МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_____ проф. Приходько С.Б.

«___» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка вебсервісу для моніторингу стану вебсайтів

Виконав: студент групи 4151

_____Бєлоножко Н.Ю.

(підпис, ПІБ)

Керівник роботи:

_____доцент кафедри ПЗАС, к.т.н.

(посада, науковий ступень вчене звання)

_____Пухалевич А.В.

(підпис, ПІБ)

Миколаїв – 2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«___» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Белоножко Нікіті Юрійовичу _____
(Прізвище, ім'я, по батькові)
1. Тема роботи: Розробка вебсервісу для моніторингу стану вебсайтів _____
_____
Керівник роботи доцент кафедри ПЗАС, к.т.н. Пухалевич Андрій Володимирович _____
Затверджені наказом ректора №257-уч від «17» березня 2023 р.
2. Термін подання роботи: 29.05.2023 р. _____
3. Вихідні дані по роботі: _____
_____
4. Перелік питань, що належать до розробки (найменування розділів)_____

Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів 1.Слайд – тема кваліфікаційної роботи; 2. Слайд – мета роботи; 3. Слайд – Функції для користувача; 4. Слайд – Діаграма варіантів використання;5. Слайд – Діаграма діяльності; 6. Слайд – Діаграма класів; 7. Слайд – Діаграма послідовності;8. Слайд – Фізична модель даних; 9. Слайд – Використані технології; 10. Слайд – результати розробки; 11. Слайд – результати розробки; 12. Слайд – результати розробки;13. Слайд – Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 20.03.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	24.03.2023	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	31.03.2023	ПП*
3.	Аналіз вимог до ПЗ	07.04.2023	ПП*
4.	Розробка архітектури ПЗ	21.04.2023	
5.	Розробка проекту ПЗ	05.05.2023	
6.	Реалізація та тестування ПЗ	12.05.2023	
7.	Підготовка розділу Результати розробки ПЗ	17.05.2023	
8.	Підготовка розділу з охорони праці	19.05.2023	ПП*
9.	Підготовка розділу ВИСНОВКИ	24.05.2023	
10.	Оформлення списку використаних джерел та додатків	26.05.2023	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	29.05.2023	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
12.	Підготовка презентації та доповіді	02.06.2023	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	05.06.2023	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	12.06.2023	

* - за результатами переддипломної практики (ПП), яка була з 01.02.2023 до 19.03.2023 р.

Студент _____ Бєлоножко Н. Ю.
(підпис) (ПІБ)

Керівник роботи _____ Пухалевич А. В.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена вирішенню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці вебсервісу для моніторингу стану вебсайтів.

Об’єктом розробки є вебсервіс для моніторингу стану вебсайтів. Проведено аналіз практичної задачі інженерії програмного забезпечення з розробки вебсервісу для моніторингу вебсайтів та проаналізовано існуючі способи моніторингу стану вебсайтів. Здійснено постановку задачі на розробку вебсервісу для моніторингу стану вебсайтів. Розроблено та випробувано вебсервіс. Проведено опис аспектів охорони праці при роботі з екранними пристроями.

Програмне забезпечення для вебсервісу було виконано за допомогою середовища розробки PhpStorm, мови програмування Php та СКБД MariaDB.

Кваліфікаційна робота викладена на 106 сторінках друкованого тексту, містить 34 рисунки, 15 таблиць, список використаних джерел із 9 найменувань, та 5 додатків.

ABSTRACT

The qualification work is dedicated to solving a practical problem in software engineering, characterized by complexity and uncertainty of conditions, namely the development of a web service for monitoring the state of websites.

The object of development is a web service for monitoring the state of websites. An analysis of the practical problem in software engineering related to the development of a web service for website monitoring has been conducted. Existing methods of website monitoring have been analyzed. The task of developing a web service for monitoring the state of websites has been formulated. The web service has been developed and tested. Descriptions of occupational safety aspects when working with display devices have been provided.

The software for the web service was developed using the PhpStorm integrated development environment, the Php programming language, and the MariaDB database management system.

The qualification work consists of 106 pages of printed text and includes 34 figures, 15 tables, a reference list with 9 sources, and 5 appendices..

ЗМІСТ

ВСТУП

Кваліфікаційна робота присвячена вирішенню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці вебсервісу для моніторингу стану вебсайтів.

Для того щоб вебсайт був ефективним і приносив максимальний дохід, необхідно забезпечити його безперебійну роботу. Якщо вебсайт часто працює з перебоями або має інші проблеми з функціональністю, це може призвести до втрати клієнтів та зменшення доходів власника сайту. Тому важливо не тільки створити вебсайт, але й регулярно проводити його моніторинг, щоб він завжди працював належним чином.

Моніторинг стану вебсайтів - це процес відстеження стану вебсайтів з метою своєчасного виявлення проблем в роботі вебсайту. Ця задача може бути вирішена шляхом автоматизації моніторингу стану вебсайтів за допомогою спеціалізованих утиліт (ping [1], curl [2] та інші).

Метою кваліфікаційної роботи є розробка вебсервісу для моніторингу стану вебсайтів.

Потрібно вирішити наступні завдання, для того, щоб досягнути поставленої мети:

1) провести аналіз практичної задачі інженерії програмного забезпечення з розробки вебсервісу для моніторингу стану вебсайтів;
2) провести аналіз існуючих способів моніторингу стану вебсайтів;
3) провести аналіз методів розробки вебсайтів;
4) створити ескізний проєкт вебсервісу для моніторингу вебсайтів;
5) створити технічний проєкт вебсервісу для моніторингу вебсайтів;
6) створити робочий проєкт вебсервісу для моніторингу вебсайтів;
7) провести тестування вебсервісу для моніторингу вебсайтів.

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З РОЗРОБКИ ВЕБСЕРВІСУ ДЛЯ МОНІТОРИНГУ СТАНУ ВЕБСАЙТІВ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз практичної задачі інженерії програмного забезпечення з розробки вебсервісу для моніторингу стану вебсайтів

Кваліфікаційна робота присвячена вирішенню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці вебсервісу для моніторингу стану вебсайтів.

Ця задача може бути вирішена шляхом автоматизації моніторингу стану вебсайтів за допомогою спеціалізованих утиліт (ping [1], curl [6] та інші).

Вирішення вказаної задачі дозволить власникам вебсайтів постійно моніторити їх стан, а саме слідкувати за тим щоб:

- підключення вебсерверу до інтернету не було порушене;
- звичайний HTTP GET запит до головної сторінки вебсайту не повертав код стану, який відповідає помилці;
- сертифікат SSL був придатний;
- індексація не була заборонена для пошукових систем метатегом robots або заголовком HTTP-відповіді;
- файл robots.txt був налаштований правильно та не забороняв індексацію.

Підключення вебсерверу до інтернету не є порушеним, якщо хост-комп’ютер, до якого потрібно отримати доступ, працює. При цьому хост є доступним за IP адресою, призначеною для доменного імені, і може приймати запити [1].

Якщо вебсайт буде повертати код стану, який відповідає помилці, користувачі не зможуть відвідувати його, або вебсайт може працювати некоректно. Код стану HTTP - частина першого рядка відповіді сервера при

запитах за протоколом HTTP. Він є цілим числом з трьох цифр. Перша цифра вказує на клас стану. За кодом відповіді зазвичай йде відділена пробілом пояснювальна фраза англійською мовою, яка пояснює людині причину саме такої відповіді. По коду відповіді клієнт дізнається про результати його запиту і визначає, що йому робити далі [2].

SSL-сертифікат - це електронний документ, який забезпечує захищеність даних, які передаються між сервером та клієнтом за допомогою протоколу HTTPS. Сертифікат підтверджує ідентичність сервера та гарантує конфіденційність передачі даних. Для забезпечення безпеки важливо мати придатний SSL-сертифікат [3].

Noindex – це правило, яке задається за допомогою тегу <meta name="robots"> або заголовку X-Robots-Tag HTTP-відповіді, і забороняє індексацію контенту пошуковими системами, підтримуючими noindex, наприклад Google. Якщо індексація буде заборонена, нові користувачі не зможуть побачити вебсайт в результатах пошуку [4].

У файлі robots.txt містяться інструкції, котрі кажуть пошуковим роботам, які URL на сайті їм дозволено обробляти. Якщо некоректно налаштувати цей файл, можна заборонити індексацію усього вебсайту. Також має місце людський фактор, можна випадково не побачити помилку у файлі [5].

1.2 Аналіз існуючих способів для моніторингу стану вебсайтів

Перевірити, що підключення вебсерверу до інтернету не було порушене можна за допомогою програми Ping. Ping - це службова комп'ютерна програма, призначена для перевірки з'єднань в мережах на основі TCP/IP. Вона дозволяє визначити час між відправленням запиту й одержанням відповіді. Програма ping є одним з основних діагностичних засобів у мережах TCP/IP і входить у постачання всіх сучасних мережевих операційних систем [1].

На рисунку 1.1 представлений результат перевірки доступності вебсайту за допомогою команди ping у консолі Linux. Відповідь сервера свідчить що підключення вебсерверу до інтернету не є порушеним.

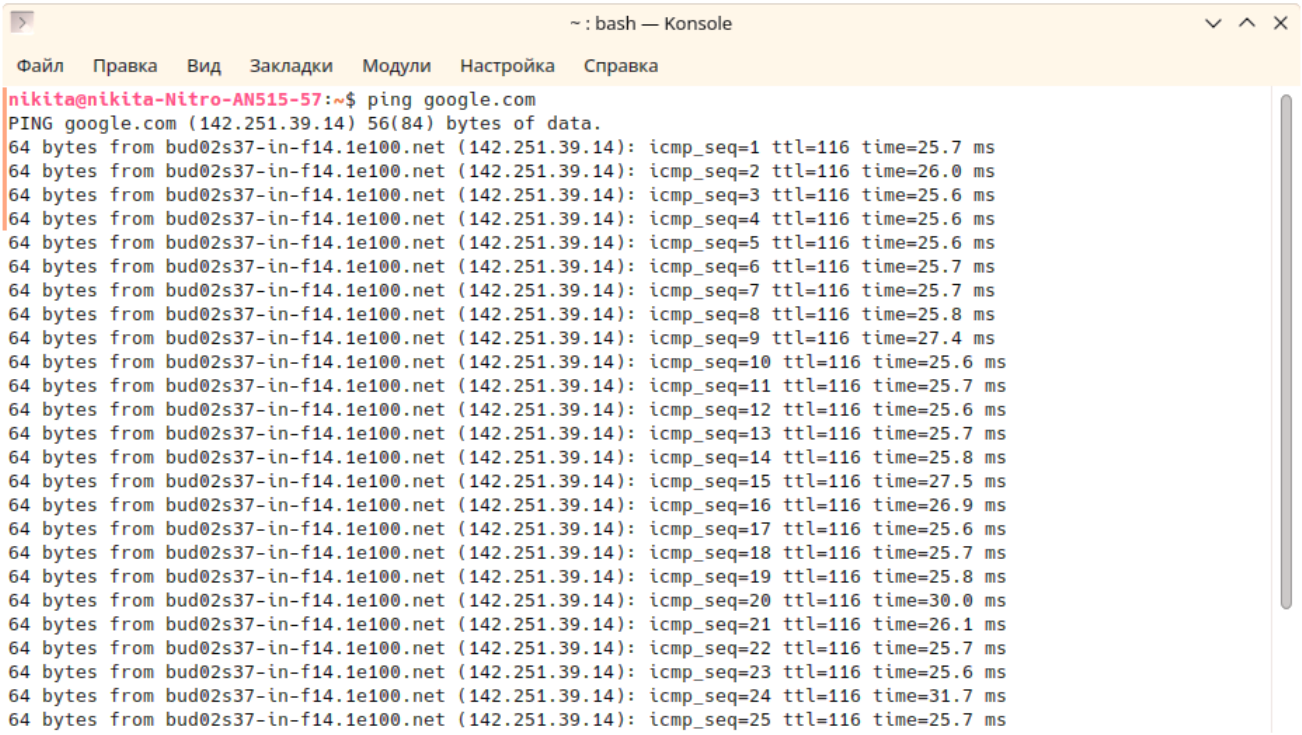


Рисунок 1.1 - Приклад роботи програми ping

Перевірити код стану, який повернув HTTP запит до вебсайту, можна за допомогою програми Curl. Це службова програма, призначена для взаємодії з серверами за допомогою різних протоколів, в тому числі протоколу HTTP.

На рисунку 1.2 показана перевірка коду стану, який повернув HTTP запит, за допомогою програми Curl. На знімку екрану видно, що код стану відповіді становить 200. Це свідчить про те, що запит був успішно оброблений сервером [2].

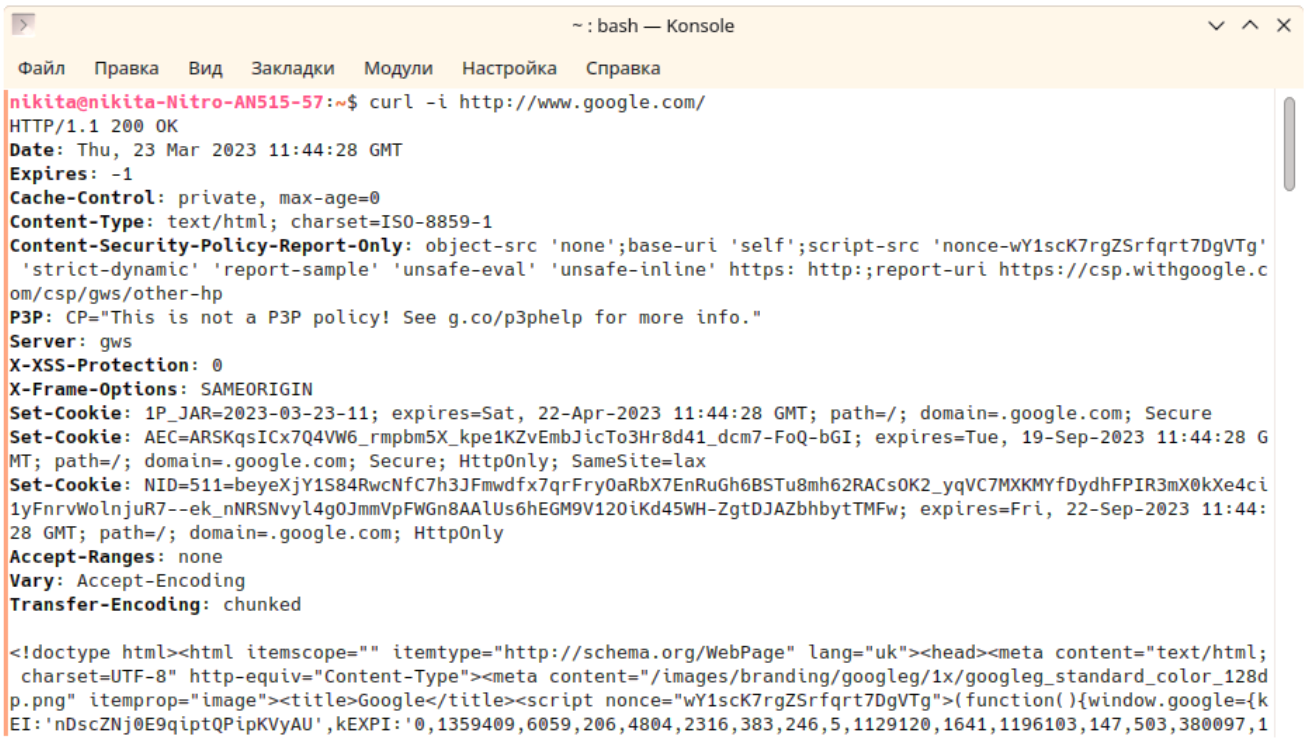


Рисунок 1.2 – Перевірка HTTP статусу за допомогою програми curl

Перевірити, що robots.txt був налаштований правильно, та не забороняє індексацію, можна за допомогою онлайн-валідатора на сайті technicalseo.com.

На рисунку 1.3 показано результати валідації файлу robots.txt за допомогою онлайн-валідатора. На знімку екрана видно, що файл robots.txt пройшов перевірку на відповідність синтаксису та містить допустимі директиви. Цей крок є важливим для забезпечення правильної індексації сайту пошуковими системами.

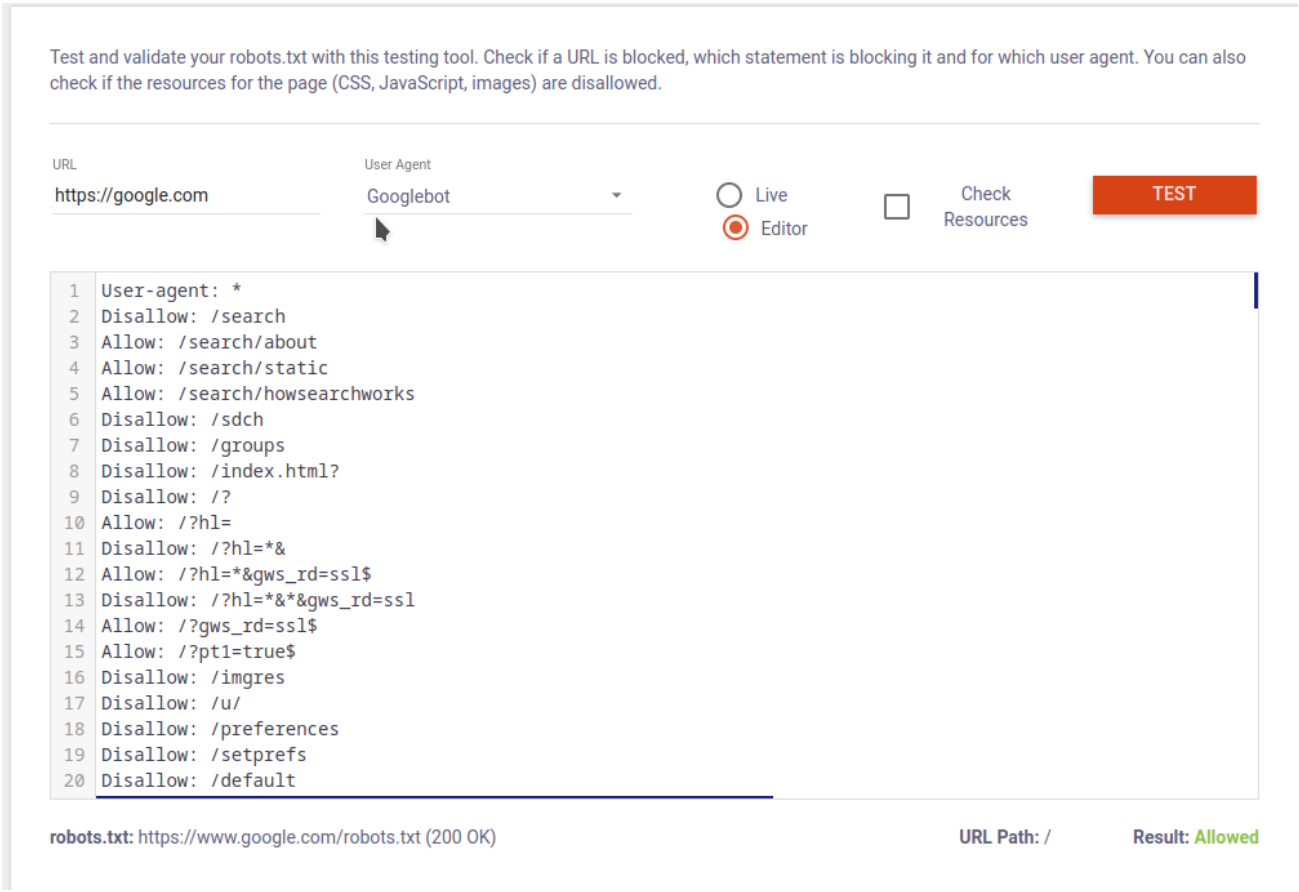


Рисунок 1.3 – Валідація robots.txt онлайн валідатором

Перевірки за допомогою наведених способів для моніторингу вебсайтів повинен робити вебмайстер у ручному режимі. Тому ці перевірки неможливо виконувати досить часто та регулярно. Швидкість ручної перевірки є низькою. Історія змін не зберігається при ручній перевірці. Також не можна виключати людський фактор, коли помилки в роботі вебсайту залишаться непоміченими.

Щоб усунути вказані недоліки, було вирішено розробити вебсервіс для моніторингу стану вебсайтів у автоматичному режимі з можливостю реєстрації користувачів, що дозволить їм отримувати повідомлення про зміну стану вебсайту.

1.3 Постановка задачі на розробку вебсервісу для моніторингу стану вебсайтів

Для забезпечення ефективного моніторингу стану вебсайтів було вирішено розробити вебсервіс, який дозволить користувачам зареєструвати свої вебсайти, автоматизувати процес моніторингу їх стану та отримувати повідомлення про зміни стану вебсайтів. Це дозволить вебмайстрам та власникам вебсайтів оперативно реагувати на проблеми, що виникають на сайтах.

В рамках постановки задачі були визначені наступні функціональні вимоги до вебсервісу:

- Реєстрація користувача.
- Авторизація користувача.
- Вихід із системи.
- Моніторинг стану вебсайту за запитом.
- Додавання вебсайту для моніторингу.
- Моніторинг стану вебсайту за розкладом.
- Оповіщення про зміну стану вебсайту.
- Збереження оповіщень про зміну стану вебсайту.
- Перегляд оповіщень про зміну стану вебсайту.
- Збереження тривалості підключення до вебсайту.
- Перегляд тривалості підключення до вебсайту.

Вимоги до складу та параметрів технічних засобів вебсерверу:

- операційна система: Linux;
- процесор: Intel Core I5-11400, 4.5GHZ, 4 ядра та більше;
- інтернет: мінімум 1Гбіт/с;
- оперативна пам’ять: 16 Гб та більше;
- жорсткий диск: для стабільної роботи вебсервісу потрібно мінімум 500 Мб вільного простору.

Вимоги до вхідних та вихідних даних:

- для реєстрації користувача вхідними даними є прізвище, ім’я, пароль та електронна пошта. Вихідними даними є запис у базі про даних про користувача;

- для авторизації користувача вхідними є електронна пошта та пароль. Вихідними даними є номер сесії, створеної сервером для користувача;

- для додавання вебсайту для моніторингу вхідними даними є адреса вебсайту. Вихідними даними є запис у базі даних про вебсайт;

- для моніторингу вебсайту вхідними даними є запис у базі даних про вебсайт. Вихідними даними є пінг, HTTP статус, інформація щодо індексації вебсайту, інформація щодо валідації файлу robots.txt, актуальність SSL-сертифікату, та запис у базу даних щодо значень стану вебсайту, якщо користувач авторизований;

- для оповіщення про зміну стану вебсайту вхідними даними є електронна адреса, адреса вебсайту та пінг, HTTP статус, інформація щодо індексації вебсайту, інформація щодо валідації файлу robots.txt, актуальність SSL-сертифікату. Вихідними даними є електронне повідомлення.

2 РОЗРОБКА ВЕБСЕРВІСУ ДЛЯ МОНІТОРИНГУ СТАНУ ВЕБСАЙТІВ

2.1 Аналіз вимог до вебсервісу для моніторингу стану вебсайтів

В даному підрозділі створено модель та специфікації варіантів використання.

2.1.1 Модель варіантів використання

Було виявлено три актори вебсервісу для моніторингу стану вебсайтів: Клієнт, Гість та Час.

Гість – це неавторизований користувач вебсервісу. Він може перевірити стан вебсайту.

Клієнт – це авторизований користувач вебсервісу, який має можливість отримувати повідомлення на електронну пошту про зміни стану вебсайтів, які він додав.

Час – неявний актор який ініціює виконання прецедентів, що виконуються за розкладом.

Акторів вебсервісу зображено на рисунку 2.1.

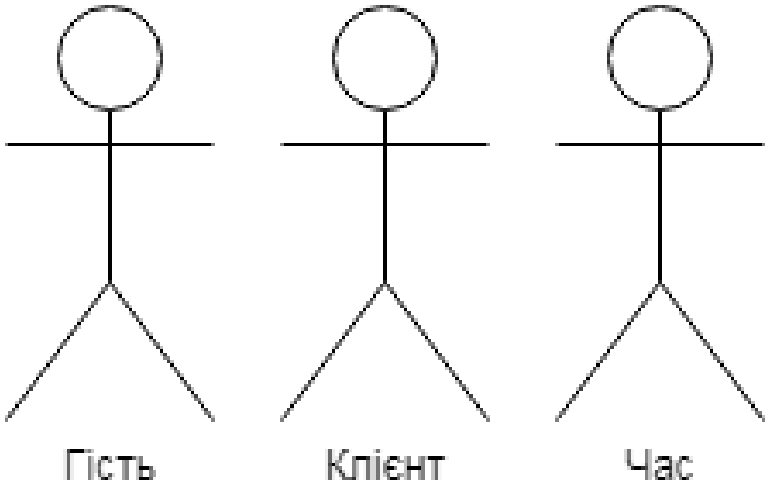


Рисунок 2.1 – Актори вебсервісу для моніторингу стану вебсайтів

Проаналізувавши особливості функціональних можливостей та вимог до вебсервісу, було відібрано та уточнено наведені нижче варіанти використання:

- Реєстрація користувача.
- Авторизація користувача.
- Вихід із системи.
- Моніторинг стану вебсайту за запитом.
- Додавання вебсайту для моніторингу.
- Моніторинг стану вебсайту за розкладом.
- Оповіщення про зміну стану вебсайту.
- Збереження оповіщень про зміну стану вебсайту.
- Перегляд оповіщень про зміну стану вебсайту.
- Збереження тривалості підключення до вебсайту.
- Перегляд тривалості підключення до вебсайту.

Усі варіанти використання наведені на діаграмі варіантів використання на рисунку 2.2.

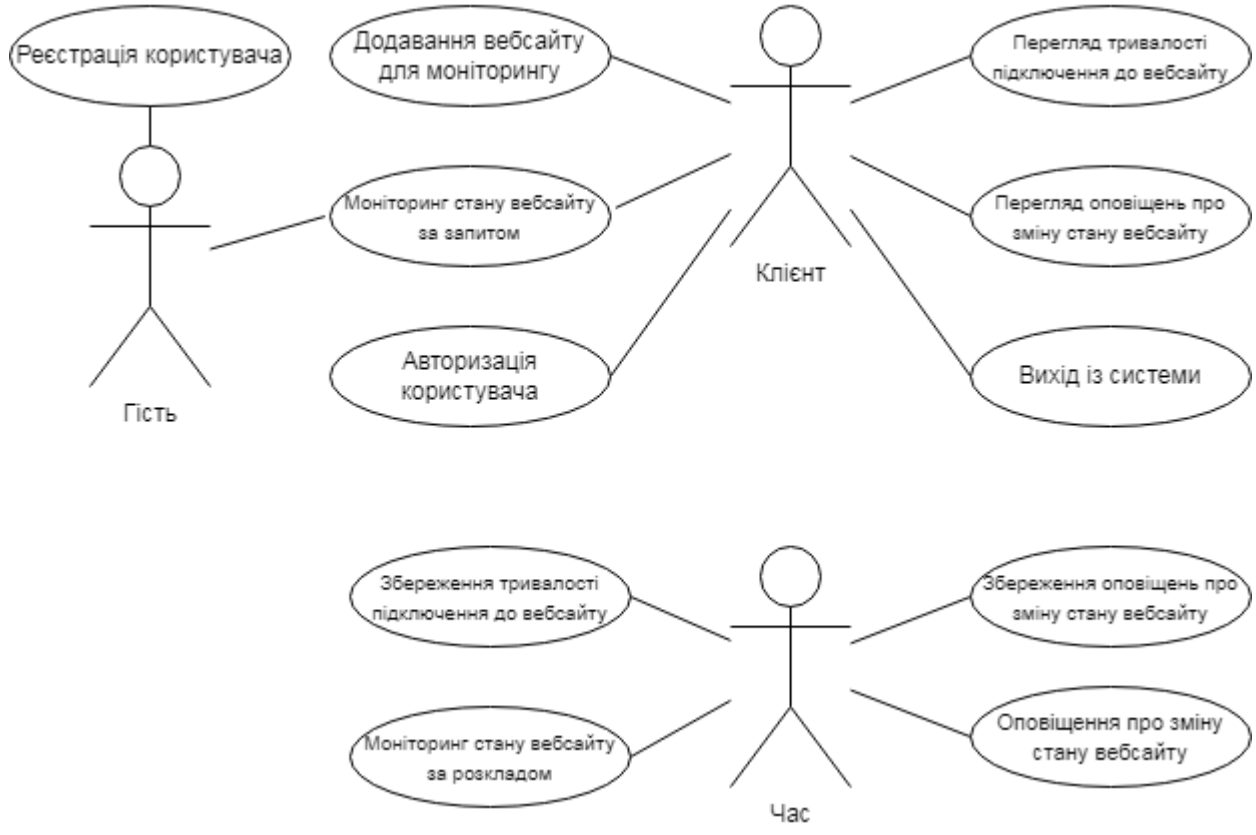


Рисунок 2.2 – Діаграма варіантів використання

Після детального аналізу діаграми варіантів використання, ми розглянемо кожен з них окремо, надавши докладний опис. В таблиці 2.1 представлено усі варіанти використання.

Таблиця 2.1 – Варіанти використання та їх короткий опис

Назва	Формулювання
1	2
Реєстрація користувача	Дозволяє створити профіль користувача електронну адресу та пароль користувача
Авторизація користувача	Дозволяє авторизуватися вказавши електронну адресу та пароль користувача
Вихід з системи	Дозволяє скасувати авторизацію на вебсервісі
Моніторинг стану вебсайту за запитом	Дозволяє переглянути стан вебсайту, вказавши його URL адресу
Додавання вебсайту для моніторингу	Дозволяє додати сайт для моніторингу за розкладом
Моніторинг стану вебсайту за розкладом	Моніторинг стану вебсайту за розкладом
Оповіщення про зміну стану вебсайту	Дозволяє серверу надсилати оповіщення електронним повідомленням про зміну стану вебсайту
Збереження оповіщень про зміну стану вебсайту	Дозволяє зберігати історію змін у базі даних
Перегляд оповіщень про зміну стану вебсайту	Дозволяє користувачу переглянути зміни у стані вебсайту, що був доданий раніше
Збереження тривалості підключення до вебсайту	Дозволяє зберігати дані про тривалість підключення до вебсайту
Перегляд тривалості підключення до вебсайту	Дозволяє побачити користувачу графік тривалості підключення до вебсайту

2.1.2 Специфікації варіантів використання

З метою уточнення вимог було створено детальні специфікації варіантів використання.

Специфікації прецедентів наведено у таблицях 2.2 –2.11.

Таблиця 2.2 – Прецедент використання «Реєстрація користувача»

Назва прецеденту	**Реєстрація користувача**
Ідентифікатор	1
Короткий опис	Даний варіант використання дозволяє клієнту зареєструватися на сайті
Головний актор	Гість
Другорядні актори	Немає
Базовий потік	1. Користувач натискає на кнопку реєстрації. 2. Користувач введе дані для реєстрації. 3. Вебсервіс авторизує нового користувача.
Передумови	Користувач повинен бути неавторизованим
Постумови	Новий користувач повинен стати авторизованим
Альтернативний потік	1. Користувач з введенною електронную адресою вже існує. 2. Користувачу відображається повідомлення про помилку.

Таблиця 2.3 – Прецедент використання «Авторизація користувача»

Назва прецеденту	**Авторизація користувача**
Ідентифікатор	2
Короткий опис	Даний варіант використання дозволяє клієнту авторизуватися на сайті
Головний актор	Клієнт
Другорядні актори	Немає

Продовження таблиці 2.3

Базовий потік	1. Користувач натискає на кнопку авторизація. 2. Користувач введе дані для авторизації. 3. Вебсервіс авторизує користувача.
Передумови	Немає
Постумови	Користувач повинен стати авторизованим
Альтернативні потоки	1. Користувач з такими даними не був знайден. 2. Відображення помилки.

Таблиця 2.4 – Прецедент використання «Моніторинг стану вебсайту за запитом»

Назва прецеденту	**Моніторинг стану вебсайту за запитом**
Ідентифікатор	3
Короткий опис	Даний варіант використання представляє користувачу докладну інформацію щодо стану вебсайту.
Головні актори	Клієнт, Гість
Другорядні актори	Немає
Базовий потік	1. Користувач ввів посилання до вебсайту. 2. Користувач натиснув кнопку відправки форми.
Передумови	Немає
Постумови	Відображення сторінки інформації щодо стану вебсайту
Альтернативні потоки	1. Вебсайт за такою адресою не знайдено. 2. Відображення помилки.

Таблиця 2.5 – Прецедент використання «Додавання вебсайту до моніторингу»

Назва прецеденту	**Додавання вебсайту до моніторингу**
Ідентифікатор	4
Короткий опис	Даний варіант використання дозволяє додати URL адресу вебсайту для моніторингу за розкладом
Головний актор	Клієнт
Другорядні актори	Немає
Базовий потік	1. Користувач авторизувався 2. Користувач ввід URL адресу вебсайту
Передумови	Користувач повинен бути авторизованим
Постумови	Вебсайт повинен бути збереженим у БД
Альтернативні потоки	1. Вебсайт за такою адресою не знайдено. 2. Відображення помилки..

Таблиця 2.6 – Прецедент використання «Моніторинг стану вебсайту за розкладом»

Назва прецеденту	**Моніторинг стану вебсайту за розкладом**
Ідентифікатор	5
Короткий опис	Даний варіант використання дозволяє вебсервісу здійснювати фоновий моніторинг вебсайтів клієнтів
Головни актор	Час
Другорядні актори	Клієнт
Базовий потік	1. Перевірка тривалості підключення до вебсайту. 2. Перевірка HTTP статусу вебсайту. 3. Перевірка заборони індексації. 4. Перевірка файлу robots.txt. 5. Перевірка придатності SSL сертифікату.

Продовження таблиці 2.6

Передумови	Настав час перевірки
Постумови	Відправлення оповіщень якщо змінився стан
Альтернативні потоки	Немає

Таблиця 2.7 – Прецедент використання «Оповіщення про зміну стану вебсайту»

Назва прецеденту	**Оповіщення про зміну стану вебсайту**
Ідентифікатор	6
Короткий опис	Даний варіант використання дозволяє відправляти електронні листи про зміни стану вебсайту, не потребуючі інших дій від клієнта
Головні актори	Час
Другорядні актори	Клієнт
Базовий потік	1. Сгенерувати повідомлення. 2. Відправити повідомлення на електронну адресу.
Передумови	Стан вебсайту змінився
Постумови	Відправлено електронного листа про зміну стану вебсайту
Альтернативні потоки	Немає

Таблиця 2.8 – Прецедент використання «Збереження оповіщень про зміну стану вебсайту»

Назва прецеденту	**Збереження оповіщень про зміну стану вебсайту**
Ідентифікатор	7
Короткий опис	Даний варіант використання дозволяє вебсервісу зберігати історію змін
Головний актор	Час
Другорядні актори	Клієнт

Продовження таблиці 2.8

Базовий потік	1. Відправити оповіщення. 2. Зберегти оповіщення у БД.
Передумови	Було створено оповіщення
Постумови	Збережене оповіщення у БД
Альтернативні потоки	Немає

Таблиця 2.9 – Прецедент використання «Перегляд оповіщень про зміну стану вебсайту»

Назва прецеденту	**Перегляд оповіщень про зміну стану вебсайту**
Ідентифікатор	8
Короткий опис	Даний варіант використання дозволяє клієнту побачити зміни в стані вебсайту в електронному листі
Головний актор	Клієнт
Другорядні актори	Немає
Базовий потік	1. Клієнт раніше додав вебсайт до моніторингу. 2. Клієнт відкрив сторінку перегляду оповіщень.
Передумови	Відбулися зміни в стані вебсайту
Постумови	Немає
Альтернативні потоки	Немає

Таблиця 2.10 – Прецедент використання «Збереження тривалості підключення до вебсайту»

Назва прецеденту	**Збереження тривалості підключення до вебсайту**
Ідентифікатор	9

Продовження таблиці 2.10

Короткий опис	Даний варіант використання дозволяє зберігати інформацію щодо тривалості підключення до вебсайту для подальшої побудови графіку
Головний актор	Час
Другорядні актори	Клієнт, Гість
Базовий потік	1. Будь-який клієнт раніше додавав вебсайт до моніторингу. 2. Користувач виконав моніторинг вебсайту за запитом.
Передумови	Цей вебсайт вже раніше перебував у фоновому моніторингу будь-якого клієнту
Постумови	Збереження результату у БД
Альтернативні потоки	Немає

Таблиця 2.11 – Прецедент використання «Перегляд тривалості підключення до вебсайту»

Назва прецеденту	**Перегляд тривалості підключення до вебсайту**
Ідентифікатор	10
Короткий опис	Даний варіант використання дозволяє користувачу побачити графік змін швидкості вебсайту
Головні актори	Гість, Клієнт
Другорядні актори	Час
Базовий потік	1. Користувач ввів посилання до вебсайту. 2. Користувач натиснув кнопку відправки форми.
Передумови	Цей вебсайт вже раніше перебував у фоновому моніторингу будь-якого клієнту
Постумови	Немає

Продовження таблиці 2.11

Альтернативні потоки	1. Вебсайт не був раніше добавлений клієнтом. 2. Перегляд актуальної тривалості підключення. 3. Порожній графік тривалості підключення.

2.2 Архітектура вебсервісу для моніторингу стану вебсайтів

Більшість мов програмування надають достатньо технологій та функціональних можливостей для створення програмного забезпечення на основі архітектури MVC.

Модель – це компонент, що використовується для опису даних, які використовуються в програмному забезпеченні та їх логіки. У моделі зазвичай використовується клас з атрибутами, і вона не має логіки взаємодії з користувачем через користувацький інтерфейс. Робота з моделями найбільш ефективна при використанні ORM-фреймворків для баз даних.

Представлення – відповідає за користувацький інтерфейс, через який користувач може взаємодіяти з програмним забезпеченням. Представлення не повинно мати майже жодної логіки для управління даними, крім їх відображення.

Контролер – це головний компонент у схемі MVC. Він відповідає за зв'язок між програмним забезпеченням, користувачем, моделлю та представленням. Контролер містить усю логіку, яка потрібна для обробки користувацьких запитів.

2.3 Проєкт вебсервісу для моніторингу стану вебсайтів

2.3.1 Ескізний проєкт вебсервісу для моніторингу вебсайтів

На основі проведеного аналізу основних вимог було розроблено ескізний проєкт. Для розробки проєкту використано мову моделювання UML з метою детального опису.

На даному етапі проєктування програмного забезпечення було використано об'єктно-орієнтовану методологію та створено наступні моделі:

- Діаграми діяльності, які ілюструють послідовність дій та взаємодію акторів у системі;
- Прототип інтерфейсу користувача, що демонструє вигляд та функціональні можливості програми на початковому етапі розробки.

Діаграми діяльності

Діаграми діяльності основний варіантів використання наведені на рисунках 2.3 – 2.4.

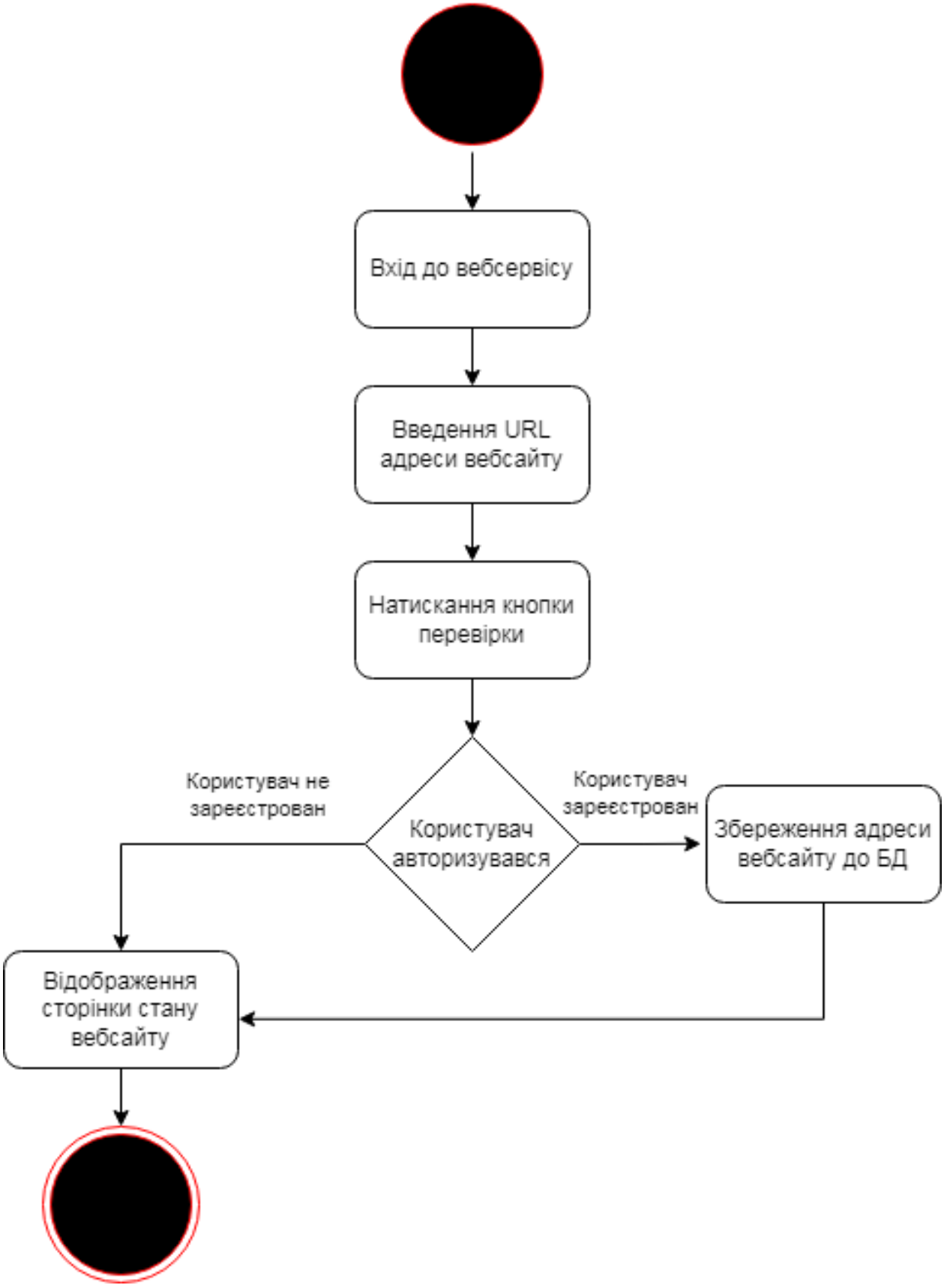


Рисунок 2.3 – Діаграма діяльності для варіанту використання «Моніторинг стану вебсайту за запитом»

Рисунок 2.4 – Діаграма діяльності для варіанту використання «Моніторинг стану вебсайту за розкладом»

Прототип інтерфейсу користувача

З метою уточнення кінцевого інтерфейсу вебсервісу для моніторингу стану вебсайтів, аналізу елементів інтерфейсу та необхідних варіантів для використання акторами, а також їх взаємозв'язку, був розроблений прототип інтерфейсу користувача.

Загалом, структура вмісту вебсторінок була поділена на три рівні для оптимального розміщення інформації:

- верхній рівень (заголовок сторінки) містить навігаційне меню, кнопки реєстрації або авторизації;

- середній рівень (контент сторінки або тіло) містить найважливішу інформацію, яку користувач буде переглядати часто, або функціональну частину сайту, з якою користувач буде активно взаємодіяти;

- нижній рівень (нижній колонтитул сторінки) містить контактну інформацію підтримки вебсайту, розробників, представників компанії тощо, а також посилання на сторінку з додатковою інформацією про сайт або значок копірайту.

З огляду на звичайний шлях читання зліва направо, першим елементом, що розміщується у голові сторінки, є назва вебсайту. Далі слідує основний функціонал, такий як авторизація та реєстрація користувача.

На головній сторінці вебсайту слід забезпечити наступні елементи для зручного використання. На головній сторінці слід розташувати поле введення URL-адреси, де користувач може ввести URL-адресу вебсайту, стан якого він бажає перевірити. Це дозволить швидко та зручно виконувати моніторинг стану різних вебсайтів. Для активації процесу перевірки стану вебсайту після введення URL-адреси на головній сторінці повинна бути присутня кнопка відправки. Користувач зможе натиснути цю кнопку, щоб запустити процес моніторингу. Забезпечення наявності цих елементів на головній сторінці сприятиме зручності використання та навігації користувачами сайту.

Прототип головної сторінки сторінка вебсайту, яку переглядає користувач після переходу на сайт, наведена на рисунку 2.5.

Рисунок 2.5 – прототип головної сторінки вебсайту

Для отримання повного доступу до основних функцій програмного забезпечення вебсайту необхідно авторизуватися за допомогою облікового запису. Щоб авторизуватися, необхідно перед цим мати зареєстрований обліковий запис.

Було прийнято рішення розмістити окрему сторінку для реєстрації користувачів, яка буде відкриватися окремо від головної сторінки. На цій сторінці буде розміщена форма реєстрації, до якої користувач буде мати доступ через натискання на кнопку реєстрації.

Прототип сторінки реєстрації користувача зображено на рисунку 2.6.

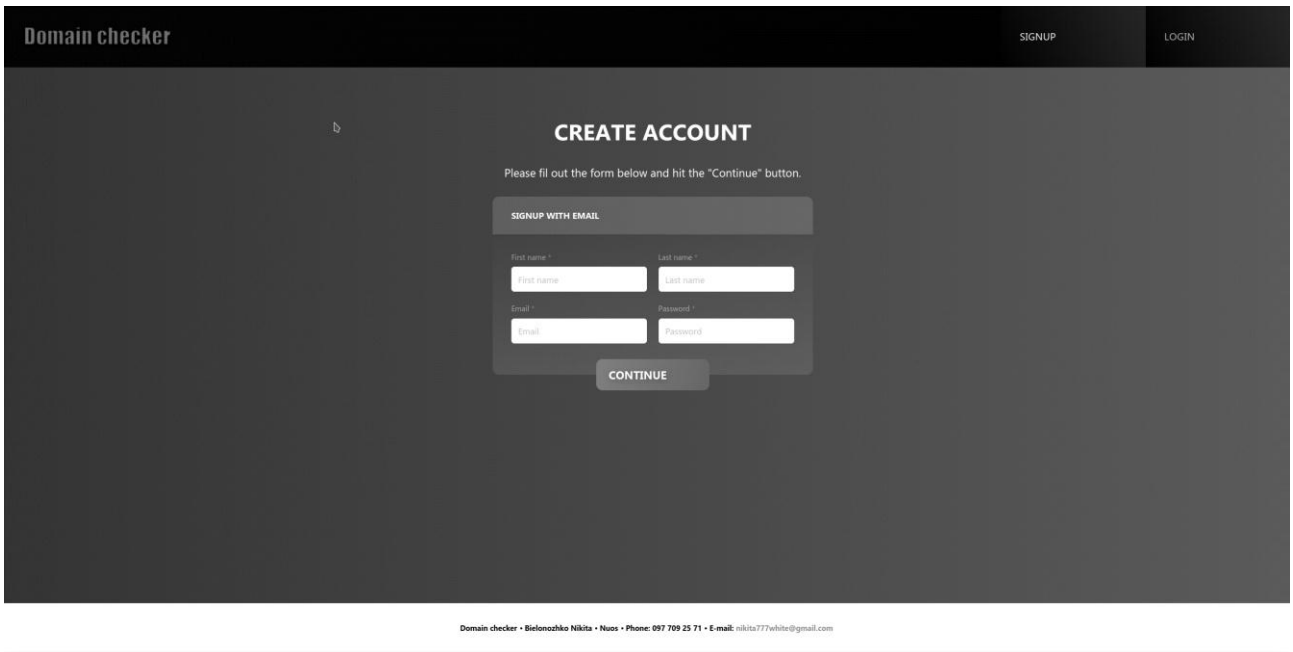


Рисунок 2.6 – прототип сторінки реєстрації користувача

Сторінка авторизації була розроблена з врахуванням принципу, який використовувався для вікна реєстрації, щоб забезпечити єдність структури сайту для користувача. Прототип сторінки з формою авторизації зображена на рисунку 2.7.

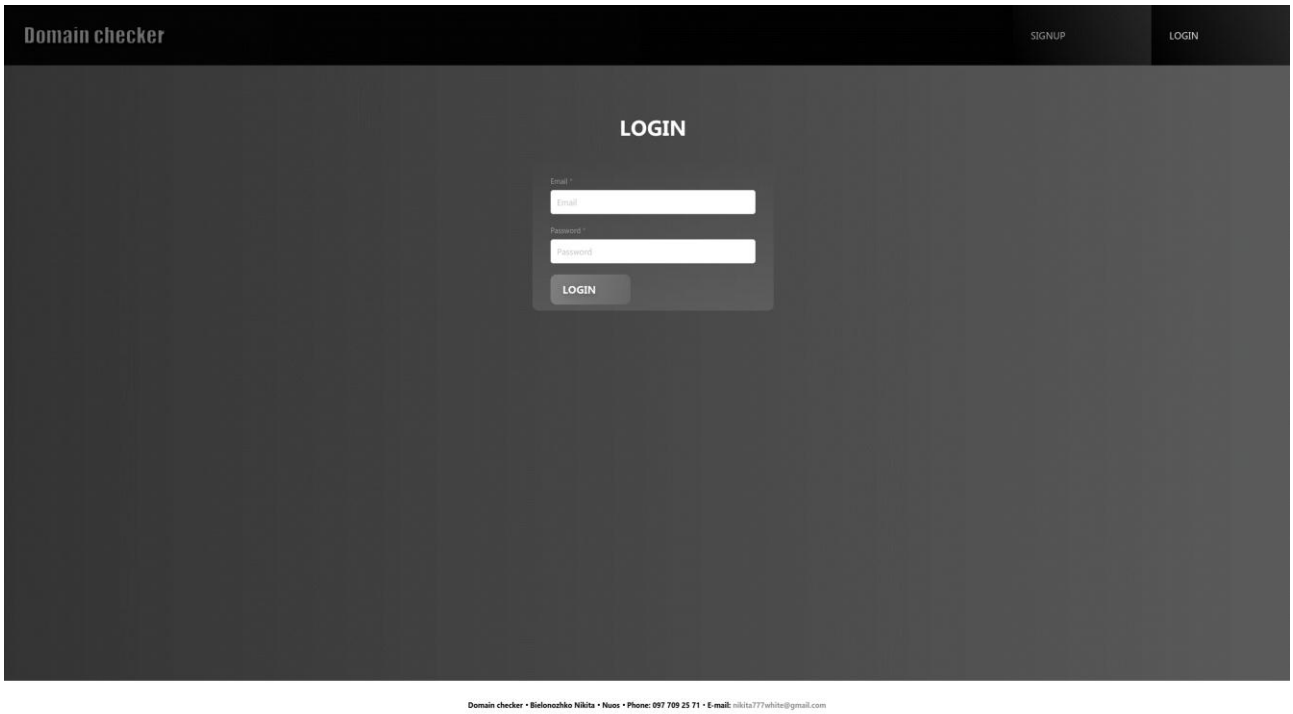


Рисунок 2.7 – прототип сторінки авторизації користувача

Для того щоб моніторити сайт, було вирішено додати строку введення HTTP адреси вебсайту у центральній верхній частині головної сторінки. Це

дозволить користувачам зручно та швидко вказувати адресу вебсайту, який вони бажають моніторити.

Після введення URL адреси вебсайту користувача буде переведено до сторінки перегляду стану вебсайту.

Прототип сторінки моніторингу стану вебсайту наведено на рисунках 2.8 - 2.9.

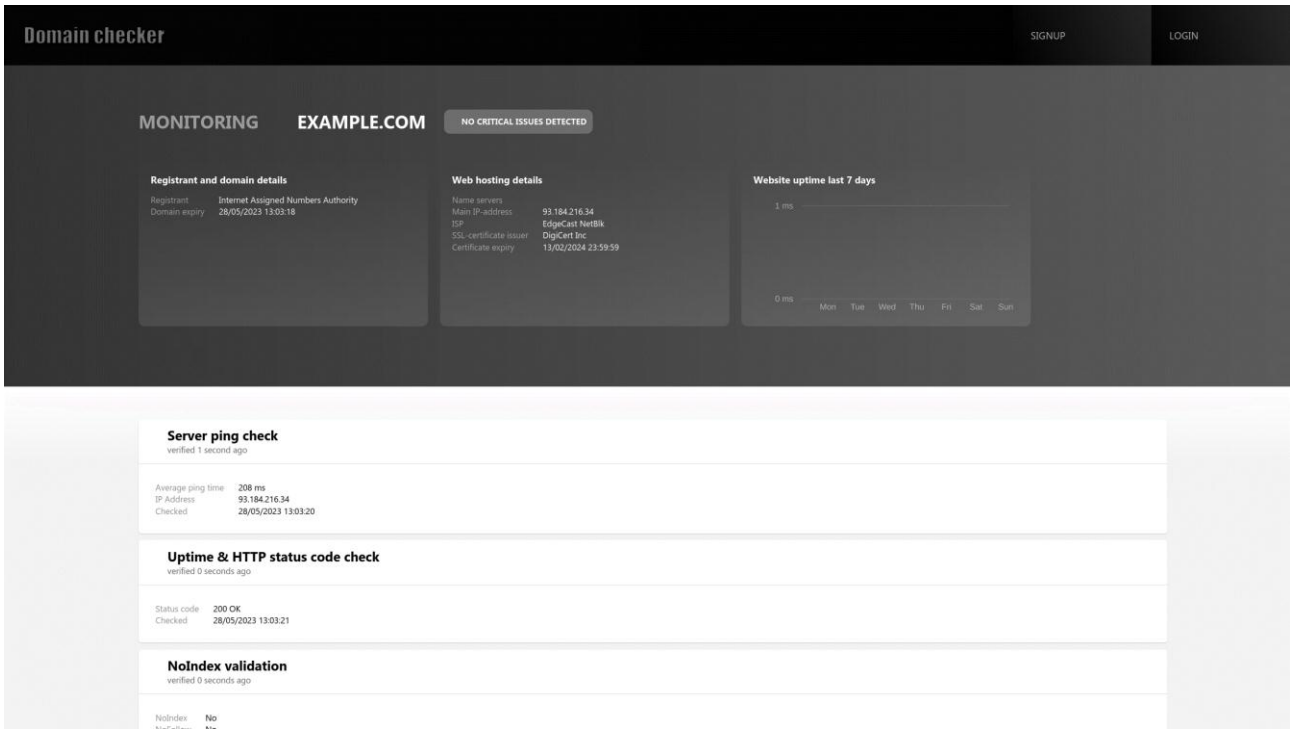


Рисунок 2.8 – прототип сторінки моніторингу стану вебсайту

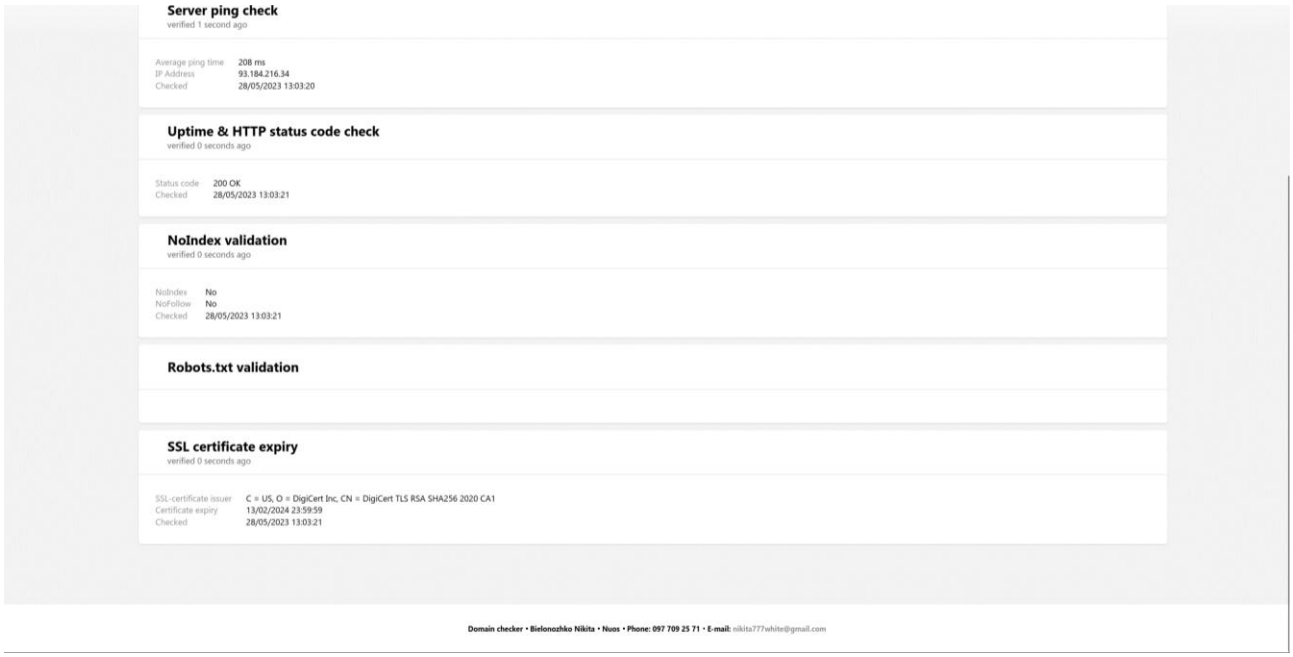


Рисунок 2.9 – продовження прототипу сторінки моніторингу стану вебсайту

На сторінці профілю користувача знаходяться лічильники сайтів, котрі не мають проблем з певним пунктом, або лічильники вебсайтів з проблемами, якщо є вебсайти, котрі мають проблеми. Також нижче знаходиться список вебсайтів, за змінами статусу яких користувач слідкує, та кнопки для перегляду стану певного вебсайту та видалення зі списку для автоматичного моніторингу.

Прототип сторінки профіля користувача приведено на рисунку 2.11.

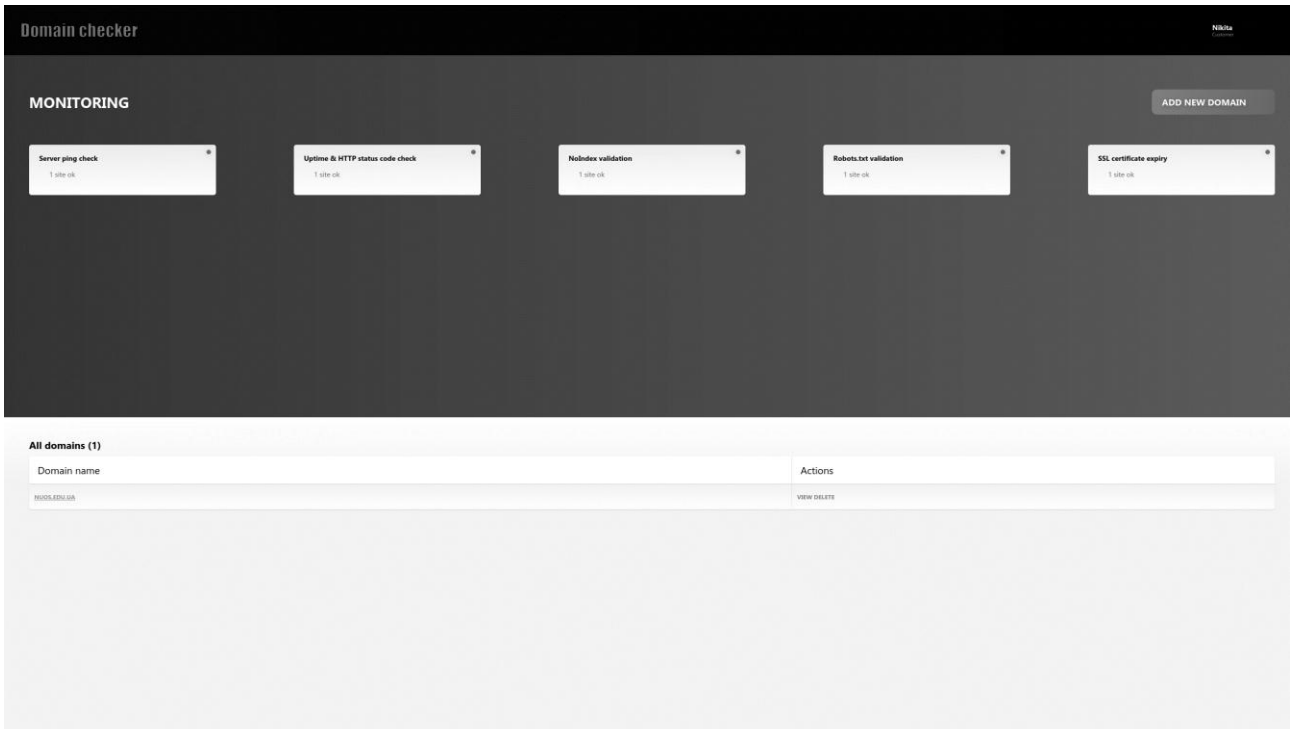


Рисунок 2.10 – прототип сторінки профіля користувача

2.3.2 Технічний проєкт вебсервісу для моніторингу вебсайтів

На основі розробки ескізного проєкту, було створено технічний проєкт вебсервісу. Під час розробки були створені статична та динамічна моделі вебсервісу, а також логічна модель даних. Після аналізу класів програмного забезпечення була підготовлена специфікація класів.

Статична модель вебсервісу

Статична модель у вигляді діаграми класів представлена на рисунку 2.11.

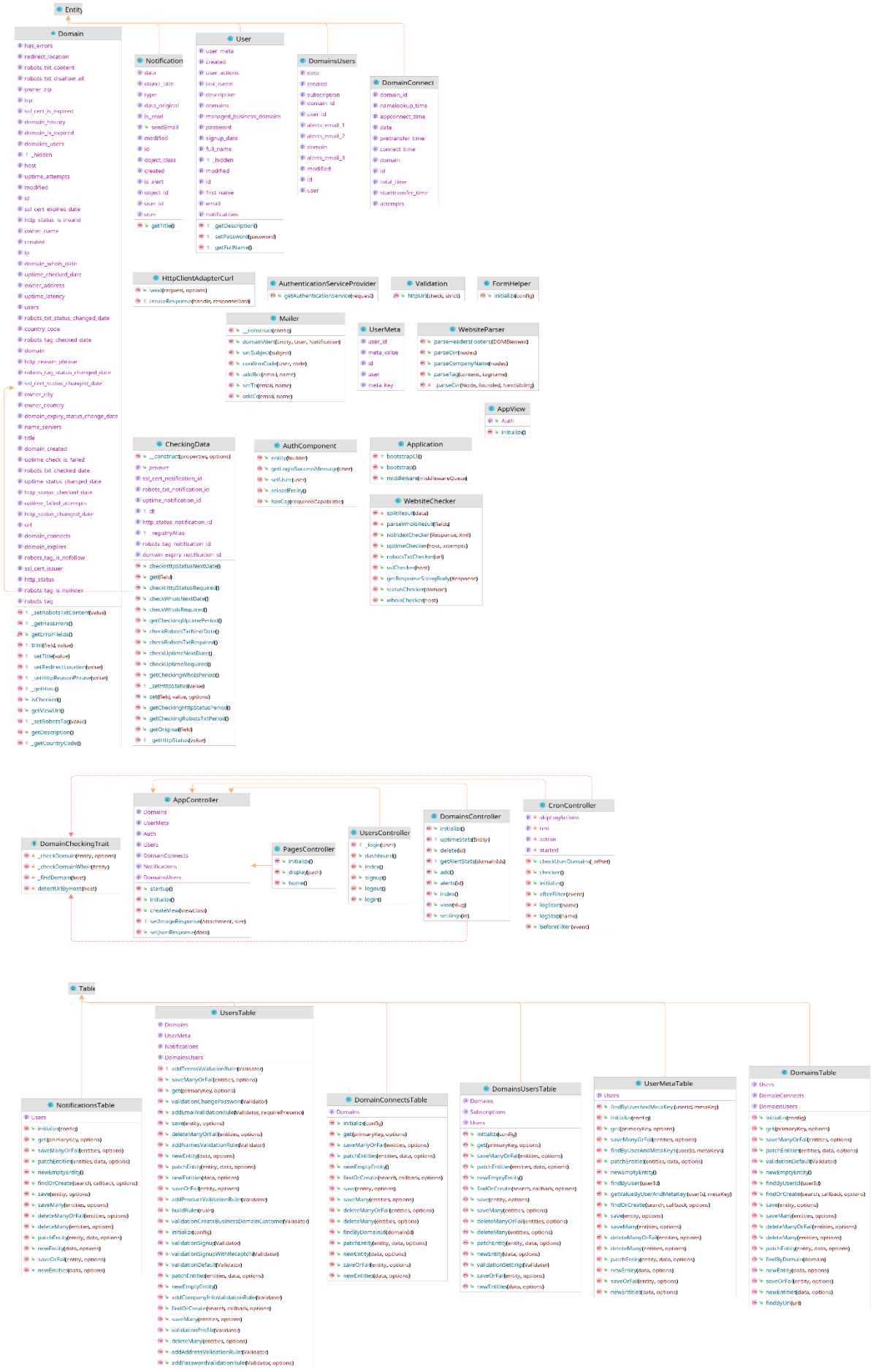


Рисунок 2.11 – статична модель у вигляді діаграми класів

Специфікація класів

Проаналізувавши класи у вигляді створення статичної та динамічної моделі, можемо проаналізувати створити специфікацію класів.

Специфікація класів вебсервісу для моніторингу стану вебсайтів приведена нижче.

Назва: Application

Відповідальність класу: Головний клас вебсервісу, відповідає за його налаштування та управління.

Функції які надає клас:

- bootstrapCli() - виконує налаштування додатку для командного рядка ініціалізує необхідні ресурси та сервіси, необхідні для виконання команд;

- bootstrap() - виконує загальне налаштування додатку, ініціалізує компоненти, сервіси та інші ресурси, необхідні для правильної роботи вебсервісу;

- middleware() - налаштовує проміжне ПЗ (middleware) для обробки запитів та відповідей вебсервісу, додає та конфігурує різні шари проміжного програмного забезпечення, такі як авторизація, обробка помилок, кешування тощо.

Назва: WebsiteChecker.

Відповідальність класу: Клас містить логіку для перевірки стану вебсайту.

Функції, які надає клас:

- noIndexChecker () – перевіряє чи не заборонена індексація на вебсайті;

- uptimeChecker() – перевіряє тривалість підключення до вебсейту;

- robotsTxtChecker() – перевіряє правильність налаштувань файлу robots.txt на заперечення індексації вебсайту;

- sslChecker() – перевіряє придатність ssl сертифікату;

- statusChecker() – перевіряє HTTP статус, що повертає вебсайт та IP адресу вебсайту.

- whoIsChecker() – отримує деталі вебхостингу.

Назва: AuthComponent.

Відповідальність класу: Компонент авторизації, відповідає за авторизацію користувачів та управління доступом у вебсервісі.

Функції які надає клас:

- allowUnauthenticated() - дозволяє доступ до вказаних дій контролера без авторизації;

- authenticate() - перевіряє вхідні дані користувача та виконує авторизацію;

- logout() - виконує вихід користувача з системи, завершує сеанс та знищує дані авторизації;

- entity() - повертає об'єкт з бази даних, що представляє авторизованого користувача;

- hasCap() - Перевіряє, чи має користувач доступ до вказаної дії вебсервісу.

Назва: Mailer.

Відповідальність класу: Клас відповідає за функціонал електронних повідомлень.

Функції, які надає клас:

- setTemplate() – дозволяє підключити заданий шаблон;

- setTo() – метод для задання електронної адреси отримувача;

- setSubject() – метод для задання теми листа;

- domainAlert() – метод в якому будуть створюватися електронні повідомлення, які буде отримувати користувач при змінах в стані вебсайту, який він моніторить;

- send() – надсилає повідомлення.

Назва: CronController.

Відповідальність класу: Службовий клас, який виконує заплановані перевірки стану збережених вебсайтів.

Функції, які надає клас:

- checkUserDomains() – робе перевірку збережених вебсайтів;

- logStart() – записує в лог час запуску запланованого завдання;

- logStop() – записує в лог час завершення запланованого завдання;

- checked() – фукнція виконує перевірки вебсайтів клієнтів за розкладом.

Назва класу: DomainsController.

Відповідальність класу: Клас відповідає за логіку роботи з моделлю Domain.

Функції які надає клас:

- add() – додає вебсайт до списку вебсайтів користувача;

- delete() – видаляє вебсайт із списку сайтів, котрі моніторить користувач;

- index() – обробляє форму на головній сторінці;

- view() – відповідає за перегляд сторінки стану вебсайту;

- getAlertStats() – підраховує кількість проблем на вебсайті;

- uptimeStats() – робить розрахунки та створює графік змін показників пінгу.

Динамічна модель варіантів використання

Визначивши сценарії варіантів використання та розробивши діаграму класів було розроблено динамічну модель.

Діаграму послідовності для варіанту використання Авторизація користувача представлено на рисунку 2.12.

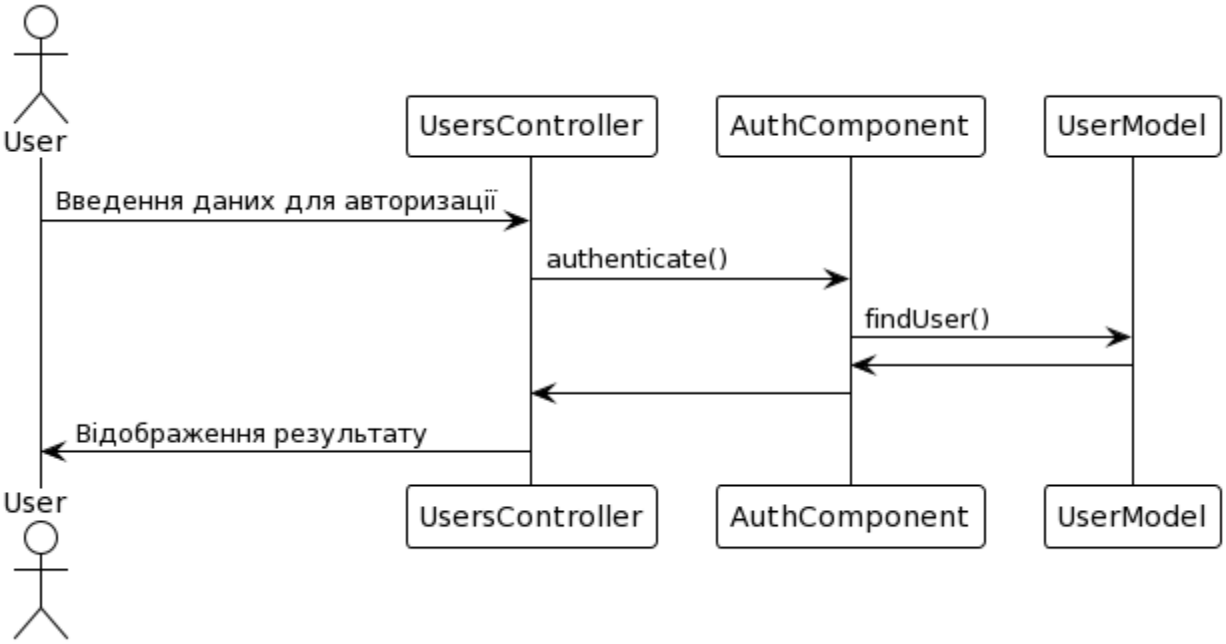


Рисунок 2.12 – діаграма послідовності для Авторизація користувача

Логічна модель даних

Щоб відобразити структури бази даних незалежно від використовуваної СКБД було побудовано Логічну модель.

Модель представлено на рисунку 2.13.

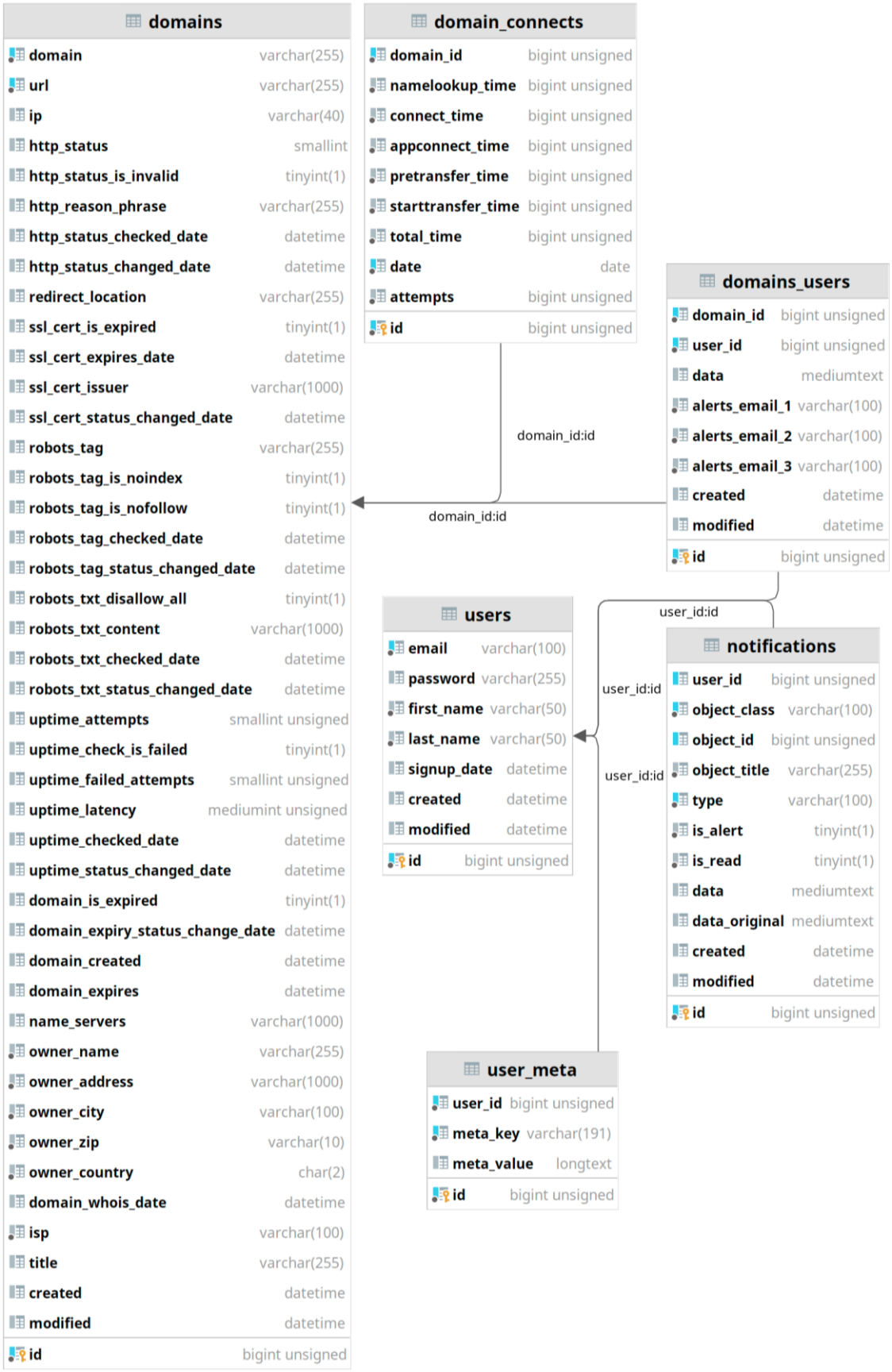


Рисунок 2.13 – Логічна модель бази даних

2.3.3 Робочий проєкт вебсервісу для моніторингу стану вебсайтів

Після розробки технічного проєкту було вибрано конкретну мову програмування та системи керування базами даних (СКБД) для реалізації вебсервісу. У цьому підрозділі обгрунтовано переваги зроблений вибір, проводено тестування різних сценаріїв використання та випробовано роботу вебсервісу для моніторингу стану вебсайтів.

Вибір мови програмування

При розробці будь-якого програмного забезпечення постає завдання вибору мови програмування та технологій, які будуть використовуватись у процесі розробки. В цьому контексті ми зосереджуємося на виборі оптимальних мов програмування та технологій для розробки вебсервісу.

Розглянемо варіанти між мовами C++, Java та PHP.

C++ є мовою програмування, яка зазвичай використовується для розробки системного та вбудованого програмного забезпечення. Основні переваги C++ полягають у його швидкості та продуктивності, що дозволяє розробляти швидкодіючі програми. Водночас, використання C++ вимагає більшого досвіду та уваги до деталей, оскільки мова дає більше можливостей та контролю, але й може призвести до складнішого коду та більшої кількості помилок.

Java є мовою програмування, яка використовується для розробки різноманітних додатків, від вебсерверів до мобільних додатків. Однією з найбільших переваг Java є її переносимість, оскільки програми, написані на Java, можуть запускатися на різних операційних системах без необхідності в перекомпіляції. Java також відома своєю надійністю та безпекою. Проте, у порівнянні з деякими іншими мовами, Java може бути менш продуктивною та вимагати більше ресурсів пам'яті.

PHP є мовою програмування, спеціально розробленою для веброзробки. Однією з головних переваг PHP є його простота вивчення та використання. Вона має широку підтримку серед спільноти розробників та велику кількість готових

компонентів і бібліотек для швидкої розробки вебдодатків. Проте, PHP має деякі недоліки, такі як менша продуктивність порівняно з C++ та обмежені можливості масштабування для дуже великих проектів.

В результаті було обрано об’єктно-орієнтовану мову програмування PHP. На цій мові програмування побудовано фреймворк CakePHP, який є гнучким інструментом для розробки вебдодатків. Фреймворк CakePHP використовує архітектуру MVC, що дозволяє ефективно організувати код та розділити логіку програми на моделі, представлення та контролери.

Завдяки фреймворку CakePHP розробка вебдодатків стає більш простою та швидкою. Він надає широкий набір готових компонентів і бібліотек, що спрощують роботу з базами даних, формами, аутентифікацією та іншими завданнями.

Використання мови програмування PHP з фреймворком CakePHP дозволяє зручно використовувати парадигму MVC, використовуючи модель для опису даних, представлення для користувацького інтерфейсу та контролер для обробки користувацьких запитів. Фреймворк CakePHP надає необхідні інструменти для розробки та організації коду у вебдодатках на основі архітектурного шаблону MVC.

Вибір системи керування базами даних

Для роботи з БД зазвичай використовують СКБД. Сьогодні існує дуже багато різноманітних СКБД та типів БД, і цей підрозділ присвячений аргументуванню вибору СКБД вебсервісу для моніторингу стану вебсайтів.

Під реляційними базами даних мають на увазі ті, які використовують мову запитів SQL. Дані в таких базах даних зберігаються у вигляді таблиць і рядків, які можна поєднувати між собою, створюючи зв'язки за допомогою ключів. Це дозволяє розширити можливості для збереження та організації даних, а також забезпечує можливість встановлення унікальності для елементів, що зберігаються в таблицях.

Нереляційні бази даних (NoSQL) відрізняються від реляційних баз даних тим, що не мають такої ж суворої структурованості. У них інформація зберігається у спеціальних документах, які використовують формат JSON або інші подібні формати пар ключ-значення. Це дозволяє зберігати дані без жорсткої схеми і реляційної моделі, що розширює можливості для гнучкого збереження та обробки даних.

Обидві бази даних, реляційні та нереляційні (NoSQL), мають свої переваги та недоліки. Однак, для розробки програмного забезпечення, в якому використовується фреймворк CakePHP, основним фактором є підтримка данного фреймворку.

Після аналізу реляційних та нереляційних баз даних, було вирішено обрати реляційну базу даних з урахуванням потреб проєкту.

Для реляційних баз даних доступні різні СКБД, такі як MySQL, PostgreSQL та Microsoft SQL Server. Обрання конкретної СКБД залежить від її можливостей та вимог проєкту, проте фреймворк CakePHP дозволяє працювати з будь-якою з них.

PostgreSQL використовується переважно в великих проєктах завдяки своєму широкому функціоналу та високому рівню захисту. Однак, у даному випадку, коли більша частина роботи виконується фреймворком, використання PostgreSQL може бути не доцільним. Також слід враховувати, що через свій великий функціонал та розташування на одному сервері, PostgreSQL може мати обмеження щодо швидкодії порівняно з іншими доступними СКБД.

MySQL є простою та швидкою СКБД, яка має вбудовані функції, необхідні для розробки вебсайтів. Вона підтримує багато мов програмування та технологій і включає в себе вбудовані засоби обмеження доступу.

Microsoft SQL Server має багато переваг, які є схожими на ті, що присутні в MySQL, а також має простий та зрозумілий інтерфейс, може бути розміщена на декількох серверах. Однак, одним з недоліків є вартість розміщення на хостингу через обмежений вибір доступних сервісів. Проте, існують

безкоштовні варіанти розміщення, які обмежені за функціоналом, доступним в СКБД.

Після аналізу наявних СКБД, було прийняте рішення використовувати MySQL у розробці вебсервісу для моніторингу стану вебсайтів, тому що вона є швидкою та оптимізованою для роботи з великими об’ємами даних.

Фізична модель даних

Фізичну модель бази даних наведено у вигляді таблиць 2.12-2.17.

Таблиця 2.12 – Структура таблиці users

Номер	Назва поля	Ключ	Тип Даних
1	id	PK	bigint (20)
2	email	FK	varchar(100)
3	password		varchar(255)
4	first_name		varchar(50)
5	last_name		varchar(50)
6	signup_date		datetime
7	created		datetime
8	modified		datetime

Таблиця 2.13 – Структура таблиці user_meta

Номер	Назва поля	Ключ	Тип
1	id	PK	bigint(20)
2	user_id	FK	bigint(20)
3	meta_key	FK	varchar(191)
4	meta_value		longtext

Таблиця 2.14 – Структура таблиці domain_connects

Номер	Назва поля	Ключ	Тип
1	id	PK	bigint(20)
2	domain_id	FK	bigint(20)
3	namelookup_time		bigint(20)
4	connect_time		bigint(20)
5	appconnect_time		bigint(20)
6	pretransfer_time		bigint(20)
7	starttransfer_time		bigint(20)
8	total_time		bigint(20)
9	date	FK	date
10	attempts		bigint(20)

Таблиця 2.15 – Структура таблиці domains

Номер	Назва поля	Ключ	Тип
1	id	PK	bigint(20)
2	domain		varchar(255)
3	url	FK	varchar(255)
4	ip		varchar(40)
5	http_status		smallint(5)
6	http_status_is_invalid		tinyint(1)
7	http_reason_phrase		varchar(255)
8	http_status_checked_date		datetime
9	http_status_changed_date		datetime
10	redirect_location		varchar(255)
11	ssl_cert_is_expired		tinyint(1)
12	ssl_cert_expires_date		datetime
13	ssl_cert_issuer		varchar(1000)
14	ssl_cert_status_changed_date		datetime

Продовження таблиці 2.15

15	robots_tag		varchar(255)
16	robots_tag_is_noindex		tinyint(1)
17	robots_tag_is_nofollow		tinyint(1)
18	robots_tag_checked_date		datetime
19	robots_tag_status_changed_date		datetime
20	robots_txt_disallow_all		tinyint(1)
21	robots_txt_content		varchar(1000)
22	robots_txt_checked_date		datetime
23	robots_txt_status_changed_date		datetime
24	uptime_attempts		smallint(5)
25	uptime_check_is_failed		tinyint(1)
26	uptime_failed_attempts		smallint(5)
27	uptime_latency		mediumint(7)
28	uptime_checked_date		datetime
29	uptime_status_changed_date		datetime
30	domain_is_expired		tinyint(1)
31	domain_expiry_status_change_date		datetime
32	domain_created		datetime
33	domain_expires		datetime
34	name_servers		varchar(1000)
35	owner_name		varchar(255)
36	owner_address		varchar(1000)
37	owner_city		varchar(100)
38	owner_zip		varchar(10)
39	owner_country		char(2)
40	domain_whois_date		datetime
41	isp		varchar(100)
42	title		varchar(255)

Продовження таблиці 2.15

43	created		datetime
44	modified		datetime

Таблиця 2.16 – Структура таблиці domains_users

Номер	Назва поля	Ключ	Тип
1	id	PK	bigint(20)
2	domain_id	FK	bigint(20)
3	user_id	FK	bigint(20)
4	data		mediumtext
5	alerts_email_1		varchar(100)
6	alerts_email_2		varchar(100)
7	alerts_email_3		varchar(100)
8	created		datetime
9	modified		datetime

Таблиця 2.17 – Структура таблиці notifications

Номер	Назва поля	Ключ	Тип
1	id	PK	bigint(20)
2	user_id	FK	bigint(20)
3	object_class		varchar(100)
4	object_id	FK	bigint(20)
5	object_title		varchar(255)
6	type		varchar(100)
7	is_alert		tinyint(1)
8	is_read		tinyint(1)
9	data		mediumtext
10	data_original		mediumtext

Родовження таблиці 2.15

11	created		datetime
12	modified		datetime

Тестування варіантів використання

Тестування були проведені за принципом чорної скриньки. Для цього були обрані основні варіанти використання. Головна мета кожного тестування є перевірка на помилки в роботі вебсервісу для моніторингу стану вебсайтів.

Тестування варіантів використання.

У ході першого тестування був протестований варіант використання «Авторизація». Проведених тестів 2.

Вхідні дані:

- введена електрона адреса;

- введений пароль;

Тести:

1) Авторизація гостя як клієнта.

Для того щоб користувач мав змогу авторизуватися, йому потрібно у шапці натиснути на кнопку авторизації, заповнити форму авторизації. Натиснувши кнопку підтвердження, очікуваним результатом буде перевірка введених користувачем даних (рисунок 2.14) та авторизувати користувача як клієнта, якщо введені дані є коректними. Результат тесту представлений на рисунку 2.15.

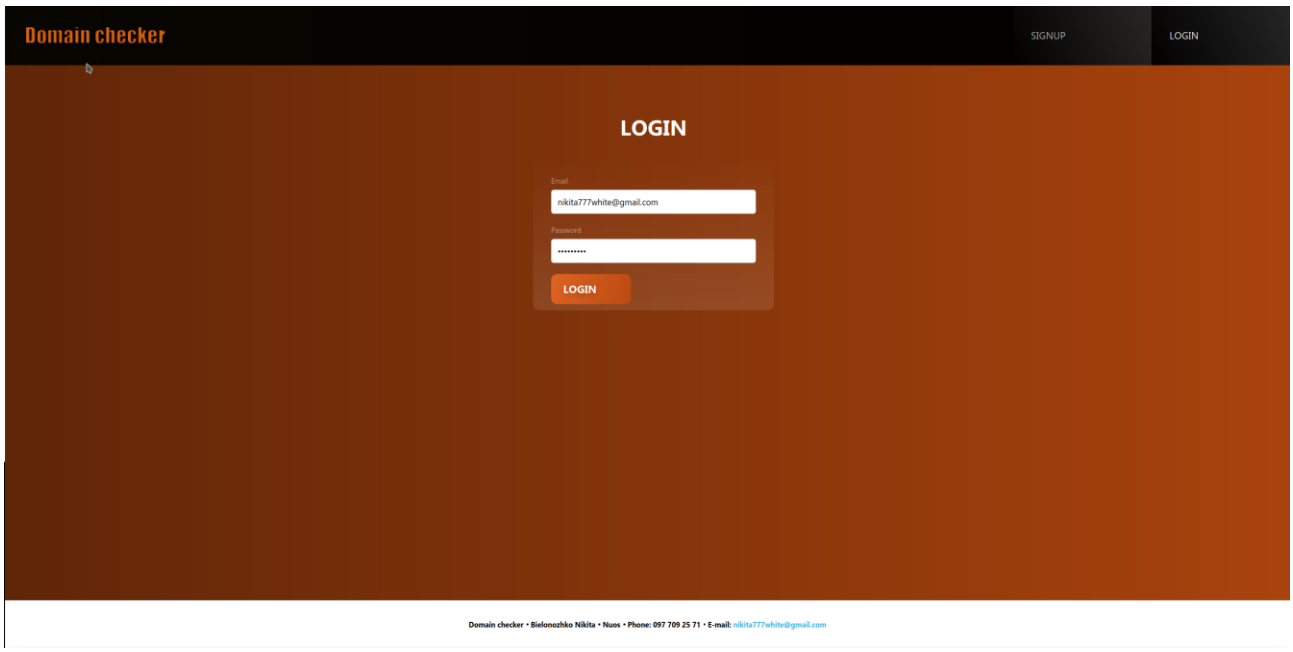


Рисунок 2.14 – Введення даних для авторизації

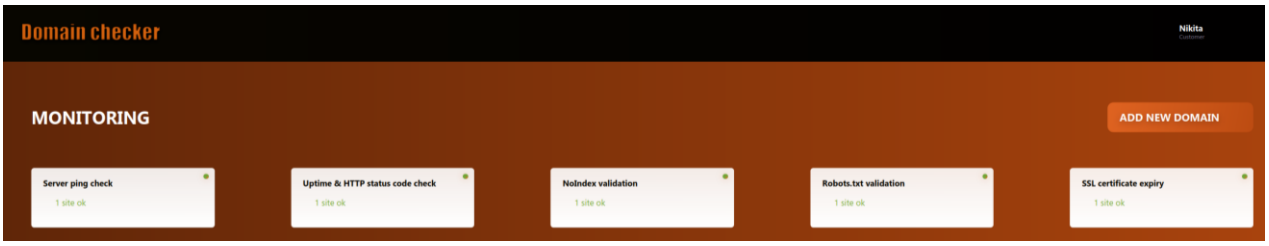


Рисунок 2.15 – Результат тесту авторизації

2) Авторизація з помилковими даними

Якщо користувач введе некоректний пароль та відправить форму, на формі повинен з’явитися помилка що свідчить про некоректний пароль. На рисунку 2.16 відображено приклад цієї помилки.

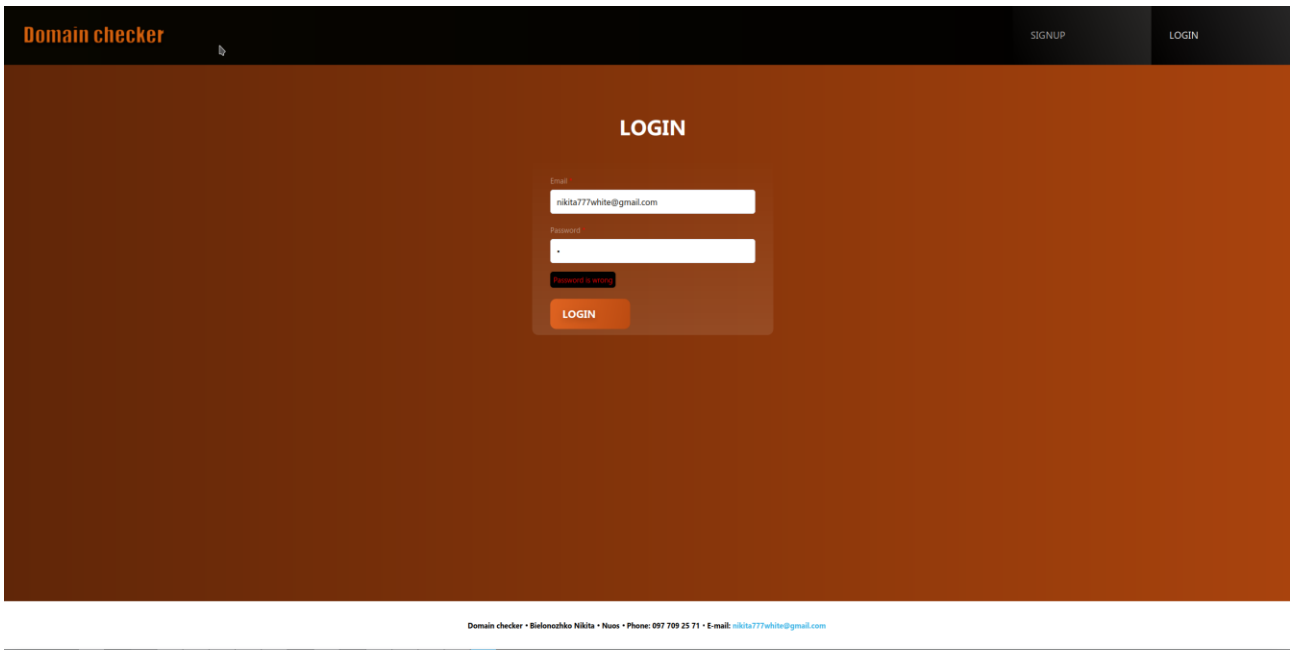


Рисунок 2.16 – Авторизація з неправильним паролем

Під час тестування Авторизації користувача помилок у роботі вебсервісу не було виявлено.

У ході другого тестування був протестований варіант використання «Моніторинг стану вебсайту за запитом».

Тести:

1) Моніторинг стану вебсайту зареєстрованим користувачем

Для ініціалізацію моніторингу стану вебсайту за запитом потрібно ввести URL адресу вебсайту (рисунок 2.17) та натиснути кнопку «Run» або натиснути клавішу Enter.

Вхідні дані:

- URL адреса сайту.

Очікуваним результатом повинні бути дані про стан вебсайту (рисунок 2.18 – 2.19).

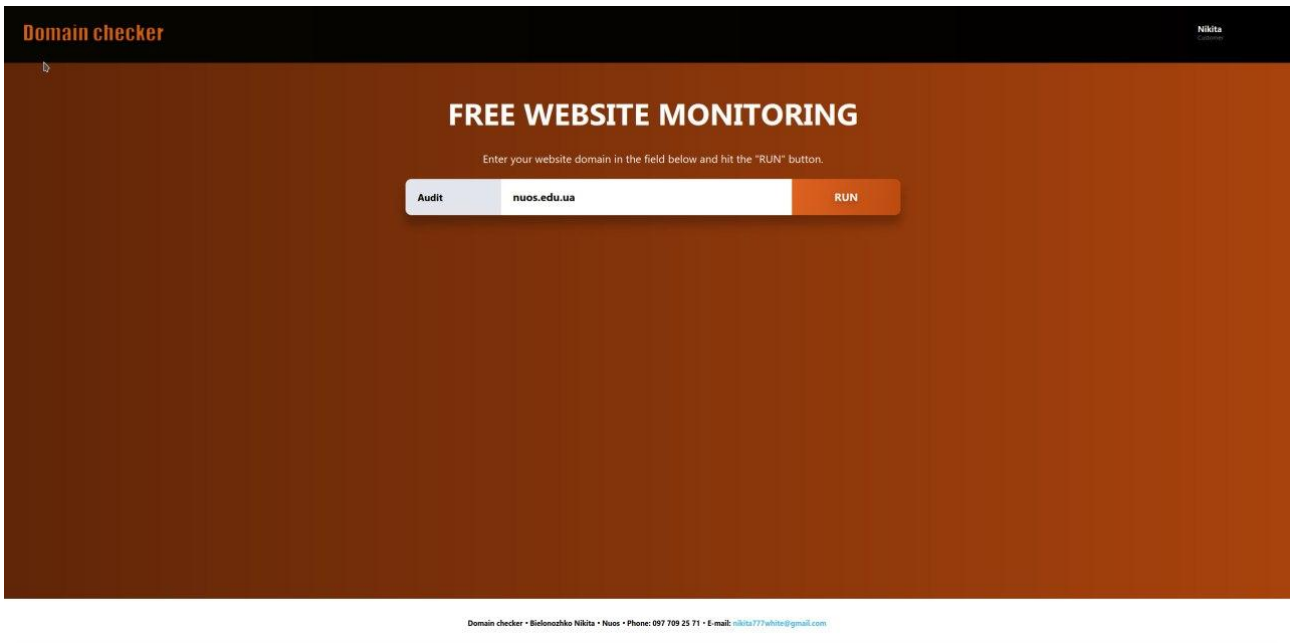


Рисунок 2.17 – Моніторинг стану вебсайту за запитом

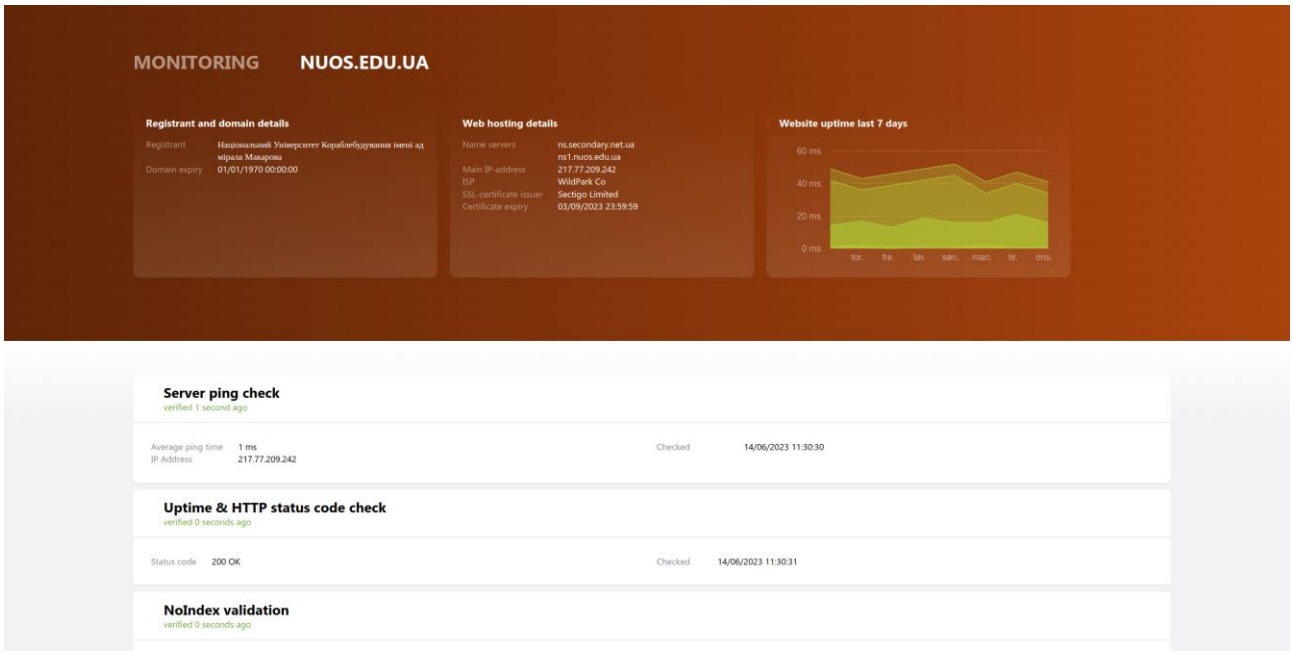


Рисунок 2.18 – Результат моніторингу вебсайту за запитом

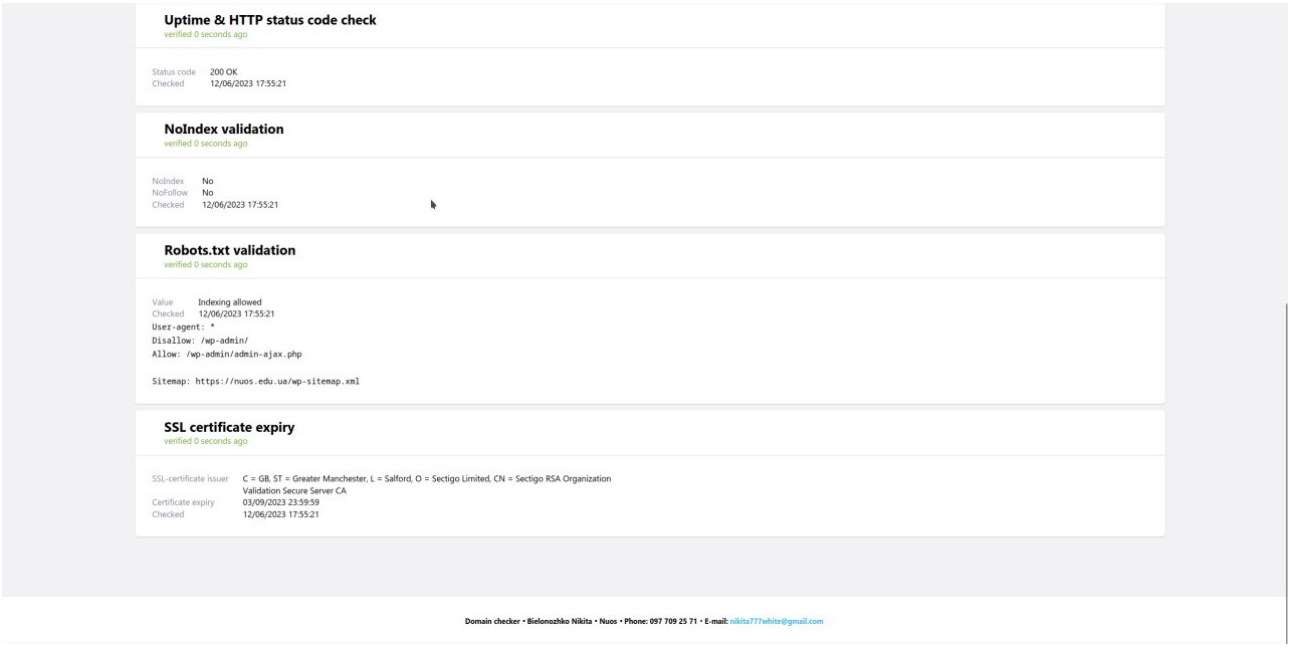


Рисунок 2.19 – Продовження результату моніторингу стану вебсайту за запитом

Під час тестування моніторингу стану вебсайтів за запитом помилок у роботі вебсервісу не було виявлено.

Випробовування роботи програмного забезпечення.

Методику випробувань для розробленого вебсервісу для моніторингу стану вебсайтів наведено в Додатку Д. На основі додатка можемо провести випробування роботи програмного забезпечення.

Для початку роботи, користувачу потрібно перейти до вебсервісу за допомогою браузеру. Користувач побачить головну сторінку вебсервісу, яка зображена на рисунку 2.20.

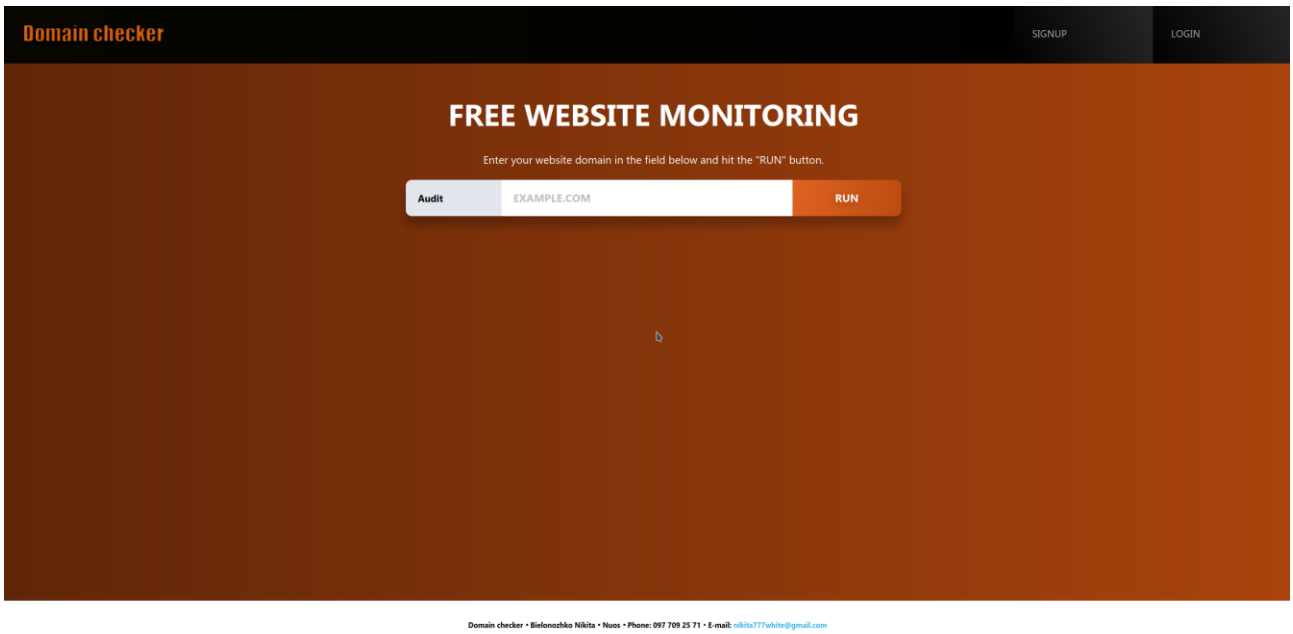


Рисунок 2.20 – Перша строрінка вебсервісу, яку побачить користувач

Проблем під час завантаження головної сторінки не відбулося.

Користувач має змогу авторизуватися чи одразу виповнити моніторинг стану вебсайту за запитом. Моніторинг стану вебсайту за запитом представлено на рисунках 2.21-2.22. Також процес реєстрації та авторизації зображено на рисунках 2.23-2.24. Перенаправлення до сторінки лічильників проблем на вебсайтах клієнта відбувається автоматично та зображено на рисунку 2.25.

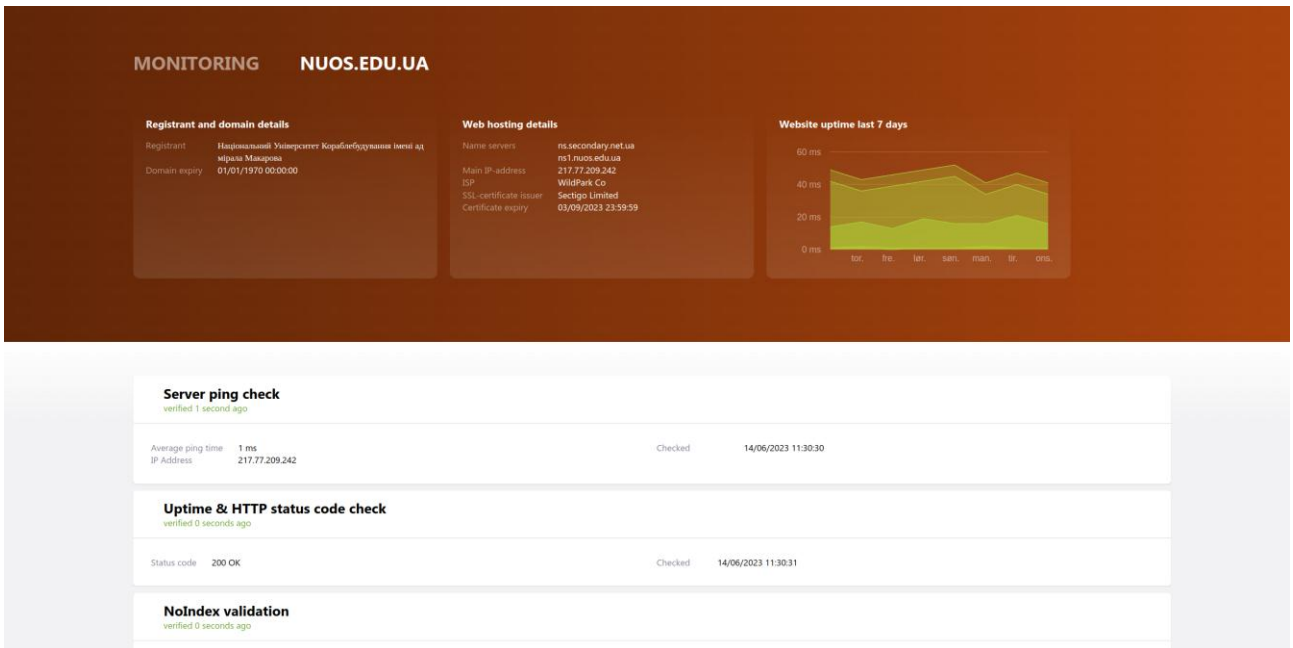


Рисунок 2.21 – Моніторинг стану вебсайту за запитом.

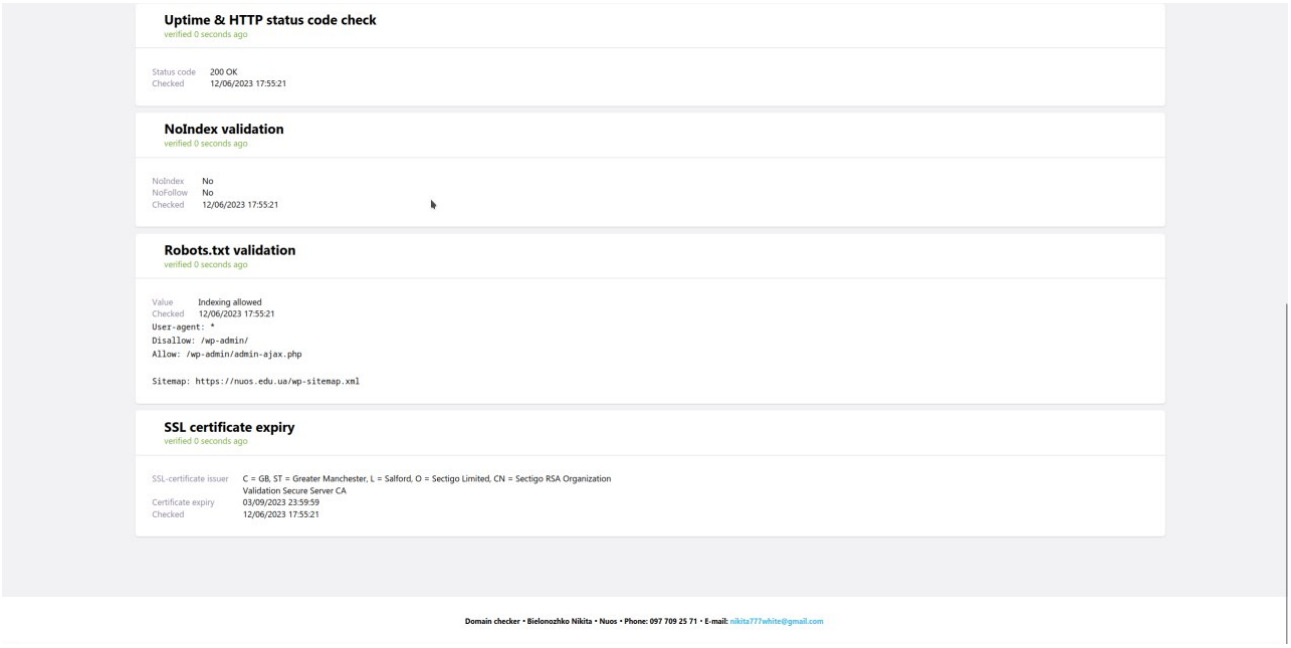


Рисунок 2.21 – Продовження сторінки моніторинг стану вебсайту за запитом

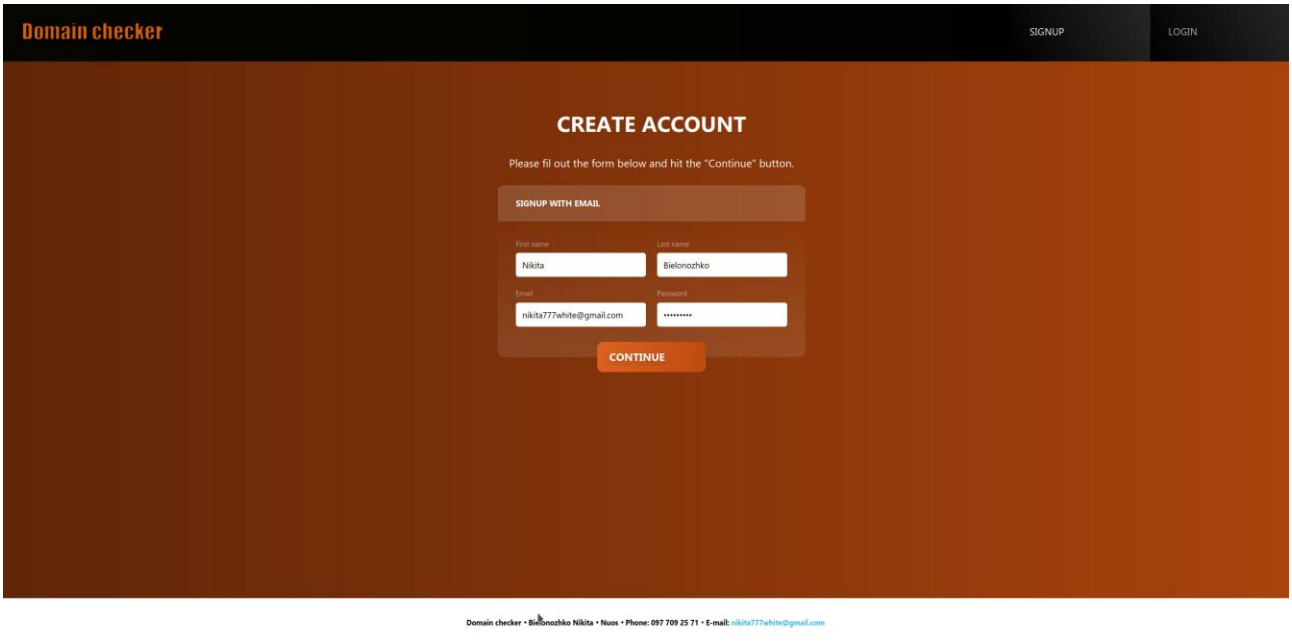
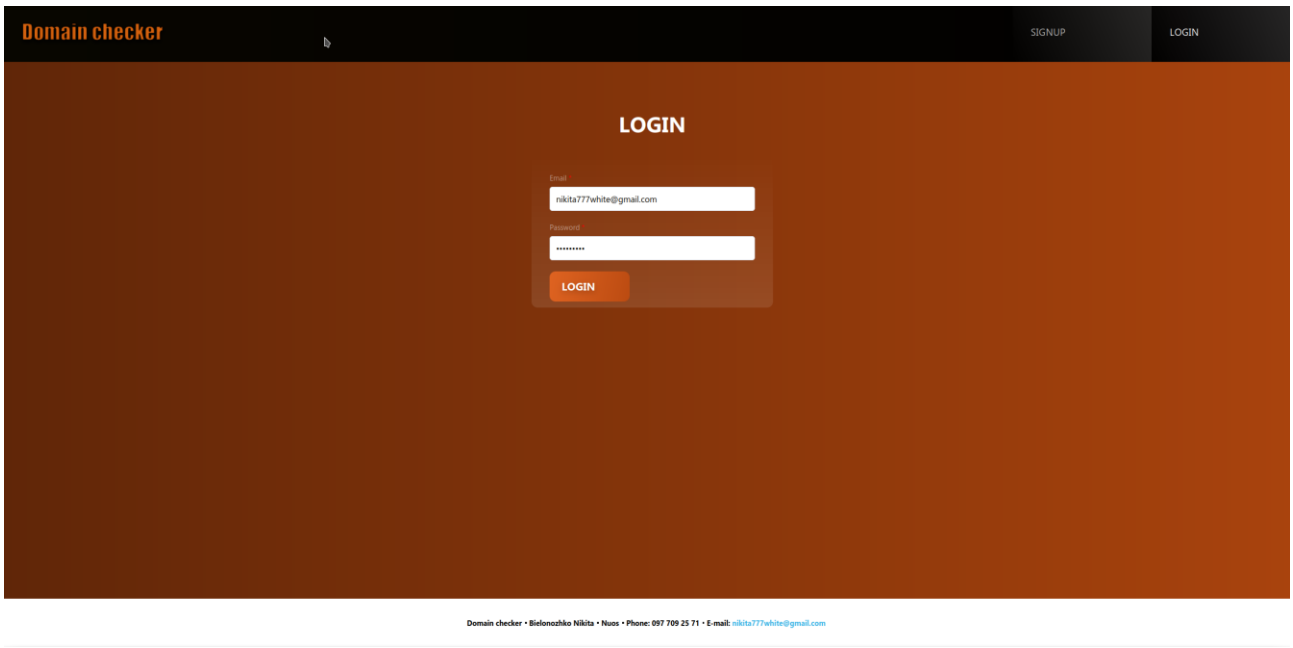


Рисунок 2.23 - Реєстрація користувача



2.24 – Авторизація користувача

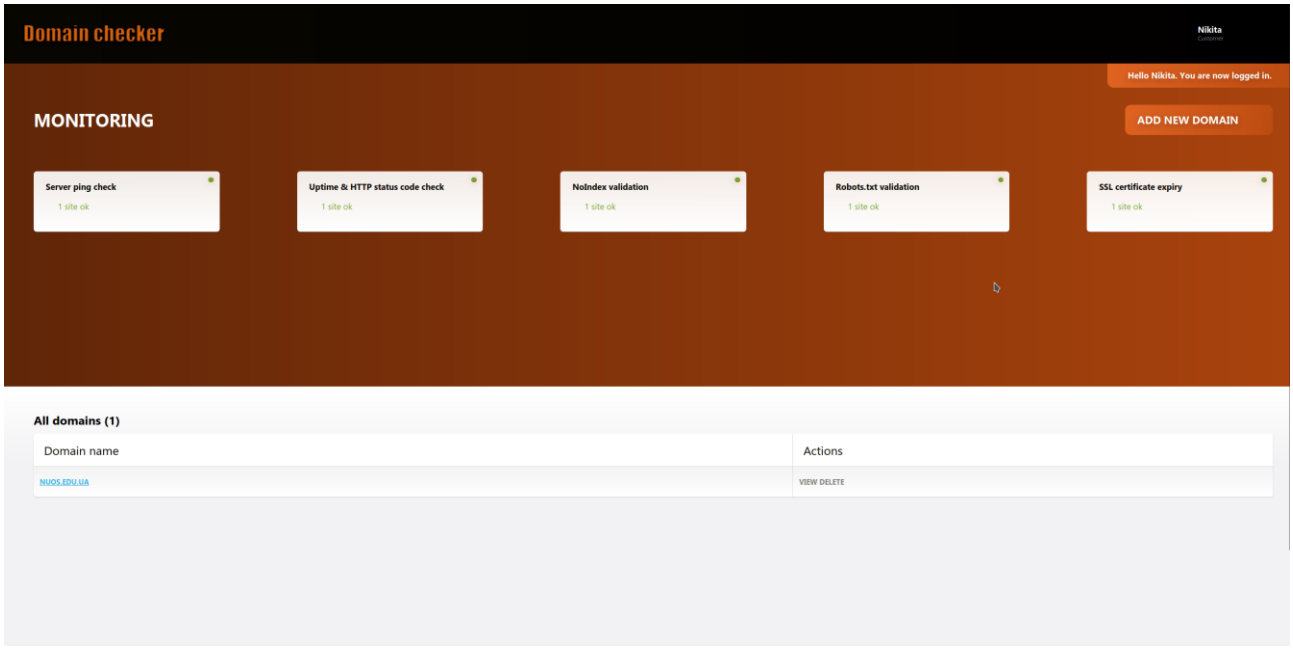


Рисунок 2.25 - Сторінка лічильників проблем вебсайтв

Після авторизації та реєстрації перенаправлення до сторінки з лічильниками проблем вебсайтів, помилок не було виявлено.

Для виходу із системи користувачу потрібно у шапці натиснути на своє ім’я та кнопку «logout» на будь-якій сторінці вебсервісу (рисунок 2.26).

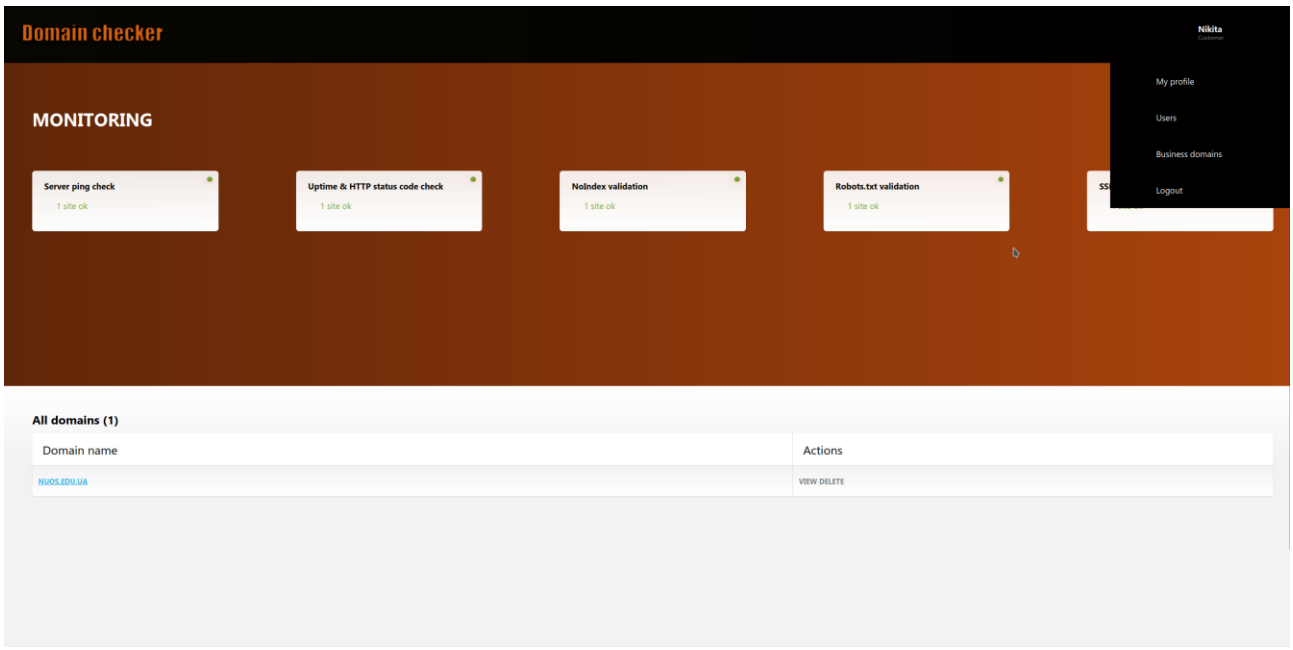


Рисунок 2.26 - Процес виходу із системи

Відкриття списку пройшло успішно.

Під час натискання кнопки виходу із системи, користувача поверне до сторінки авторизації та користувач вийде із системи (рисунок 2.27).

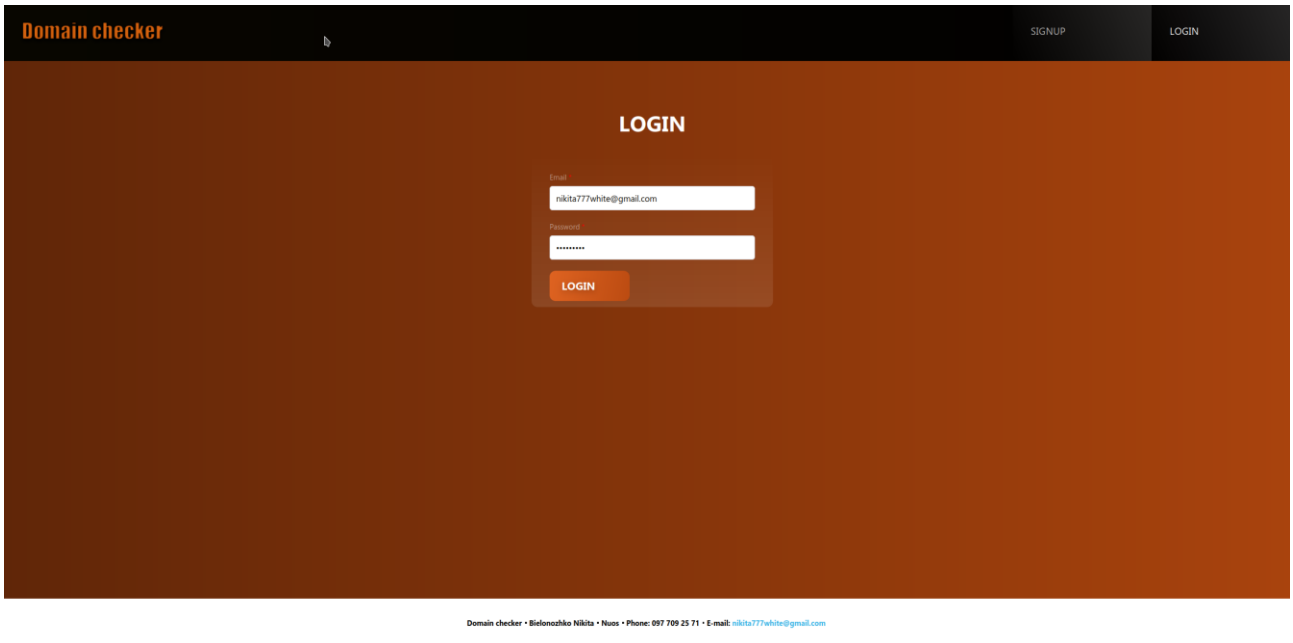


Рисунок 2.27 - Переадресація до сторінки авторизації користувача

Переадресація та вихід із системи пройшло успішно. У процесі жодних помилок не було виявлено.

На рисунках 2.28-2.31 представлено результати роботи моніторингу стану вебсайту за розкладом.

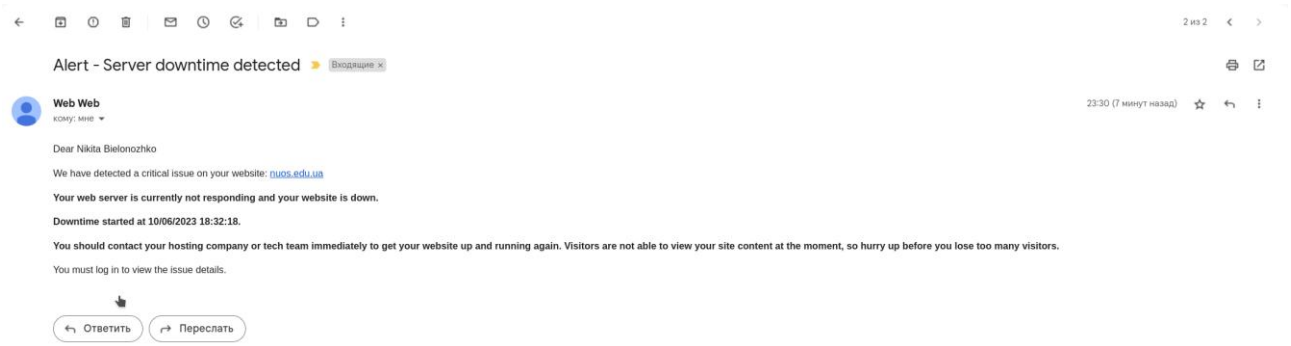


Рисунок 2.28 – Оповіщення про пурешення з’єднання

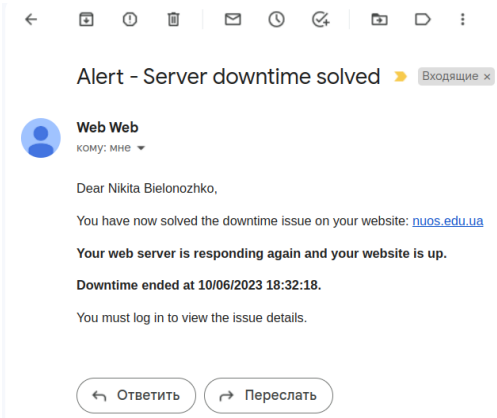


Рисунок 2.29 - Оповіщення про відновлене з’єднання з вебсервісом

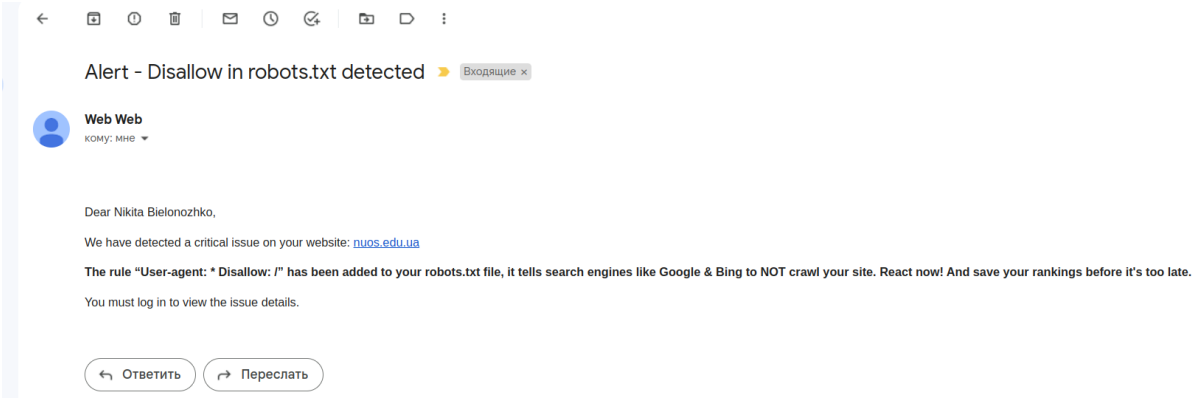


Рисунок 2.30 - Оповіщення про заперечення індексації файлом robots.txt

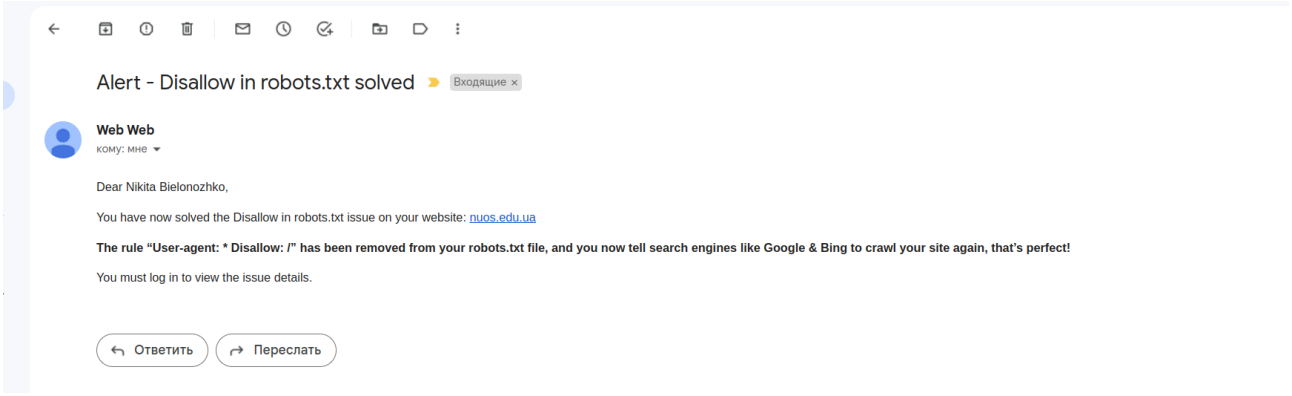


Рисунок 2.31 - Оповіщення що проблему індексації файлом robots.txt вирішено

Під час проведення усіх тестів помилок не було виявлено, тому можна зробити висновок що вебсервіс для моніторингу стану вебсайтів працює без помилок. Випробовування вебсервісу було проведено успішно.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ВЕБСЕРВІСУ ДЛЯ МОНІТОРИНГУ СТАНУ ВЕБСАЙТІВ

В кваліфікаційній роботі було розроблено вебсервіс для моніторингу стану вебсайтів.

Розроблений вебсервіс був реалізований на об’єктно-орієнтованій мові програмування PHP з використання фреймворку CakePHP та СКБД MySQL. Програмне забезпечення складається із 3993 рядків PHP коду, та 3587 рядків CSS коду. Технічне завдання наведено у Додатку А. Частину коду реалізованого вебсервісу представлено у Додатку Б. Опис програми наведено у Додатку В.

У Додатку Г представлено інструкцію користувача для роботи з вебсервісом.

Було проведено випробовування вебсервісу, використовуючи методику вказану у Додатку Д. Було зроблено висновок що програмний продукт є цілком працездатним.

4 ОХОРОНА ПРАЦІ

4.1 Основні положення

Охорона праці – це система, що поєднує правові, соціально-економічні, організаційно-технічні, санітарно-гігієнічні та профілактичні заходи, спрямовані на збереження життя, здоров'я та працездатності людини під час трудової діяльності.

Охорона праці включає різні аспекти, включаючи інженерно-технічні заходи, що спрямовані на зменшення ризиків у робочому середовищі шляхом впровадження нових технологій та використання менш небезпечних матеріалів. Крім того, охорона праці включає соціальні заходи, спрямовані на компенсацію завданої шкоди внаслідок нещасних випадків та на захист прав працівника.

Усі права працівника на безпечні умови праці захищаються законом України “Про охорону праці” [7].

Охорона праці відіграє визначну роль у забезпеченні безпеки та благополуччя працівників під час виконання трудових обов'язків. Вона втілює систему інноваційних заходів, які забезпечують захист від потенційних ризиків і ризикових факторів, враховуючи правові, соціально-економічні, організаційно-технічні, санітарно-гігієнічні і профілактичні аспекти.

Інженерно-технічне завдання охорони праці виявляється у впровадженні передових технологій та інноваційних рішень для забезпечення безпеки та мінімізації ризиків для працівників. Це охоплює широкий спектр заходів, таких як використання ергономічних робочих місць, застосування безпечних матеріалів, впровадження систем вентиляції та освітлення, а також організацію навчань та тренінгів з питань безпеки та здоров'я.

Соціальне завдання охорони праці необхідно розглядати як більш широкий і комплексний підхід, спрямований на покращення якості життя працівників та їх соціального благополуччя. В рамках цього завдання враховуються не тільки

відшкодування шкоди, що була завдана через нещасні випадки або професійні захворювання, але й забезпечення адекватного медичного обслуговування, компенсаційних виплат, професійної реабілітації, соціального захисту та підтримки працівників. Крім того, унікальність полягає в розробці та впровадженні інноваційних програм і політик, спрямованих на попередження нещасних випадків та професійних захворювань, а також на підвищення свідомості та участі працівників у процесі охорони праці. Застосування сучасних технологій, використання аналітичних інструментів та систем управління допомагають покращити ефективність охорони праці та забезпечити високий рівень безпеки та благополуччя працівників.

4.2 Аналіз шкідливих та небезпечних факторів під час роботи з екранними пристроями

Існують небезпечні та шкідливі фактори роботи з екранними пристроями які потрібно знати. До головних можна віднести роботу з самим екранним пристроєм та робоче місце працівника [8].

Нижче наведено деякі з них:

1.1) Візуальний вплив під час роботи з комп'ютером: Використання комп'ютера може призводити до напруження очей, сухості, подразнення та втоми. Тривале спрямоване уваги на екран може спричиняти втому очей, головний біль та навіть проблеми зі зором. Рекомендується регулярно робити перерви, виконувати спеціальні вправи для очей та налаштовувати яскравість та контрастність екрану на комфортний рівень. При довготривалому використанні комп'ютера також корисно забезпечити достатнє освітлення приміщення та використовувати захисні фільтри для екрану, якщо потрібно.

1.2) Постійний напружений стан під час роботи з комп'ютером: Використання комп'ютера може призводити до постійного стресу, особливо в ситуаціях, коли необхідно бути постійно на зв'язку або доступним. Це може негативно впливати на психічне здоров'я та загальний стан організму. Для

забезпечення ергономічної безпеки рекомендується встановити режими відпочинку, вимкнути сповіщення, коли вони необхідні, і створити звичку розслаблятися і відпочивати від комп'ютера. Важливо регулярно проводити паузи, займатися фізичними вправами, виконувати розтяжку і релаксаційні техніки, щоб зняти накопичений стрес і зберегти психічне благополуччя.

1.3) Негативний вплив на позу під час роботи з комп'ютером: Використання комп'ютера може призводити до некомфортної пози тіла, особливо коли людина схиляється вперед або має незручне положення рук і шиї під час роботи. Це може призвести до появи болю в шиї, спині та плечах. Для запобігання цим проблемам важливо дотримуватися правильної пози: тримати спину прямою, руки розслабленими на робочому столі, дотримуватися оптимальної відстані від монітора, а також робити перерви для розтягування і вправ для м'язів.

1.4) Ефект електромагнітного випромінювання: Мобільні пристрої випромінюють електромагнітне випромінювання, яке може мати потенційно шкідливі наслідки для здоров'я в довгостроковій перспективі. Хоча наукові дослідження не дають однозначних доказів щодо шкідливості електромагнітного випромінювання на низьких рівнях, деякі люди можуть відчувати негативний вплив. Зменшити експозицію можна, використовуючи навушники або вбудовані функції гучномовця, а також обмежуючи тривалість розмов та тримаючи пристрій подалі від тіла.

Норми мікроклімату зазначені у санітарних нормах мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [9].

Температура навколишнього повітря може варіюватися залежно від сезону та інших факторів. Для забезпечення комфортних умов необхідно використовувати додаткові засоби контролю мікроклімату, такі як кондиціонери, обігрівачі та інші пристрої. Це дозволить уникнути неприємних температурних відхилень і забезпечити оптимальні умови для праці та відпочинку.

Забезпечення ергономічної безпеки праці передбачає дотримання простих правил. Щоб забезпечити правильне положення спини, рекомендується

нахилити її назад під невеликим кутом, що сприятиме збільшенню кута між тулубом і стегнами. Це сприятиме покращенню кровообігу та зменшенню навантаження на хребет. Руки повинні бути розслабленими й вільно опущеними вздовж боків, а передпліччя й кисті мають бути паралельними до підлоги. Стегна повинні бути у прямому куті до тулуба, а коліна - у прямому куті до стегон. Це допоможе забезпечити правильну позу та комфортні умови під час виконання роботи.

Максимальний час безперервної роботи з комп'ютером не повинен перевищувати 2 години на день. Через кожні 7 хвилин рекомендується робити коротку перерву тривалістю 10-15 секунд. Під час перерви можна звернути погляд у вікно або на спеціально розміщену картину з приємним пейзажем, що сприяє відпочинку очей. Важливо, щоб картина мала переважно неяскраві та заспокійливі кольори, такі як зелений чи блакитний. Крім того, рекомендується виконувати прості вправи для очей, щоб зберегти їх здоров'я та покращити кровообіг. Це допоможе зменшити негативний вплив тривалого роботи за комп'ютером і підтримати ергономічну безпеку.

2) Вимоги безпеки після закінчення роботи:

2.1) Після завершення роботи з комп'ютером, важливо виконати наступні кроки з метою збереження безпеки та організації робочого середовища. Відключіть комп'ютер та всі його пристрої від електромережі, за винятком обладнання, яке потребує постійного живлення, таких як факсимільні апарати, мережеві сервери та інше. Приберіть своє робоче місце, впорядкуйте розташування обладнання та документів. Забезпечте достатнє провітрення приміщення для підтримання комфортної атмосфери. Також, зверніть увагу на особисту гігієну, помийте руки та обличчя, щоб підтримувати чистоту та гігієну.

2.2) Зберегти інформацію на комп’ютері.

2.3) Прибрати робоче місце.

ВИСНОВКИ

В кваліфікаційній роботі вирішено практичну задачу інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме задачу розробки вебсервісу для моніторингу вебсайтів.

Було проведено аналіз практичної задачі інженерії програмного забезпечення з розробки вебсервісу для моніторингу вебсайтів та проаналізовано існуючі способи моніторингу стану вебсайтів. Здійснено постановку задачі на розробку вебсервісу для моніторингу стану вебсайтів.

Було також створено ескізний, технічний, та робочий проєкти. Здійснене тестування вебсервісу для моніторингу стану вебсайтів.

Проведено опис аспектів охорони праці при роботі з екранними пристроями.

Розроблене технічне завдання наведено у Додатку А. Частину коду реалізованого вебсервісу представлено у Додатку Б. Опис програми наведено у Додатку В. У Додатку Г представлено інструкцію користувача для роботи з вебсервісом.

Після виконання випробовувань за програмою та методикою, наведеною в Додатку Д, було визначено, що вебсервіс не має ніяких помилок, що можуть свідчити про некоректну роботу програмного продукту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Определение Ping. URL: https://www.techtarget.com/searchnetworking/definition/ping (дата звернення 28.04.2023).

2. curl - бібліотека та інструмент командного рядка для передачі даних за допомогою різних протоколів. URL: https://curl.se/ (дата звернення 01.05.2023).

3. Список кодів стану HTTP. URL: https://uk.wikipedia.org/wiki/Список_кодів_стану_HTTP (дата звернення 29.04.2023).

4. Що таке SSL? URL: https://hostiq.ua/ukr/info/what-is-ssl/ (дата звернення 30.04.2023).

5. Заборона індексування сторінок. URL: https://developers.google.com/search/docs/crawling-indexing/block-indexing?hl=ua (дата звернення 02.05.2023).

6. Вступ до файлу robots.txt. URL: https://developers.google.com/search/docs/crawling-indexing/robots/intro?hl=ua (дата звернення 01.05.2023).

7. Про охорону праці: Закон України від 21.11.2002 р. № 229-IV. Дата оновлення: 04.02.2021. URL: https://zakon.rada.gov.ua/laws/main/2694-12#Text. (дата звернення: 22.05.2023).

8. Про затвердження Вимог щодо безпеки та захисту здоров’я працівників: Закон України від 14.02.2018 № 207 Дата оновлення: 17.06.2015. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#n14 (дата звернення: 22.05.2023).

9. Санітарні норми мікроклімату виробничих приміщень (ДСН 3.3.6.042-99). URL: https://zakon.rada.gov.ua/rada/show/va042282-99#Text (дата звернення: 22.05.2023).

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

1 Вступ

1.1 Назва програмного продукту

Вебсервіс для моніторингу стану вебсайтів.

1.2 Сфера застосування

Вебсервіс призначений для автоматичного моніторингу стану вебсайтів та відправлення оповіщень користувачам. Це дозволить швидко і зручно дізнаватися про зміни в стані вебсайту.

2 Підстава для розробки

Підставою для розробки даного програмного забезпечення є завдання на кваліфікаційну роботу.

3 Призначення розробки

3.1 Мета розробки програмного забезпечення

Мета розробки - оперативно дізнаватися якщо з вебсайтом сталася проблема.

3.2 Призначення програмного забезпечення

3.2.1 Експлуатаційне призначення

Автоматизація процесів моніторингу стану вебсайту, зменшення обсягу ручної роботи та впливу людського фактора.

3.2.2 Функціональне призначення

Програмне забезпечення повинно забезпечувати спрощення процесу моніторингу за станом вебсайтів шляхом автоматизації, та генерувати повідомлення про зміни стану вебсайту.

4 Технічні вимоги до програмного забезпечення

4.1 Вимоги до функціональних характеристик

Програмне забезпечення має реалізовувати наступні функції:

- Перевірка тривалості підключення до вебсайту.
- Перевірка HTTP статусу вебсайту.
- Перевірка заборони індексації.
- Перевірка файлу robots.txt.
- Перевірка придатності SSL сертифікату.
- Побудова графіку тривалості підключення до вебсайту.
- Моніторинг стану вебсайтів за розкладом.
- Оповіщення користувачів про зміни у стані їх вебсайтів.

4.2 Вимоги до надійності

Програмне забезпечення має виконувати свої функції, при умові стабільної роботи комп’ютера і стабільного з’єднання з інтернет мережею, без виникнення збоїв.

Якщо виник збій під час моніторингу стану вебсайтів за розкладом, отриманий прогрес має бути відкинутий та процес моніторингу стану вебсайтів за розкладом початися з початку.

4.3 Умови використання

Необхідним є комп’ютер з доступом до інтернет мережі, який використовується в сухому приміщенні з температурою 20 градусів Цельсія (+-5 градусів Цельсія) для стабільної роботи обладнання.

4.4 Вимоги до складу та параметрів технічних засобів

Для стабільної роботи вебсервісу, сервер повинен мати наступні мінімальні характеристики:

- процесор: Intel Core I5-11400, 4.5GHZ, 4 ядра та більше;
- інтернет: мінімум 1 Гбіт/с;
- оперативна пам’ять: 16 Гб та більше;
- жорсткий диск: для стабільної роботи вебсервісу потрібно мінімум 500 Мб вільного простору.

4.5 Вимоги до інформаційної і програмної сумісності

Вимоги до інформаційної і програмної сумісності вебсервісу:

- операційна система Ubuntu 22.04;
- PHP 8.0;
- MySQL 8.

4.6 Вимоги до маркування та упаковки

Вимоги до маркування та упаковки відсутні.

4.7 Вимоги до транспортування і зберігання

Вимоги до транспортування і зберігання відсутні.

4.8 Спецілальні вимоги

Спеціальні вимоги до вебсервісу відсутні.

5 Вимоги до документації програми

До складу програмної документації повинні входити:

- Технічне завдання.
- Текст програми.
- Опис програми.
- Інструкція користувача.

- Програма та методика випробовувань.

6 Техніко-економічні показники

Техніко-економічні показники не розраховуються.

7 Стадії та етапи розробки

Етапи розробки вебсервісу наведено в таблиці А.1.

Таблиця А.1 – Етапи розорбки вебсервісу

Номер	Назва етапів роботи	Терміни виконання
1.	Підготовка розділу ВСТУП	24.03.2023
2.	Аналіз практичної задачі інженерії ПЗ	31.03.2023
3.	Аналіз вимог до ПЗ	07.04.2023
4.	Розробка архітектури ПЗ	21.04.2023
5.	Розробка проекту ПЗ	05.05.2023
6.	Реалізація та тестування ПЗ	12.05.2023
7.	Підготовка розділу Результати розробки ПЗ	17.05.2023
8.	Підготовка розділу з охорони праці	19.05.2023
9.	Підготовка розділу ВИСНОВКИ	24.05.2023
10.	Оформлення списку використаних джерел та додатків	26.05.2023
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	29.05.2023
12.	Підготовка презентації та доповіді	02.06.2023
13.	Попередній захист роботи на засіданні кафедри ПЗАС	05.06.2023
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	12.06.2023

8 Порядок контролю і приймання

Контроль проводиться викладачем на кажному етапі роботи.

Приймання проводиться відповідно до програми і методики випробовувань.

ДОДАТОК Б ТЕКСТ ПРОГРАМИ

CronController.php

```
<?php
declare(strict_types=1);

namespace App\Controller;

use App\Mailer\Mailer;
use App\Model\Entity\Notification;
use App\Utility\CheckingData;
use Cake\Core\Configure;
use Cake\Database\Exception\MissingConnectionException;
use Cake\Event\EventInterface;
use Cake\I18n\FrozenDate;
use Cake\I18n\FrozenTime;
use Cake\Log\Log;
use Cake\ORM\Query;
use Cake\Utility\Hash;

class CronController extends AppController {

	use DomainCheckingTrait;

	private string $action;

	private ?string $test;

	private int $started;

	private array $skipLogActions = [];

	public function initialize(): void {
		parent::initialize();
		$this->disableAutoRender();
		Configure::write('debug', true);

		$this->Auth->setConfig('requireIdentity', false);
```

```
            $this->action = $this->getRequest()->getParam('action');
            $this->started = time();
      }

      public function beforeFilter(EventInterface $event) {
            if (!in_array($this->action, $this->skipLogActions, true))
                  $this->logStart(str_replace('/cron/', '', $this->getRequest()-
>getRequestTarget()));

            return parent::beforeFilter($event);
      }

      public function afterFilter(EventInterface $event) {
            if (!in_array($this->action, $this->skipLogActions, true))
                  $this->logStop(str_replace('/cron/', '', $this->getRequest()-
>getRequestTarget()));

            return parent::afterFilter($event);
      }

      private function logStart(string $name): void {
            Log::info("Cron job started: {$name} {$this->started}");
      }

      private function logStop(string $name): void {
            Log::info("Cron job finished: {$name} {$this->started}");
      }

      public function checkUserDomains($_offset = 0) {
            $this->loadModel('DomainConnects');
            $NotificationsTable = $this->loadModel('Notifications');

            $Query = $this->Domains->find()
                  ->contain([
                        'Users' => function (Query $Query) {
                              return $Query->select(['id', 'email',
'first_name', 'last_name']);
                        },
                        'DomainConnects' => function (Query $Query) {
                              return $Query->where(['DomainConnects.date' => new
FrozenDate()]);
                        },
                  ]);
```

```
                /** @var \App\Model\Entity\Domain[] $Entities */
                $Entities = $Query->all();

            foreach ($Entities as $Entity) {
                $id = $Entity->id;

                $url = $Entity->url;
                $options = [
                    'uptime' => false,
                    'status' => false,
                    'robots_txt' => false,
                    'whois' => false,
                ];
                /** @var \App\Utility\CheckingData[] $checkingData */
                $checkingData = [];
                $notificationIds = [];
                foreach ($Entity->users as $User) {
                    /** @var \App\Model\Entity\DomainsUsers $DomainsUsers */
                    $DomainsUsers = $User->_joinData;
                    $isInit = null === $DomainsUsers->data;
                    if ($isInit) {
                        $DomainsUsers->data = [];
                    }
                    $CheckingData = new CheckingData($DomainsUsers->data,
['markNew' => $isInit, 'markClean' => !$isInit]);

                    if ($CheckingData->checkUptimeRequired()) {
                        $options['uptime'] = true;
                        $notificationIds []= $CheckingData-
>uptime_notification_id;
                    }
                    if ($CheckingData->checkHttpStatusRequired()) {
                        $options['status'] = true;
                        $options['whois'] = true;
                        $notificationIds []= $CheckingData-
>http_status_notification_id;
                        $notificationIds []= $CheckingData-
>ssl_cert_notification_id;
                        $notificationIds []= $CheckingData-
>robots_tag_notification_id;
                    }
                    if ($CheckingData->checkRobotsTxtRequired()) {
```

```
                        $options['robots_txt'] = true;
                        $notificationIds []= $CheckingData-
>robots_txt_notification_id;
                  }
                  $checkingData[$User->id] = $CheckingData;
            }

            $this->_checkDomain($Entity, $options);

            $this->Users->loadInto($Entity->users, ['Notifications' =>
function (Query $Query) use ($notificationIds) {
                  return $Query->where(['Notifications.id IN' =>
$notificationIds]);
            }]);

            foreach ($Entity->users as $User) {
                  /** @var \App\Model\Entity\DomainsUsers $DomainsUsers */
                  $DomainsUsers = $User->_joinData;
                  $CheckingData = $checkingData[$User->id];
                  /** @var \App\Model\Entity\Notification[] $notifications
*/
                  $notifications = Hash::combine($User->notifications,
'{*}.type', '{*}');

                  if ($CheckingData->checkUptimeRequired()) {
                        $fields = CheckingData::UPTIME_FIELDS;
                        $data = $Entity->extract($fields);
                        $CheckingData->set($data);
                        [$isError, $date] = [$Entity-
>uptime_check_is_failed, $CheckingData->uptime_checked_date];
                        if (null !== $isError) {
                              $type = $isError ?
Notification::TYPE_DOMAIN_UPTIME_CHECK_FAILED :
Notification::TYPE_DOMAIN_UPTIME_CHECK_SUCCESS;
                              $isNew = null === $CheckingData-
>getOriginal('uptime_check_is_failed');
                              if (!isset($notifications[$type])) {
                                    $Notification = $NotificationsTable-
>newEmptyEntity()->set(['object_class' => 'Domains', 'object_id' => $id,
'object_title' => $url]);
                                    $Notification->sendEmail = $isError ||
!$isNew;
```

```
                                        $Notification->is_alert =
$Notification->sendEmail;
                                        $Notification->type = $type;
                                        $Notification->data_original = !$isNew
? $CheckingData->extractOriginal($fields) : null;
                                        $notifications[$type] = $Notification;
                                  }
                                  $Notification = $notifications[$type];
                                  $dates = $Notification->data['dates'] ?? [];
                                  array_unshift($dates, $date);
                                  $Notification->data = $data + ['dates' =>
$dates];
                            }
                      }

                      if ($CheckingData->checkHttpStatusRequired()) {
                            $fields = CheckingData::HTTP_STATUS_FIELDS;
                            $data = $Entity->extract($fields);
                            $CheckingData->set($data);
                            [$isError, $date] = [$Entity-
>http_status_is_invalid, $CheckingData->http_status_checked_date];
                            if (null !== $isError) {
                                  $type = $isError ?
Notification::TYPE_DOMAIN_HTTP_STATUS_CHECK_FAILED :
Notification::TYPE_DOMAIN_HTTP_STATUS_CHECK_SUCCESS;
                                  $isNew = null === $CheckingData-
>getOriginal('http_status_is_invalid');
                                  if (!isset($notifications[$type])) {
                                        $Notification = $NotificationsTable-
>newEmptyEntity()->set(['object_class' => 'Domains', 'object_id' => $id,
'object_title' => $url]);
                                        $Notification->sendEmail = $isError ||
!$isNew;
                                        $Notification->is_alert =
$Notification->sendEmail;
                                        $Notification->type = $type;
                                        $Notification->data_original = !$isNew
? $CheckingData->extractOriginal($fields) : null;
                                        $notifications[$type] = $Notification;
                                  }
                                  $Notification = $notifications[$type];

                                  $dates = $Notification->data['dates'] ?? [];
```

```
                                    array_unshift($dates, $date);
                                    $Notification->data = $data + ['dates' =>
$dates];
                              }

                              $fields = CheckingData::SSL_CERT_FIELDS;
                              $data = $Entity->extract($fields);
                              $CheckingData->set($data);
                              $data += ['http_status_checked_date' => $Entity-
>http_status_checked_date];
                              [$isError, $date] = [$Entity->ssl_cert_is_expired,
$CheckingData->http_status_checked_date];
                              if (null !== $isError) {
                                    $type = $isError ?
Notification::TYPE_DOMAIN_SSL_CERT_STATUS_CHECK_FAILED :
Notification::TYPE_DOMAIN_SSL_CERT_STATUS_CHECK_SUCCESS;
                                    $isNew = null === $CheckingData-
>getOriginal('ssl_cert_is_expired');
                                    if (!isset($notifications[$type])) {
                                          $Notification = $NotificationsTable-
>newEmptyEntity()->set(['object_class' => 'Domains', 'object_id' => $id,
'object_title' => $url]);
                                          $Notification->sendEmail = $isError ||
!$isNew;
                                          $Notification->is_alert =
$Notification->sendEmail;
                                          $Notification->type = $type;
                                          $Notification->data_original = !$isNew
? $CheckingData->extractOriginal($fields) : null;
                                          $notifications[$type] = $Notification;
                                    }
                                    $Notification = $notifications[$type];
                                    $dates = $Notification->data['dates'] ?? [];
                                    array_unshift($dates, $date);
                                    $Notification->data = $data + ['dates' =>
$dates];
                              }

                              $fields = CheckingData::ROBOTS_TAG_FIELDS;
                              $data = $Entity->extract($fields);
                              $CheckingData->set($data);
                              $data += ['http_status' => $Entity->http_status];
```

```
                              [$isError, $date] = [$Entity-
>robots_tag_is_noindex, $CheckingData->robots_tag_checked_date];
                              if (null !== $isError) {
                                    $type = $isError ?
Notification::TYPE_DOMAIN_ROBOTS_TAG_CHECK_FAILED :
Notification::TYPE_DOMAIN_ROBOTS_TAG_CHECK_SUCCESS;
                                    $isNew = null === $CheckingData-
>getOriginal('robots_tag_is_noindex');
                                    if (!isset($notifications[$type])) {
                                          $Notification = $NotificationsTable-
>newEmptyEntity()->set(['object_class' => 'Domains', 'object_id' => $id,
'object_title' => $url]);
                                          $Notification->sendEmail = $isError ||
!$isNew;
                                          $Notification->is_alert =
$Notification->sendEmail;
                                          $Notification->type = $type;
                                          $Notification->data_original = !$isNew
? $CheckingData->extractOriginal($fields) : null;
                                          $notifications[$type] = $Notification;
                                    }
                                    $Notification = $notifications[$type];
                                    $dates = $Notification->data['dates'] ?? [];
                                    array_unshift($dates, $date);
                                    $Notification->data = $data + ['dates' =>
$dates];
                              }
                        }

                        if ($CheckingData->checkRobotsTxtRequired()) {
                              $fields = CheckingData::ROBOTS_TXT_FIELDS;
                              $data = $Entity->extract($fields);
                              $CheckingData->set($data);
                              [$isError, $date] = [$Entity-
>robots_txt_disallow_all, $CheckingData->robots_txt_checked_date];
                              if (null !== $isError) {
                                    $type = $isError ?
Notification::TYPE_DOMAIN_ROBOTS_TXT_CHECK_FAILED :
Notification::TYPE_DOMAIN_ROBOTS_TXT_CHECK_SUCCESS;
                                    $isNew = null === $CheckingData-
>getOriginal('robots_txt_disallow_all');
                                    if (!isset($notifications[$type])) {
```

```
                                        $Notification = $NotificationsTable-
>newEmptyEntity()->set(['object_class' => 'Domains', 'object_id' => $id,
'object_title' => $url]);
                                        $Notification->sendEmail = $isError ||
!$isNew;
                                        $Notification->is_alert =
$Notification->sendEmail;
                                        $Notification->type = $type;
                                        $Notification->data_original = !$isNew
? $CheckingData->extractOriginal($fields) : null;
                                        $notifications[$type] = $Notification;
                                  }
                                  $Notification = $notifications[$type];
                                  $dates = $Notification->data['dates'] ?? [];
                                  array_unshift($dates, $date);
                                  $Notification->data = $data + ['dates' =>
$dates];
                            }
                      }

                      $User->notifications = array_values($notifications);
                      foreach ($notifications as $Notification) {
                            if ($Notification->isDirty()) {
                                  $User->setDirty('notifications');
                                  break;
                            }
                      }

                      $DomainsUsers->data = $CheckingData->toArray();
                      $User->setDirty('_joinData', $DomainsUsers-
>isDirty('data'));
                      $Entity->extendDirty('users', $User->isDirty());
                }

                $this->Domains->saveOrFail($Entity);

                foreach ($Entity->users as $User) {
                      /** @var \App\Model\Entity\DomainsUsers $DomainsUsers */
                      $DomainsUsers = $User->_joinData;
                      $CheckingData = $checkingData[$User->id];
                      foreach ($User->notifications as $Notification) {
                            $type = $Notification->type;
```

```
                        if (in_array($type,
[Notification::TYPE_DOMAIN_UPTIME_CHECK_FAILED,
Notification::TYPE_DOMAIN_UPTIME_CHECK_SUCCESS], true))
                            $CheckingData->uptime_notification_id =
$Notification->id;
                        elseif (in_array($type,
[Notification::TYPE_DOMAIN_HTTP_STATUS_CHECK_FAILED,
Notification::TYPE_DOMAIN_HTTP_STATUS_CHECK_SUCCESS], true))
                            $CheckingData->http_status_notification_id =
$Notification->id;
                        elseif (in_array($type,
[Notification::TYPE_DOMAIN_SSL_CERT_STATUS_CHECK_FAILED,
Notification::TYPE_DOMAIN_SSL_CERT_STATUS_CHECK_SUCCESS], true))
                            $CheckingData->ssl_cert_notification_id =
$Notification->id;
                        elseif (in_array($type,
[Notification::TYPE_DOMAIN_ROBOTS_TAG_CHECK_FAILED,
Notification::TYPE_DOMAIN_ROBOTS_TAG_CHECK_SUCCESS], true))
                            $CheckingData->robots_tag_notification_id =
$Notification->id;
                        elseif (in_array($type,
[Notification::TYPE_DOMAIN_ROBOTS_TXT_CHECK_FAILED,
Notification::TYPE_DOMAIN_ROBOTS_TXT_CHECK_SUCCESS], true))
                            $CheckingData->robots_txt_notification_id =
$Notification->id;
                        elseif (in_array($type,
[Notification::TYPE_DOMAIN_EXPIRATION_CHECK_FAILED,
Notification::TYPE_DOMAIN_EXPIRATION_CHECK_SUCCESS], true))
                            $CheckingData->domain_expiry_notification_id
= $Notification->id;
                    }
                    $DomainsUsers->data = $CheckingData->toArray();
                    $User->setDirty('_joinData', $DomainsUsers-
>isDirty('data'));
                    $Entity->extendDirty('users', $User->isDirty());
                }
                $this->Domains->saveOrFail($Entity);

                foreach ($Entity->users as $User) {
                    foreach ($User->notifications as $Notification) {
                        if (!$Notification->sendEmail)
                            continue;
```

```
                                $Mailer = new Mailer();
                                $Mailer->domainAlert($Entity, $User,
$Notification);
                            }
                        }
                    }
                }
```

Domain.php

```
<?php
declare(strict_types=1);

namespace App\Model\Entity;

use Cake\ORM\TableRegistry;
use Cake\Utility\Security;

/**
 * @property int $id
 * @property string $domain
 * @property string $url
 * @property string|null $ip
 * @property int|null $http_status
 * @property bool|null $http_status_is_invalid
 * @property string|null $http_reason_phrase {@uses
\App\Model\Entity\Domain::_setHttpReasonPhrase()}
 * @property \Cake\I18n\FrozenTime|null $http_status_checked_date
 * @property \Cake\I18n\FrozenTime|null $http_status_changed_date
 * @property string|null $redirect_location {@uses
\App\Model\Entity\Domain::_setRedirectLocation()}
 * @property bool|null $ssl_cert_is_expired
 * @property \Cake\I18n\FrozenTime|null $ssl_cert_expires_date
 * @property string|null $ssl_cert_issuer
 * @property \Cake\I18n\FrozenTime|null $ssl_cert_status_changed_date
 * @property string|null $robots_tag {@uses
\App\Model\Entity\Domain::_setRobotsTag()}
 * @property bool|null $robots_tag_is_noindex
 * @property bool|null $robots_tag_is_nofollow
 * @property \Cake\I18n\FrozenTime|null $robots_tag_checked_date
```

```
 * @property \Cake\I18n\FrozenTime|null $robots_tag_status_changed_date
 * @property bool|null $robots_txt_disallow_all
 * @property string|null $robots_txt_content {@uses
\App\Model\Entity\Domain::_setRobotsTxtContent()}
 * @property \Cake\I18n\FrozenTime|null $robots_txt_checked_date
 * @property \Cake\I18n\FrozenTime|null $robots_txt_status_changed_date
 * @property int|null $uptime_attempts
 * @property bool|null $uptime_check_is_failed
 * @property int|null $uptime_failed_attempts
 * @property int|null $uptime_latency
 * @property \Cake\I18n\FrozenTime|null $uptime_checked_date
 * @property \Cake\I18n\FrozenTime|null $uptime_status_changed_date
 * @property \Cake\I18n\FrozenTime|null $domain_created
 * @property bool|null $domain_is_expired
 * @property \Cake\I18n\FrozenTime|null $domain_expiry_status_change_date
 * @property \Cake\I18n\FrozenTime|null $domain_expires
 * @property array|null $name_servers
 * @property string $owner_name
 * @property string $owner_address
 * @property string $owner_city
 * @property string $owner_zip
 * @property string $owner_country
 * @property \Cake\I18n\FrozenTime|null $domain_whois_date
 * @property string $isp
 * @property string|null $title {@uses \App\Model\Entity\Domain::_setTitle()}
 * @property \Cake\I18n\FrozenTime|null $created
 * @property \Cake\I18n\FrozenTime|null $modified
 *
 * @property \App\Model\Entity\User[] $users
 * @property \App\Model\Entity\DomainsUsers[] $domains_users
 * @property \App\Model\Entity\DomainConnect[] $domain_connects
 *
 * @property-read bool $has_errors {@uses
\App\Model\Entity\Domain::_getHasErrors()}
 * @property-read string $host {@uses \App\Model\Entity\Domain::_getHost()}
 * @property-read string $country_code {@uses
\App\Model\Entity\Domain::_getCountryCode()}
 */
class Domain extends Entity {

      protected $_hidden = ['users'];

      public const CHECKING_UPTIME = 15*MINUTE;
```

```
public const CHECKING_STATUS = 24*HOUR;
public const CHECKING_ROBOTS_TXT = 24*HOUR;

public function isChecked(): bool {
	$fields = [
		'http_status_checked_date',
		'robots_tag_checked_date',
		'robots_txt_checked_date',
		'uptime_checked_date',
		'safe_browsing_checked_date',
		'phishtank_checked_date',
		'dns_records_checked_date',
	];

	foreach ($fields as $field) {
		if (null !== $this->get($field))
			return true;
	}

	return false;
}

public static function getErrorFields(): array {
	return [
		'uptime_check_is_failed',
		'http_status_is_invalid',
		'robots_tag_is_noindex',
		'robots_txt_disallow_all',
		'ssl_cert_is_expired',
	];
}

protected function _getHasErrors(): bool {
	foreach (static::getErrorFields() as $field) {
		if ($this->get($field))
			return true;
	}

	return false;
}

protected function _getHost(): string {
	return parse_url($this->url, PHP_URL_HOST);
```

```
	}

	protected function _setTitle(?string $value): ?string {
		return null === $value ? null : $this->trim('title', $value);
	}

	protected function _setHttpReasonPhrase(?string $value): ?string {
		return null === $value ? null : $this->trim('http_reason_phrase',
$value);
	}

	protected function _setRedirectLocation(?string $value): ?string {
		return null === $value ? null : $this->trim('redirect_location',
$value);
	}

	protected function _setRobotsTag(?string $value): ?string {
		return null === $value ? null : $this->trim('robots_tag', $value);
	}

	protected function _setRobotsTxtContent(?string $value): ?string {
		return null === $value ? null : $this->trim('robots_txt_content',
$value);
	}

	protected function trim(string $field, string $value): string {
		$length = TableRegistry::getTableLocator()->get($this->getSource())-
>getSchema()->getColumn($field)['length'];
		if (mb_strlen($value) > $length)
			$value = mb_substr($value, 0, $length-1) . '…';

		return $value;
	}

	protected function _getCountryCode() {
		$db = getIp2locationSource();
		$records = $db->lookup($this->ip, \IP2Location\Database::ALL);

		return $records['countryCode'] ?? '';
	}

	public function getViewUrl(): array {
```

```
            return ['controller' => 'Domains', 'action' => 'view', '?' => ['url'
=> $this->url]];
      }

      public function getDescription(): string {
            $this->requireProperties(['id', 'url']);
            return "#{$this->id} \"{$this->url}\"";
      }
}
```

DomainConnects.php

```
<?php
declare(strict_types=1);

namespace App\Model\Entity;

/**
 * @property int $id
 * @property int $domain_id
 * @property int $namelookup_time
 * @property int $connect_time
 * @property int $appconnect_time
 * @property int $pretransfer_time
 * @property int $starttransfer_time
 * @property int $total_time
 * @property \Cake\I18n\FrozenDate $date
 * @property int $attempts
 *
 * @property \App\Model\Entity\Domain $domain
 */
class DomainConnect extends Entity {

}
```

DomainsUsers.php

```
<?php
declare(strict_types=1);
```

```
namespace App\Model\Entity;

/**
 * @property int $id
 * @property int $domain_id
 * @property int $user_id
 * @property array|null $data
 * @property string $alerts_email_1
 * @property string $alerts_email_2
 * @property string $alerts_email_3
 * @property \Cake\I18n\FrozenTime|null $created
 * @property \Cake\I18n\FrozenTime|null $modified
 *
 * @property \App\Model\Entity\Domain $domain
 * @property \App\Model\Entity\User $user
 * @property \App\Model\Entity\Subscription|null $subscription
 */
class DomainsUsers extends Entity {

}
```

Notification.php

```
<?php
declare(strict_types=1);

namespace App\Model\Entity;

/**
 * @property int $id
 * @property int|null $user_id
 * @property string $object_class
 * @property int|null $object_id
 * @property string $object_title
 * @property string $type
 * @property bool $is_alert
 * @property bool $is_read
 * @property array|null $data
 * @property array|null $data_original
 * @property \Cake\I18n\FrozenTime|null $created
 * @property \Cake\I18n\FrozenTime|null $modified
```

```
 *
 * @property \App\Model\Entity\User|null $user
 */
class Notification extends Entity {

	public bool $sendEmail = false;

	public const TYPE_DOMAIN_UPTIME_CHECK_SUCCESS =
'domain_uptime_check_success';
	public const TYPE_DOMAIN_UPTIME_CHECK_FAILED =
'domain_uptime_check_failed';
	public const TYPE_DOMAIN_HTTP_STATUS_CHECK_SUCCESS =
'domain_http_status_check_success';
	public const TYPE_DOMAIN_HTTP_STATUS_CHECK_FAILED =
'domain_http_status_check_failed';
	public const TYPE_DOMAIN_SSL_CERT_STATUS_CHECK_SUCCESS =
'domain_ssl_cert_status_check_success';
	public const TYPE_DOMAIN_SSL_CERT_STATUS_CHECK_FAILED =
'domain_ssl_cert_status_check_failed';
	public const TYPE_DOMAIN_ROBOTS_TAG_CHECK_SUCCESS =
'domain_robots_tag_check_success';
	public const TYPE_DOMAIN_ROBOTS_TAG_CHECK_FAILED =
'domain_robots_tag_check_failed';
	public const TYPE_DOMAIN_ROBOTS_TXT_CHECK_SUCCESS =
'domain_robots_txt_check_success';
	public const TYPE_DOMAIN_ROBOTS_TXT_CHECK_FAILED =
'domain_robots_txt_check_failed';
	public function getTitle(): string {
		$isNew = null === $this->data_original;
		switch ($this->type) {
			case static::TYPE_DOMAIN_UPTIME_CHECK_SUCCESS: return $isNew ?
__('Server downtime checked') : __('Server downtime solved');
			case static::TYPE_DOMAIN_UPTIME_CHECK_FAILED: return
__('Server downtime detected');
			case static::TYPE_DOMAIN_HTTP_STATUS_CHECK_SUCCESS: return
$isNew ? __('HTTP status code checked') : __('HTTP status code solved');
			case static::TYPE_DOMAIN_HTTP_STATUS_CHECK_FAILED: return
__('HTTP status code change detected');
			case static::TYPE_DOMAIN_SSL_CERT_STATUS_CHECK_SUCCESS: return
$isNew ? __('SSL certificate checked') : __('SSL certificate solved');
			case static::TYPE_DOMAIN_SSL_CERT_STATUS_CHECK_FAILED: return
__('Expired SSL certificate detected');
```

```
                case static::TYPE_DOMAIN_ROBOTS_TAG_CHECK_SUCCESS: return
$isNew ? __('NoIndex checked') : __('NoIndex solved');
                case static::TYPE_DOMAIN_ROBOTS_TAG_CHECK_FAILED: return
__('NoIndex detected');
                case static::TYPE_DOMAIN_ROBOTS_TXT_CHECK_SUCCESS: return
$isNew ? __('Disallow in robots.txt checked') : __('Disallow in robots.txt
solved');
                case static::TYPE_DOMAIN_ROBOTS_TXT_CHECK_FAILED: return
__('Disallow in robots.txt detected');
            }

            return ucfirst(str_replace('_', ' ', $this->type));
      }
}
```

User.php

```
<?php
declare(strict_types=1);

namespace App\Model\Entity;

use Authentication\PasswordHasher\DefaultPasswordHasher;

/**
 * @property int $id
 * @property string $email
 * @property string|null $password
 * @property string $first_name
 * @property string $last_name
 * @property \Cake\I18n\FrozenTime|null $signup_date
 * @property \Cake\I18n\FrozenTime|null $created
 * @property \Cake\I18n\FrozenTime|null $modified
 *
 * @property string $full_name {@uses \App\Model\Entity\User::_getFullName()}
 * @property string $description {@uses
\App\Model\Entity\User::_getDescription()}
 *
 * @property \App\Model\Entity\Notification[] $notifications
 * @property \App\Model\Entity\UserAction[] $user_actions
 * @property \App\Model\Entity\UserMeta[] $user_meta
```

```
 * @property \App\Model\Entity\BusinessDomain[] $managed_business_domains
 * @property \App\Model\Entity\Domain[] $domains
 */
class User extends Entity {

	protected $_hidden = ['password'];


	protected function _setPassword(string $password) {
		$hasher = new DefaultPasswordHasher();

		return $hasher->hash($password);
	}

	protected function _getFullName(): string {
		if (!$this->has(['first_name', 'last_name']))
			return '';

		return trim($this->first_name.' '.$this->last_name);
	}

	protected function _getDescription(): string {
		return "#{$this->id} \"{$this->full_name}\" <{$this->email}>";
	}
}
```

UserMeta.php

```
<?php
declare(strict_types=1);

namespace App\Model\Entity;

/**
 * @property int $id
 * @property int $user_id
 * @property string $meta_key
 * @property array|null $meta_value
 *
 * @property \App\Model\Entity\User $user
 */
```

```
class UserMeta extends \Diplom\Model\Entity\Meta {

}
```

DomainConnectsTable.php

```
<?php
declare(strict_types=1);

namespace App\Model\Table;

use Cake\ORM\Query;

/**
 * @property \App\Model\Table\DomainsTable&\Cake\ORM\Association\BelongsTo
$Domains
 *
 * @method \App\Model\Entity\DomainConnect newEmptyEntity()
 * @method \App\Model\Entity\DomainConnect newEntity(array $data, array $options
= [])
 * @method \App\Model\Entity\DomainConnect[] newEntities(array $data, array
$options = [])
 * @method \App\Model\Entity\DomainConnect get($primaryKey, $options = [])
 * @method \App\Model\Entity\DomainConnect findOrCreate($search, ?callable
$callback = null, $options = [])
 * @method \App\Model\Entity\DomainConnect
patchEntity(\Cake\Datasource\EntityInterface $entity, array $data, array
$options = [])
 * @method \App\Model\Entity\DomainConnect[] patchEntities(iterable $entities,
array $data, array $options = [])
 * @method \App\Model\Entity\DomainConnect|false
save(\Cake\Datasource\EntityInterface $entity, $options = [])
 * @method \App\Model\Entity\DomainConnect
saveOrFail(\Cake\Datasource\EntityInterface $entity, $options = [])
 * @method
\App\Model\Entity\DomainConnect[]|\Cake\Datasource\ResultSetInterface|false
saveMany(iterable $entities, $options = [])
 * @method \App\Model\Entity\DomainConnect[]|\Cake\Datasource\ResultSetInterface
saveManyOrFail(iterable $entities, $options = [])
```

```
 * @method
\App\Model\Entity\DomainConnect[]|\Cake\Datasource\ResultSetInterface|false
deleteMany(iterable $entities, $options = [])
 * @method \App\Model\Entity\DomainConnect[]|\Cake\Datasource\ResultSetInterface
deleteManyOrFail(iterable $entities, $options = [])
 */
class DomainConnectsTable extends Table {

      public function initialize(array $config): void {
            parent::initialize($config);
            $this->belongsTo('Domains');
      }

      public function findByDomainId(int $domainId): Query {
            return $this->find()->where([$this-
>aliasField('domain_id')=>$domainId]);
      }
}
```

DomainsTable.php

```
<?php
declare(strict_types=1);

namespace App\Model\Table;

use Cake\ORM\Query;

/**
 * @property \App\Model\Table\UsersTable&\Cake\ORM\Association\BelongsToMany
$Users
 * @property \App\Model\Table\DomainsUsersTable&\Cake\ORM\Association\HasMany
$DomainsUsers
 *
 * @method \App\Model\Entity\Domain newEmptyEntity()
 * @method \App\Model\Entity\Domain newEntity(array $data, array $options = [])
 * @method \App\Model\Entity\Domain[] newEntities(array $data, array $options =
[])
 * @method \App\Model\Entity\Domain get($primaryKey, $options = [])
 * @method \App\Model\Entity\Domain findOrCreate($search, ?callable $callback =
null, $options = [])
```

```
 * @method \App\Model\Entity\Domain patchEntity(\Cake\Datasource\EntityInterface
$entity, array $data, array $options = [])
 * @method \App\Model\Entity\Domain[] patchEntities(iterable $entities, array
$data, array $options = [])
 * @method \App\Model\Entity\Domain|false save(\Cake\Datasource\EntityInterface
$entity, $options = [])
 * @method \App\Model\Entity\Domain saveOrFail(\Cake\Datasource\EntityInterface
$entity, $options = [])
 * @method \App\Model\Entity\Domain[]|\Cake\Datasource\ResultSetInterface|false
saveMany(iterable $entities, $options = [])
 * @method \App\Model\Entity\Domain[]|\Cake\Datasource\ResultSetInterface
saveManyOrFail(iterable $entities, $options = [])
 * @method \App\Model\Entity\Domain[]|\Cake\Datasource\ResultSetInterface|false
deleteMany(iterable $entities, $options = [])
 * @method \App\Model\Entity\Domain[]|\Cake\Datasource\ResultSetInterface
deleteManyOrFail(iterable $entities, $options = [])
 *
 * @mixin \Cake\ORM\Behavior\TimestampBehavior
 * @property \App\Model\Table\DomainConnectsTable&\Cake\ORM\Association\HasMany
$DomainConnects
 */
class DomainsTable extends Table {

      public function initialize(array $config): void {
            $this->getSchema()->setColumnType('name_servers', 'json');
            $this->addBehavior('Timestamp');
            $this->belongsToMany('Users')
                  ->setSaveStrategy('append');
            $this->hasMany('DomainsUsers')
                  ->setSaveStrategy('append');
            $this->hasMany('DomainConnects')
                  ->setSaveStrategy('append');
      }

      /**
       * @param \App\Validation\Validator|\Cake\Validation\Validator $Validator
       * @return \App\Validation\Validator
       */
      public function validationDefault(\Cake\Validation\Validator $Validator):
\Cake\Validation\Validator {
            return parent::validationDefault($Validator)
                  ->requirePresence($field='domain', 'create')-
>notEmptyString($field)
```

```
                ->requirePresence($field='url', 'create')-
>notEmptyString($field)->unique($field)
                ;
      }

      public function findByDomain(string $domain): Query {
            return $this->find()->where([$this->aliasField('domain')=>$domain]);
      }

      public function findByUrl(string $url): Query {
            return $this->find()->where([$this->aliasField('url')=>$url]);
      }

      public function findByUserId(int $userId): Query {
            return $this->find()
                  ->innerJoinWith('Users', function (Query $Query) use ($userId)
{
                        return $Query->where(['Users.id'=>$userId]);
                  });
      }
}
```

DomainsUsersTable.php

```
<?php
declare(strict_types=1);

namespace App\Model\Table;

use App\Model\Entity\DomainsUsers;
use App\Validation\Validator;

/**
 * @property \App\Model\Table\DomainsTable&\Cake\ORM\Association\BelongsTo
$Domains
 * @property \App\Model\Table\UsersTable&\Cake\ORM\Association\BelongsTo $Users
 *
 * @method \App\Model\Entity\DomainsUsers newEmptyEntity()
 * @method \App\Model\Entity\DomainsUsers newEntity(array $data, array $options
= [])
```

```
 * @method \App\Model\Entity\DomainsUsers[] newEntities(array $data, array
$options = [])
 * @method \App\Model\Entity\DomainsUsers get($primaryKey, $options = [])
 * @method \App\Model\Entity\DomainsUsers findOrCreate($search, ?callable
$callback = null, $options = [])
 * @method \App\Model\Entity\DomainsUsers
patchEntity(\Cake\Datasource\EntityInterface $entity, array $data, array
$options = [])
 * @method \App\Model\Entity\DomainsUsers[] patchEntities(iterable $entities,
array $data, array $options = [])
 * @method \App\Model\Entity\DomainsUsers|false
save(\Cake\Datasource\EntityInterface $entity, $options = [])
 * @method \App\Model\Entity\DomainsUsers
saveOrFail(\Cake\Datasource\EntityInterface $entity, $options = [])
 * @method
\App\Model\Entity\DomainsUsers[]|\Cake\Datasource\ResultSetInterface|false
saveMany(iterable $entities, $options = [])
 * @method \App\Model\Entity\DomainsUsers[]|\Cake\Datasource\ResultSetInterface
saveManyOrFail(iterable $entities, $options = [])
 * @method
\App\Model\Entity\DomainsUsers[]|\Cake\Datasource\ResultSetInterface|false
deleteMany(iterable $entities, $options = [])
 * @method \App\Model\Entity\DomainsUsers[]|\Cake\Datasource\ResultSetInterface
deleteManyOrFail(iterable $entities, $options = [])
 *
 * @mixin \Cake\ORM\Behavior\TimestampBehavior
 */
class DomainsUsersTable extends Table {

      public function initialize(array $config): void {
            parent::initialize($config);
            $this->setEntityClass(DomainsUsers::class);
            $this->getSchema()->setColumnType('data', 'json');
            $this->addBehavior('Timestamp');
            $this->belongsTo('Users');
            $this->belongsTo('Domains');
      }

      public function validationSettings(Validator $Validator): Validator {
            for ($i = 1; $i <= 3; $i++) {
                  $Validator
                        ->requirePresence($field="alerts_email_{$i}")-
>allowEmptyString($field)->maxLengthFromTable($field, $this)->email($field)
```

```
                        ;
            }

            return $Validator;
      }
}
```

NotificationsTable.php

```
<?php
declare(strict_types=1);

namespace App\Model\Table;

/**
 * @property \App\Model\Table\UsersTable&\Cake\ORM\Association\BelongsTo $Users
 *
 * @method \App\Model\Entity\Notification newEmptyEntity()
 * @method \App\Model\Entity\Notification newEntity(array $data, array $options
= [])
 * @method \App\Model\Entity\Notification[] newEntities(array $data, array
$options = [])
 * @method \App\Model\Entity\Notification get($primaryKey, $options = [])
 * @method \App\Model\Entity\Notification findOrCreate($search, ?callable
$callback = null, $options = [])
 * @method \App\Model\Entity\Notification
patchEntity(\Cake\Datasource\EntityInterface $entity, array $data, array
$options = [])
 * @method \App\Model\Entity\Notification[] patchEntities(iterable $entities,
array $data, array $options = [])
 * @method \App\Model\Entity\Notification|false
save(\Cake\Datasource\EntityInterface $entity, $options = [])
 * @method \App\Model\Entity\Notification
saveOrFail(\Cake\Datasource\EntityInterface $entity, $options = [])
 * @method
\App\Model\Entity\Notification[]|\Cake\Datasource\ResultSetInterface|false
saveMany(iterable $entities, $options = [])
 * @method \App\Model\Entity\Notification[]|\Cake\Datasource\ResultSetInterface
saveManyOrFail(iterable $entities, $options = [])
```

```
 * @method
\App\Model\Entity\Notification[]|\Cake\Datasource\ResultSetInterface|false
deleteMany(iterable $entities, $options = [])
 * @method \App\Model\Entity\Notification[]|\Cake\Datasource\ResultSetInterface
deleteManyOrFail(iterable $entities, $options = [])
 *
 * @mixin \Cake\ORM\Behavior\TimestampBehavior
 */
class NotificationsTable extends Table {

      public function initialize(array $config): void {
            parent::initialize($config);
            $this->getSchema()->setColumnType('data', 'json');
            $this->getSchema()->setColumnType('data_original', 'json');
            $this->addBehavior('Timestamp');
            $this->belongsTo('Users');
      }
}
```

UsersMetaTable.php

```
<?php
declare(strict_types=1);

namespace App\Model\Table;

use Cake\ORM\Query;

/**
 * @property \App\Model\Table\UsersTable&\Cake\ORM\Association\BelongsTo $Users
 *
 * @method \App\Model\Entity\UserMeta newEmptyEntity()
 * @method \App\Model\Entity\UserMeta newEntity(array $data, array $options =
[])
 * @method \App\Model\Entity\UserMeta[] newEntities(array $data, array $options
= [])
 * @method \App\Model\Entity\UserMeta get($primaryKey, $options = [])
 * @method \App\Model\Entity\UserMeta findOrCreate($search, ?callable $callback
= null, $options = [])
```

```
 * @method \App\Model\Entity\UserMeta
patchEntity(\Cake\Datasource\EntityInterface $entity, array $data, array
$options = [])
 * @method \App\Model\Entity\UserMeta[] patchEntities(iterable $entities, array
$data, array $options = [])
 * @method \App\Model\Entity\UserMeta|false
save(\Cake\Datasource\EntityInterface $entity, $options = [])
 * @method \App\Model\Entity\UserMeta
saveOrFail(\Cake\Datasource\EntityInterface $entity, $options = [])
 * @method
\App\Model\Entity\UserMeta[]|\Cake\Datasource\ResultSetInterface|false
saveMany(iterable $entities, $options = [])
 * @method \App\Model\Entity\UserMeta[]|\Cake\Datasource\ResultSetInterface
saveManyOrFail(iterable $entities, $options = [])
 * @method
\App\Model\Entity\UserMeta[]|\Cake\Datasource\ResultSetInterface|false
deleteMany(iterable $entities, $options = [])
 * @method \App\Model\Entity\UserMeta[]|\Cake\Datasource\ResultSetInterface
deleteManyOrFail(iterable $entities, $options = [])
 *
 * @mixin \Cake\ORM\Behavior\TimestampBehavior
 */
class UserMetaTable extends Table {

	public function initialize(array $config): void {
		parent::initialize($config);
		$this->setEntityClass('UserMeta');
		$this->getSchema()->setColumnType('meta_value', 'json');
		$this->addBehavior('Timestamp');

		$this->belongsTo('Users');
	}

	public function getValueByUserAndMetaKey(int $userId, string $metaKey) {
		/** @var \App\Model\Entity\UserMeta $Entity */
		$Entity = $this->findByUserAndMetaKey($userId, $metaKey)->first();

		return null !== $Entity ? $Entity->meta_value : null;
	}

	public function findByUser(int $userId): Query {
		return $this->find()->innerJoinWith('Users', function (Query $Query)
use ($userId) {
```

```
                return $Query->where(['Users.id'=>$userId]);
            });
        }

        public function findByUserAndMetaKey(int $userId, string $metaKey): Query
{
            return $this->findByUser($userId)->where(['meta_key'=>$metaKey]);
        }

        public function findByUserAndMetaKeys(int $userId, array $metaKeys): Query
{
            return $this->findByUser($userId)->where(['meta_key
IN'=>$metaKeys]);
        }
}
```

UserTable.php

```
<?php
declare(strict_types=1);

namespace App\Model\Table;

use Cake\Http\Client;
use Cake\ORM\RulesChecker;
use Cake\ORM\TableRegistry;
use Cake\Utility\Hash;
use App\Validation\Validator;

/**
 * @property \App\Model\Table\NotificationsTable&\Cake\ORM\Association\HasMany
$Notifications
 * @property \App\Model\Table\UserMetaTable&\Cake\ORM\Association\HasMany
$UserMeta
 * @property \App\Model\Table\DomainsTable&\Cake\ORM\Association\BelongsToMany
$Domains
 * @property \App\Model\Table\DomainsUsersTable&\Cake\ORM\Association\HasMany
$DomainsUsers
 *
 * @method \App\Model\Entity\User newEmptyEntity()
 * @method \App\Model\Entity\User newEntity(array $data, array $options = [])
```

```
 * @method \App\Model\Entity\User[] newEntities(array $data, array $options =
[])
 * @method \App\Model\Entity\User get($primaryKey, $options = [])
 * @method \App\Model\Entity\User findOrCreate($search, ?callable $callback =
null, $options = [])
 * @method \App\Model\Entity\User patchEntity(\Cake\Datasource\EntityInterface
$entity, array $data, array $options = [])
 * @method \App\Model\Entity\User[] patchEntities(iterable $entities, array
$data, array $options = [])
 * @method \App\Model\Entity\User|false save(\Cake\Datasource\EntityInterface
$entity, $options = [])
 * @method \App\Model\Entity\User saveOrFail(\Cake\Datasource\EntityInterface
$entity, $options = [])
 * @method \App\Model\Entity\User[]|\Cake\Datasource\ResultSetInterface|false
saveMany(iterable $entities, $options = [])
 * @method \App\Model\Entity\User[]|\Cake\Datasource\ResultSetInterface
saveManyOrFail(iterable $entities, $options = [])
 * @method \App\Model\Entity\User[]|\Cake\Datasource\ResultSetInterface|false
deleteMany(iterable $entities, $options = [])
 * @method \App\Model\Entity\User[]|\Cake\Datasource\ResultSetInterface
deleteManyOrFail(iterable $entities, $options = [])
 *
 * @mixin \Cake\ORM\Behavior\TimestampBehavior
 */
class UsersTable extends Table {

      public function initialize(array $config): void {
            $this->addBehavior('Timestamp');

            $this->hasMany('Notifications')
                  ->setSaveStrategy('append');
            $this->hasMany('UserMeta')
                  ->setDependent(true);
            $this->belongsToMany('Domains')
                  ->setSaveStrategy('append');
      }

      /**
       * @param \App\Validation\Validator|\Cake\Validation\Validator $Validator
       * @return \App\Validation\Validator
       */
      public function validationDefault(\Cake\Validation\Validator $Validator):
\Cake\Validation\Validator {
```

```
            $Validator = parent::validationDefault($Validator);

            return $Validator;
      }

      public function addEmailValidationRules(Validator $Validator,
$requirePresence = true): Validator {
            return $Validator->requirePresence($field='email',
$requirePresence)->notEmptyString($field)->maxLengthFromTable($field, $this)-
>email($field)->unique($field);
      }

      public function addPasswordValidationRules(Validator $Validator, array
$options = []): Validator {
            $passwordMode = Hash::get($options, 'passwordMode', true);
            $confirmMode = Hash::get($options, 'confirmMode', true);
            return $Validator
                  ->requirePresence($field='password', $passwordMode)
                        ->notEmptyString($field, null,
is_bool($passwordMode)?!$passwordMode:$passwordMode)
                        ->minLength($field, 8)
                  ->add($field, 'strict-password', ['rule'=>function ($value,
array $context) {
                        if (!preg_match('/[a-zA-Z]/', $value) ||
!preg_match('/\d/', $value))
                              return __d('validation', 'Your password must
contain consist of both numbers and letters.');

                        return true;
                  }])
                  ->requirePresence('password_confirm', $confirmMode)
                  ->add('password_confirm', 'compareWithPassword', [
                        'rule' => ['compareWith', $field],
                        'message' => __d('validation', 'The passwords you
entered are not identical.')
                  ])
                  ;
      }

      public function addNamesValidationRules(Validator $Validator): Validator {
            return $Validator
                  ->requirePresence($field='first_name')-
>maxLengthFromTable($field, $this)
```

```
                ->requirePresence($field='last_name')-
>maxLengthFromTable($field, $this)
                ;
      }

      public function validationSignup(Validator $Validator): Validator {
            $this->addEmailValidationRules($Validator);
            $this->addNamesValidationRules($Validator);
            $this->addPasswordValidationRules($Validator,
['confirmMode'=>false]);
            $this->addTermsValidationRules($Validator);

            return $Validator;
      }
}
```

ДОДАТОК В ОПИС ПРОГРАМИ

1.3 Мови програмування

Для розробки вебсервісу було використано фреймворк CakePHP, реалізований мовою програмування PHP.

Інтегроване середовище розробки, яке було використано під час розробки – JetBrains PhpStorm 2023.

2 Функціональне призначення

2.1 Класи розв'язувальних завдань та призначення програми

Призначенням розроблюваного вебсервісу є поліпшення процесу моніторингу стану вебсайтів та отриманню оповіщень про зміни, не потребуючи додаткових дій від користувача.

Воно повинно реалізовувати наступні функції:

- Реєстрація користувача.
- Авторизація користувача.
- Вихід із системи.
- Моніторинг стану вебсайту за запитом.
- Додавання вебсайту для моніторингу.
- Моніторинг стану вебсайту за розкладом.
- Оповіщення про зміну стану вебсайту.
- Збереження оповіщень про зміну стану вебсайту.
- Перегляд оповіщень про зміну стану вебсайту.
- Збереження тривалості підключення до вебсайту.
- Перегляд тривалості підключення до вебсайту.

2.2 Функціональні обмеження на застосування

Одним з найпоголовніших функціональних обмежень розробленого вебсервісу для моніторингу стану вебсайтів є необхідність підключення до мережі інтернету.

3 Опис логічної структури

3.1 Використовувані методи

Під час розробки та аналізу вебсервісу для моніторингу стану вебсайтів було використано об’єктно-орієнтовані методи та об’єткно-орієнтовану мову програмування PHP.

3.2 Алгоритм програми

Під час створення вебсервісу використано архітектуру MVC. Вебсервіс розподіляється на серверну та клієнтську частини, де браузер виступає в ролі клієнта. Сервером служить комп'ютер або комп'ютери, на яких розташований код програми та база даних.

Надалі наведено послідовність взаємодії клієнта з сервером та сервера з базою даних:

1. Користувач у браузері заходить на вебсайт.

2. Браузер відправляє запит на сервер.

3. Сервер, під управлінням Apache, обробляє запит і передає його далі до вебсервера, де програмне забезпечення обробляє запит.

4. У разі необхідності, вебсервер відправляє запит до бази даних та отримує результат взаємодії з СКБД.

5. Програмне забезпечення формує відповідь для запиту та надсилає її від вебсервера до браузеру користувача.

3.3 Зв’язок вебсервісу з іншими програмами

Розроблений вебсервіс взаємодіє з базою даних MySQL та Apache сервером в операційній системі.

4 Використувані технічні засоби

мінімальні Гб *5 Виклик і завантаження*

Запуск ПЗ здейснюється двома спобсобами:

1) за допомогою введення адреси в браузері

2) під час виконання моніторингу за розкладом на сервері

6 Вхідні та вихідні дані

Вхідні дані програми повинну вводитися користувачем на формах сайту, або завантажуватися з бази даних.

Вихідні дані програми мають відображатися на сайті, або зберігатися в файл.

ДОДАТОК Г ІНСТРУКЦІЯ КОРИСТУВАЧА

Для роботи з вебсервісом необхідно перейти до нього за допомогою веббраузера який підтримує HTML 5.

При переході до вебсервісу перше, що побачить користувач – це головна сторінка (рисунок Г.1).

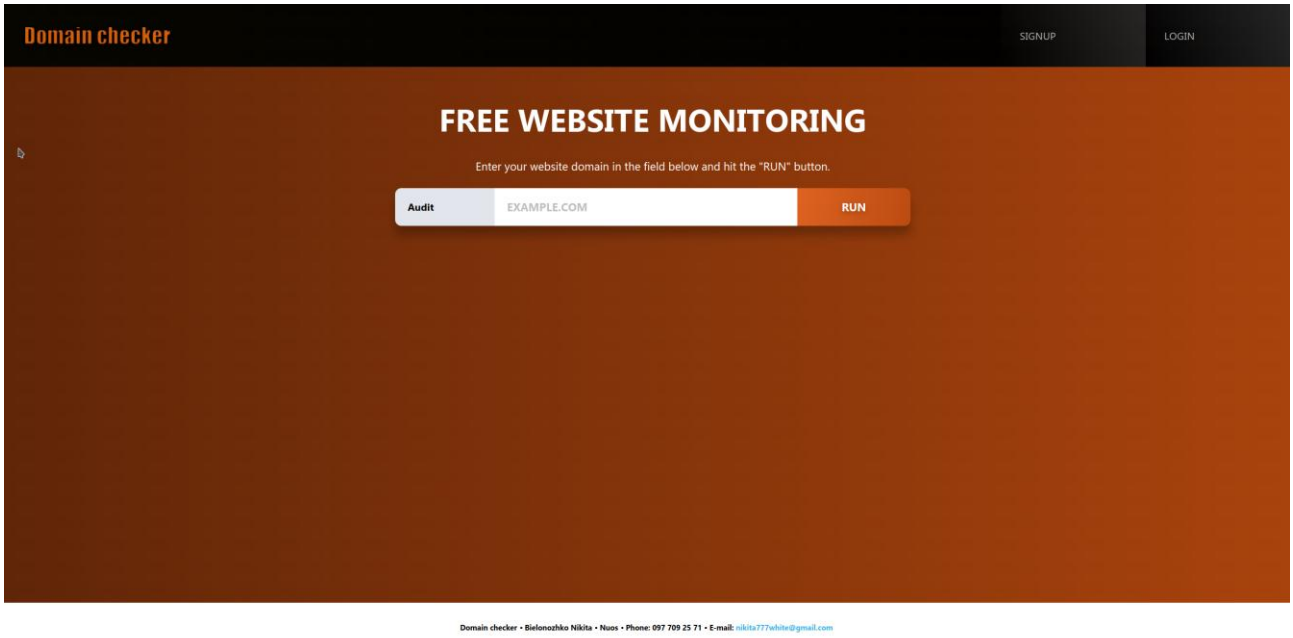


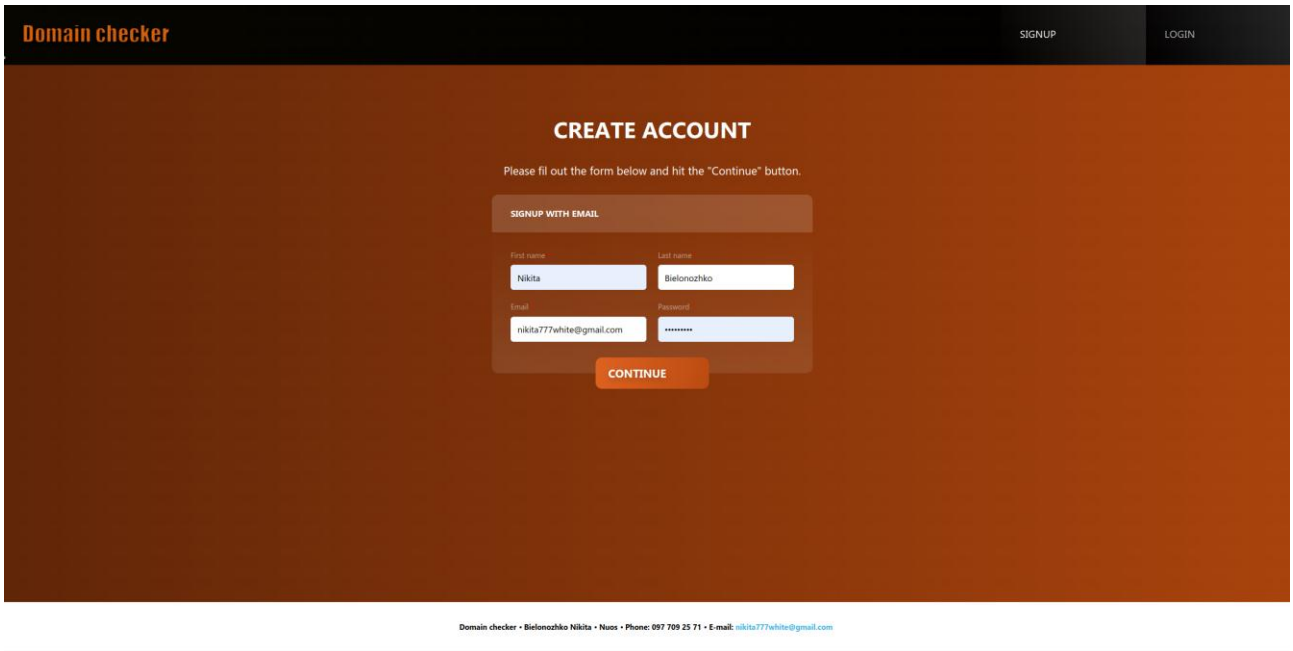
Рисунок Г.1 – Головна сторінка вебсервісу

На цій сторінці розміщена форма для введення адреси вебсайту, для якого користувач бажає виконати моніторинг стану вебсайту за запитом, а також розташовані кнопки авторизації та реєстрації. Внизу сторінки є контактні дані розробника.

Для того, щоб виконати авторизацію або реєстрацію, потрібно натиснути на відповідну кнопку у верхньому правому куті екрану.

Для ініціювання моніторингу стану вебсайту за запитом потрібно ввести URL адресу вебсайту до форми, що розміщена у верхній центральній частині сторінки.

Для реєстрації користувача потрібно заповнити форму, котра з’явиться після натискання кнопки реєстрації. Після заповнення форми потрібно натиснути на кнопку «Continue» (рисунок Г.2).



Г.2 – Реєстрація користувача

За одною адресою електронної пошти можна зареєструвати лише одного користувача.

Щоб авторизувати зареєстрованого користувача, потрібно заповнити форму авторизації та натиснути на кнопку «Login» (рисунок Г.3).

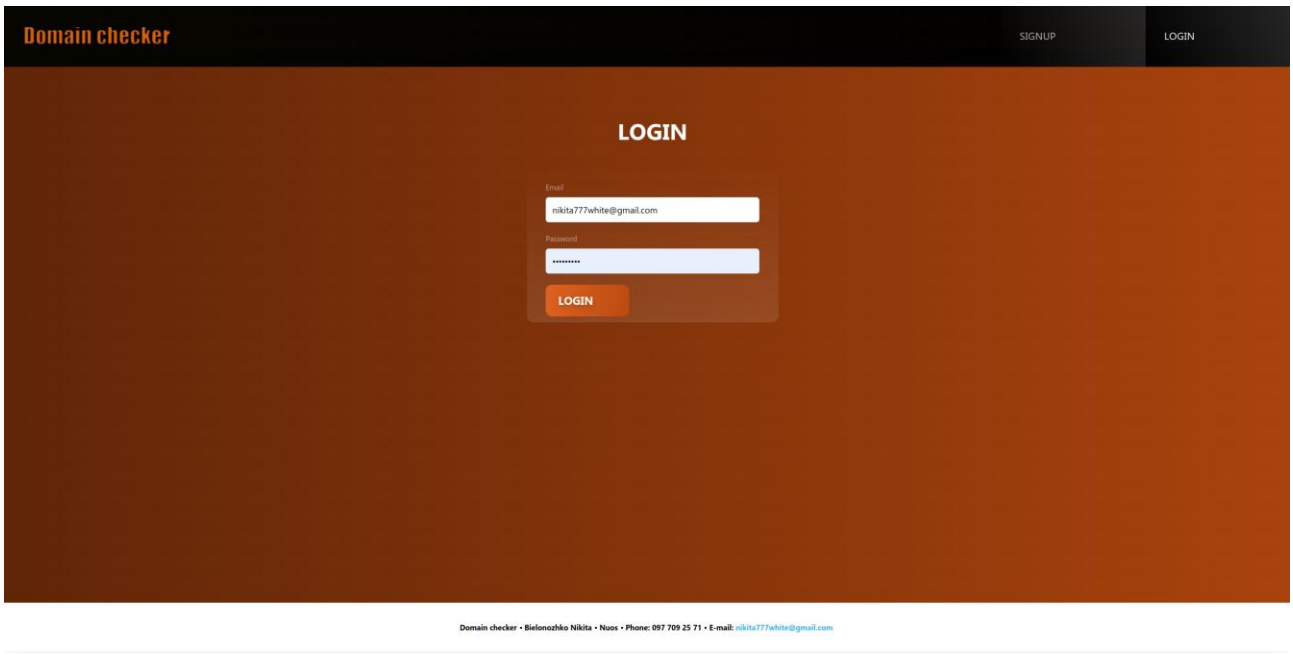


Рисунок Г.3 – Авторизація користувача

Після того, як користувач авторизувався, він має можливість передивитися збережні вебсайти та побачити лічильники проблем з вебсайтами, а також отримувати оповіщення про зміни стану його вебсайтів. Збереженні вебсайти представлено на рисунку Г.4.

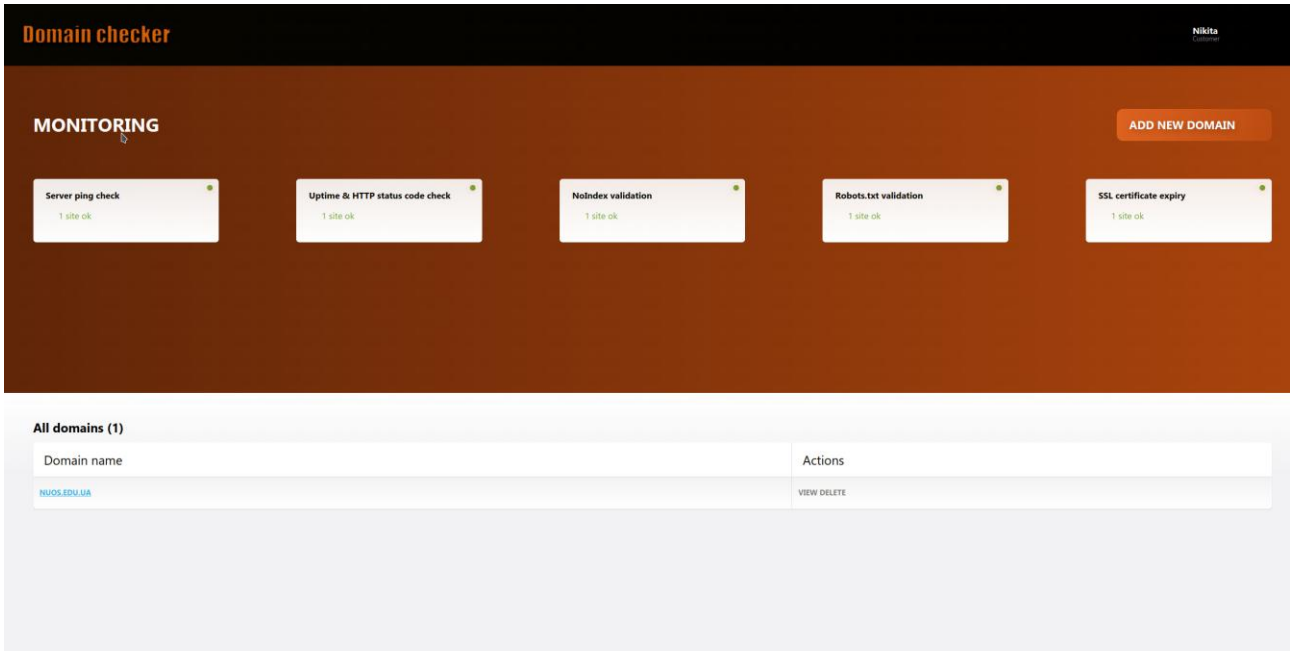


Рисунок Г.4 – Перегляд збережених вебсайтів користувача

Також користувач може натиснути на адресу вебсайту, щоб здійснити моніторинг стану вебсайту за запитом. Якщо користувач натисне кнопку «Add

new domain», він зможе додати новий вебсайт до моніторингу та відразу здійснити моніторинг стану цього вебсайту за запитом.

Для того, щоб користувач міг здійснити моніторинг стану вебсайту за запитом з головної сторінки, потрібно заповнити форму URL адресою сайту та натиснути кнопку «Run». Дії моніторингу стану вебсайту за запитом відображено на малюнках Г.5-Г.6.

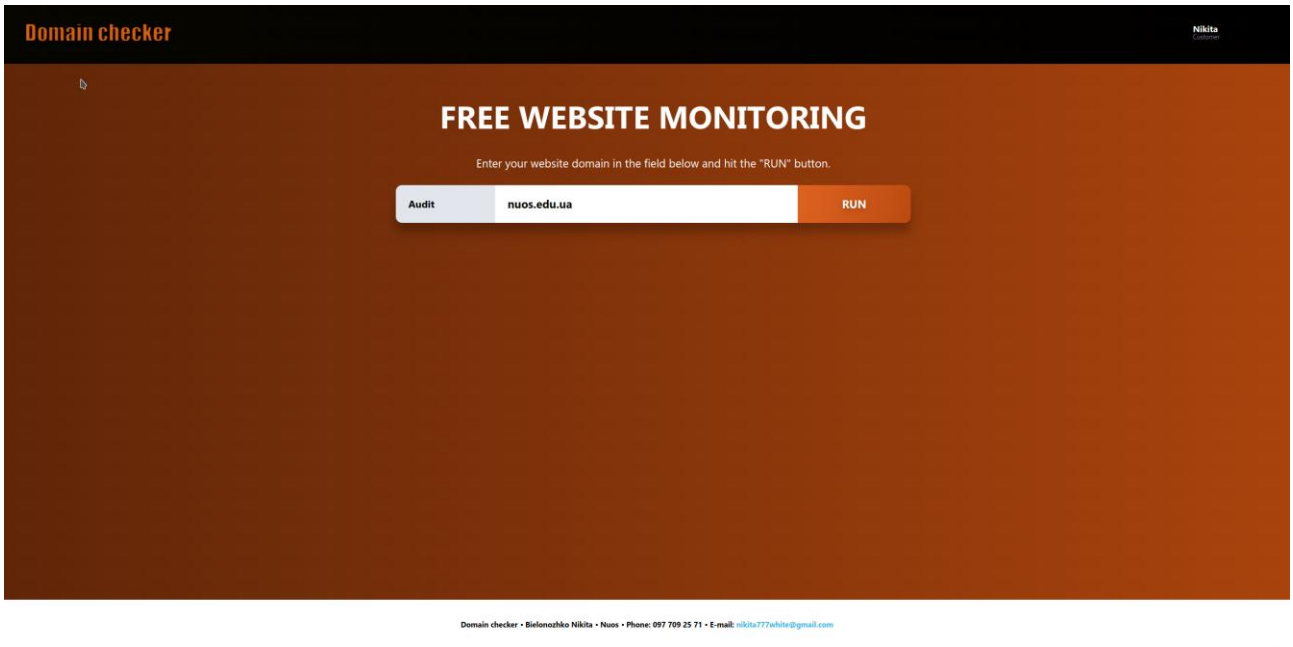


Рисунок Г.5 – Введення URL адреси вебсайту для моніторингу за запитом

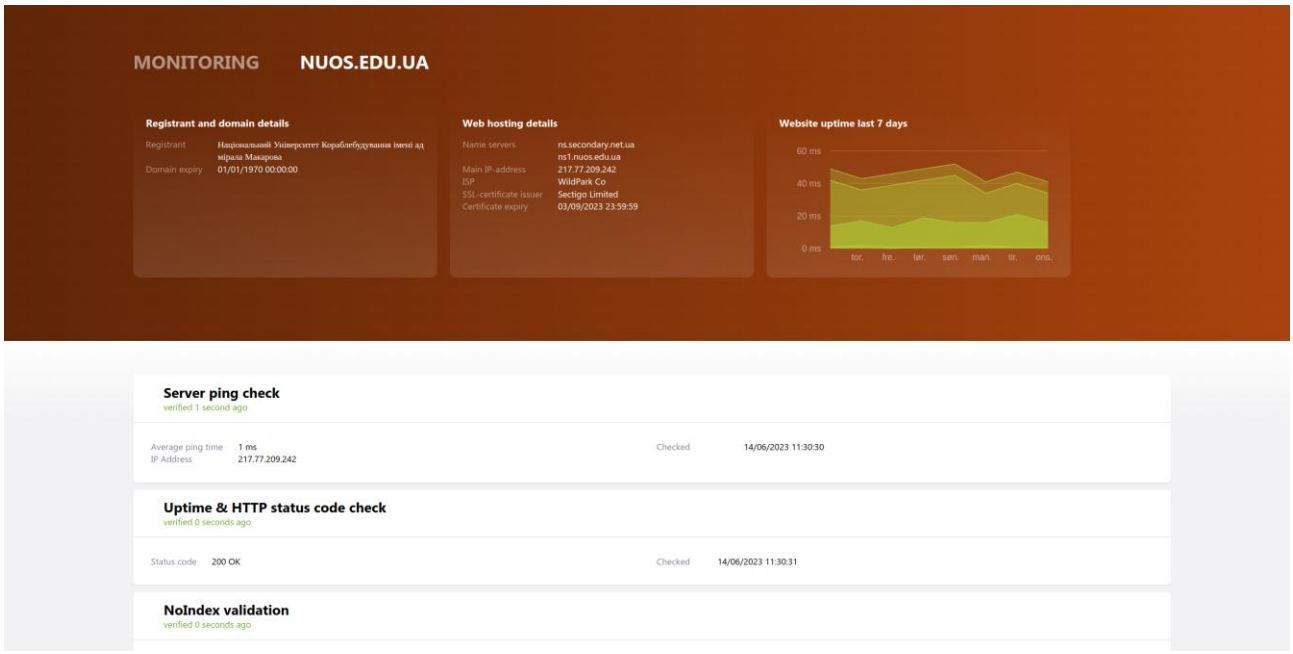


Рисунок Г.6 – Моніторинг стану вебсайту за запитом

ДОДАТОК Д ПРОГРАМА ТА МЕТОДИКА ВИПРОБОВУВАНЬ ВЕБСЕРВІСУ

1 Об’єкт випробовувань

Об’єктом випробовувань є вебсервіс для моніторингу стану вебсайтів.

2 Мета випробовувань

Метою випробовувань є перевірка працездатності вебсервісу. Перевірка передбачає тестування правильності виконання функцій вебсервісу за встановленими вимогами в технічному завданні.

3 Вимоги до програмного забезпечення

Під час проведення випробовувань вебсервісу, функціональні характеристики повинні відповідати вимогам, які було викладено у пункті «Постановка задачі».

4 Вимоги до документації програми

До складу програмної документації повинні входити:

- Технічне завдання.
- Текст програми.
- Опис програми.
- Інструкція користувача.
- Програма та методика випробовувань.

5 Засоби та порядок випробовувань

5.1 Засоби випробувань

Вимоги до складу і параметрів технічних засобів:

- Процесор від 2.0 GHZ.

- Інтернет 80 мбіт/с.
- Оперативна пам’ять від 6 Гб.
- Роздільна здатність екрана: 1080 х 1920 пікселів.

До складу програмних засобів, які мають використовуватися під час випробовувань належать:

- Браузер Chrome версії 112.
- Браузер Firefox версії 112.

5.2 Порядок проведення випробовувань

Порядок проведення випробовувань:

- Перевірка вимог щодо функціональних характеристик.
- Перевірка вимог до програмної документації.