на платформі необхідно вказати банківську картку чи рахунок [57].

Серед відомих продуктів Google Cloud Platform можна виокремити наступні:

- App Engine: Платформа для розгортання застосунків у хмарі.
- BigQuery: Масштабована аналітична система для великих обсягів даних, яка працює як інфраструктура як послуга.
- Bigtable: Масштабована NoSQL база даних, що пропонується як інфраструктура як послуга.
- Cloud AutoML: Набір інструментів для машинного навчання, який дозволяє розробникам з мінімальним досвідом у галузі машинного навчання створювати моделі нейронних мереж.
- Cloud Datastore: Документо-орієнтована база даних у хмарі.
- Cloud Functions: Функція як послуга, яка активується відповідно до подій у хмарі.

- Cloud Machine Learning Engine: Платформа для тренування та запуску машинно-навчальних моделей, особливо у TensorFlow.
- Cloud Pub/Sub: Сервіс для публікації та підписки на потоки даних, дозволяючи застосункам обмінюватися інформацією через систему публікації/підписки.
- Compute Engine: Надає віртуальні машини як інфраструктура як послуга.
- Kubernetes Engine: Служба для автоматизації розгортання, масштабування та управління контейнеризованими застосунками за допомогою Kubernetes.
- Google Genomics: Платформа для аналізу геномних даних у хмарі.
- Google Video Intelligence та Cloud Vision для аналізу зображень та відео.
- Storage: Надає доступ до зберігання даних через REST API [57].

Панель керування від Google Cloud Platform зображено на рисунку 2.2.

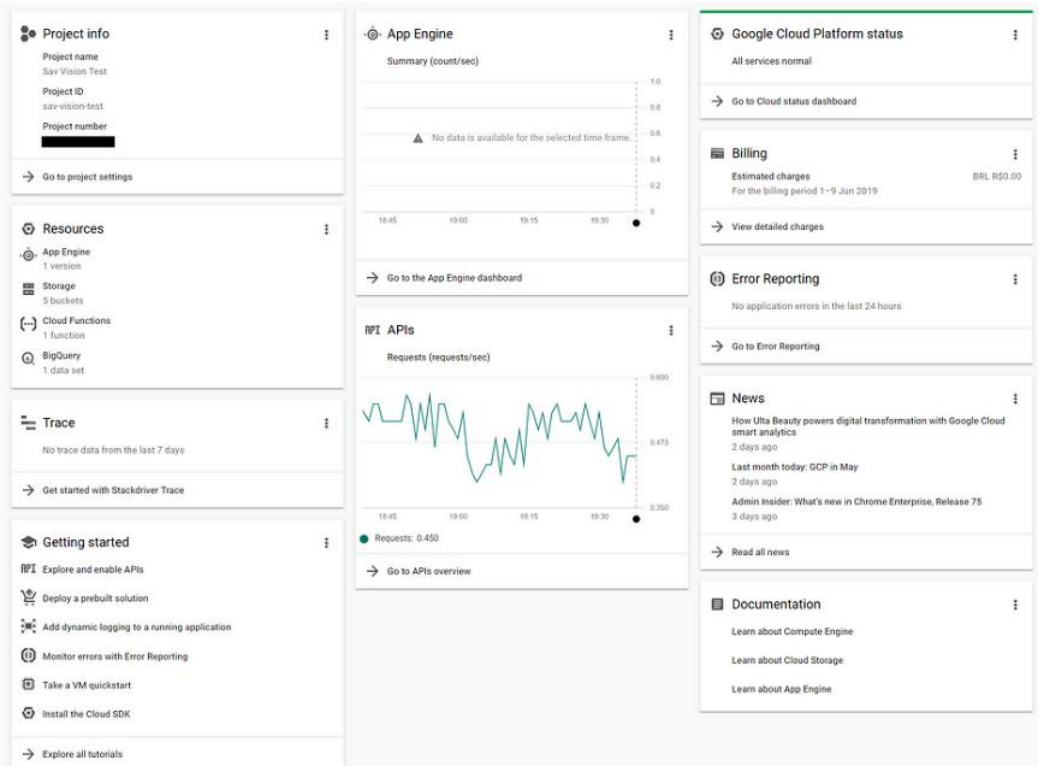

Рисунок 2.2 – Панель керування Google Cloud Platform

Google Cloud Platform також пропонує різні типи обчислювальних середовищ, включаючи інфраструктуру як послугу (IaaS), платформу як послугу (PaaS) та безсерверні обчислення [57].

2.2.3 Microsoft Azure

Microsoft Azure, відома також як Azure, представляє собою хмарну платформу і інфраструктуру від Microsoft, розроблену спеціально для фахівців у сфері розробки хмарних обчислень. Ця платформа націлена на полегшення процесу створення онлайн-додатків [26].

Azure складається з трьох основних технологічних груп, кожна з яких пропонує унікальний набір інструментів для розробників. Платформа не тільки сприяє розробці хмарних додатків, але й може використовуватися для програм, які функціонують локально на персональних комп’ютерах користувачів [59].

Платформа Windows Azure містить наступні ключові компоненти:

- Windows Azure: Цей компонент забезпечує середовище виконання для додатків, базоване на операційних системах та Windows Server, і надає простір для зберігання даних. Він функціонує на віртуальних машинах за допомогою технології, схожої на Hyper-V.
- Обчислення: Відповідає за виконання обчислювальних завдань, необхідних для розміщення додатків.
- Зберігання: Забезпечує зберігання даних в хмарному середовищі.
- SQL Azure: Пропонує сервіси реляційної бази даних, які можна використовувати у хмарному середовищі [59].

Додатково, платформа Windows Azure AppFabric надає додаткову функціональність, яка допомагає інтегрувати та управляти послугами у хмарі [59].

Крім технології .NET, розробка додатків для Azure може включати використання мов програмування, таких як Java, PHP, C/C++ або Python. Один із ключових інструментів у цьому процесі — це SDK емулятор хмари. Додатки, що працюють на емуляторі, мають доступ до бібліотек, що зареєстровані в місцевому глобальному кеші збірок (GAC), що дозволяє легко реєструвати та налаштовувати комп'ютери [59].

На рисунку 2.3 можна побачити панель керування Microsoft Azure.

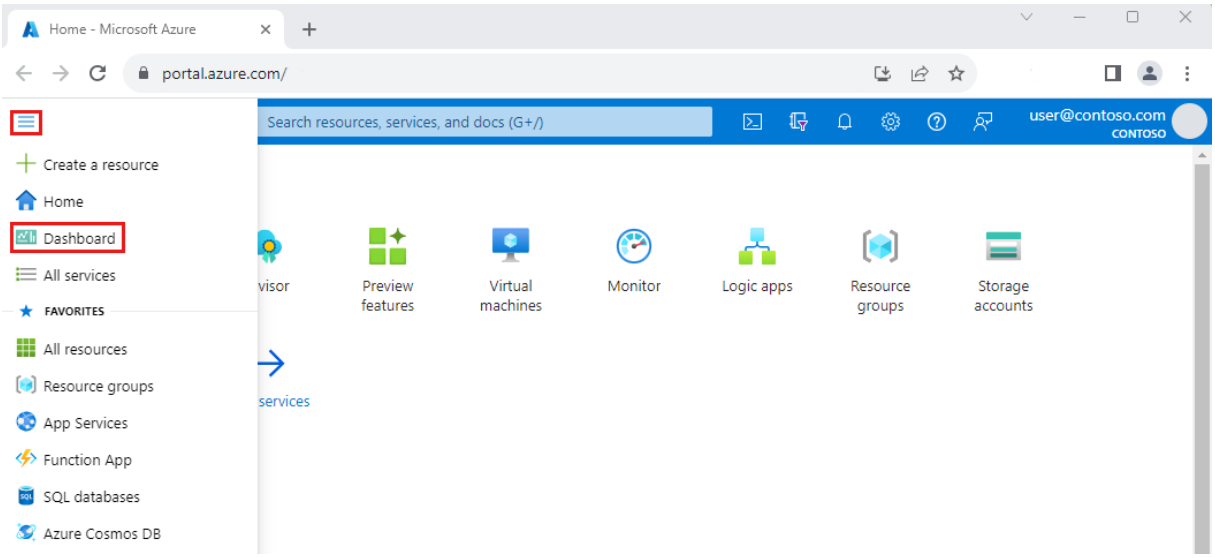

Рисунок 2.3 – Панель керування Microsoft Azure

Цей емулятор дозволяє зберігати діагностичну інформацію в консолі або використовувати Windows Azure Diagnostics. Вся ця інформація зберігається у спеціальній таблиці в Windows Azure Storage. Всі процеси, що запущені на емуляторі, мають права адміністратора, тоді як ті, що працюють безпосередньо на платформі, обмежені правами стандартного користувача Windows. Варто зазначити, що емулятор не повністю відтворює поведінку балансування навантаження, яке використовується на платформі Windows Azure [45].

2.2.4 Платформа Anvil.works

Anvil.works є інноваційною платформою, яка дозволяє з нуля розробляти веб-застосунки, використовуючи лише Python. Це рішення вирізняється своєю унікальною здатністю інтегрувати як фронтенд, так і бекенд розробку в єдиному середовищі, значно спрощуючи процес розробки [47].

Додаток Anvil складається з:

- Інтерфейсу користувача.
- Клієнтського коду Python, який виконується у веб-браузері.
- Серверного коду Python, який працює на серверах Anvil.
- Вбудованої бази даних (таблиці даних), в якій зберігаються ваші дані.

- Додаткового коду Python, запущеного на вашому комп'ютері/сервері, який також може взаємодіяти з вашою програмою, але цей компонент не є обов'язковим [47].

Використання Python як для серверної, так і для клієнтської частини знижує поріг входження та спрощує розробку і є вагомою перевагою платформи над іншими конкурентами. Також, Anvil пропонує візуальний редактор форм з «drag-and-drop» інтерфейсом для дизайну та легку інтеграцію з базами даних, API та сторонніми бібліотеками [47]. На рисунку 2.4 зображено середовище розробки Anvil.works.

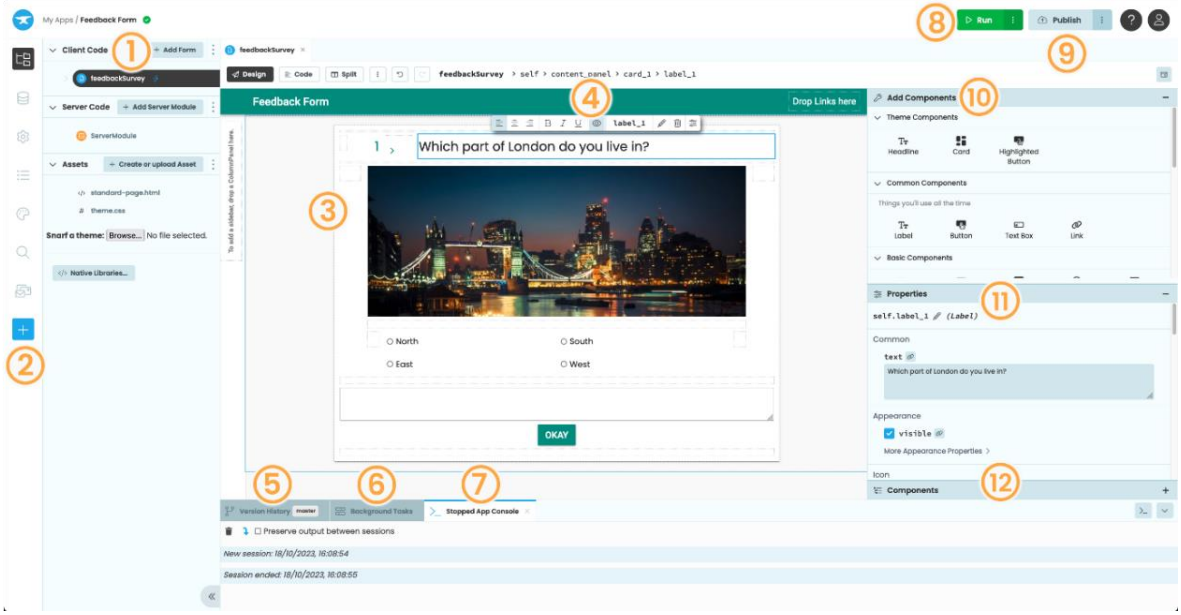

Рисунок 2.4 – Середовище розробки Anvil.works

Розглянемо детальніше компоненти середовища розробки зображені на рисунку 2.4:

1. Браузер програми знаходиться в лівій бічній панелі, і його можна знайти, натиснувши кнопку програми в меню бічної панелі. Він дозволяє вибрати, яку частину програми ви редагуєте.

2. Меню бічної панелі – це місце, де ви налаштовуєте програму, знаходите логи програми і додаєте попередньо створені бібліотеки та інтеграції, як-от таблиці даних, керування користувачами тощо.

3. Редактор форм відображає, як виглядатиме ваш інтерфейс користувача, і дозволяє перетягувати компоненти, щоб створити свій інтерфейс користувача.

4. Палітра об’єктів дозволяє швидко й легко редагувати властивості компонента, які найчастіше редагуються.

5. Панель «Історія версій» відображає історію змін вашої програми, дозволяє створювати нові гілки та клонувати вашу програму за допомогою Git.

6. Панель фонових завдань показує всі запущені фонові завдання.

7. Консоль програми – з’явиться після запуску програми. Вона показує вихідні дані програми а також помилки.

8. Кнопка «Запустити» запускає програму в редакторі та відображає консоль програми. Натиснувши меню з крапками, ви можете запустити програму в режимі розділеного перегляду або в новому вікні.

9. Кнопка «Опублікувати» дозволяє опублікувати вашу програму публічно або приватно. Тут ви можете додавати та налаштовувати середовища для своєї програми.

10. Панель інструментів – це місце, де ви вибираєте нові компоненти для додавання до свого інтерфейсу користувача.

11. Панель властивостей дозволяє налаштувати компонент, який ви вибрали в редакторі форм (ви можете вибрати компоненти, натиснувши на них).

12. Дерево компонентів показує ієрархію компонентів у поточній формі та дозволяє переміщувати компоненти у формі.

Окрім редактора форм де «будується» інтерфейс додатку, в Anvil є вікно написання коду, його зображено на рисунку 2.5.

Редактор коду дозволяє писати код додатку який виконуватиметься на клієнтській стороні (тобто в браузері). З точки зору написання коду процес надзвичайно простий та швидкий, кожен елемент інтерфейсу має готові функції які можна додати в код додатку натисканням мишки, також працює автодоповнення. Крім того, не потрібно перезапускати або компілювати додаток, щойно написаний рядок коду одразу ж працюватиме в додатку, щоправда доведеться оновити сторінку з додатком [47].

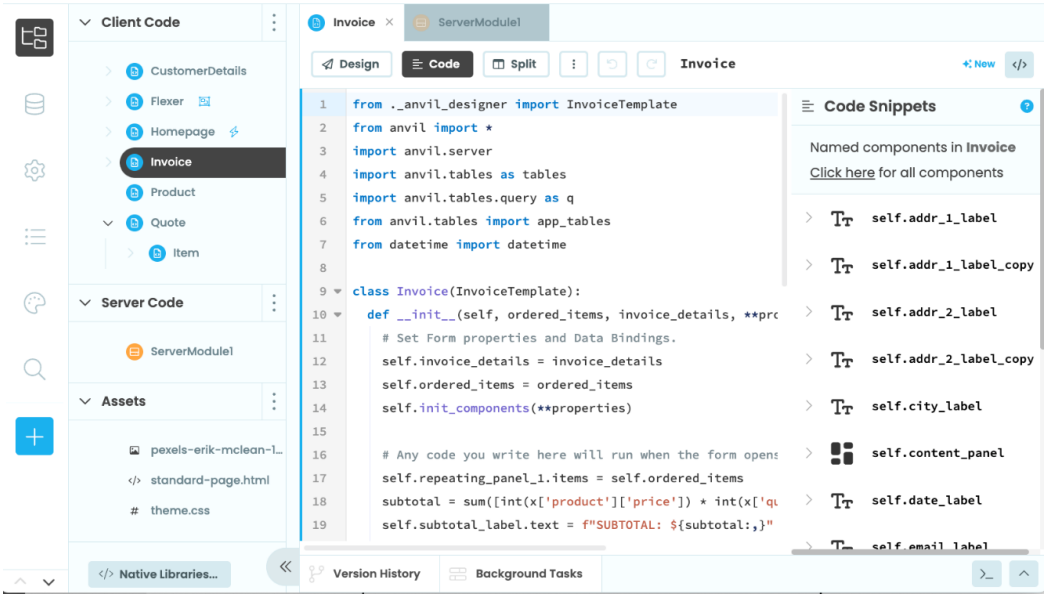

Рисунок 2.5 – Редактор коду Anvil.works

Окрім клієнтського коду Anvil також дає можливість запускати частину коду на серверній стороні. Функції, написані в серверному модулі, можна викликати з коду на стороні клієнта, надавши функції декоратор @anvil.server.callable, а потім викликавши anvil.server.call("<function-name>", <arg-name>) з коду форми [47].

Також, доволі корисним є те що Anvil має вбудовану базу даних де можна зберігати дані потрібні для роботи додатку а також записувати логи про роботу програми. Саму базу даних можна увімкнути та підключити до додатка за допомогою кількох кліків мишкою, а детальна документація допоможе з її інтеграцією в додаток. Через веб-інтерфейс (який показано на рисунку 2.6) можна переглядати та редагувати створені таблиці, рядки та поля.

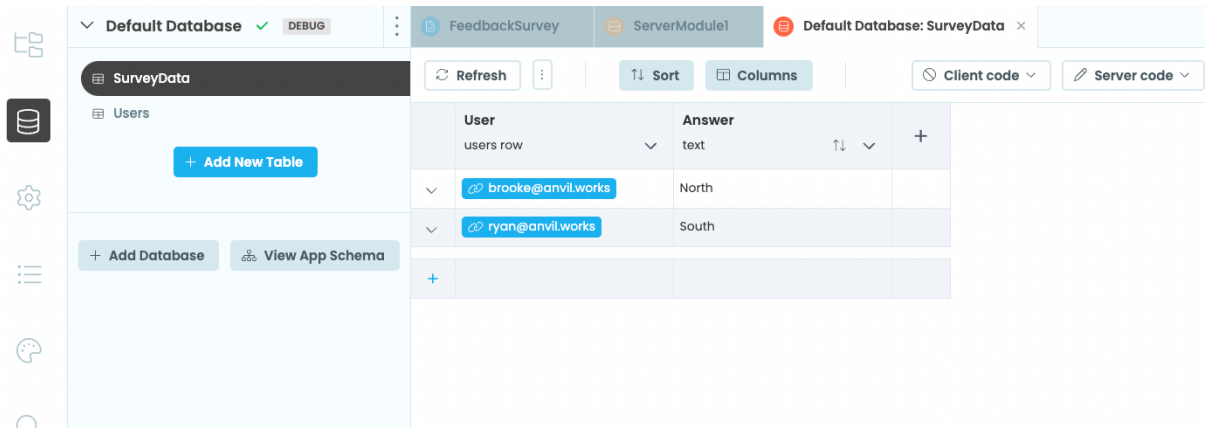

Рисунок 2.6 – Веб-інтерфейс керування базою даних в Anvil

Окрім вище перелічених функцій, Anvil має нативну підтримку:

- Менеджменту користувачів, який додає можливість створювати сесії окремим користувачам та відмежовувати дані.
- Інтеграції з Google, Facebook та Microsoft сервісами, що дозволить користувачам входити в додаток за допомогою акаунтів вищезгаданих компаній. Втім, розробник не отримує доступу до жодної інформації, а дані використані під час входу зберігаються як зашифрований хеш.
- Інтеграції з платіжним процесором Stripe, що дозволить виставляти рахунки користувачам та проводити інші платіжні операції.
- Файлового менеджера, що дозволяє користувачам завантажувати файли в додаток, а додатку, в свою чергу, обробляти файли.
- Надсилання листів електронною поштою.
- Uplink – це технологія Anvil яка дозволяє підключати веб-додаток на їх серверах до будь-якого іншого сервера/комп'ютера та виконувати частину коду віддалено [47].

Платформа Anvil.works максимально спрощує процес від ідеї до прототипу та подальшого розгортання застосунку, що робить її ідеальною для швидкої розробки та першого релізу [47].

2.3 Інструменти для моніторингу статусу доменів

Моніторинг статусу доменів є ключовим аспектом управління доменними іменами, особливо для систем, які займаються автоматизацією процесів реєстрації, відновлення та відстеження життєвого циклу доменів. Ефективний моніторинг дозволяє виявляти домени, що незабаром звільняться, та автоматично реєструвати їх. Для цього використовуються різноманітні інструменти та API, надані доменними реєстраторами. Останній варіант є більш привабливим оскільки окрім перевірки статусу домена його можна одразу зареєструвати в доменного реєстратора [27].

Розглянемо найбільш популярні сервіси для моніторингу статусів доменів:

- DomainTools. Цей сервіс пропонує розширені інструменти для моніторингу доменів, включаючи відстеження змін WHOIS-даних, повідомлення про закінчення терміну дії доменів та аналіз історії доменів. Оплата доступу до їхнього сервісу відбувається на основі кількості використаних API запитів [50]. Сайт сервісу DomainTools зображено на рисунку 2.7.

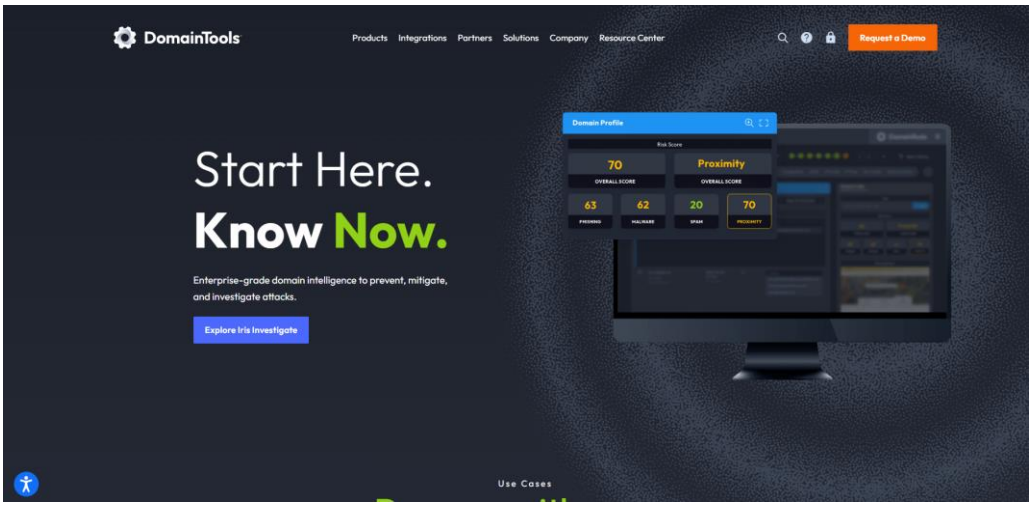

Рисунок 2.7 – Головна сторінка веб-сайту DomainTools

- Dynadot’s Domain Monitoring дозволяє користувачам встановлювати сповіщення для доменів, які їх цікавлять, надаючи інформацію про зміни статусу та доступність для реєстрації. Нажаль в цього сервісу досить високі ціни на реєстрацію доменів [51]. Головна сторінка сайту зображена на рисунку 2.8.

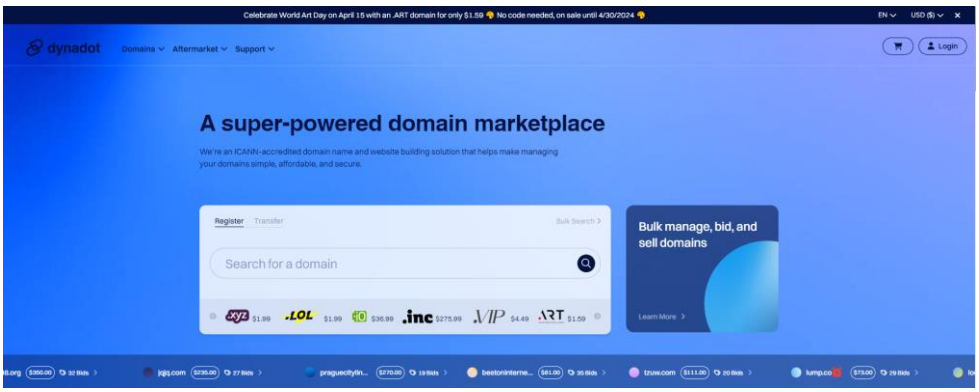

Рисунок 2.8 – Головна сторінка сайту Dynadot

- GoDaddy API. Надає широкий спектр функцій для управління доменами, включаючи пошук, реєстрацію та відновлення доменів. Це дозволяє

автоматизувати процеси покупки та відновлення доменів на основі моніторингу статусу. Однак собівартість домена на цьому сайті також значно вища за конкурентів [56]. Вигляд головної сторінки сайту GoDaddy можна побачити на рисунку 2.9.

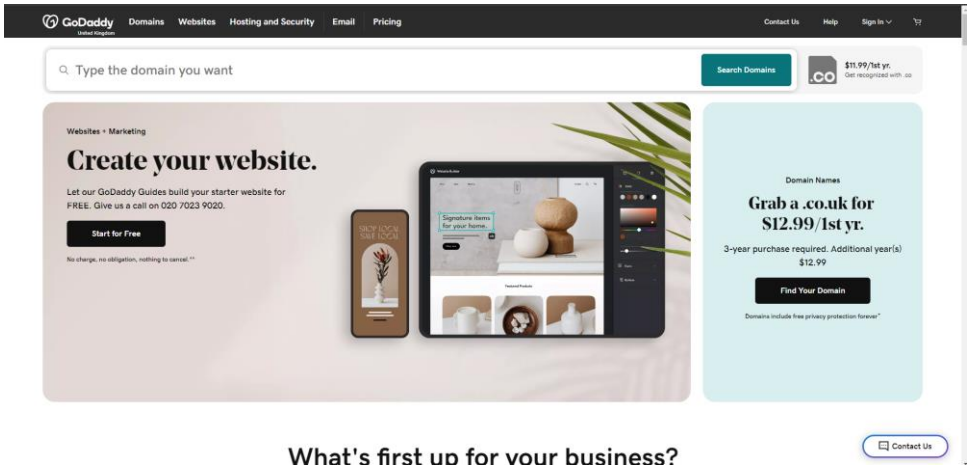

Рисунок 2.9 – Головна сторінка веб-сайту GoDaddy

- Namecheap пропонує API, яке дозволяє виконувати різноманітні операції, пов’язані з доменами, включаючи пошук, реєстрацію, та моніторинг статусу доменів. Використання Namecheap API для моніторингу дозволить системі автоматично визначати, коли домени стають доступними для бекордерингу, та вживати необхідних дій для їх реєстрації. Низькі ціни на реєстрацію доменів дозволять сформувати меншу ціну для кінцевого користувача системи моніторингу доменів [60]. Веб-сайт Namecheap.com зображено на рисунку 2.10.

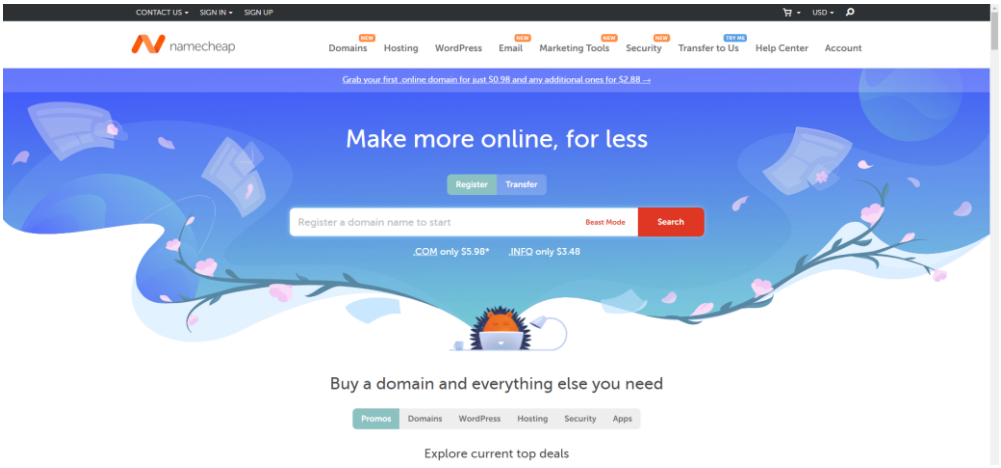

Рисунок 2.10 – Веб-сайт Namecheap.com

Вибір системи моніторингу є одним з ключових аспектів системи, оскільки він визначає можливості та обмеження майбутньої системи.

2.4 Порівняльний аналіз засобів для реалізації системи

Для створення інноваційної та конкурентоспроможної системи необхідно проаналізувати варіанти запропоновані для кожного компоненту системи. Проведемо порівняльний аналіз мов програмування, хостингових платформ та інструментів для моніторингу статусів доменів. Результат аналізу популярних мов програмування для мережевих застосунків подано у таблиці 2.1.

Таблиця 2.1 – Порівняльний аналіз мов програмування

Критерій	**Python**	**Java**	**JavaScript**	**Go**	**C#**
Простота вивчення	5	3	4	2	1
Бібліотеки та фреймворки	5	4	3	2	1
Гнучкість	5	3	4	2	1
Спільнота та підтримка	5	4	3	2	1
Виконання скриптів	5	2	4	3	1
Навчальні ресурси	5	4	3	2	1

Порівняння проводилось за наступними критеріями:

- Простота вивчення: Цей критерій оцінює, наскільки легко новачкам почати використовувати мову, враховуючи такі фактори як синтаксис, доступність ресурсів для навчання та інтуїтивність основних концепцій.
- Бібліотеки та фреймворки: Оцінює наявність та якість сторонніх бібліотек і фреймворків, які розширюють функціональність мови та полегшують розробку складних додатків.
- Гнучкість: Відноситься до можливості мови адаптуватися до різних стилів програмування та розробки, включаючи процедурну, об'єктно-орієнтовану та функціональну парадигми.

- Спільнота та підтримка: Оцінює наявність офіційної та неофіційної документації, туторіалів, книг та інших ресурсів для навчання.
- Виконання скриптів: Відноситься до здатності мови легко та швидко виконувати невеликі програми або скрипти, які використовуються для різноманітних задач, таких як автоматизація, обробка даних, тестування та інше.
- Навчальні ресурси: Цей критерій визначає наявність та якість освітніх матеріалів, які допомагають новачкам та досвідченим розробникам освоїти мову програмування.

Оцінювання проводилося використовуючи шкалу від 1 до 5, де 1 - найнижчий бал, а 5 – найвищий.

Таблиця 2.1 ілюструє, чому Python є кращим вибором для системи моніторингу та реєстрації доменних імен, особливо з огляду на його простоту вивчення та багатий набір інструментів.

У таблиці 2.2 подано результати порівняльного аналізу популярних платформ хостингу. Було використано шкалу оцінювання від 1 до 4, де 1 - найнижчий бал, а 4 – найвищий.

Таблиця 2.2 – Порівняльний аналіз хостингових платформ

Критерій	**Anvil.works**	**AWS**	**Google Cloud Platform**	**Microsoft Azure**
Легкість використання	4	2	3	1
Швидкість розгортання	4	2	3	1
Інтеграція веб-розробки	4	1	2	3
Підтримка новачків	4	2	3	1
Ціноутворення	4	1	2	3

Порівняння було проведено за наступними критеріями:

- Легкість використання: Інтуїтивно зрозумілий інтерфейс та прості у використанні інструменти, що знижують поріг входу для новачків та забезпечують швидке впровадження проєктів.

- Швидкість розгортання: Швидкість з якою можливо розгортати веб-додатки.
- Інтеграція веб-розробки: Зручність інтеграції фронтенду і бекенду в єдиній платформі.
- Підтримка новачків: Наявність документації, навчальних матеріалів а також форуму для користувачів.
- Ціноутворення: Тарифні плани які пропонує платформа.

Аналогічним чином було проведено порівняння систем для моніторингу статусів доменів:

- Ціни на популярні TLD: Оцінює доступність та конкурентоспроможність цін на популярні верхні рівні доменні імена.
- Можливість реєстрації доменів через API: Визначає, чи можуть користувачі автоматизувати реєстрацію доменів через програмний інтерфейс.
- Можливість менеджменту доменів: Оцінює інструменти та функції, що дозволяють управляти своїми доменами.
- Безпека платформи: Включає наявність заходів безпеки, таких як двофакторна автентифікація, захист від шахрайства, і надійність сервісу в цілому.

Таблиця 2.3 – Порівняльний аналіз інструментів для моніторингу статусу доменів

Критерій	**Namecheap**	**DomainTools**	**GoDaddy**	**Dynadot**
Ціни на популярні TLD	4	2	3	1
Можливість реєстрації доменів через API	4	-	3	2
Можливість менеджменту доменів через API	4	3	2	1
Безпека платформи	4	2	3	1

В результаті порівняльного аналізу було обрано наступні рішення для розробки системи: мова програмування – python; хостингова платформа – Anvil.works; система моніторингу статусів доменів – Namecheap API

2.5 Висновок до другого розділу

В другому розділі кваліфікаційної роботи досліджено популярні мови програмування для веб-додатків та технологічні платформи для розміщення цих додатків.

В ході дослідження мов програмування, для розробки системи було вирішено обрати мову програмування Python через простоту синтаксису, гнучкість, швидкість на різноманіття фреймворків.

Платформою для створення системи було обрано Anvil.works, через простоту та швидкість розробки, а також наявність безкоштовного тарифного плану який покриває весь необхідний функціонал та здешевить розробку.

Щодо сервісу моніторингу доменів, то тут вибір падає на Namecheap API. Встановлених сервісом API-лімітів буде достатньо для потреб розроблюваної системи, а низькі ціни на реєстрацію доменів дозволять сформувати конкурентоспроможну ціну для кінцевого користувача.

3 РОЗРОБКА ТА ТЕСТУВАННЯ СИСТЕМИ МОНІТОРИНГУ ДОМЕНІВ

3.1 Початкові налаштування середовища розробки Anvil

Для початку роботи з середовищем Anvil необхідно перейти на їхній веб-сайт за адресою https://anvil.works/ та створити безкоштовний акаунт, натиснувши на кнопку «Start building». Головна сторінка веб-сайту Anvil зображена на рисунку 3.1.

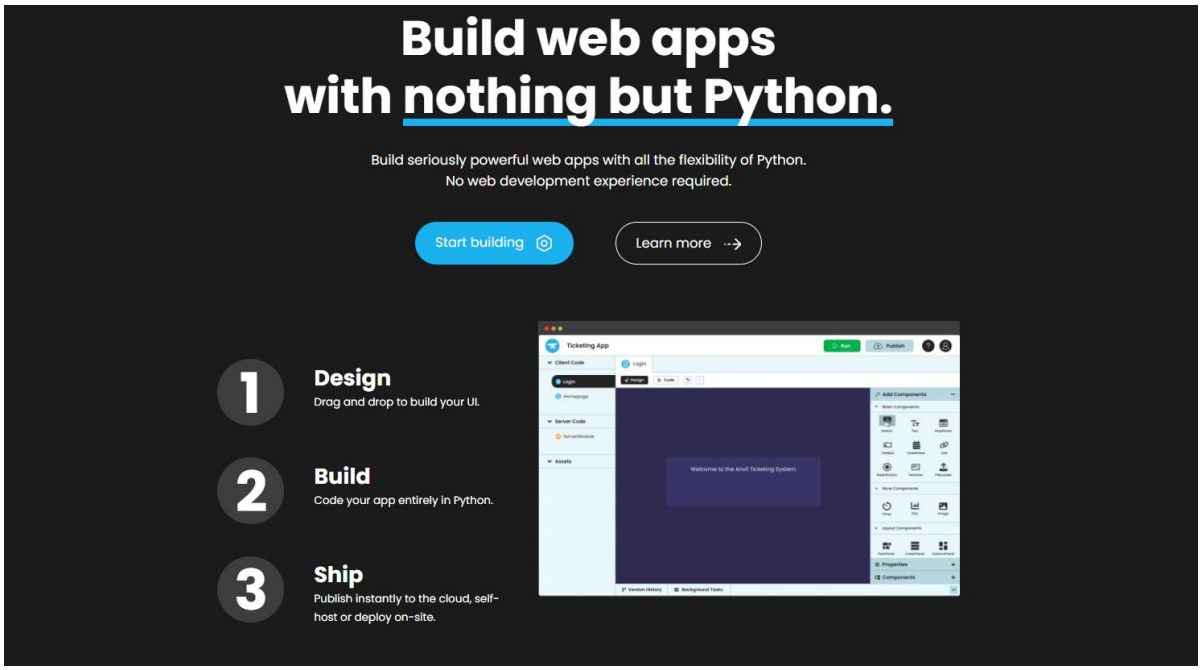

Рисунок 3.1 – Головна сторінка веб-сайту Anvil

Після створення акаунту на веб-сайті Anvil користувач перенаправляється на свою панель керування. В лівій частині екрану можна знайти кнопки для створення нового додатку (порожнього додатку або додатку з готових шаблонів), а також посилання на корисні ресурси, такі як: приклади додатків створених на Anvil, документація платформи а також навчальні статті створені розробниками платформи. В правій частині екрану користувач може знайти список своїх додатків та швидко перейти до редагування обраного додатку. Панель керування Anvil зображено на рисунку 3.2.

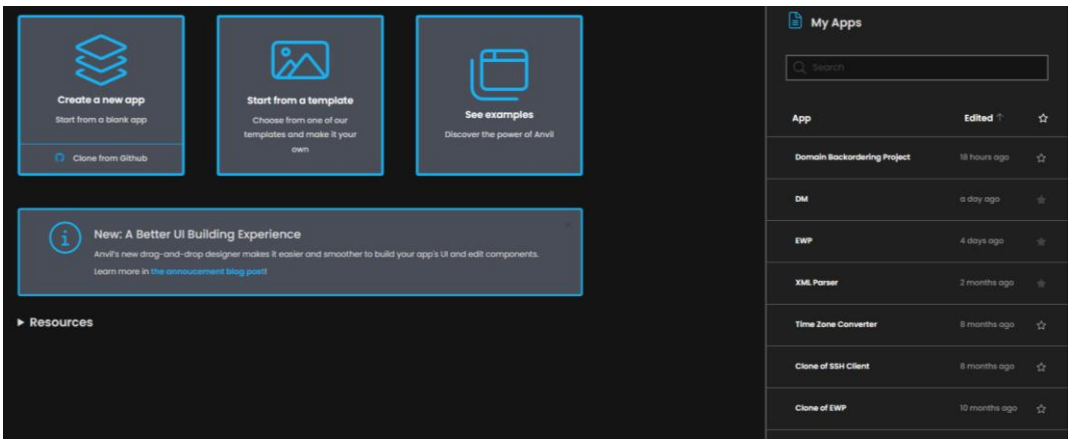

Рисунок 3.2 – Панель керування Anvil

Для початку роботи над новим проєктом необхідно обрати одну з опцій: створення з готового шаблону або порожній додаток. У випадку створення додатку за шаблоном у вашому акаунті створиться проєкт з готовою веб-формою, базою даних та базовим кодом, в залежності від обраного шаблону (список доступних шаблонів зображено на рисунку 3.3).

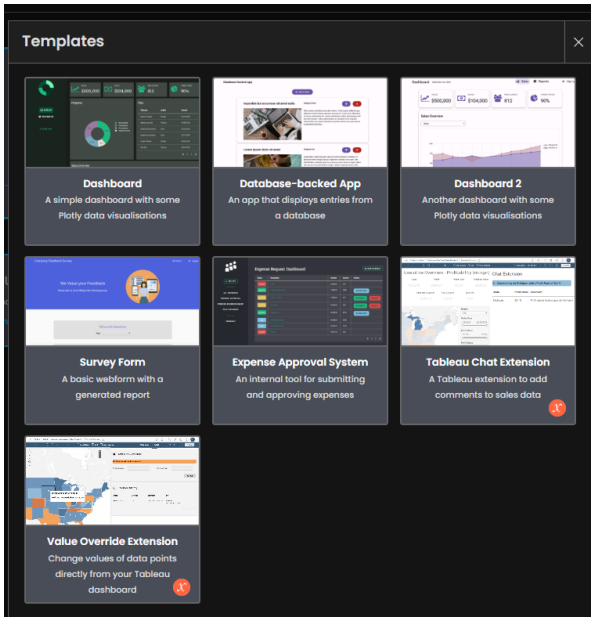

Рисунок 3.3 – Список доступних шаблонів для додатків Anvil

Якщо жоден з запропонованих шаблонів не підходить під потреби користувача то варто обрати порожній проєкт, в цьому випадку середовище лише запропонує обрати тему додатка (його загальний візуальний вигляд) і не

доведеться переробляти шаблон під власні потреби. Меню з вибором візуальних тем для додатку зображено на рисунку 3.4.

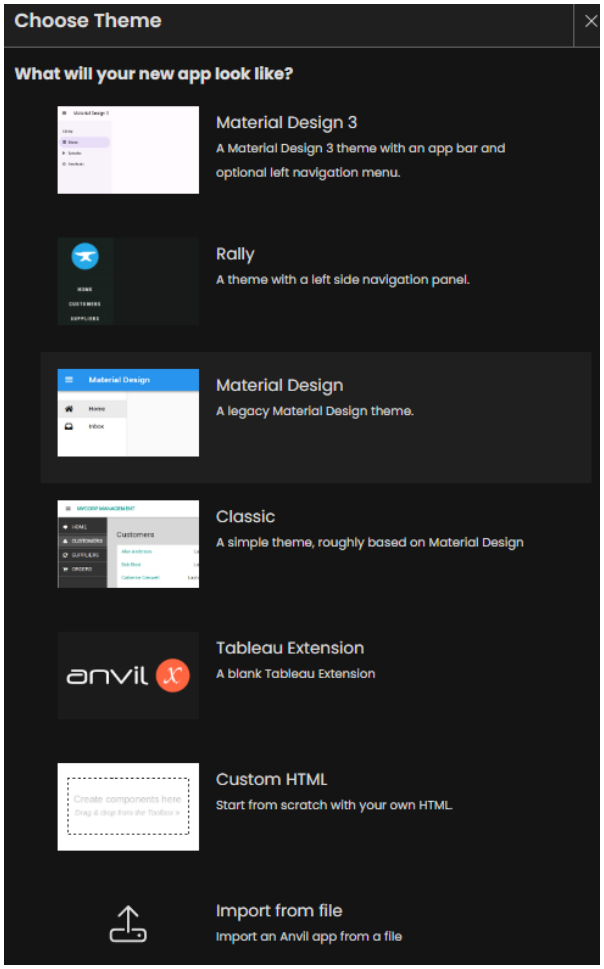

Рисунок 3.4 – Меню вибору тем для створення нового проєкту

Після вибору теми для нового додатку, користувач побачить середовище розробки Anvil (рисунок 3.5) і зможе приступити до роботи. Форма додатку (знаходиться по центру екрану) представляє собою віртуальну сітку на якій можна зручно розміщувати, групувати та змінювати компоненти. Для створення візуального оформлення додатку достатньо переносити компоненти з палітри компонентів на форму додатку простим перетягуванням миші. Праворуч від форми знаходиться палітра компонентів та меню редагування властивостей компонентів. Ліворуч можна знайти меню з налаштуваннями додатку, середовища та додатковими бібліотеками для додатку.

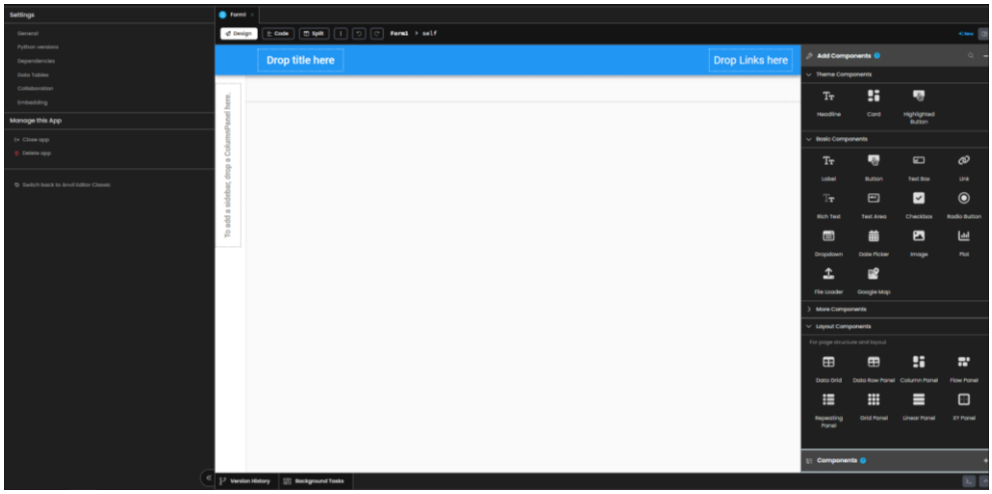

Рисунок 3.5 – Середовище розробки Anvil

Натиснувши на назву додатку у лівому верхньому кутку або на шестірню в лівій боковій панелі можна перейти в меню з основними налаштуваннями додатку. На рисунку 3.6 зображено меню з налаштуваннями додатку. Перш за все змінимо наступні параметри: 1) назва додатку – ця назва використовується для розрізнення додатків в своїй панелі керування; 2) заголовок додатку – назва яка відображатиметься в вікні браузера при відкриванні додатку; 3) опис додатку – опис який відображатиметься в пошукових системах; 4) іконка додатку – зображення яке відображатиметься в вікні браузера при відкриванні додатку.

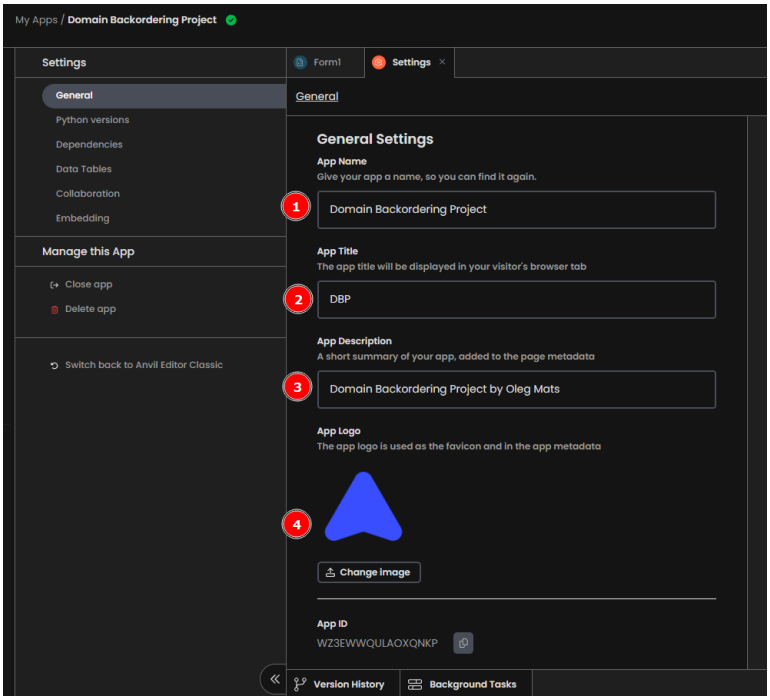

Рисунок 3.6 – Меню налаштувань додатку

При натисканні на іконку палітри в лівому боковому меню переходимо на вкладку де можна редагувати та додавати кольори для швидкого доступу (рисунок 3.7). Для кожного елемента інтерфейсу можна вказувати код кольору в шістнадцятковій системі, але завдяки цьому меню можна створити собі прості назви для конкретних кодів кольорів, які Anvil розпізнаватиме так само як і шістнадцяткові коди. Саме завдяки таким особливостям розробляти додатки на Anvil дуже зручно та швидко. Обираємо бажану кольорову гаму для нашого додатка.

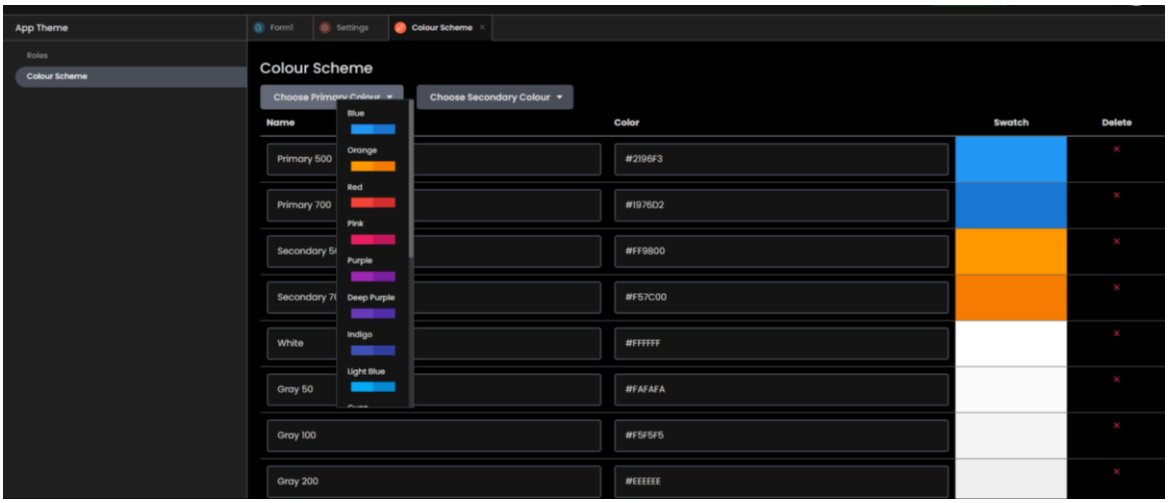

Рисунок 3.7 – Меню з палітрою кольорів для швидкого доступу

Щоб додаток отримав унікальну URL адресу та його можна було повноцінно використовувати з будь-якого пристрою, додаток необхідно опублікувати. Для цього слід використати кнопку «Publish» в верхній правій частині екрану. У вікні що відкриється (рисунок 3.8) можна буде вибрати безкоштовну публікацію (додаток отримає унікальний піддомен кореневого домену anvil.app) або з’єднати Anvil з власним доменом (але ця опція вимагає купівлі одного з платних тарифних планів Anvil).

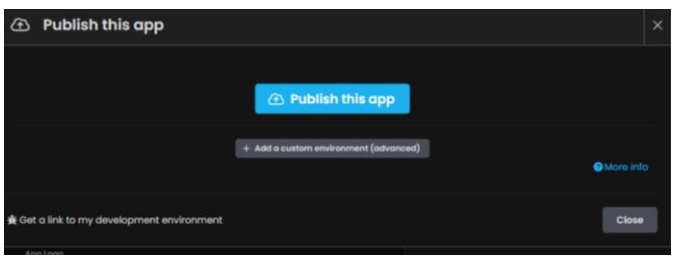

Рисунок 3.8 – Вікно з вибором опції публікації додатку

Після вибору безкоштовного публікування потрібно вказати бажану URL адресу додатку (піддомен) (рисунок 3.9), обрати потрібні опції публікації (такі як база даних, функція Uplink, тощо). Фінальне вікно налаштувань публікації показане на рисунку 3.10.

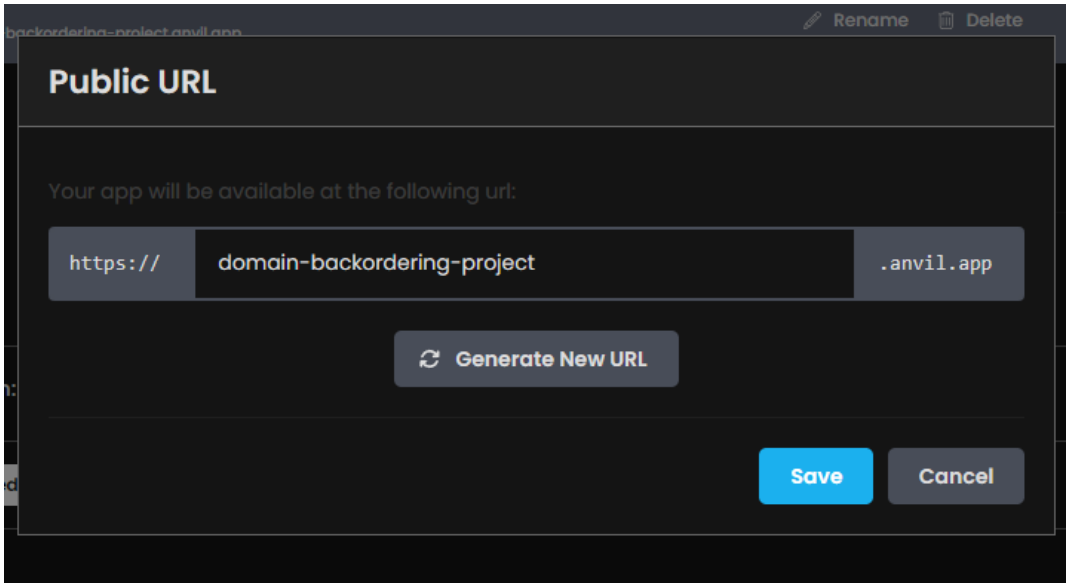

Рисунок 3.9 – Вікно вибору URL адреси додатку

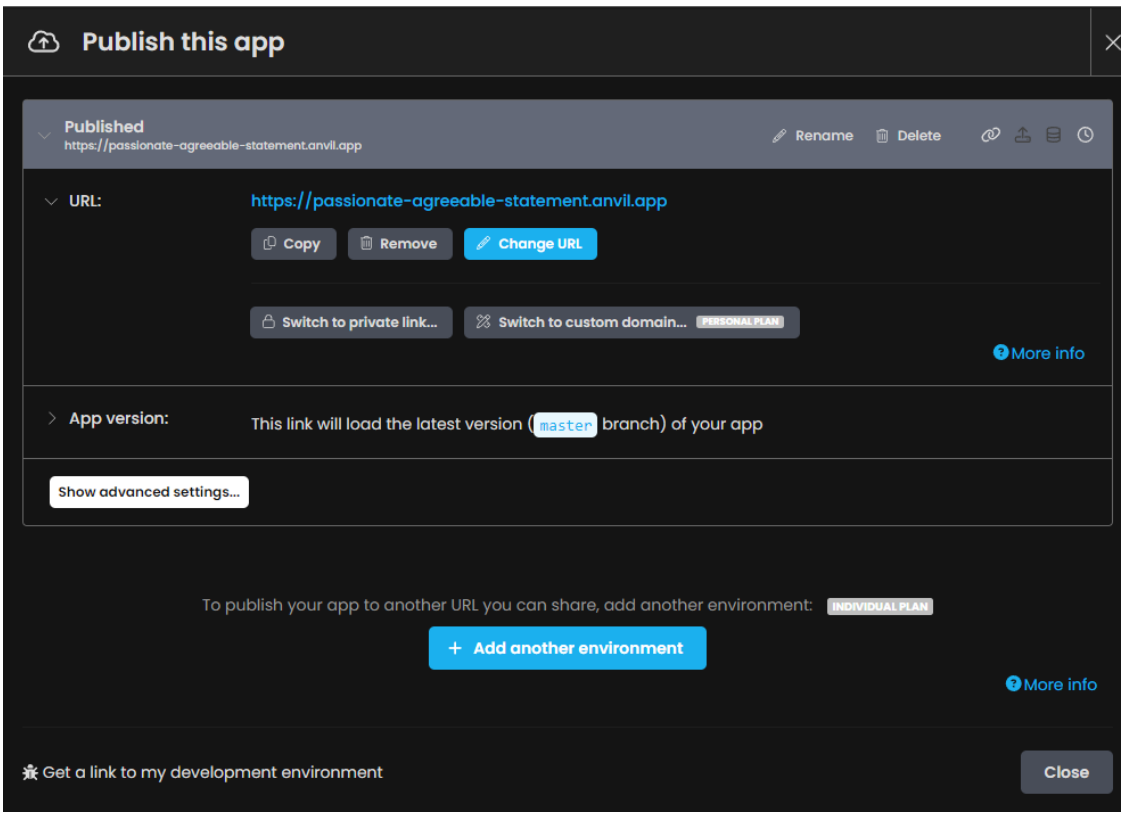

Рисунок 3.10 – Фінальне вікно налаштувань публікації додатку

Після того як додаток було опубліковано його можна буде відкрити у будь-якому браузері перейшовши за URL адресою вказаною на попередньому кроці. Якщо на форму додатку не додавалися жодні елементи то порожня форма виглядатиме так як зображено на рисунку 3.11.

Рисунок 3.11 – Порожня форма опублікованого додатку

Продумавши взаємодію користувача з додатком, розміщуємо всі необхідні елементи інтерфейсу (написи, списки, кнопки та поля введення) на формі додатку. Для цього просто перетягуємо елементи з палітри компонентів на форму, розміщуємо їх та змінюємо розміри елементів так як нам потрібно. Для реалізації інтерфейсу користувача знадобилося: 5 написів, 2 кнопки, 1 лемент відображення даних, 1 текстове поле введення та 1 текстова область введення. Вигляд елементів розміщених на формі подано на рисунку 3.12.

Одним з основних елементів додатку буде поле введення тексту куди користувач зможе вводити домени, статус яких необхідно моніторити. Для зрозумілості було зроблено напис-підказку над полем введення, а також задано текст для порожнього поля введення за допомогою властивості «placeholder». Властивість «placeholder» та її вигляд на полі введення показано на рисунку 3.13.

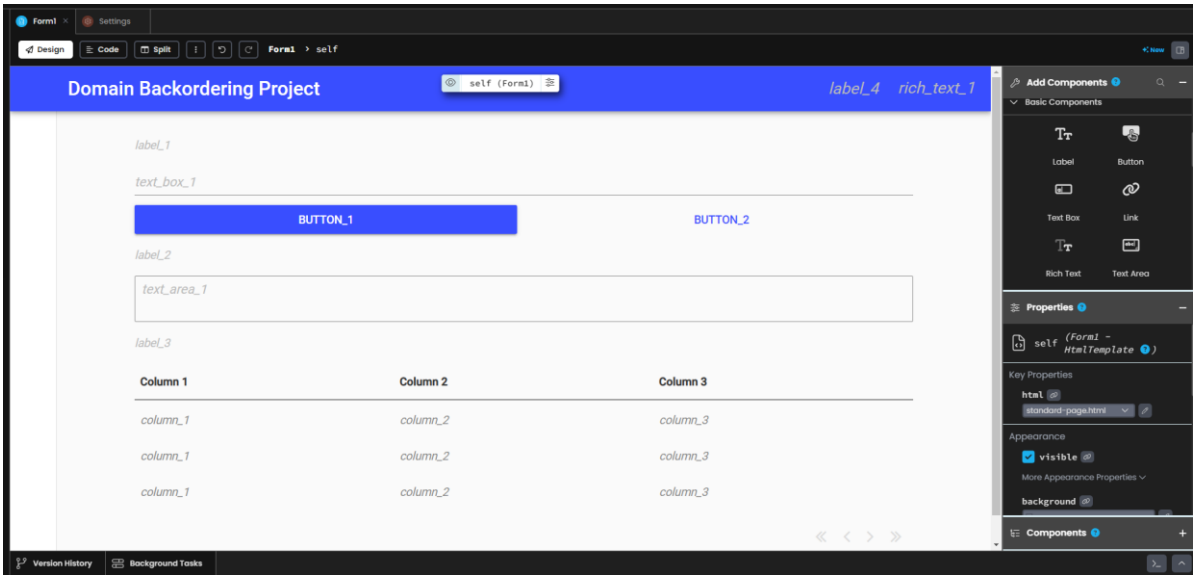

Рисунок 3.12 – Форма додатку з усіма розміщеними елементами

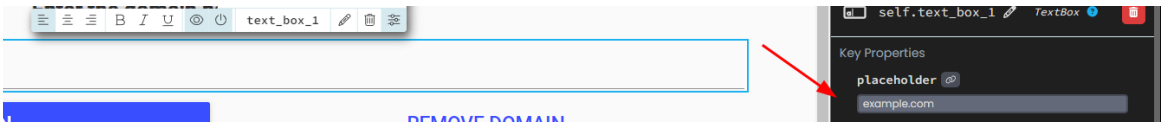

Рисунок 3.13 – Властивість «placeholder» поля введення

Також було задано текст для написів на кнопках, за допомогою однойменної властивості. Властивість «text» показано на рисунку 3.14.

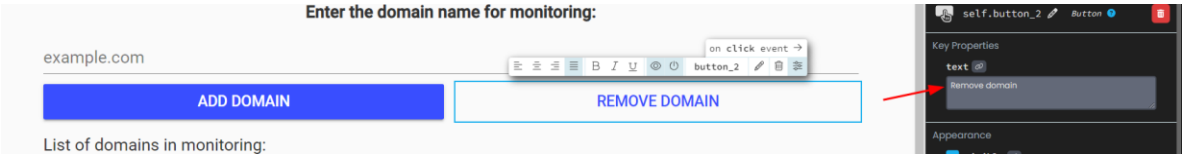

Рисунок 3.14 – Властивість кнопки «text»

Область введення було використано як елемент де відображатимуться всі доменні імена які користувач додав і статуси яких перевіряються системою. Оскільки для додавання доменів використовуватиметься поле введення, взаємодія користувача з областю введення не є бажаною, тож її було відключено за допомогою властивості «enabled». Також було додано текст (властивістю «placeholder») який відображатиметься коли область буде порожньою (в системі не буде доменів для моніторингу). Властивості області введення зображено на рисунку 3.15.

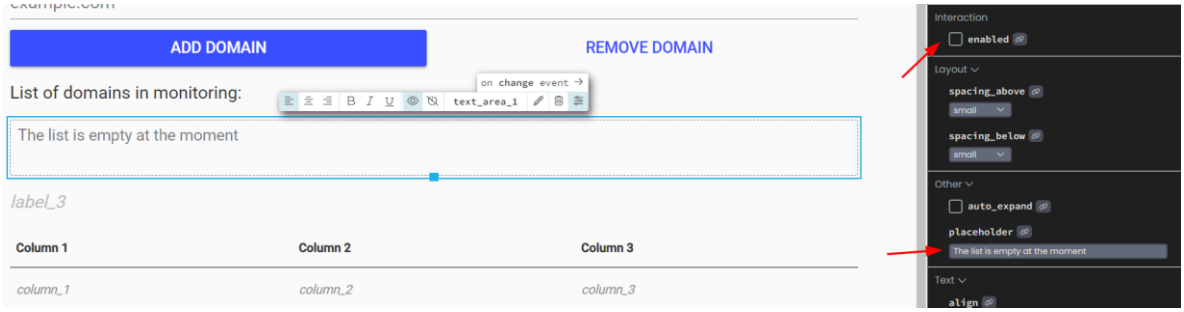

Рисунок 3.15 – Область введення з властивостями «enabled» та «placeholder»

Фінальний вигляд інтерфейсу користувача, після того як всі елементи розміщено на формі та властивості елементів були вказані, показано на рисунку 3.16.

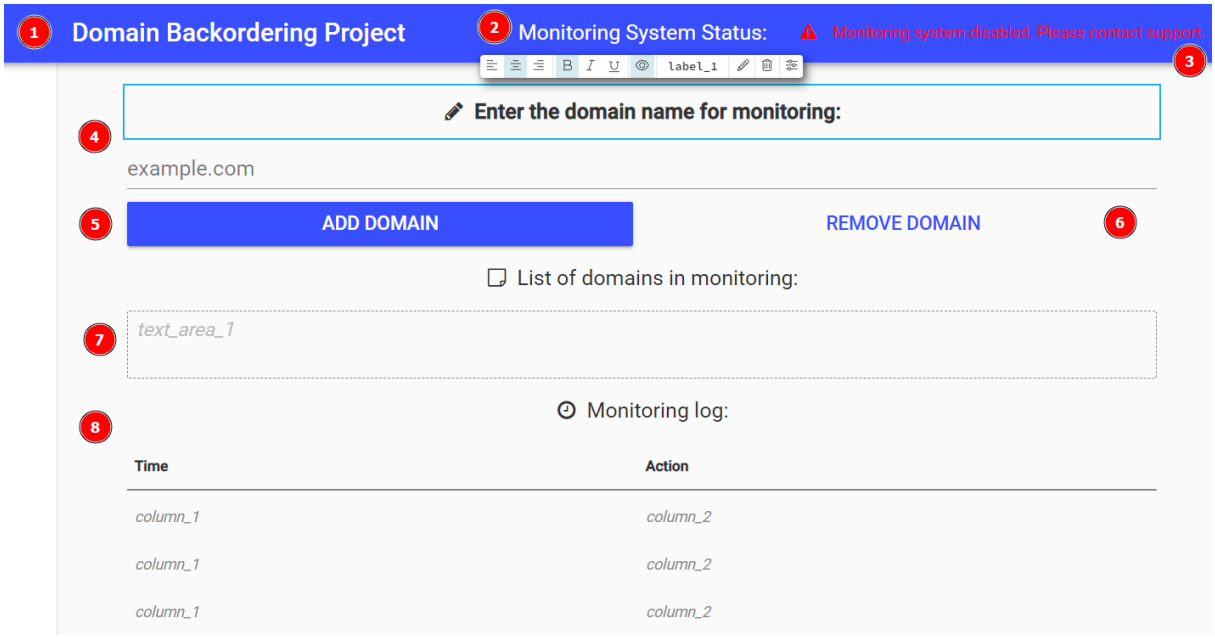

Рисунок 3.16 – Фінальний вигляд інтерфейсу користувача

Розглянемо детальніше елементи створеного інтерфейсу користувача:

1. назва додатку;

2. статус системи моніторингу;

3. напис який може динамічно змінюватися, і, власне, вказує на сам статус системи моніторингу;

4. поле введення доменів для моніторингу;

5. кнопка додавання домену до списку моніторингу;

6. кнопка видалення домену зі списку моніторингу;

7. область де відображатимуться всі домени які зараз перевіряються системою;

8. табличка з інформацією про останні дії в системі (додавання/видалення доменів а також коли домени ставали доступними/реєструвались).

Варто зауважити що напис зі статусом системи моніторингу може використовуватись в кількох випадках:

- Основним застосуванням цього елементу є налагодження роботи додатку в процесі розробки. Цей простий напис є надзвичайно корисним для перевірки статусу серверної частини системи. При відкриванні клієнтської частини системи, додаток відправлятиме коротке повідомлення на сервер, у відповідь на яке він повинен отримати відповідь. Якщо відповідь отримано то статус відобразить відповідне повідомлення зображене на рисунку 3.17. В протилежному випадку додаток покаже повідомлення про те що з серверною частиною є проблеми.
- Також цей елемент інтерфейсу можна використовувати для комунікації з користувачами, у випадку технічних робіт в системі або при проблемах з оплатою послуг, коли моніторинг доменів користувача призупиняється.

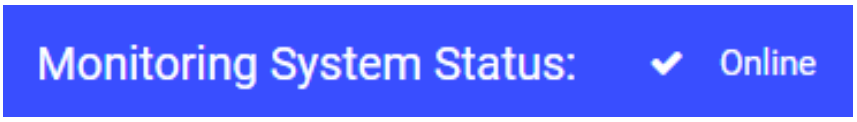

Рисунок 3.17 – Статус при належній роботі системи моніторингу

Створений користувацький інтерфейс є мінімалістичним та інтуїтивно зрозумілим, водночас його функцій повністю вистачить для повноцінного користування системою моніторингу.

3.2 Початок роботи з NamecheapAPI

API Namecheap дозволяє створювати веб-програми та програми для настільних ПК, які інтегруються з вашим обліковим записом Namecheap. Це дає змогу

програмно виконувати такі операції, як пошук домену, реєстрація домену, покупка SSL, тощо [60].

У Namecheap є робоче та тестове серверні середовища. Середовище тестового сервера називається пісочницею. Namecheap рекомендує зареєструвати акаунт в пісочниці та тестувати свої додатки у ній, що є абсолютно безкоштовно. Функції пісочниці є ідентичними до робочого середовища, за винятком того що усі покупки та операції є віртуальними. Для інтеграції в робоче середовище після пісочниці достатньо буде лише змінити URL-адресу, ім’я користувача API та ключ API [60].

Для того щоб почати використовувати Namecheap API потрібно увійти в свій обліковий запис на сайті namecheap.com, перейти у вкладку «Профіль» а далі в «Інструменти». В секції «Бізнес та інструменти розробників» потрібно натиснути на кнопку «Управління» що знаходиться навпроти опції «Namecheap API Access». Вкладка «Інструменти» зображена на рисунку 3.18.

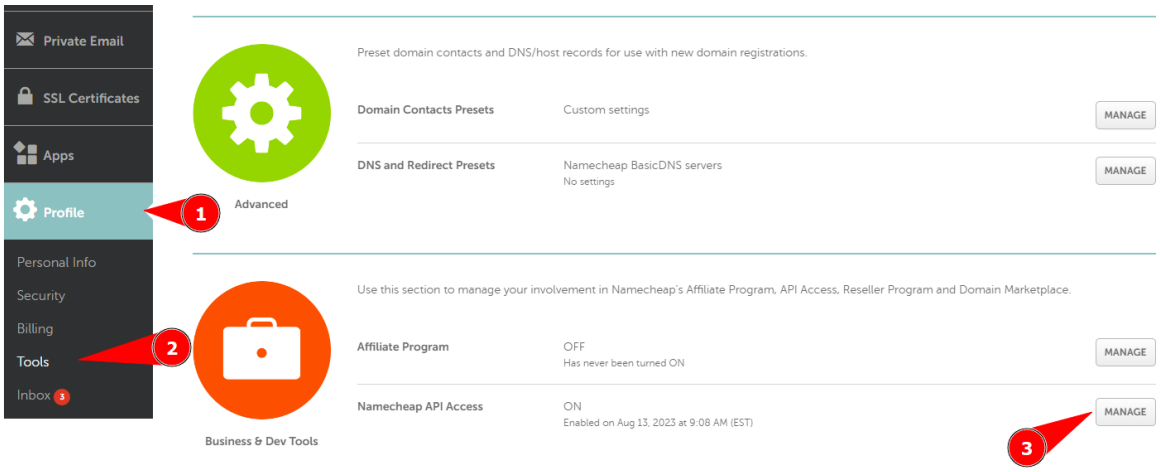

Рисунок 3.18 – Вкладка «Інструменти» в акаунті Namecheap

Панель управління API зображено на рисунку 3.19.

На сторінці «Управління» API необхідно увімкнути цю функцію для свого акаунту та скопіювати згенерований API ключ що з’явиться після цього. Також необхідно додати принаймні одну IP-адресу в список виключень, для того щоб мати змогу працювати з API. IP-адреса повинна належати комп’ютеру або серверу з якого надсилатимуться API запити, всі запити що не належатимуть цій IP адресі відхилятимуться серверами Namecheap [60].

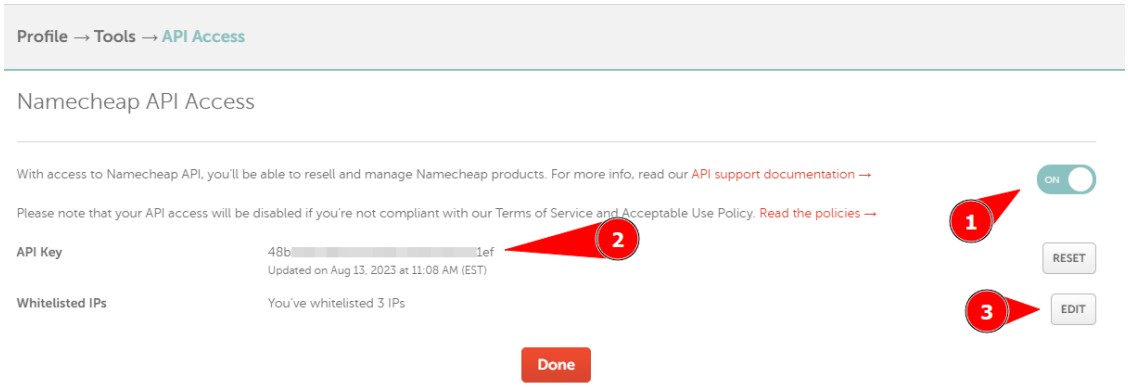

Рисунок 3.19 – Панель управління API

Перевірити роботу сервісу можна за допомогою наступної команди перевірки доступності домена вставленої в адресний рядок браузера:

```
https://api.sandbox.namecheap.com/xml.response?ApiUser=ncuser
&ApiKey=apikey&UserName=ncuser&ClientIp=121.22.123.22&Command=name
cheap.domains.check&DomainList=domain1.com
```

Кожна команда повинна складатися з наступних частин:

- https://api.sandbox.namecheap.com/xml.response? – адреса API сервісу.
- ApiUser – ім’я користувача API.
- ApiKey – ключ користувача API.
- UserName – ім’я акаунту з якого здійснюватиметься команда (зазвичай воно повинно бути таким самим як і ApiUser).
- ClientIp – IP адреса машини з якої надіслано запит.
- Command – специфічна команда для API запиту. Усі доступні команди можна переглянути на відповідній сторінці довідника Namecheap: https://www.namecheap.com/support/api/methods/.

У відповідь на API запит користувач отримає відповідну інформацію в залежності від викликаної команди. Це можуть бути певні дані, повідомлення про успіх або код помилки.

Команда «domains.check» буде основною в системі моніторингу доменів, оскільки вона викликатиметься найчастіше і перевірятиме доступність домену.

Результат виконання команди на перевірку доступності домену подану у лістингу 3.1.

Лістинг 3.1 – Результат виконання команди перевірки доступності для зареєстрованого домену

```
<ApiResponse xmlns="http://api.namecheap.com/xml.response"
Status="OK">
     <Errors/>
     <Warnings/>
     <RequestedCommand>namecheap.domains.check</RequestedCommand>
     <CommandResponse Type="namecheap.domains.check">
     <DomainCheckResult Domain="domain1.com" Available="false"
ErrorNo="0" Description="" IsPremiumName="false"
PremiumRegistrationPrice="0" PremiumRenewalPrice="0"
PremiumRestorePrice="0" PremiumTransferPrice="0" IcannFee="0"
EapFee="0.0"/>
     </CommandResponse>
     <Server>PHX01APIEXT01</Server>
     <GMTTimeDifference>--4:00</GMTTimeDifference>
     <ExecutionTime>0.326</ExecutionTime>
     </ApiResponse>
```

В лістингу 3.2 подано результат виконання для домену який є доступним для реєстрації.

Лістинг 3.2 – Результат виконання команди перевірки доступності для доступного домену

```
<ApiResponse xmlns="http://api.namecheap.com/xml.response"
Status="OK">
     <Errors/>
     <Warnings/>
     <RequestedCommand>namecheap.domains.check</RequestedCommand>
     <CommandResponse Type="namecheap.domains.check">
     <DomainCheckResult Domain="domain1488.com" Available="true"
ErrorNo="0" Description="" IsPremiumName="false"
PremiumRegistrationPrice="0" PremiumRenewalPrice="0"
PremiumRestorePrice="0" PremiumTransferPrice="0" IcannFee="0"
EapFee="0.0"/>
     </CommandResponse>
     <Server>PHX01APIEXT03</Server>
     <GMTTimeDifference>--4:00</GMTTimeDifference>
     <ExecutionTime>0.332</ExecutionTime>
     </ApiResponse>
```

Проаналізувавши дві відповіді API сервера, можна зробити висновок що при перевірці доступності домену варто опиратись на поле «Available». Якщо значення цього поля дорівнюватиме true то це означає що домен доступний, якщо false – домен не доступний для реєстрації.

Наведених вище дій достатньо для того щоб перейти до наступного етапу розробки з використанням Namecheap API.

3.3 Написання серверної частини додатку та тестування системи

Останнім етапом розробки є написання серверної частини додатку та тестування створеної системи. Оскільки безкоштовний тарифний план Anvil не підтримує серверні модулі, а принцип роботи системи вимагає того щоб серверна частина постійно була запущеною та мала доступ до мережі, для роботи системи необхідно буде використовувати веб-хостинг або будь-який комп'ютер з доступом до мережі інтернет. З метою забезпечення автономності та неперервної роботи системи було прийнято рішення розмістити серверну частину додатку на хостинговому сервері.

Щоб підготувати сервер для роботи додатку, необхідно увійти в cPanel акаунт хостингу, та з головної сторінки перейти в розділ для роботи з python додатками (головна сторінка cPanel з виділеним меню для роботи з python додатками зображена на рисунку 3.20).

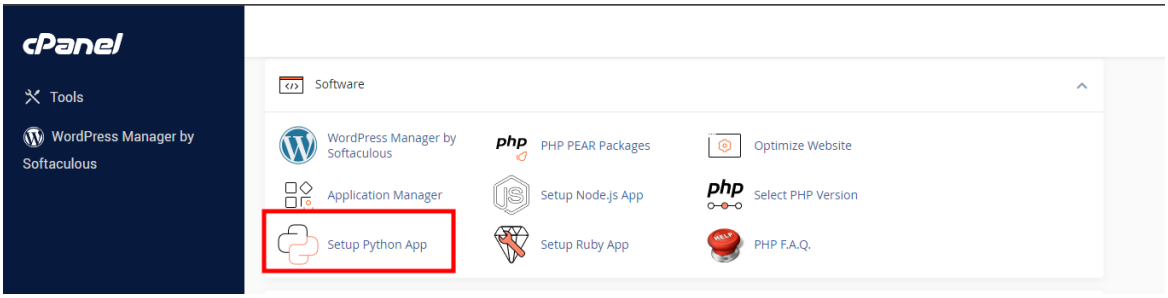

Рисунок 3.20 – Головна сторінка cPanel

В вищезгаданому меню необхідно створити новий додаток, для цього потрібно натиснути на відповідну кнопку «Створити додаток» (рисунок 3.21),

обрати місце розташування файлів додатку заповнивши відповідні поля cPanel та натиснути кнопку «Створити» (рисунок 3.22).

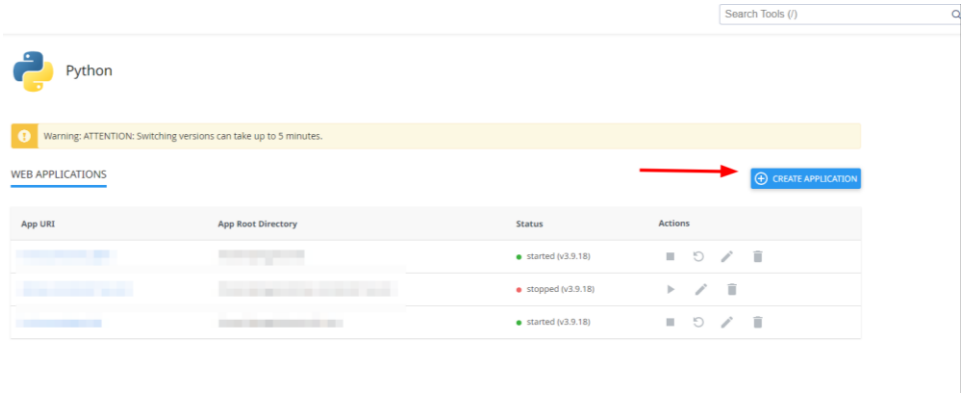

Рисунок 3.21 – Меню для роботи з python додатками

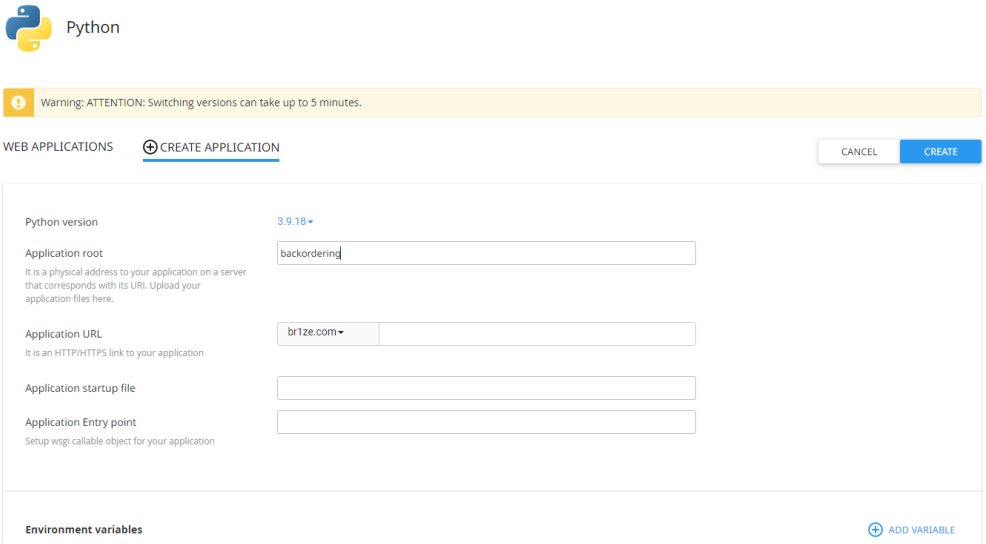

Рисунок 3.22 – Меню створення нового додатка python

Після того як новий додаток успішно створено, над полями введення з’явиться команда доступу до віртуального середовища, яку слід зберегти (команду зображено на рисунку 3.23). Ця команда дозволить перейти в директорію додатку через вбудований в cPanel термінал. Термінал cPanel зображено на рисунку 3.24.

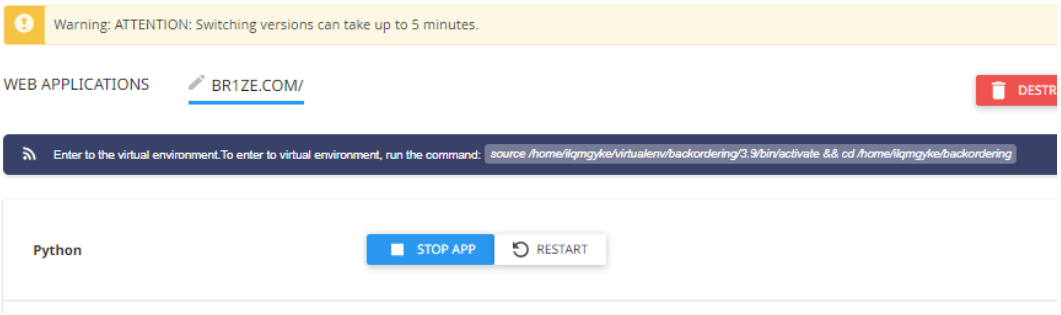

Рисунок 3.23 – Команда для доступу до віртуального середовища

Рисунок 3.24 – Термінал cPanel

За допомогою системи керування пакунками python (PIP) встановлюємо необхідні модулі, а саме: requests та anvil-uplink. Модулі встановлюються за допомогою наступних команд введених в термінал:

```
pip install requests
pip install anvil-uplink
```

Після цього необхідно створити файл де зберігатиметься виконуваний код нашого додатку, у відповідній директорії в cPanel. Базовий код для серверної частини додатку наведено в лістингу 3.2.

Лістинг 3.3 – Базовий код для серверної частини додатку Anvil

```
import anvil.server
anvil.server.connect("<your Server Uplink key>")
@anvil.server.callable
def get_data(name):
  print(f"Hello from your own machine, {name}!")
  return  [1, 2, 4, 8]
anvil.server.wait_forever()
```

Коли роботу базового коду було налагоджено, можна приступити безпосередньо до написання коду системи моніторингу. Система моніторингу

складається з трьох основних функцій: перевірка статусу доменів (лістинг 3.4), реєстрація домену та основна функція моніторингу, що являє собою постійний цикл.

Лістинг 3.4 – Функція перевірки статусу доменів

```
def check_domain_availability(domains):
	Domain_list = ",".join(domains)
	payload = {'ApiUser': ApiUser, 'ApiKey': ApiKey, 'UserName'
: UserName, 'Command' : CommandCheck, 'ClientIp' : ClientIp,
'DomainList' : Domain_list}
	AvailRequest = requests.get(URL, params=payload)
	logger.debug (f"{get_curtime()}: API Response =
\n{AvailRequest.text}")
	parsed_results = parse_check_xml(AvailRequest.text)
	for domain, status in parsed_results:
		logger.debug (f"Domain: {domain}, Status: {status}")
	return (parsed_results)
```

Окрім цих функцій було реалізовано систему сповіщень про вдалу реєстрацію домену, а також отримання неочікуваної помилки від API сервісу. Сповіщення надсилаються на задану електронну пошту. Також, додано систему логування, всі повідомлення що генеруються в результаті роботи серверного коду логуються в окремі файли (що зберігаються на сервері в директорії додатку) розміром до 5МБ. Як тільки розмір файлу стає більшим за заданий ліміт, додаток створює новий файл. Щоб не перевантажувати хостинговий сервер, кількість файлів з логами на сервері обмежена до 10. При потребі в додаткових файлах, додаток видаляє найстаріший з них і записує інформацію в новий. Повний код серверної частини системи подано в додатку В.

Відповідно до серверної частини, були написані функції для елементів інтерфейсу клієнтської частини додатку. Код клієнтської частини системи подано в додатку Д.

Клієнтський додаток доступний за посиланням: https://domain-backordering-project.anvil.app/

Взаємодія з додатком відбувається наступним чином:

1. Користувач відкриває сторінку додатку (рисунок 3.25), і перевіряє статус роботи системи моніторингу.

2. Користувач вводить доменне ім’я яке потрібно «зловити».

3. Користувач натискає на кнопку «Додати домен». Якщо домен не відповідає правилам синтаксису, то користувач отримує відповідну помилку.

4. Після цього домен додається в базу даних додатку, і його статус перевіряється серверною частиною з певним інтервалом. Якщо статус домена вказує на те що домен недоступний для реєстрації, він залишається в базі даних і моніторинг продовжується.

5. Як тільки домен стає доступним для реєстрації, система автоматично реєструє його та надсилає повідомлення про успішну реєстрацію на електронну пошту (домен видаляється з бази даних). В разі непередбачуваних проблем з реєстрацією, система надсилає повідомлення про помилку (домен залишається в базі даних, і система намагатиметься реєструвати його з певними інтервалами).

6. Якщо користувач передумав реєструвати доменне ім’я, то його можна видалити зі списку моніторингу ввівши в поле для введення та натиснувши на відповідну кнопку.

Всі дії в додатку (додавання, видалення та реєстрація домену) записуються в окрему таблицю бази даних і відображаються внизу сторінки в секції «Monitoring log» (рисунок 3.26).

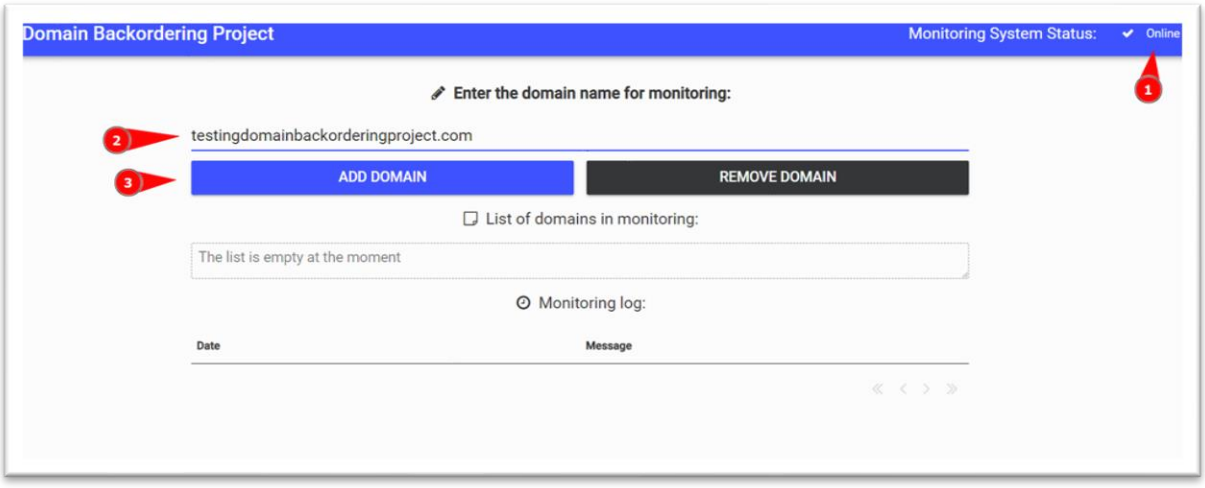

Рисунок 3.25 – Головна сторінка створеного додатку

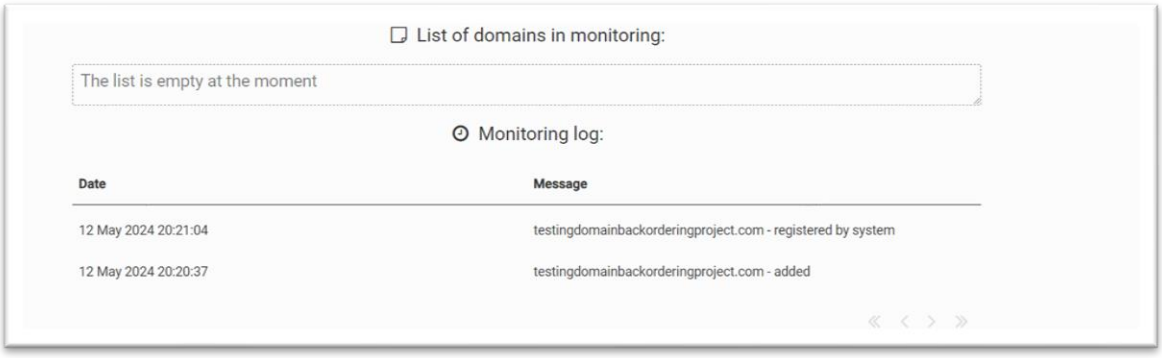

Рисунок 3.26 – Секція з історією дій в додатку

Для тестування додатку доступне доменне ім’я було введено в відповідне поле веб-форми. Додаток спрацював очікувано, та через кілька секунд зареєстрував доменне ім’я, видалив його з бази даних та надіслав відповідне повідомлення на електронну пошту. Приклад повідомлення на електронній пошті зображено на рисунку 3.27. Доменне ім’я в акаунті Namecheap зображено на рисунку 3.28.

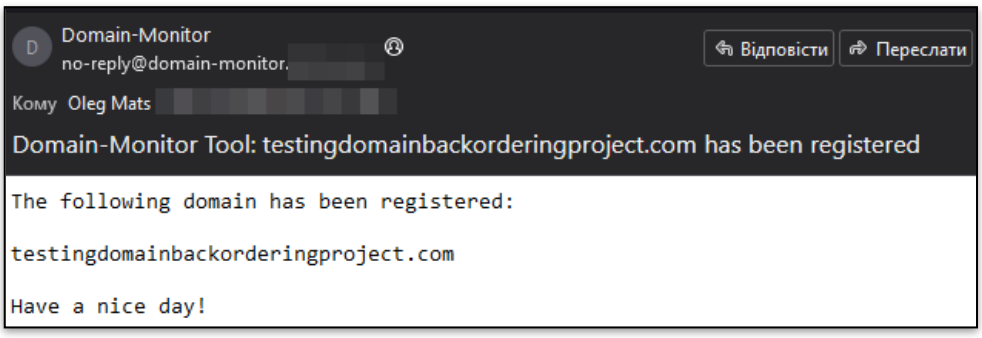

Рисунок 3.27 – Електронне повідомлення про успішну реєстрацію

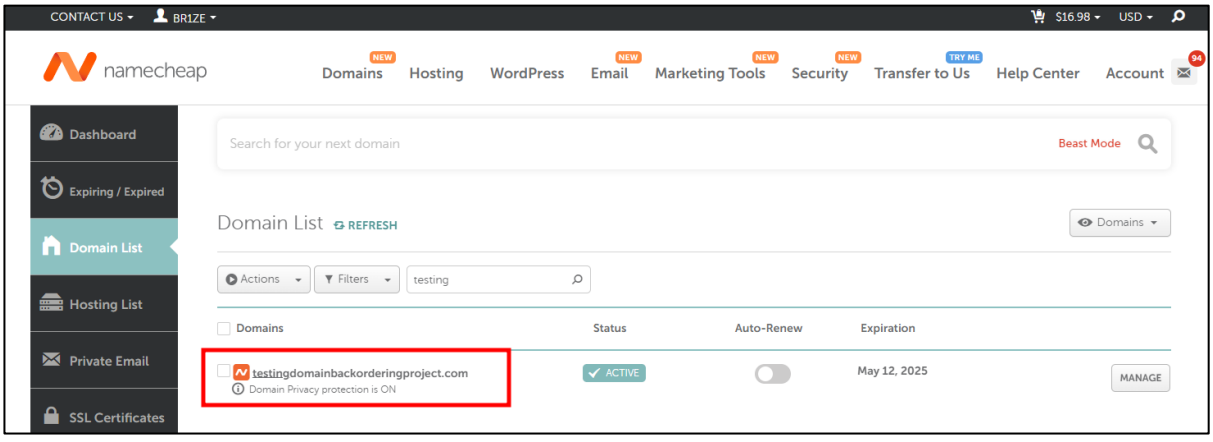

Рисунок 3.28 – Зареєстроване доменне ім’я в акаунті Namecheap

Також було протестовано негативний сценарій. Одна з функцій серверної частини була змінена, таким чином що додаток реєстрував доменне ім’я яке не доступне для реєстрації і, відповідно, отримував помилку від API сервісу. Доменне ім’я не було зареєстровано, воно залишилось у базі даних і система намагалася й далі реєструвати його.

3.4 Висновок до третього розділу

Розроблена система моніторингу та реєстрації доменних імен представляє собою високоефективний інструмент, який забезпечує користувачам перевагу в реєстрації високовартісних доменів в інтернеті.

У таблиці 3.1 наведено результати порівняльного аналізу з відомими системами моніторингу доменів: NameJet, SnapNames та DropCatch. Було використано шкалу оцінювання від 1 до 4, де 1 - найнижчий бал, а 4 – найвищий.

Таблиця 3.1 – Порівняльний аналіз запропонованого рішення та відомих систем моніторингу доменних імен

Критерій	Створена система	NameJet	SnapNames	DropCatch
Ціна реєстрації .com домена	4 ($10.50)	2($59+)	1($100+)	3($40+)
Зручність інтерфейсу	4	2	3	1
Функціонал	2	4	3	1

Порівняння проводилося за наступними критеріями:

- Ціна за реєстрацію .com: Цей показник відображає вартість реєстрації одного домену з популярним розширенням .com через кожний з сервісів. Відповідно до отриманих даних можна зробити висновок й про інші розширення.
- Зручність інтерфейсу: Оцінюється наскільки інтуїтивно зрозумілий і легкий у використанні інтерфейс кожного сервісу.
- Функціонал: Цей пункт порівнює різноманітність та глибину функцій, які кожен сервіс надає для моніторингу та перехоплення доменів.

Використовуючи API Namecheap, система гарантує отримання найнижчих цін, що робить її привабливою для кінцевих користувачів, які прагнуть оптимізувати свої витрати. Для прикладу, в схожих сервісах ціна моніторингу та реєстрації домену в зоні .com обійдеться більше ніж у $40. В запропонованому рішенні, собівартість реєстрації домену .com складає $10.50, це дозволить поставити меншу комісію та сформувати вигідну ціну для користувачів які шукають економічно ефективні рішення.

Завдяки автоматичному моніторингу статусів доменів та здатності швидко реагувати на зміни, система не лише підвищує шанси користувачів на успішну реєстрацію бажаних доменів, але й зменшує ручний внесок і потенційні людські помилки. Розміщення серверної частини на надійному хостингу Namecheap додатково підсилює стабільність та безпеку процесів, оскільки система працюватиме постійно та матиме доступ до мережі 24/7.

В порівнянні з конкурентними системами, рішення вирізняється здатністю до гнучкого налаштування під індивідуальні потреби користувачів та відмінною підтримкою через інтегровані засоби відстеження та логування дій.

Всі наведені переваги роблять систему не лише технологічно просунутою, але й дружньою до користувача, що є ключовими факторами для вибору на користь запропонованого рішення серед аналогічних пропозицій на ринку.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Підвищення стійкості роботи підприємств приладобудівної галузі у воєнний час

Війна стала значним структурним шоком для економіки України та кожного регіону держави. Прогнозування термінів і наслідків військової агресії є неможливим, тому стратегічні цілі держави та регіонів потребують корегування відповідно до умов воєнного часу і майбутньої відбудови [13].

У воєнних умовах необхідно переглянути усталені підходи до стратегування та розв'язання нагальних соціально-економічних, гуманітарних і безпекових проблем на всіх рівнях управління, включаючи регіональний [13].

Війна на виснаження вимагає від України та її регіонів здійснення обережної економічної політики. Міжнародні резерви слід використовувати ощадливо, а бюджетні ресурси спрямовувати на фінансування військових цілей і забезпечення життєдіяльності в умовах воєнного стану [13].

З урахуванням військових дій, територію України можна розділити на тимчасово окуповані, звільнені після окупації, прифронтові з високою ймовірністю загрози окупації, фронтові, де тривають бойові дії, та регіони глибокого тилу [13].

Збройна агресія Російської Федерації насамперед вплинула на реальний сектор економіки, особливо на підприємства в тимчасово окупованих, звільнених після окупації, та прифронтових регіонах, що забезпечували значну частку внутрішнього промислового виробництва та експорту [13].

Завдання тилових регіонів включає збереження стратегічно важливих виробництв і робочих місць, забезпечення роботи релокованих підприємств, підтримку експортоорієнтованих підприємств, а також виробництва соціально значущих товарів і продукції військового призначення. Важливо стимулювати перехід до виробництва військової продукції [13].

У воєнний час першочерговими завданнями є максимальне збереження людей та підприємств, виконання гуманітарних місій. Успіхи плану

реконструкції залежать від того, наскільки буде збережено українців, економіку та інституції. Україна та її союзники повинні структурувати економіку воєнного часу відповідно до безпекових ризиків [13].

Тернопільщина, як відносно безпечний регіон, виконує роль «глибокого тилу»: допомагає переміщеним підприємствам, формує програми зайнятості для переміщених осіб, забезпечує логістичну та цифрову мобільність, транспортні коридори для ввезення гуманітарної допомоги, а також будує житло для внутрішньо переміщених осіб, зберігаючи економічний потенціал регіону [13].

Тилові регіони надають гуманітарну та технічну допомогу прифронтовим і фронтовим регіонам, забезпечуючи евакуацію виробництв, де це можливо, і підтримуючи зв'язок з іншими територіями країни [13].

Як тиловий регіон, Тернопільщина повинна також зосередити зусилля на забезпеченні заходів національного спротиву, антитерористичного забезпечення, належного функціонування систем оповіщення та сховищ. Ризиком є затягування і ескалація військових дій, руйнування інфраструктури, окупація регіону [13].

Стійкість роботи промислового об’єкта в умовах надзвичайних ситуацій визначається здатністю виробляти продукцію в запланованих обсягах та номенклатурі, а при слабких і середніх руйнуваннях відновлювати виробництво у мінімальні терміни [13].

Стійкість роботи підприємства приладобудівної галузі залежить від захисту і нормального функціонування таких елементів:

- виробничий персонал;
- будівлі і споруди з технологічним устаткуванням;
- системи постачання енергією, водою, паливом, устаткуванням;
- виробничі і кооперативні зв’язки з іншими об’єктами [13].

Тому стійкість підприємства приладобудівної галузі в умовах надзвичайних ситуацій визначається такими факторами:

- захист робітників та службовців від зброї масового ураження;
- здатність інженерно-технічного комплексу протистояти вражаючим факторам;

- надійність системи постачання об’єкта;
- захищеність об’єкта від вторинних вражаючих факторів;
- безперервність управління виробництвом і цивільною обороною;
- підготовленість об’єкта до проведення рятувальних та інших невідкладних робіт і відновлення виробництва [13].

Ці фактори визначають основні шляхи підвищення стійкості роботи в умовах надзвичайних ситуацій, зокрема:

- захист робітників та службовців;
- захист основних виробничих фондів;
- підвищення надійності управління виробництвом;
- забезпечення стійкості постачання;
- підготовка до відновлення виробництва [13].

Захист робітників та службовців в умовах надзвичайних ситуацій мирного і воєнного часу є головною задачею, оскільки робітники та службовці є головною продуктивною силою економіки [13].

Військові конфлікти супроводжуються руйнуванням будівель і знищенням основної продуктивної сили. Тому основним завданням є забезпечення захисту робітників та службовців і членів їхніх родин [13].

Захист робітників та службовців здійснюється трьома способами:

- укриття у захисних спорудах;
- евакуація робітників, службовців і членів їхніх родин;
- використання засобів індивідуального захисту [13].

Захист засобів виробництва включає підвищення фізичної опірності будівель, захист технологічного устаткування, засобів зв’язку та інших засобів виробничого процесу [13].

Методика оцінки стійкості будівель і технологічного устаткування до вражаючих факторів ядерного вибуху виконується за трьома основними показниками:

- вплив ударної хвилі ядерного вибуху;
- світлове випромінювання на предмет виникнення пожеж;
- радіація на предмет захисту виробничого персоналу [13].

Підготовка до відновлення порушеного виробництва передбачає планування відбудовних робіт, підготовку ремонтних бригад, створення запасів матеріалів і устаткування [13].

Підвищення стійкості роботи об’єкта у воєнний час і в умовах надзвичайних ситуацій досягається завчасним проведенням комплексу інженерно-технологічних, технологічних і організаційних заходів, спрямованих на зниження впливу вражаючих факторів і швидке відновлення виробництва [13].

Інженерно-технічні заходи включають підвищення стійкості будівель, устаткування, комунально-енергетичних систем. Технологічні заходи спрямовані на зміну технологічного процесу для прискорення виробництва. Організаційні заходи передбачають розробку дій керівного складу, захист робітників та службовців, відновлення виробництва і випуск продукції на збережених потужностях [13].

Технологічні заходи спрямовані на підвищення стійкості об’єкта шляхом зміни технологічних процесів, що прискорює виробництво продукції та усуває можливість виникнення вторинних вражаючих факторів [13].

Організаційні заходи включають розробку та планування дій керівного складу, штабу, служб та формувань цивільного захисту для забезпечення захисту працівників підприємства, виконання невідкладних робіт, відновлення виробництва та випуску продукції на наявних потужностях [13].

4.2 Загальні вимоги безпеки з охорони праці для користувачів ПК

Сучасний розвиток технічного та технологічного стану виробництва передбачає постійну автоматизацію та оптимізацію виробничих процесів. Сьогодні, напевно, важко уявити компанію, господарська діяльність в якій би здійснювалась без використання комп’ютерної техніки. Через масовий характер робіт, що виконуються працівниками за допомогою комп’ютера, законодавством України чітко врегульовано норми та вимоги до використання комп’ютерної техніки на підприємстві, безпосередньо й охорона праці при роботі з комп’ютером [12].

Приміщення, в яких планується установка та подальша робота з комп'ютером, повинні відповідати проєктній 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 характеристики електромагнітного, ультрафіолетового та інфрачервоного полів [12].

Конкретні показники зазначених санітарних норм подані 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 [12].

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 [12].

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 [12].

Роботодавець, який використовує найману працю робітників, повинен забезпечити відповідність їхніх робочих місць комфортним та безпечним умовам [12].

Площа одного робочого місця повинна бути не менше 6 $м^2$, а обсяг – не менше 20 $м^3$. При розміщенні робочих місць необхідно дотримуватись таких вимог:

- природне світло повинно падати збоку, переважно зліва;
- відстань від робочого місця до стін зі світловим прорізами повинна складати не менше 1 м;
- відстань між бічними поверхнями відеотерміналів має бути не меншою за 1,2 м;
- відстань між тильною поверхнею одного відеотермінала та екрана
- іншого не повинна бути меншою 2,5 м, а прохід між рядами робочих
- місць – не меншим одного метра [12].

Висота робочої поверхні столу для відеотермінала має бути в межах 68-80 см, а ширина повинна забезпечувати можливість використання операцій у зоні досяжності моторного поля (рекомендовані розміри столу: висота – 72,5 см, ширина – 60-140 см, глибина – 80-100 см) [12].

Робоче сидіння (сидіння, стілець, крісло) працівника на обчислювальній техніці повинно бути підйомно-поворотним, плоским, спереду закругленим, а для усунення статичного напруження м'язів рук улаштоване стаціонарними або змінними підлокітниками [12].

Екран відеотермінала та клавіатура мають розташовуватися на оптимальній відстані від очей працівника, але не ближче 60 см, з урахуванням розміру алфавітно-цифрових знаків та символів [12].

Організація робочого місця з ЕОМ для управління технологічними обладнаннями має передбачати:

- достатній простір для людини-оператора;
- вільну досяжність органів ручного управління в зоні моторного поля: відстань по висоті – до 133 см, по глибині – 40-50 см;
- розташування екрана відеотермінала в робочій зоні, яке забезпечувало б зручність зорового спостереження у вертикальній площині під кутом плюс-мінус 30° від лінії зору оператора;
- можливість повертання екрана відеотермінала навколо горизонтальної та вертикальної осі [12].

При потребі високої концентрації уваги під час виконання робіт з високим рівнем напруженості суміщені робочі місця з відеотерміналами та персональними ЕОМ необхідно відділяти одне від одного перегородками висотою 1,5 - 2 м [12].

4.3 Висновок до четвертого розділу

В четвертому розділі наукової роботи описано методи підвищення стійкості роботи підприємств приладобудівної галузі в умовах воєнного часу.

Також описано вимоги до робочого місця при роботі з персональним комп’ютером, які встановлені законодавством України та державними санітарними нормами.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було створено автоматизовану систему для супроводу процесів реєстрації та життєвого циклу доменних імен.

В першому розділі кваліфікаційної роботи освітнього рівня «Магістр»:

- Подано структуру системи доменних імен (DNS).
- Розглянуто основні етапи життєвого циклу доменного імені.
- Висвітлено проблеми пов’язані з життєвим циклом доменів, з якими стикаються їх власники.
- Досліджено відомі на даний час системи для автоматичної реєстрації доменів.
- Обґрунтовано потребу в системах для супроводу процесів реєстрації та життєвого циклу доменних імен.
- Сформовано вимоги до автоматизованої системи.

В другому розділі кваліфікаційної роботи:

- Описано відомі засоби для реалізації систем супроводу процесів реєстрації та життєвого циклу доменних імен.
- Досліджено популярні мови програмування, хостингові платформи та системи моніторингу доменнних імен.
- Подано порівняльний аналіз засобів для реалізації системи.

В третьому розділі кваліфікаційної роботи:

- Спроєктовано та розроблено автоматизовану систему для супроводу процесів реєстрації та життєвого циклу доменних імен.
- Запропоновано економічно вигідне рішення для супроводу процесів реєстрації та життєвого циклу доменних імен.
- Протестовано автоматичну реєстрацію доступного доменного імені.

У розділі «Охорона праці та безпека в надзвичайних ситуаціях» описано методи підвищення стійкості роботи підприємств приладобудівної галузі в умовах воєнного часу. Також описано вимоги до робочого місця при роботі з персональним комп’ютером, які встановлені законодавством України та державними санітарними нормами.

ПЕРЕЛІК ДЖЕРЕЛ

1. Альбітц П. DNS and BIND Cookbook - В.: O'Reilly Media, 2002. - 240 с. - ISBN 978-0596004101.

2. Баер В. Cyberworld Security - В.: Springer, 2010. - 300 с. - ISBN 978-3642124321.

3. Бікс Б. Практика та теорія DNS - СПб.: Питер, 2017. - 360 с. - ISBN 978-5-4461-0920-3.

4. Блейк С. Основи веб-хостингу - К.: Видавництво "Альфа", 2019. - 300 с. - ISBN 978-617-7280-24-3.

5. Брайан У. Програмування мовою Java - В.: Науковий світ, 2020. - 504 с. - ISBN 978-966-10-5879-7

6. Браун Е. Вивчаємо JavaScript: керівництво по створенню сучасних веб-сайтів. 3 видання - В.: Вільямс, 2017. - 368 с. - ISBN 978-617-7812-55-4

7. Васильєв О. Мова програмування Go - В.: Навчальна книга - Богдан, 2022. - 432 с. - ISBN 978-617-7812-22-6

8. Васильєв О. Програмування мовою Python - В.: Навчальна книга - Богдан, 2019. - 504 с. - ISBN 978-966-10-5611-3

9. Віксі П. Managing Mission-Critical Domains and DNS - В.: "No Starch Press", 2018. - 272 с. - ISBN 978-1593278065.

10. Вільямс П. Хостинг для малого бізнесу - СПб.: Питер, 2017. - 310 с. - ISBN 978-5-4461-0823-7.

11. Гаррісон Л. Оптимізація веб-хостингів - Х.: Харківський національний університет, 2020. - 330 с. - ISBN 978-966-7009-98-5.

12. Грибан В. Г. Охорона праці. Навчальний посібник – Київ: Центр учбової літератури, 2009. – 280 с.

13. Демиденко Г.П. Безпека життєдіяльності: навч.посіб. – К.: НТУУ «КШ», 2008. – 287 с.

14. Джексон Г. Хостинг для малого бізнесу - СПб.: Питер, 2017. - 310 с. - ISBN 978-5-4461-0823-8.

15. Джексон К. Розуміння DNS та системи доменних імен - М.: Видавництво "Техніка", 2018. - 290 с. - ISBN 978-5-001-20641-8.

16. Джонсон М. Java: Основи програмування - К.: Видавництво "Знання", 2019. - 504 с. - ISBN 978-617-7480-00-5.

17. Доменне ім'я - Wikipedia. URL: https://uk.wikipedia.org/wiki/Доменне_ім'я (дата звернення: 22.02.2024)

18. Ендрюс П. Програмування на Python: від початківця до професіонала - М.: Видавництво "Просвітництво", 2018. - 460 с. - ISBN 978-5-010-30742-3.

19. Картер Б. Програмування на Go: практичний посібник - Л.: Літера, 2021. - 480 с. - ISBN 978-966-03-9522-6.

20. Касперський Є. Cybersecurity for Beginners - В.: "Red&Black", 2017. - 208 с. - ISBN 978-1911452034.

21. Кемпбелл С. Оптимізація хостинг-рішень - Х.: Харківський національний університет, 2020. - 330 с. - ISBN 978-966-7009-99-8.

22. Крокер С. DNS та безпека Інтернету - Л.: Літера, 2020. - 310 с. - ISBN 978-966-03-9081-5.

23. Лаура Д. The Global War for Internet Governance - В.: Yale University Press, 2014. - 296 с. - ISBN 978-0300199475.

24. Лі Дж. DNS імена та адреси - Х.: Харківський національний університет, 2019. - 250 с. - ISBN 978-966-7009-62-6.

25. Лоу Н. Securing the Internet of Things - В.: Syngress, 2017. - 154 с. - ISBN 978-0128045053.

26. Мартінес Т. Вибір хостинг-провайдера - М.: Наука, 2018. - 320 с. - ISBN 978-5-02-038739-5.

27. Міллер Р. Бізнес-моделі для доменних імен - М.: Наука, 2018. - 410 с. - ISBN 978-5-02-038754-7.

28. Морісон Л. C#: Ефективне програмування - Х.: Харківський національний університет, 2019. - 450 с. - ISBN 978-966-7009-61-9.

29. Московіц Д. DNS and BIND - В.: O'Reilly Media, 2006. - 642 с. - ISBN 978-0596100575.

30. Оре Г. Системи доменних імен (DNS) - К.: Видавництво "Альфа", 2019. - 320 с. - ISBN 978-617-7280-01-2.

31. Петерсон Л. DNS та управління Інтернетом - М.: Наука, 2018. - 280 с. - ISBN 978-5-02-038837-7.

32. Петров М. Інтеграція DNS у корпоративні мережі - СПб.: Питер, 2020. - 340 с. - ISBN 978-5-4461-1001-7.

33. Реєстратор доменних імен - Wikipedia. URL: https://uk.wikipedia.org/wiki/Реєстратор_доменних_імен (дата звернення: 05.11.2023)

34. Ремплінг Б. DNS For Dummies - В.: Wiley. John Wiley & Sons, LTD, 2020. - 368 с. - ISBN 9780764516832

35. Роні Е. - Посібник з життєвого циклу системи доменних імен (DNS) - В.: Cmp Books, 2018. - 645 с. - ISBN 978-0879305154

36. Система Доменних Імен - Wikipedia. URL: https://uk.wikipedia.org/wiki/Система_Доменних_Імен (дата звернення: 14.01.2024)

37. Скіт Д. C# in Depth - В.: Manning, 2019. - 528 с. - ISBN 9781617294532

38. Соломон М. DNS Security Management - В.: Wiley, 2017. - 384 с. - ISBN 978-1119328275.

39. Стоун Б. Продається все. Джефф Безос та ера Amazon - В.: Наш Формат, 2023. - 400 с. - ISBN 978-617-8120-51-1

40. Томпсон Б. DNS та Інтернет-протоколи - К.: Видавництво "Освіта", 2019. - 320 с. - ISBN 978-617-7585-01-5.

41. Френк Дж. JavaScript для веб-розробників - СПб.: Питер, 2017. - 430 с. - ISBN 978-5-4461-0790-5.

42. Фриз М.Є., Млинко Б.Б. Умовні лінійні випадкові процеси з дискретним часом та їх властивості - Вісник Хмельницького національного університету. Серія: Технічні науки, 2022 (309), № 3. С. 7-12.

43. Харпер Дж. Правове регулювання доменних імен - В.: Науковий світ, 2021. - 380 с. - ISBN 978-966-10-7226-9.

44. Харрісон Г. Веб-хостинг для початківців - К.: Видавництво "Альфа", 2019. - 300 с. - ISBN 978-617-7280-24-1.

45. Шнайдер В. Mastering Windows Server 2016 Hyper-V - В.: Sybex, 2016. - 648 с. - ISBN 978-1119286187.

46. Amazon Web Services - Wikipedia. URL: https://uk.wikipedia.org/wiki/Amazon_Web_Services (дата звернення: 02.02.2024)

47. Anvil Docs website. URL: https://anvil.works/docs/overview (дата звернення: 11.04.2024)

48. Domain name - Wikipedia. URL: https://en.wikipedia.org/wiki/Domain_name (дата звернення: 02.10.2023)

49. Domain Name System - Wikipedia. URL: https://en.wikipedia.org/wiki/Domain_name (дата звернення: 19.11.2023)

50. DomainTools website. URL: https://www.domaintools.com/ (дата звернення: 11.02.2024)

51. Dynadot website. URL: https://www.dynadot.com/ (дата звернення: 05.03.2024)

52. Fryz M., Mlynko B. Properties of Stationarity and Cyclostationarity of Conditional Linear Random Processes - IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET). Lviv-Slavske, Ukraine: IEEE, 2020. P. 166-170.

53. Fryz M., Mlynko B. Property Analysis of Conditional Linear Random Process as a Mathematical Model of Cyclostationary Signal - 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2022). Ternopil, Ukraine: CEUR Workshop Proceedings, 2022. Vol. 3309. P. 77-82.

54. Fryz M., Mlynko B. Property analysis of multivariate conditional linear random processes in the problems of mathematical modelling of signals - Technol. Audit Prod. Reserv. 2022. Vol. 3, № 2(65). P. 29-32.

55. Fryz M., Mlynko B. Determination of the characteristic function of discrete-time conditional linear random process and its application - Sci. J. TNTU, 2023. Vol. 109, № 1. P. 16-23.

56. GoDaddy website. URL: https://www.godaddy.com/ (дата звернення: 13.01.2024)

57. Google Cloud Platform - Wikipedia. URL: https://en.wikipedia.org/wiki/Google_Cloud_Platform (дата звернення: 22.02.2024)

58. IP-адреса - Wikipedia. URL: https://uk.wikipedia.org/wiki/IP-адреса (дата звернення: 22.02.2024)

59. Microsoft Azure - Wikipedia. URL: https://en.wikipedia.org/wiki/Microsoft_Azure (дата звернення: 01.03.2024)

60. Namecheap website. URL: https://www.namecheap.com/ (дата звернення: 19.03.2024)

ДОДАТКИ

Додаток А

Теза конференції

УДК 004.738.5
Мац О. – ст. гр.СНнм-61
Тернопільський національний технічний університет імені Івана Пулюя

ПОРІВНЯЛЬНИЙ АНАЛІЗ ПЕРЕХОПЛЕННЯ ПРОТЕРМІНОВАНИХ ДОМЕНІВ НАД РЕЄСТРАЦІЄЮ НОВИХ ДОМЕННИХ ІМЕН

Науковий керівник: к.т.н., доцент Млинко Б. Б.

Mats O.
Ternopil Ivan Puluj National Technical University

COMPARATIVE ANALYSIS OF EXPIRED DOMAINS DROP-CATCHING OVER REGISTRATION OF NEW DOMAIN NAMES

Supervisor: Ph.D., Assoc. Prof. Bogdana Mlynko

Ключові слова: домен, перехоплення, протермінований.
Keywords: domain, drop-catching, expired.

У сучасному цифровому світі, де Інтернет став невід'ємною частиною практично всіх аспектів життя та бізнесу, доменні імена відіграють ключову роль у визначенні присутності та ідентичності в Інтернеті. Однак, питання їх вибору і управління вимагає ретельного аналізу та вибору стратегій, що найкраще відповідають потребам користувачів та бізнесу. Це питання не лише важливе для власників веб-проєктів, але й для інтернет-спільноти в цілому.

Перехоплення доменного імені (англ. Drop-Catching) – це процес реєстрації протермінованих доменних імен за допомогою автоматизованих систем протягом короткого періоду часу (навіть долі секунди) після їх видалення реєстром. Простіше кажучи, автоматична реєстрація доменів в момент їх видалення. Drop-Catching виконується різними компаніями, відомими як "drop-catchers" [1].

Здебільшого перехоплення використовується власниками доменів, які намагаються повернути свій домен в якого сплив термін дії і не хочуть платити великі комісії доменним реєстраторам. Також перехоплення застосовується інвесторами, які хочуть отримати найкращі домени з метою перепродажу [2].

В ході дослідженні було визначено ряд переваг перехоплення доменів:

- Органічний трафік: Домени, які стають доступними для реєстрації, часто мають історію і наявний органічний трафік. Це може бути важливим фактором для веб-проєктів, оскільки він дозволяє отримувати відвідувачів навіть без активних маркетингових кампаній [3].
- Цінність домену: Деякі протерміновані домени можуть бути цінними самі по собі через їхню легкість запам'ятовування, короткість або ключові слова, які вони містять. Перехоплення таких доменів може стати вигідним інвестиційним рішенням [1].
- Уникнення конфліктів: Перехоплення дозволяє уникнути конфліктів з наявними доменами, особливо якщо вони вже використовуються у власницькій або торговій діяльності. Доволі часто бізнеси які банкрутують або закриваються залишають свої домени без поновлення [4].
- Економія часу та ресурсів: Замість пошуку нових доменних імен, перехоплення дозволяє швидко і ефективно вибрати вже наявні, що економить час та ресурси [4].

Однак, варто пам'ятати що часто перехоплення дозволяє зберегти історію та авторитет домену. Це може вплинути на SEO та інші аспекти веб-сайту як позитивно так і негативно. Перед перехопленням домену варто ретельно дослідити його статуси WHOIS, історію, блокування у різноманітних чорних списках та базах даних спаму [1].

Підсумовуючи результати порівняльного аналізу встановлено, що перехоплення дозволяє отримати доступ до доменів з історією, які можуть мати вже налагоджений органічний трафік і відомі у веб-середовищі. Крім того, ці домени можуть бути цінними самі по собі через свою унікальність та потенційну вартість у бізнесових та маркетингових аспектах. Важливою перевагою перехоплення є також можливість зменшення конкуренції за нові домени, що дозволяє ефективніше використовувати доступні ресурси для досягнення поставлених цілей у доменному просторі Інтернету.

Література

1. Drop-Catching – ICANNWiki. URL: https://icannwiki.org/Drop-Catching
2. Jones M. Domain Name Investing: Make Money Online And Run Your Own Home Business By Buying And Selling Premium Domains In Your Spare Time! CreateSpace IPP. 2019, 98.
3. Kesmodel, D. The Domain Game: How People Get Rich From Internet Domain Names. Xlibris. 2008, 212.
Silver J. E. Domain Names: How to Choose & Protect a Great Name for Your Website. Nolo. 2014, 250.

Додаток Б

Теза конференції

УДК 004.738.5
Мац О. – ст. гр.СНнм-61
Тернопільський національний технічний університет імені Івана Пулюя

ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ АВТОМАТИЗОВАНОЇ СИСТЕМИ ДЛЯ СУПРОВОДУ ПРОЦЕСІВ РЕЄСТРАЦІЇ ТА ЖИТТЄВОГО ЦИКЛУ ДОМЕННИХ ІМЕН

Науковий керівник: к.т.н., доцент Млинко Б. Б.

Mats O.
Ternopil Ivan Puluj National Technical University

SELECTION OF TECHNOLOGIES FOR THE DEVELOPMENT OF AN AUTOMATED SYSTEM TO ACCOMPANY REGISTRATION PROCESSES AND DOMAIN NAMES LIFE CYCLE

Supervisor: Ph.D., Assoc. Prof. Bogdana Mlynko

Ключові слова: мова програмування, система, реєстратор.
Keywords: programming language, system, registrar.

Розробка автоматизованої системи для супроводу процесів реєстрації та життєвого циклу доменних імен вимагає глибокого розуміння потреб користувачів та технічних вимог до системи. Це дослідження окреслює вимоги, які повинні бути задоволені для ефективної роботи такої системи а також способи їх реалізації.

До автоматизованої системи для супроводу процесів реєстрації та життєвого циклу доменних імен можна поставити ряд стандартних вимог як і для будь-якої інформаційної системи що знаходиться в мережі інтернет: зручний та зрозумілий інтерфейс, безпека даних, надійність, швидкість, масштабованість, гнучкість, тощо [1].

Для розробки автоматизованої системи та задоволення поставлених вимог, нам було важливо врахувати різноманітні аспекти технологій. Ключовими етапами розробки є вибір мови програмування, вибір технологічного стеку для розробки, а також інструменту для моніторингу статусу доменів.

Вибір мови програмування може значно вплинути на продуктивність застосунку, легкість розробки, а також можливості для майбутнього розширення. Серед доступних опцій мови програмування Python або JavaScript (з використанням фреймворків як Django або React.js) можуть бути відмінними виборами для реалізації функціональності системи [2].

Вибір технологічного стеку для розробки автоматизованої системи супроводу процесів реєстрації та життєвого циклу доменних імен є важливим етапом, який визначає як швидкість розробки, так і ефективність й масштабованість системи, легкість обслуговування системи та її безпеку. В цьому контексті, важливо розглянути як традиційні, так і інноваційні рішення, серед яких платформа Anvil.works виступає як обнадійливий кандидат [1].

Моніторинг статусу доменів є останньою, але не менш важливою частиною системи. Ефективний моніторинг дозволяє виявляти домени, що незабаром звільняться, та автоматично реєструвати їх. Для цього можна використовувати різноманітні інструменти та API, в тому числі надані доменними реєстраторами [2].

Згідно з дослідженням, використання платформи акредитованого доменного реєстратора дозволить не лише бути впевненим в надійності отриманих даних про статуси доменів, але й мінімізувати затримку між виявленням вільного домену та його автоматичною реєстрацією. Окрім даних про надійність та безпеку доменного реєстратора, не в останню чергу була звернута увага на ціни на послуги реєстрації доменів, оскільки це формуватиме ціну на послуги системи для кінцевого користувача.

В ході порівняльного аналізу мов програмування, для розробки системи було обґрунтовано вибір мови програмування Python, через простоту синтаксису, гнучкість для розробки, швидкість та різноманіття фреймворків.

Платформою для створення системи було обґрунтовано вибір Anvil.works на підставі наступних переваг: простоти експлуатації та швидкості розробки, а також наявність безкоштовного тарифного плану який покриває весь необхідний функціонал та здешевляє розробку.

Щодо сервісу моніторингу доменів, то після аналізу багатьох альтернатив, вибір було зосереджено на Namecheap API. Наданих сервісом можливостей виявилось достатньо для потреб розроблюваної системи, а низькі ціни на реєстрацію доменів дозволяють сформувати конкурентну ціну на послуги системи.

Література

1. Jones M. Domain Name Investing: Make Money Online And Run Your Own Home Business By Buying And Selling Premium Domains In Your Spare Time! CreateSpace IPP. 2019, 98.

2. Kesmodel, D. The Domain Game: How People Get Rich From Internet Domain Names. Xlibris. 2008, 212.

Додаток В

Лістинг коду серверної частини системи

```
import anvil.server
import anvil.tables as tables
import anvil.tables.query as q
from anvil.tables import app_tables

import smtplib
from email.utils import formataddr
from email.message import EmailMessage

import time
from time import strftime
from datetime import datetime, timedelta

import re
import json
import requests

import logging
import logging.handlers as handlers

#============= Current Time for logging (in UTC)
=============
def get_curtime ():
    utc_time = datetime.utcnow()
    current_time = utc_time.strftime("%d %b %Y %H:%M:%S")
    return (current_time)

#============= Logging feature =============
logger = logging.getLogger('my_app')
logger.setLevel(logging.DEBUG)
logHandler = handlers.RotatingFileHandler("log",
maxBytes=5000000, backupCount=10)
logHandler.setLevel(logging.DEBUG)
logHandler.namer = lambda name: name + ".log"
logger.addHandler(logHandler)

logger.debug("======================START==================
==")
logger.debug("Version 1.0")
logger.debug(f"Started at {get_curtime()}")

#============= Anvil Settings =============
anvil.server.connect("token")

#Option to check app connection during the page load
@anvil.server.callable
def check_connection():
  return 22

#============= API Settings =============
time_sleep = 20 #The interval for checking (in seconds)
CommandCheck = "namecheap.domains.check"
```

```
    CommandRegister = "namecheap.domains.create"
    ClientIp = "91.245.111.213"
    UserName = "Username"
    ApiUser = " Username "
    ApiKey = "ApiKey"
    URL = "https://api.namecheap.com/xml.response"

    #Domain registration details
    Years = 1
    AuxBillingFirstName = "John"
    AuxBillingLastName = "Smith"
    AuxBillingAddress1 = "8939 S.cross Blv"
    AuxBillingStateProvince = "CA"
    AuxBillingPostalCode = "90045"
    AuxBillingCountry = "US"
    AuxBillingPhone = "+1.6613102107"
    AuxBillingEmailAddress = "john@gmail.com"
    AuxBillingOrganizationName = "NC"
    AuxBillingCity = "CA"
    TechFirstName = "John"
    TechLastName = "Smith"
    TechAddress1 = "8939 S.cross Blvd"
    TechStateProvince = "CA"
    TechPostalCode = "90045"
    TechCountry = "US"
    TechPhone = "+1.6613102107"
    TechEmailAddress = "john@gmail.com"
    TechOrganizationName = "NC"
    TechCity = "CA"
    AdminFirstName = "John"
    AdminLastName = "Smith"
    AdminAddress1 = "8939 cross Blvd"
    AdminStateProvince = "CA"
    AdminPostalCode = "9004"
    AdminCountry = "US"
    AdminPhone = "+1.6613102107"
    AdminEmailAddress = "joe@gmail.com"
    AdminOrganizationName = "NC"
    AdminCity = "CA"
    RegistrantFirstName = "John"
    RegistrantLastName = "Smith"
    RegistrantAddress1 = "8939 S.cross Blvd"
    RegistrantStateProvince = "CS"
    RegistrantPostalCode = "90045"
    RegistrantCountry = "US"
    RegistrantPhone = "+1.6613102107"
    RegistrantEmailAddress = "jo@gmail.com"
    RegistrantOrganizationName = "NC"
    RegistrantCity = "CA"
    AddFreeWhoisguard = "yes"
    WGEnabled = "yes"

    #============= Parsing XML output to get Domain and it's
Status =============
    def parse_check_xml(xml_data):
```

```
        pattern = r'<DomainCheckResult Domain="(.*?)"
Available="(.*?)"'
        matches = re.findall(pattern, xml_data, re.DOTALL)

        results =  []
        for match in matches:
            domain, status = match
            results.append((domain, status))
        return results

      #============= Parsing XML output to get status of domain
registration =============
    def parse_register_xml(xml_data):
      match = re.search(r'Registered="( [^"]*)"', xml_data)
      if match:
            return match.group(1)
      else:
            return None

    #=============== Check domain availability and returning
parsed results as: domain, status =================
    def check_domain_availability(domains):
      Domain_list = ",".join(domains)
      payload = {'ApiUser': ApiUser, 'ApiKey': ApiKey, 'UserName'
: UserName, 'Command' : CommandCheck, 'ClientIp' : ClientIp,
'DomainList' : Domain_list}
      AvailRequest = requests.get(URL, params=payload)
      logger.debug (f"{get_curtime()}: API Response =
\n{AvailRequest.text}")
      parsed_results = parse_check_xml(AvailRequest.text)
      for domain, status in parsed_results:
            logger.debug (f"Domain: {domain}, Status: {status}")
      return (parsed_results)

    #=============== Registers a domain ================
    def register_domain(domain):
      logger.debug (f"Registering {domain}\n")
      payload = {'ApiUser': ApiUser,
      'ApiKey': ApiKey,
      'UserName' : UserName,
      'Command' : CommandRegister,
      'ClientIp' : ClientIp,
      'DomainName' : domain,
      'Years': Years,
        'AuxBillingFirstName': AuxBillingFirstName,
        'AuxBillingLastName': AuxBillingLastName,
        'AuxBillingAddress1': AuxBillingAddress1,
        'AuxBillingStateProvince': AuxBillingStateProvince,
        'AuxBillingPostalCode': AuxBillingPostalCode,
        'AuxBillingCountry': AuxBillingCountry,
        'AuxBillingPhone': AuxBillingPhone,
        'AuxBillingEmailAddress': AuxBillingEmailAddress,
        'AuxBillingOrganizationName': AuxBillingOrganizationName,
        'AuxBillingCity': AuxBillingCity,
        'TechFirstName': TechFirstName,
       'TechLastName': TechLastName,
```

```
          'TechAddress1': TechAddress1,
          'TechStateProvince': TechStateProvince,
          'TechPostalCode': TechPostalCode,
          'TechCountry': TechCountry,
          'TechPhone': TechPhone,
          'TechEmailAddress': TechEmailAddress,
          'TechOrganizationName': TechOrganizationName,
          'TechCity': TechCity,
          'AdminFirstName': AdminFirstName,
          'AdminLastName': AdminLastName,
          'AdminAddress1': AdminAddress1,
          'AdminStateProvince': AdminStateProvince,
          'AdminPostalCode': AdminPostalCode,
          'AdminCountry': AdminCountry,
          'AdminPhone': AdminPhone,
          'AdminEmailAddress': AdminEmailAddress,
          'AdminOrganizationName': AdminOrganizationName,
          'AdminCity': AdminCity,
          'RegistrantFirstName': RegistrantFirstName,
          'RegistrantLastName': RegistrantLastName,
          'RegistrantAddress1': RegistrantAddress1,
          'RegistrantStateProvince': RegistrantStateProvince,
          'RegistrantPostalCode': RegistrantPostalCode,
          'RegistrantCountry': RegistrantCountry,
          'RegistrantPhone': RegistrantPhone,
          'RegistrantEmailAddress': RegistrantEmailAddress,
          'RegistrantOrganizationName': RegistrantOrganizationName,
          'RegistrantCity': RegistrantCity,
          'AddFreeWhoisguard': AddFreeWhoisguard,
          'WGEnabled': WGEnabled}

       AvailRequest = requests.get(URL, params=payload)
       logger.debug (f"{get_curtime()}: API Response =
\n{AvailRequest.text}")
       registration_results = parse_register_xml(AvailRequest.text)
       if registration_results == "true":
              logger.debug(f"Registered {domain}. Sending email\n")
              send_email(domain)
              logger.debug(f"{get_curtime()}: Sending Success
email.")
       else:
              send_error_email(domain)
              logger.debug(f"{get_curtime()}: Sending Error
email.")
       return (registration_results)

     #=============== The infinite loop that's constantly checking
domain list================
     def monitor_domains():
       while True:
              data_row = app_tables.dm_maindb.get(M_index=22)
              monitored_domains = json.loads(data_row
["M_domainObject"])
              logger.debug (f"{get_curtime()}: DB loaded. List:")
              logger.debug (monitored_domains)
```

```
            results =
check_domain_availability(monitored_domains)

            for domain, status in results:
                  if status == "true":
                        registration_status =
register_domain(domain)
                        if registration_status == "true":
                              monitored_domains.remove(domain)

      app_tables.dm_logdb.add_row(Date=get_curtime(),
Message=f"{domain} - registered by system",
Counter=len(app_tables.dm_logdb.search()))..0
                              data = monitored_domains
                              jsonString = json.dumps(data)
                              data_row.update(M_domainObject =
jsonString)
                              logger.debug(f"{get_curtime()}:
{domain} removed from the monitoring list.")
                        else:
                              logger.debug(f"{get_curtime()}:
{domain} was not registered due to the error.")
                              pass

                  else:
                        logger.debug(f"{get_curtime()}:
{domain} still under monitoring.")
            logger.debug(f"============== {get_curtime()}:
Sleeping now ==============")
            time.sleep(time_sleep)

     #=============== EMAIL sending =================

     def send_email(domain):
       msg = EmailMessage()
       msg ['From'] = formataddr(('Domain-Monitor', 'no-
reply@domain-monitor.br1ze.com'))
       msg ['To'] = formataddr(('Full Name, 'email@gmail.com'))
       msg ['Subject'] = (f"Domain-Monitor Tool: {domain} has been
registered")
       msg.set_content(f"""The following domain has been
registered:
     {domain}

     Have a nice day!""")
       sending_email(msg)
       return (msg)

     def send_error_email (domain):
       msg = EmailMessage()
       msg ['From'] = formataddr(('Domain-Monitor', 'no-
reply@domain-monitor.br1ze.com'))
       msg ['To'] = formataddr(('Oleg Mats',
'olik.mats@gmail.com'))
       msg ['Subject'] = (f"Domain-Monitor Tool: {domain} was NOT
    registered")
```

```
      msg.set_content(f"""The following domain was NOT registered
due to error:

    {domain}

    Please check logs!""")
      sending_email(msg)
      return

    def sending_email(message):
      smtp_server = "server.web-hosting.com"
      port = 587
      username = "no-reply@domain-monitor.domain.com"
      password = "password"

      server = smtplib.SMTP(smtp_server, port)
      try:
          logger.debug("Logging in to the email...")

          status_code, response = server.ehlo()
          logger.debug(f" [*] Echoing the server: {status_code}
{response}")

          status_code, response = server.starttls()
          logger.debug(f" [*] StartingTLS connection:
{status_code} {response}")

          status_code, response = server.login(username, password)
          logger.debug(f" [*] Logging in: {status_code}
{response}")
          server.send_message(message)

      except Exception as e:
          logger.debug(e)
      finally:
          server.quit()
          logger.debug(f"Email sent")

    monitor_domains()
    anvil.server.wait_forever()
```

Додаток Д

Лістинг коду клієнтської частини системи

```
from ._anvil_designer import Form1Template
from anvil import *
import anvil.server
import anvil.tables as tables
import anvil.tables.query as q
from anvil.tables import app_tables
import re
import json
import time
from time import strftime
from datetime import datetime, timedelta

class Form1(Form1Template):
  def __init__(self, **properties):
    # Set Form properties and Data Bindings.
    self.init_components(**properties)

    global monitored_domains
    monitored_domains =  []

    data_row = app_tables.dm_maindb.get(M_index=22)
    monitored_domains = json.loads(data_row
["M_domainObject"])
    if data_row ["M_domainObject"] == " []":
      pass
    else:
      self.update_list()

    self.repeating_panel_1.items =
app_tables.dm_logdb.search(tables.order_by("Counter",
ascending=False))
    try:
      if anvil.server.call('check_connection') == 22:
        self.make_online()

      else:
        pass
    except anvil.server.NoServerFunctionError:
      pass
    except anvil.server.RuntimeUnavailableError:
      pass
    except anvil.server.UplinkDisconnectedError:
      pass
    # Any code you write here will run before the form opens.

#====================DOMAIN VALIDAION====================
  def extract_domain(self, url):
    url = url.strip()
    # Deletes spaces and other symbols
    clean_url = re.sub(r'https?://|www\.|/.*', '', url)
    return clean_url

 def isValidDomain(self, str_input):
```

```
            regex = r"^((?!-) [A-Za-z0-9-]{1,63}(?<!-)\.)+ [A-Za-
z]{2,6}"
            p = re.compile(regex)

            if (str_input == None):
              # If the string is empty return false
                return False

            if p.match(str_input):
              # Return if the string matched the ReGex
                return True
            else:
                return False

        def isDuplicateDomain(self, new_domain):
          if new_domain in monitored_domains:
            return True
          else:
            return False

      #====================UI functions====================
        def clear_domain_field(self):
          self.domain_input.text = ""

        def make_online (self):
          self.label_5.text = "Online"
          self.label_5.icon = "fa:check"
          self.label_5.foreground = "white"

        def make_offline (self):
          self.label_5.text = "Monitoring system disabled. Please
contact support."
          self.label_5.icon = "fa:warning"
          self.label_5.foreground = ""

        def update_list(self):
          data_row = app_tables.dm_maindb.get(M_index=22)
          monitored_domains_list = json.loads(data_row
["M_domainObject"])
          result_list =  []
          result = ""
          for item in monitored_domains_list:
            result_list.append(item)

          for item in result_list:
            result = item + "\n" + result
          self.text_area_1.text = result

        def get_curtime (self):
          utc_time = datetime.utcnow()
          current_time = utc_time.strftime("%d %b %Y %H:%M:%S")
          return (current_time)

      #====================ADD DOMAIN====================
       def button_1_click(self, **event_args):
```

```
        new_domain =
self.extract_domain(str(self.domain_input.text))

        if self.isValidDomain(str(new_domain)) == False:
          alert("The domain name is invalid, please double-check
it and try again")
        else:
          data_row = app_tables.dm_maindb.get(M_index=22)
          data = monitored_domains

          if self.isDuplicateDomain(new_domain):
            alert("The domain name is already added to the
monitoring list. Please refresh the page")
          else:
            data.append(new_domain)
            #print(data)
            jsonString = json.dumps(data)
            data_row.update(M_domainObject = jsonString) #write
data into DB for backend: DM_mainDB
            app_tables.dm_logdb.add_row(Date=self.get_curtime(),
Message=f"{new_domain} - added",
Counter=len(app_tables.dm_logdb.search())) #write data for DM_Log

            #self.text_area_1.text = data #update Text Area to
show actual list
            self.update_list()
            self.clear_domain_field()
        pass

    #====================REMOVE DOMAIN====================
      def button_2_click(self, **event_args):
          try:
            domain = self.domain_input.text

            data_row = app_tables.dm_maindb.get(M_index=22)
            monitored_domains = json.loads(data_row
["M_domainObject"])
            monitored_domains.remove(domain)
            data = monitored_domains
            jsonString = json.dumps(data)
            data_row.update(M_domainObject = jsonString)
            self.update_list()
            app_tables.dm_logdb.add_row(Date=self.get_curtime(),
Message=f"{domain} - removed using the web-form",
Counter=len(app_tables.dm_logdb.search()))
          except Exception as e:
            alert("Error: Please double-check the domain you
entered. It doesn't look like the domain is present in the list")
      pass
```