Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Розробка методу та засобів реалізації вебсистеми обліку ставків Вінницької області»

Виконав: студент IV курсу
групи 1ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Миронюк О.В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Войтко В. В.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ОТ Городецька О. С.
(прізвище та ініціали)

Допущено до захисту

Зав. Кафедри ПЗ Романюк О.Н.
«06» червня 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Рівень вищої освіти перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення

УЗГОДЖЕНО
НАЧАЛЬНИК ДЕРЖАВНОЇ
ЕКОЛОГІЧНОЇ ІНСПЕКЦІЇ У
ВІННИЦЬКІЙ ОБЛАСТІ
Дубовий Ю.В.
«_21_» _березня_ 2025 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«_21_» _березня_ 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Миронюку Олександру Володимировичу

1. Тема роботи – «Розробка методу та засобів реалізації вебсистеми обліку ставків Вінницької області».

Керівник роботи: Войтко Вікторія Володимирівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. №97.

2. Термін подання студентом роботи: 2 червня 2025 року.

3. Вихідні дані до роботи: середовище розробки – Visual Studio, система управління базами даних – Microsoft SQL Server; мова запитів – SQL; мови розробки – HTML, CSS, JavaScript, C#; технології Razor Pages, Redis; операційна система – Windows 8/10/11.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз розвитку автоматизованих вебсистем обліку ставків та постановка задач дослідження; розробка методів, моделі та алгоритмів програмного продукту; розробка

програмних модулів вебсистеми обліку ставків вінницької області; тестування програми; висновки; список використаних джерел; додатки; ілюстративна частина.

5. Перелік ілюстративного матеріалу: титульний слайд – автор, науковий керівник бакалаврської кваліфікаційної роботи, тема; мета, об'єкт та предмет дослідження; задачі дослідження; наукова новизна; практична цінність отриманих результатів; аналіз аналогів; модель роботи системи; блок схема алгоритму роботи програмного застосунку для сторінок «Ставки» та «Орендарі»; блок-схема алгоритму роботи модуля імпорту ставків; використані технології; функціонал вебсистеми; апробація та публікації результатів роботи; впровадження; фінальний слайд.

6. Консультанти розділів бакалаврської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1 – 4	Войтко В. В., к.т.н., доцент кафедри ПЗ	24.03.25	01.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз розвитку автоматизованих вебсистем обліку ставків та постановка задач дослідження	25.03.25 – 03.04.2025	Виконано
2	Розробка методів, моделі та алгоритмів роботи вебсистеми	03.04.25 – 20.04.2025	Виконано
3	Розробка програмних модулів вебсистеми обліку ставків вінницької області	20.04.25 – 03.05.2025	Виконано
4	Тестування програми	03.05.25 – 23.05.2025	Виконано
5	Оформлення матеріалів до захисту БКР	23.05.2025 – 30.05.25	Виконано

Студент _____ (підпис) **Мироннюк О. В.** (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Войтко В. В.** (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається зі 119 сторінок формату А4, на яких є 82 рисунки, 2 таблиці, список використаних джерел містить 25 найменувань.

У бакалаврській кваліфікаційній роботі було проведено аналіз сфери систем обліку ставків. Проведено аналіз аналогів, визначено їх переваги і недоліки та доведено актуальність розробки власної вебсистеми.

Розроблена вебсистема дозволяє вести облік ставків, зображаючи їх на інтерактивній мапі та спостерігати за договорами оренди і дозволами на водокористування з щомісячним нагадуванням про ті, які закінчаться протягом наступного місяця. Удосконалено методи зберігання інформації через кешування та нагадувань через електронну пошту.

Вебсистему було реалізовано на мовах програмування C# та JavaScript, каскадних стилях CSS та мові гіпертекстової розмітки HTML. Середовищем розробки було обрано Visual Studio. Для кешування використано Redis.

У результаті виконання бакалаврської кваліфікаційної роботи отримано автоматизовану вебсистему, що підвищить зручність обліку ставків та сприятиме більш швидкій роботі з даними ставків.

Отримані в бакалаврській кваліфікаційній роботі результати можна використати для покращення процесу обліку ставків.

Ключові слова: вебсистема, автоматизація, водні ресурси.

ABSTRACT

The Bachelor's qualification thesis consists of 119 A4 pages, including 82 drawings and 2 tables. The list of used sources includes 25 denominations.

The thesis presents an analysis of the domain of pond accounting systems. Existing analogues were reviewed, their advantages and disadvantages were identified, and the relevance of developing a custom web system was substantiated.

The developed web system enables accounting of ponds, displaying them on an interactive map, and monitoring lease agreements and water usage permits with monthly reminders for those expiring within the next month. Information storage methods were improved through caching, and reminders were implemented via email.

The web system was developed using C# and JavaScript programming languages, CSS for styling, and HTML for markup. Visual Studio was chosen as the development environment, and Redis was used for caching.

As a result of the Bachelor's qualification thesis, an automated web system was created to improve the convenience of pond accounting and facilitate faster data processing.

The outcomes of this Bachelor's qualification thesis can be used to enhance the pond accounting process.

Keywords: web system, automation, water resources.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сучасний етап розвитку управління природними ресурсами вимагає вдосконалення методів моніторингу та обліку водних об’єктів, зокрема ставків. Однією з основних проблем у сфері управління водоймами є відсутність автоматизованих систем, які б дозволяли точно фіксувати географічні, екологічні та економічні параметри [1].

Традиційні методи обліку ставків часто покладаються на ручний збір даних, що може призводити до помилок, неефективного використання ресурсів і ускладнює швидке реагування на порушення [2]. У цьому контексті використання геоінформаційних систем (ГІС), які здатні інтегрувати просторові дані, відкриває нові можливості для точного аналізу стану водойм, моніторингу орендних угод та виявлення екологічних загроз.

Ставки, як об’єкти управління, мають досить складну структуру. Вона включає в себе географічні координати, гідрологічні параметри, такі як глибина, площа та об’єм, стан заростання, інформацію про орендарів і навіть історію порушень. Автоматизовані системи допомагають зібрати всі ці дані, візуалізувати їх на карті та створювати звіти, що значно економить час у порівнянні з традиційними ручними методами [3].

Важливість точності даних полягає в їхньому впливі на прийняття стратегічних рішень, які стосуються раціонального використання водних ресурсів. Наприклад, точний моніторинг замулення ставків допомагає уникнути їх перетворення на болота, а відстеження договорів оренди гарантує своєчасне оновлення умов використання водойм.

Існуючі системи обліку ставків часто стикаються з проблемами, такими як відсутність інтеграції з ГІС, недостатня автоматизація та брак деталізації даних. Більшість комерційних рішень націлені на великі підприємства, що робить їх недоступними для місцевих органів влади.

Тому актуальною є розробка власної автоматизованої вебсистеми для обліку ставків.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою дослідження є покращення управління водними ресурсами шляхом розробки програмного забезпечення для обліку ставків, яке дозволить ефективно збирати, аналізувати та візуалізувати дані про водні об'єкти.

Задачі дослідження:

- розробити модель автоматизованої вебсистеми обліку ставків, яка міститиме основний функціонал системи;
- розробка механізму імпорту даних з Excel;
- створення механізмів експорту даних у форматах PDF/Excel;
- реалізація інтерактивної карти з використанням бібліотеки Leaflet.js;
- реалізувати функціонал нагадування через електронну пошту;
- розробити модуль статистики;
- інтеграція модулів у єдину систему з інтуїтивним інтерфейсом;
- здійснити тестування вебсистеми відповідно до поставлених задач.

Об’єкт дослідження – процес розробки вебсистеми для управління водними ресурсами на прикладі ставків Вінницької області.

Предмет дослідження – методи та засоби автоматизації обліку та моніторингу ставків.

Методи дослідження. У дослідженні було використано методи:

- методи і технології модульної розробки вебсистем для програмної реалізації модулів системи;
- методи візуалізації даних результатів звітності для забезпечення отримання інформативної аналітики роботи вебсистеми;
- методи кешування для пришвидшення роботи вебсистеми;

– методи тестування для перевірки працездатності роботи програми.

Новизна отриманих результатів:

1. Подальшого розвитку набув метод кешування даних у системі за допомогою Redis, який, на відміну від існуючих рішень, базується на високопродуктивному зберіганні в пам’яті та асинхронному TTL-контролі, що значно зменшує час відповіді на запити, які часто зустрічаються, забезпечує зниження навантаження на базу даних і підвищує масштабованість додатка.

2. Подальшого розвитку набув метод формування системи нагадувань через електронну пошту, що реалізована з використанням протоколу OAuth 2.0 та Gmail API і, на відміну від традиційних SMTP-рішень, забезпечує безпечну авторизацію без зберігання паролів, автоматичну обробку черги повідомлень і можливість персоналізації листів відповідно до налаштувань користувача.

Практична цінність отриманих результатів. Практична цінність отриманих результатів полягає у можливості використання розробленої вебсистеми для автоматизації процесів обліку ставків, зокрема, дозволяє користувачам отримати всю необхідну інформацію про ставки у вигляді інтерактивної мапи з позначеними ставками з використанням Redis для кешування, можливістю експортувати цю інформацію у Excel/PDF формати, отримати різноманітну статистику про водні ресурси у системі та отримувати сповіщення на емейл про договори оренди і дозволи на водокористування, що скоро закінчаться.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській кваліфікаційній роботі, отримані автором особисто. У науковій праці[4], опублікованій у співавторстві, автору належать такі результати: таблиця порівняння функціональних характеристик аналогів, алгоритм роботи програми. У науковій роботі [5], що також була опублікована у співавторстві, автору належать визначені високорівневі вимоги до розробки вебсистеми.

Апробація матеріалів бакалаврської кваліфікаційної роботи. Матеріали бакалаврської кваліфікаційної роботи доповідалися на конференції: «LIV

Всеукраїнська науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2025)».

Публікації. Результати роботи були опубліковані в матеріалах LIV Всеукраїнської науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії (2025) [4, 5].

Структура та обсяг роботи. Бакалаврська кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, що містить 25 найменувань, 5 додатків. Робота містить 53 ілюстрації та 2 таблиці.

1 АНАЛІЗ РОЗВИТКУ АВТОМАТИЗОВАНИХ ВЕБСИСТЕМ ОБЛІКУ СТАВКІВ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану технологій обліку ставків

Сучасний етап розвитку технологій обліку водних об’єктів вимагає постійного вдосконалення методів збору, обробки та аналізу даних про ставки. Хоча традиційні методи, такі як електронні таблиці та ручне введення інформації, все ще використовуються, вони часто обмежені через відсутність автоматизації, низьку деталізацію та високу ймовірність помилок. У цьому контексті впровадження спеціалізованих вебсистем стає дедалі більш актуальним.

Одним із ключових аспектів розвитку сучасних технологій обліку є зростання потужності обчислювальних систем і вдосконалення алгоритмів обробки даних. Сучасні комп'ютери та технології баз даних дозволяють швидко і ефективно обробляти великі обсяги інформації, що є необхідним для централізованого обліку водних об’єктів. Автоматизовані системи не лише забезпечують зберігання даних, але й надають можливість їх комплексного аналізу, що значно підвищує оперативність прийняття управлінських рішень.

Використання спеціалізованих інформаційних систем обліку ставків має значні переваги в порівнянні з традиційними методами. Основною перевагою є підвищена точність і своєчасність отримання даних, що дозволяє оперативно моніторити стан водних об’єктів – від визначення їх розмірів і обсягів до оцінки рівня заростання чи забруднення. Інформація, яку надають автоматизовані системи, дозволяє точніше відображати поточний стан водних ресурсів і зменшує ризик помилок, які часто виникають при ручному введенні даних [3].

Сучасні вебсистеми обліку ставків можуть інтегруватися з геоінформаційними сервісами, що відкриває нові можливості для візуалізації розташування ставок на карті та проведення просторового аналізу. Це дозволяє враховувати географічні особливості регіону, визначати зони з підвищеним

ризиком екологічних порушень і швидко реагувати на критичні зміни в стані водних об’єктів [4].

Крім того, автоматизовані системи можуть об'єднувати різні типи даних: технічні характеристики, історичну інформацію, поточні показники стану, що дозволяє проводити глибокий аналіз трендів і оцінювати ефективність впроваджених заходів щодо збереження водних ресурсів. Це, в свою чергу, сприяє прийняттю більш обґрунтованих управлінських рішень [5].

Отже, аналіз сучасного стану технологій обліку ставків підтверджує необхідність переходу від традиційних методів до інноваційних інформаційних систем, які забезпечують більш точне, оперативне та комплексне управління даними. Використання досвіду провідних компаній у галузі розробки програмного забезпечення та впровадження сучасних технологій дозволить створити конкурентоспроможне рішення, що підвищить ефективність управління водними ресурсами та сприятиме екологічній безпеці регіону.

1.2 Порівняльний аналіз аналогів

Використання технологій управління базами даних та геоінформаційних систем (ГІС) стає все більш популярним в останні роки, і ці технології активно інтегруються в різні додатки та системи, зокрема в системи обліку природних ресурсів. Розробка програмних засобів для обліку водних об’єктів, таких як ставки, забезпечить ефективність та точність управління цими ресурсами, допоможе швидше виявляти порушення та оптимізувати використання водних об’єктів. До найбільш відомих рішень належать:

- ArcGIS;
- Держводагентство України;
- HydroMonitor;
- QGIS;
- AQUARIUS.

Одним із прикладів використання ГІС-технологій для управління водними ресурсами є програмний застосунок ArcGIS (рис. 1.1) – універсальна платформа для створення мап та управління природними ресурсами [6]. Він дозволяє проводити детальний аналіз водних об’єктів, особливо ставків, завдяки інтеграції з географічними даними, що забезпечує високу ефективність управління.

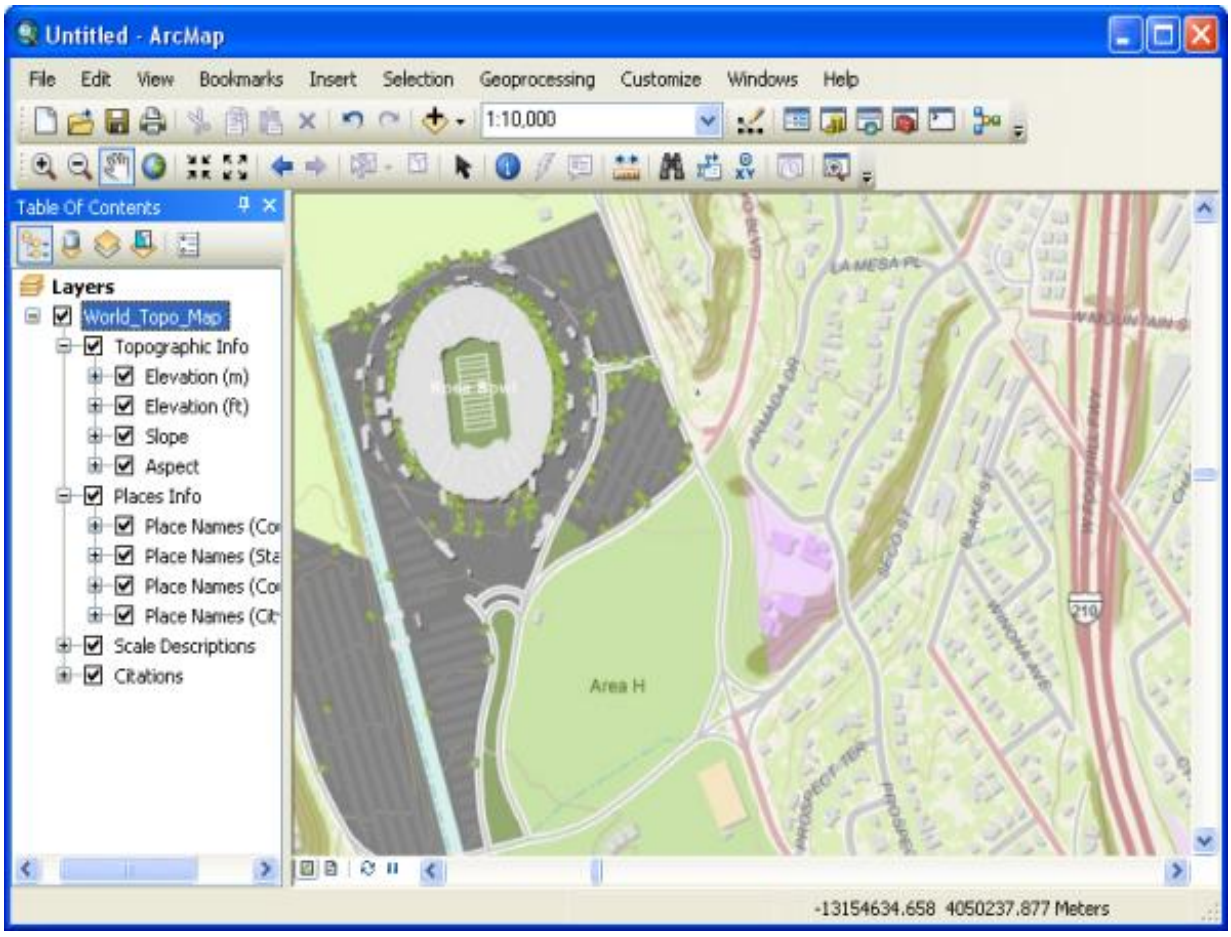


Рисунок 1.1 – Приклад роботи застосунку ArcGIS

Іншим прикладом є національний сервіс Держводагенства України, що зображено на рисунку 1.2. Він є державною системою обліку водних ресурсів, яка містить дані про водні об’єкти, в тому числі стави [7]. Однак вона має обмежений функціонал щодо деталізації параметрів, оренди та порушень. Основна перевага полягає в офіційних даних, але є недолік у вигляді не зручного інтерфейсу та недостатньої автоматизації.

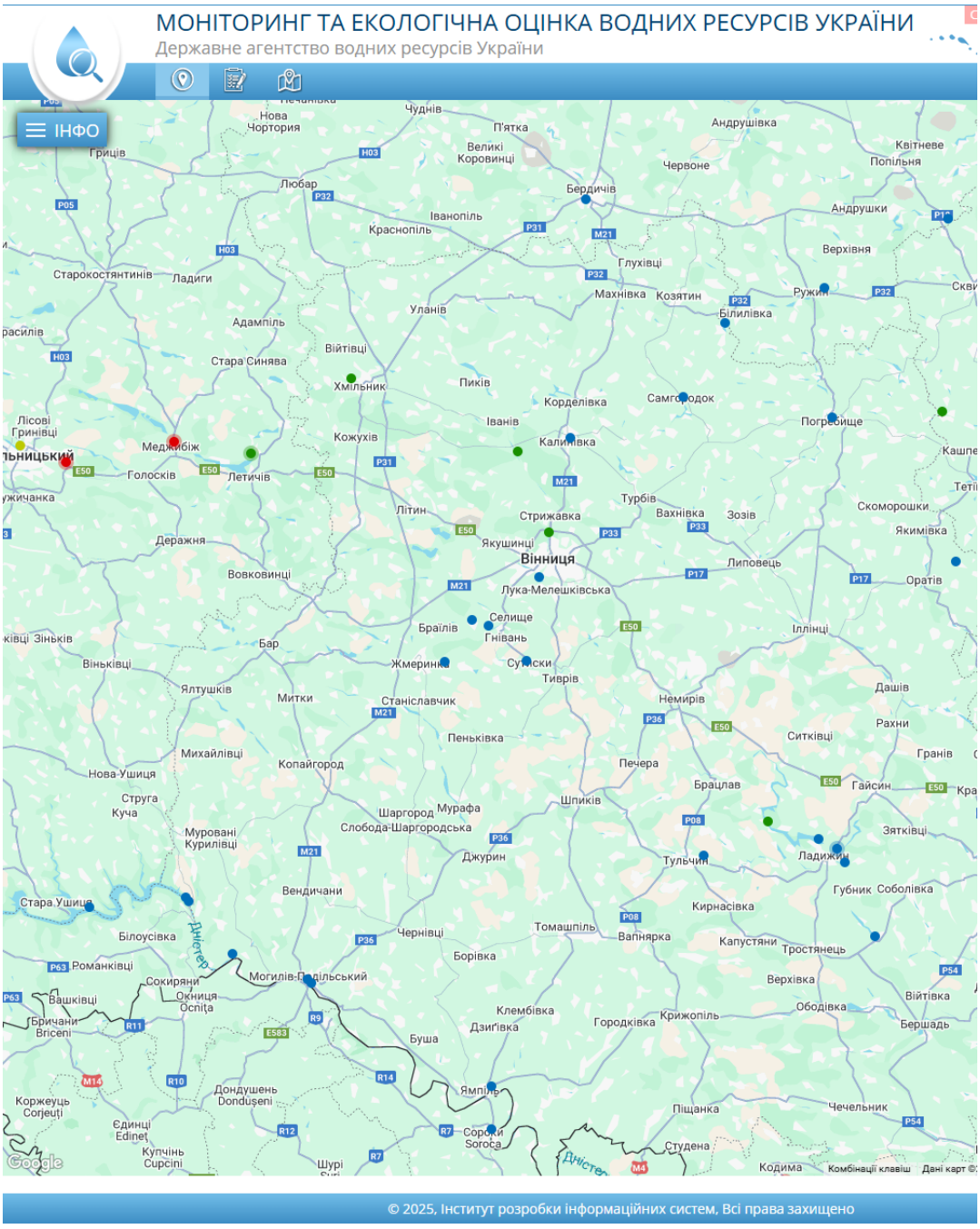


Рисунок 1.2 – Приклад роботи застосунку Держводагентства України

HydroMonitor (рис. 1.3) – це програмне забезпечення для гідрологічного моніторингу, яке допомагає аналізувати стан водойм, включаючи рівень води, забруднення та інші параметри [8]. Основна перевага полягає в його спеціалізації на водних ресурсах, хоча воно не має можливостей для управління орендою та порушеннями.

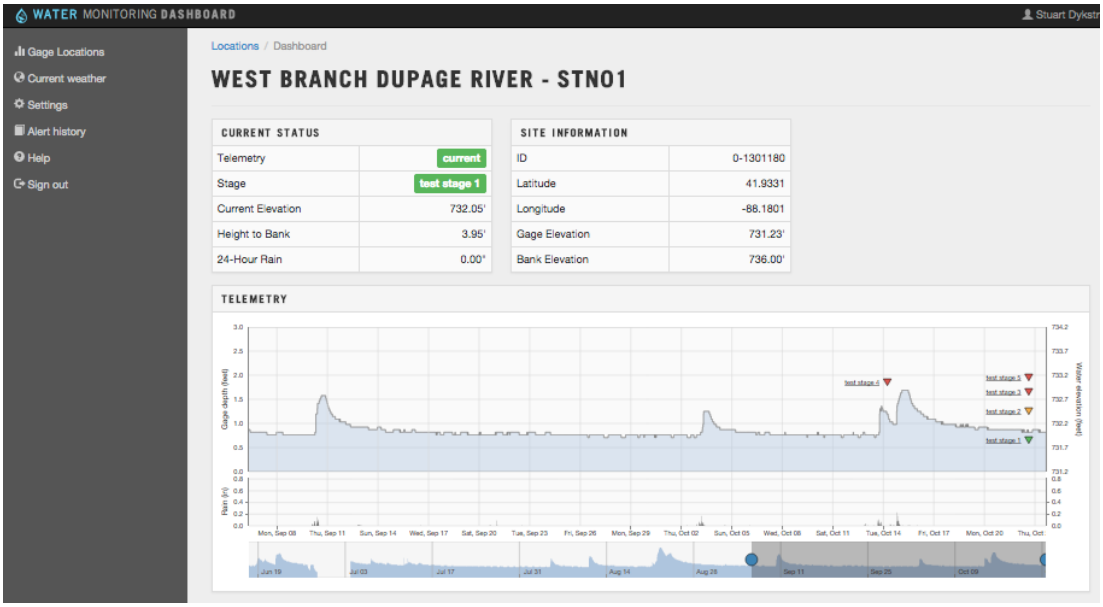


Рисунок 1.3 – Приклад роботи застосунку HydroMonitor

QGIS з модулем водних ресурсів (рис. 1.4) є відкритою ГІС-платформою, що пропонує додаткові модулі для управління водними ресурсами [9]. Платформа дозволяє користувацьке модифікування, однак для її налаштування потрібні технічні навички. Основною перевагою є гнучкість, але також варто зазначити відсутність спеціалізованих функцій для обліку саме ставків.

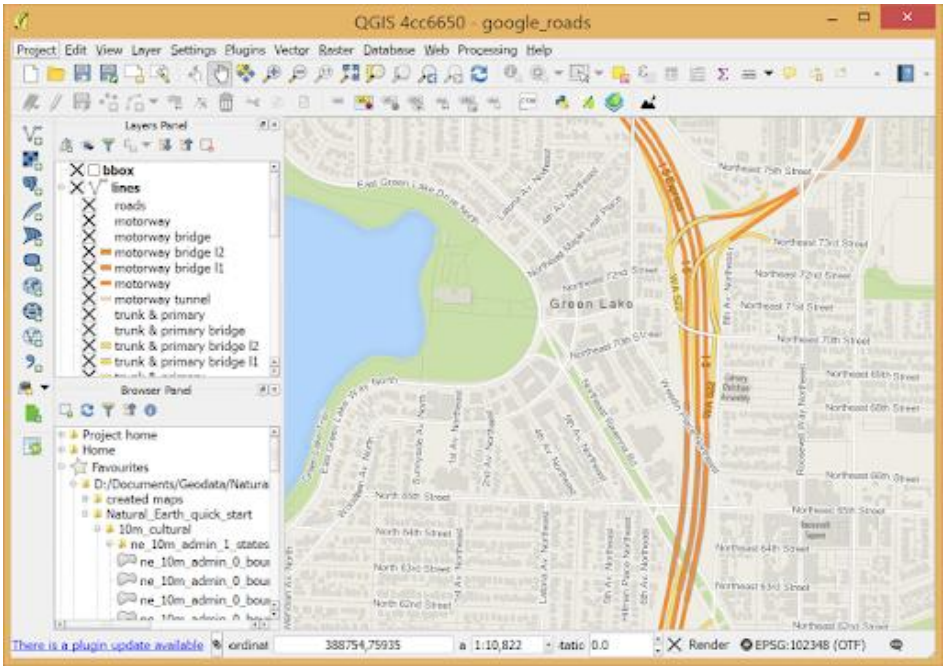


Рисунок 1.4 – Приклад роботи застосунку QGIS

AQUARIUS (рис. 1.5) – це програмне рішення для управління водними ресурсами, яке дозволяє збирати, аналізувати та візуалізувати гідрологічні дані[10]. Основною перевагою є інтеграція з датчиками та ГІС, але бракує інструментів для відстеження порушень та управління орендою.

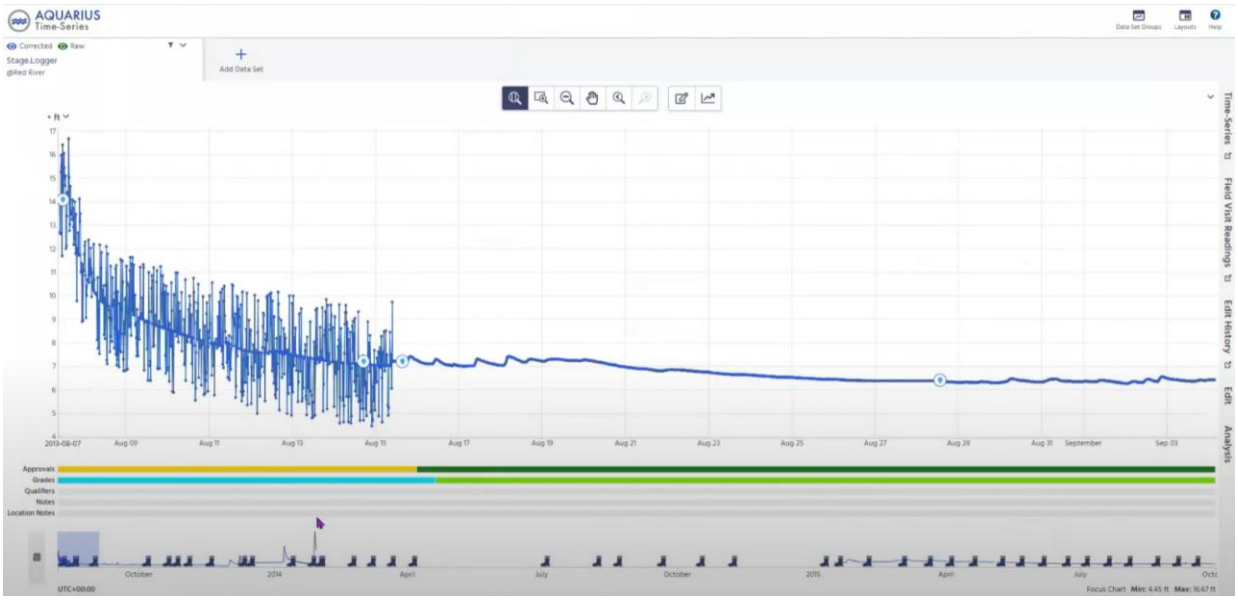


Рисунок 1.5 – Приклад роботи застосунку AQUARIUS

Розробка програмних засобів для обліку ставків з використанням геоінформаційних систем вимагає досвіду в роботі з базами даних, геоінформаційними системами та електронними таблицями. Це передбачає створення внутрішньої системи, здатної збирати, аналізувати та візуалізувати дані про ставки, з урахуванням таких параметрів, як географічні координати, розміри, стан водойми, орендарі та порушення. Частина інтерфейсу користувача передбачає розробку інтуїтивно зрозумілих інструментів, які дозволять користувачам ефективно взаємодіяти з системою, вводити дані, генерувати звіти та відстежувати стан водних ресурсів. Розробка програмного засобу для обліку ставків з використанням ГІС відкриває нові можливості для покращення управління водними ресурсами. Завдяки точним і детальним даним, адміністративні органи зможуть ефективніше контролювати стан водних об’єктів, швидко реагувати на порушення та оптимізувати використання природних ресурсів.

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерії	ArcGIS	Держвод-агентство	Hydro Monitor	QGIS	AQUARIUS	Власна розробка
Деталізовані дані водойми	+	+	+	-	+	+
Експорт/імпорт	+	-	+	+	+	+
Нагадування	-	-	-	-	-	+
Кадастрові дані	+	+	-	+	+	+
Відстеження порушень	-	+	-	-	-	+
Загальна оцінка	60%	60%	40%	40%	60%	100%

Згідно з таблицею порівняльних характеристик, створення власних програмних засобів для обліку ставків Вінницької області з використанням геоінформаційних систем є доцільною. Розроблений продукт зможе усунути недоліки наявних рішень, такі як обмежений функціонал для адміністративного обліку та відсутність автоматизації нагадувань. Нова система буде надавати розширені можливості для детального моніторингу водних об’єктів, включаючи відстеження порушень та управління орендою у реальному часі.

Отже, розробка програмних засобів для обліку ставків відкриває перед органами управління природними ресурсами широкі можливості для підвищення ефективності контролю та запобігання екологічним загрозам. Завдяки інтеграції з ГІС, точній візуалізації даних на карті та автоматизації нагадувань, адміністративні структури зможуть швидко приймати обґрунтовані рішення, що є критично

важливим для збереження екологічного балансу та раціонального використання водних об’єктів.

1.3 Аналіз методів розв’язання задачі

Розробка програмних засобів для обліку ставків Вінницької області, що базуються на вебтехнологіях, вимагає комплексного підходу до вирішення завдань, пов'язаних із збором, обробкою, аналізом та візуалізацією даних про водні об’єкти. Традиційні методи, такі як використання електронних таблиць, дозволяють зберігати інформацію, але часто стикаються з проблемами, пов'язаними з відсутністю інтеграції між різними джерелами даних, низькою оперативністю та високою ймовірністю помилок. У цьому контексті застосування спеціалізованих вебсистем для обліку ставків стає дедалі більш актуальним.

Одним із ключових аспектів розробки сучасних рішень є використання реляційних систем управління базами даних (СУБД) та сучасних серверних технологій, які дозволяють ефективно обробляти великі обсяги інформації. Цей підхід забезпечує швидкий доступ до даних, автоматизований аналіз та можливість оперативного моніторингу стану водних об’єктів, що є критично важливим для прийняття управлінських рішень. Інтеграція з геоінформаційними системами дозволяє відображати дані в зручному для користувача форматі, що підвищує ефективність роботи з інформацією.

Альтернативним підходом є використання хмарних технологій, які дозволяють зберігати та обробляти дані в реальному часі. Застосування хмарних рішень забезпечує масштабованість, високу доступність та захист інформації, що є важливим при роботі з великою кількістю даних з різних джерел. Проте цей метод може вимагати значних початкових інвестицій і ретельного вирішення питань безпеки даних. Найефективнішим методом, що дозволяє досягти високої точності та оперативності, є поєднання сучасних реляційних баз даних, інтеграції з ГІС-сервісами та використання хмарних технологій для обробки даних. Такий підхід забезпечує комплексну обробку інформації, дозволяє автоматично моніторити стан

водних об’єктів, формувати аналітичні звіти та проводити просторовий аналіз, що сприяє прийняттю обґрунтованих управлінських рішень. Отже, аналіз методів розв’язання задачі обліку ставків демонструє, що традиційні підходи, засновані на електронних таблицях, мають суттєві обмеження, тоді як використання сучасних вебсистем, інтегрованих з геоінформаційними та хмарними технологіями, є більш ефективним.

1.4 Постановка задач для розробки автоматизованої вебсистеми обліку ставків

Після аналізу переваг і недоліків існуючих методів обліку ставків, що ґрунтуються на ручному введенні даних в електронних таблицях, були визначені наступні завдання, які потрібно виконати для успішної розробки вебсистеми обліку ставків Вінницької області:

- розробити модель автоматизованої вебсистеми обліку ставків, яка міститиме основний функціонал системи;
- розробка алгоритмів для автоматизації збору та аналізу даних про ставки;
- створення механізмів експорту даних у форматах PDF/Excel;
- реалізація інтерактивної карти з використанням бібліотеки Leaflet.js;
- реалізувати функціонал нагадування через електронну пошту;
- розробити модуль статистики
- інтеграція модулів у єдину систему з інтуїтивним інтерфейсом;
- здійснити тестування вебсистеми відповідно до поставлених задач.

1.5 Висновки

У першому розділі було розглянуто стан існуючих методів та програмних засобів для обліку водних об’єктів. Були розглянуті програмні засоби такі як ArcGIS, Держводагентство України, HydroMonitor, QGIS, AQUARIUS. Було проаналізовано переваги та недоліки кожного з наведених програмних засобів.

Після аналізу переваг і недоліків існуючих рішень було визначено завдання для подальшої розробки власного програмного забезпечення для обліку ставків Вінницької області. Ці завдання включають створення інтегрованої бази даних, вебсистеми з можливістю оперативного пошуку за фільтрами по розробленій БД, інтеграцію з геоінформаційними системами, розробку модулів для імпорту та експорту інформації про ставки, а також функціонал нагадування про договори оренди, терміни яких скоро закінчуються. Таким чином, було підтверджено доцільність розробки власного програмного рішення, яке забезпечить оперативний моніторинг стану водних об’єктів і підвищить ефективність управління водними ресурсами у Вінницькій області. На основі отриманої інформації було складено перелік завдань, необхідних для реалізації власної вебсистеми обліку ставків.

2 РОЗРОБКА МЕТОДІВ, МОДЕЛІ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних даних системи

Для розробки програмного продукту для обліку та управління ставками у Вінницькій області буде використано сучасний технологічний стек на базі платформи .NET. Основними інструментами розробки стануть мова програмування C# [11], фреймворк ASP.NET [12] для створення вебзастосунків, Entity Framework [13] для роботи з базами даних, Razor Pages [14] для динамічного інтерфейсу та СУБД Microsoft SQL Server (MSSQL) для надійного зберігання інформації. C# – це сучасна об’єктно-орієнтована мова, яка забезпечує високу продуктивність, типізованість та інтеграцію з різноманітними бібліотеками, що робить її ідеальним вибором для розробки складних систем управління даними. ASP.NET дозволяє створювати масштабовані вебдодатки з підтримкою сучасних стандартів безпеки та автентифікації. Entity Framework спрощує взаємодію з базою даних через ORM (Object-Relational Mapping), автоматизуючи генерацію SQL-запитів та управління схемами даних. Razor Pages забезпечує гнучке створення інтерфейсів із використанням синтаксису C# у HTML-розмітці, що дозволяє швидко розробляти сторінки з динамічним контентом. MSSQL є надійною реляційною СУБД, яка підтримує великі обсяги даних, транзакції та складні запити.

Розробка системи включає кілька важливих етапів. Перший етап – це проєктування архітектури застосунку та розробка базових модулів збереження інформації про певні сутності у базі даних та реалізація можливості імпорту/експорту інформації. Для створення моделей даних та міграцій використовується Entity Framework, що забезпечує синхронізацію між об’єктами C# та таблицями MSSQL

Другий етап – розробка інтерактивного інтерфейсу користувача. Модуль управління даними про ставки реалізується за допомогою ASP.NET Razor Pages, де кожна сторінка відповідає за відображення та редагування інформації про

географічні координати, параметри водойми, стан заростання, замулення та інших даних.

Для візуалізації даних на карті використовуються геоінформаційні системи. Завдяки бібліотеці Leaflet, можна відображати ставки на інтерактивній карті, де користувачі мають можливість переглядати деталі кожного об’єкта, а також фільтрувати дані за районами або станом водойми. Ця функціональність реалізована через інтеграцію JavaScript з ASP.NET Razor Pages, що дозволяє динамічно оновлювати контент без перезавантаження сторінок.

Уся інформація щодо ставка зберігається в MSSQL з використанням Entity Framework, а звіти формуються за допомогою бібліотек QuestPDF та ClosedXML для експорту в Excel/PDF.

Для кешування інформації про ставки використовується Redis.

Отже, розробка системи обліку ставків на основі C#, ASP.NET, Entity Framework, Razor Pages та MSSQL є комплексним рішенням, яке об'єднує сучасні технології для автоматизації управління водними ресурсами. Використання Razor Pages забезпечує зручний інтерфейс, Entity Framework спрощує роботу з даними, а MSSQL гарантує надійність і швидкість. Інтеграція з ГІС та механізми в реальному часі роблять цю систему інструментом, який не лише підвищує ефективність адміністрування, але й сприяє збереженню екологічного балансу регіону.

2.2 Розробка моделі системи

Щоб краще розуміти роботу модулів вебсистеми обліку ставків Вінницької області вирішено розробити модель роботи системи у вигляді UML діаграми діяльності (рис. 2.1). Відповідно до діаграми, для початку роботи користувач має обрати сторінку (Ставки, Орендар, Договори тощо). На сторінці «Ставки» він може імпортувати, експортувати та фільтрувати ставки, що відображаються. З специфічних для одного ставка дій є його редагування та зображення на мапі. На сторінках з орендарями, договорами орендарів та дозволами на водовикористання є функціонал додати, редагувати та видалити відповідно орендаря, договір оренди

та дозвіл на водовикористання. Також є сторінки «Договори оренди, що закінчуються» та «Дозволи, що закінчуються», на яких користувач може фільтрувати договори та дозволи відповідно.

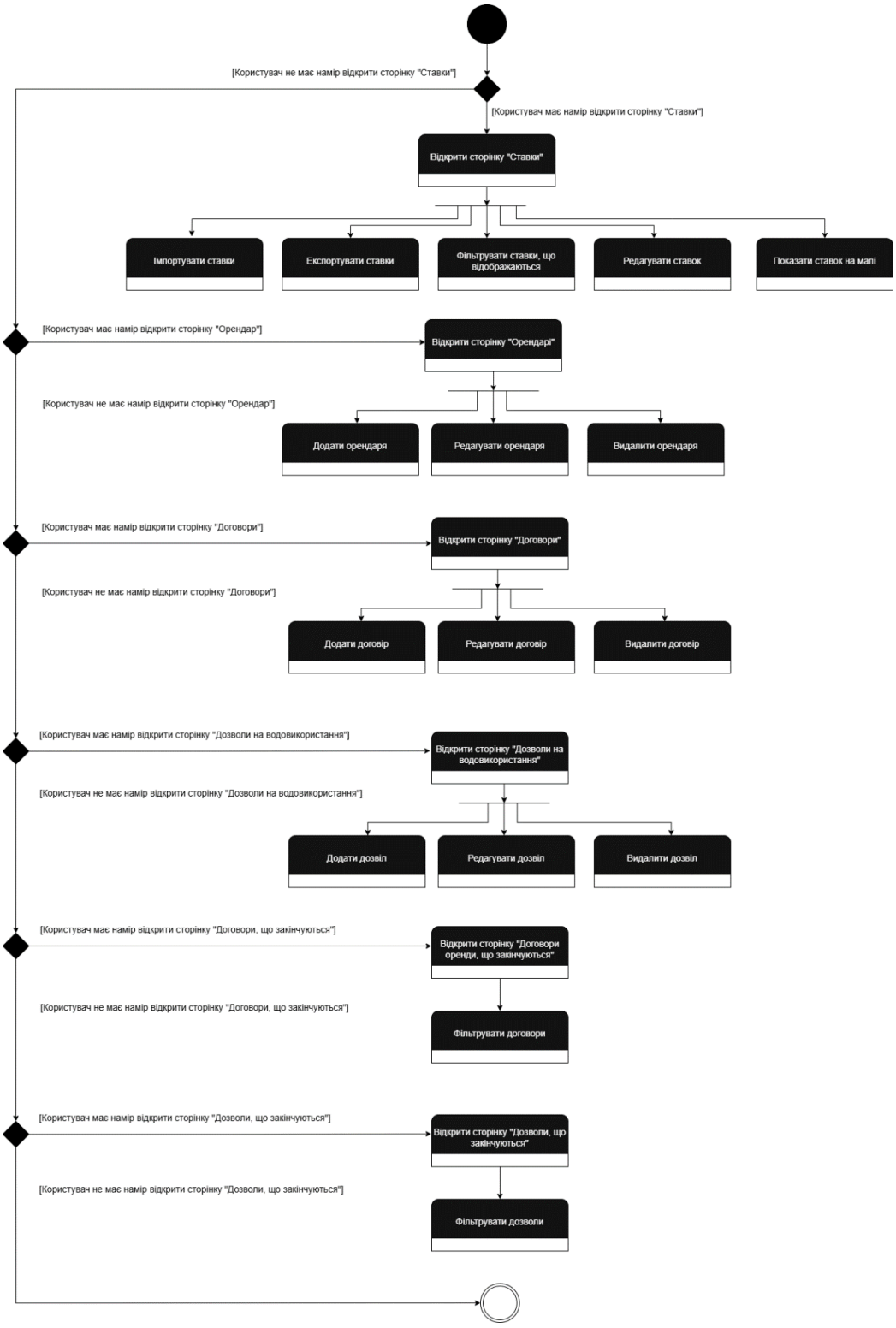


Рисунок 2.1 – Модель роботи системи у вигляді діаграми діяльності

Діаграма діяльності була розроблена у вебзастосунку Draw.IO [15].

2.3 Розробка алгоритмів роботи системи

Для розробки модулів вебсистеми обліку ставків Вінницької області необхідно визначити загальний алгоритм роботи застосунку та розробити відповідну до нього блок-схему (рис. 2.2).

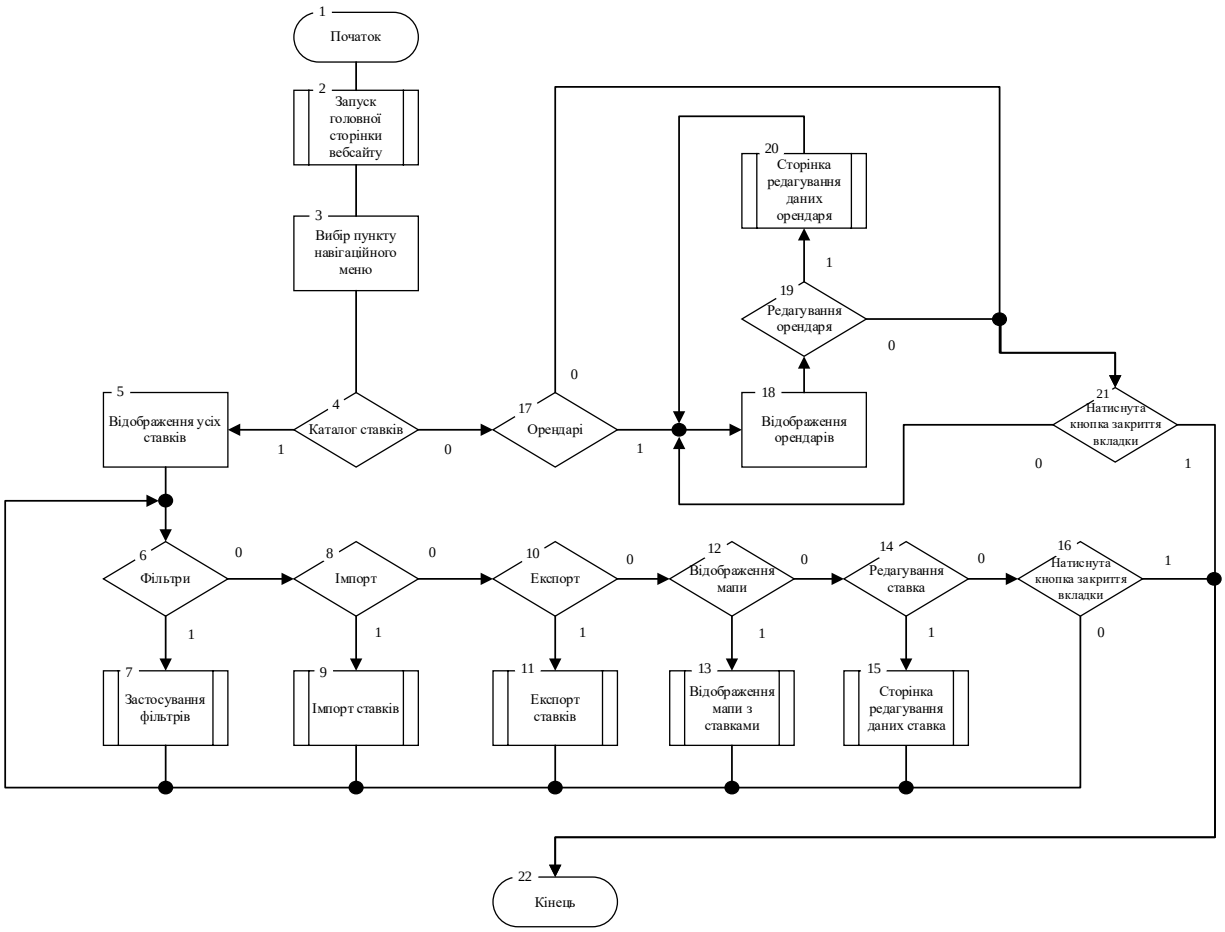


Рисунок 2.2 – Блок-схема алгоритму роботи програмного застосунку для сторінок «Ставки» та «Орендарі»

Згідно алгоритму, робота застосунку починається з завантаження головної сторінки вебсайту, і подальший вибір сторінки для роботи. При виборі сторінки з ставками є можливість обрати фільтрування ставків, що відображаються,

імпотрувати ставки з Excel-таблиці, експортувати у Excel таблицю або PDF файл та відобразити обраний ставок на мапі.

Для кращого розуміння роботи імпорту ставків розроблено блок-схему, що репрезентує роботу цього модуля (рис. 2.3).

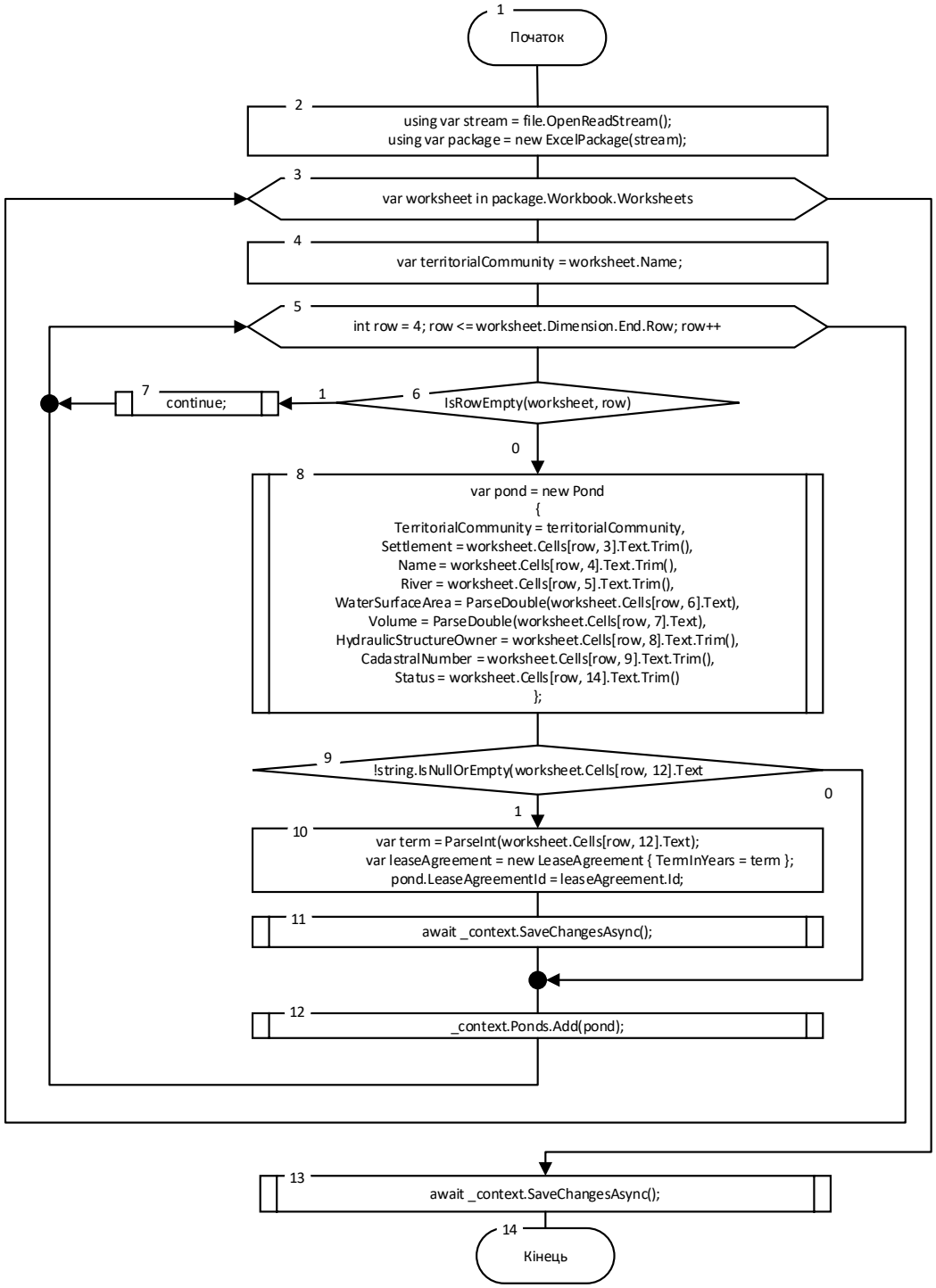


Рисунок 2.3 – Блок-схема алгоритму роботи модуля імпорту ставків

Спочатку ініціалізуються сторонні модулі для роботи з файлами та для роботи з Excel таблицями. Після чого, йде прямий перепис даних з кожної колонки кожного стовбця до відповідної властивості ставка (ім’я, територіальна громада, поселення, річка, об’єм, кадастровий номер та інші).

2.4 Розробка методу кешування даних у системі за допомогою Redis

1. Користувач заходить на сторінку "Ponds" або натискає кнопку "Export". Після чого роутинг ASP .NET перенаправляє запит до методу OnGetAsync або OnPostExportAsync відповідної PageModel.

2. Для побудови ключа кешу викликається метод BuildCacheKey, який використовує StringBuilder. Він додає до базового префіксу "PondList" сегменти ":name={Name}" та ":size={Size}" за наявності значень, а наприкінці приєднує суфікс (наприклад, ":list"). Нарешті, StringBuilder.ToString() повертає готовий ключ, наприклад PondList:name=River:size=20:list.

3. За допомогою інтерфейсу IDistributedCache виконується асинхронний виклик _cache.GetStringAsync(cacheKey). У Redis це надсилає команду GET <cacheKey>. Redis виконує пошук ключа в оперативній пам’яті та повертає рядок JSON або null, якщо ключ не знайдено чи минув TTL.

4. Якщо Redis повернув непустий рядок, застосовується JsonConvert.DeserializeObject<List<Pond>>(cachedData) з пакету Newtonsoft.Json. Таким чином ми одразу отримуємо список об’єктів Pond без жодного звернення до бази даних.

5. Якщо _cache.GetStringAsync повернув null, викликається сервіс IPondService.GetFilteredPondsAsync(Name, Size), який через Entity Framework Core робить SQL-запит до бази та повертає актуальні дані. Отриманий список серіалізується в JSON за допомогою JsonConvert.SerializeObject, і далі через _cache.SetStringAsync(cacheKey, serialized) (команда SET в Redis) зберігається в кеші.

6. Для звичайного рендерингу Razor-сторінки використовується властивість Ponds, і метод OnGetAsync повертає Page().

7. Для експорту викликається OnPostExportAsync(exportType), де за допомогою PondExportHelper.ExportToExcel або PondExportHelper.ExportToPdf формується масив байтів файлу.

8. Після операцій створення, редагування чи видалення запису ставка виконується _cache.RemoveAsync(cacheKey). Це викликає в Redis команду DEL <cacheKey>, яка видаляє застарілий запис із оперативної пам’яті.

Блок-схема алгоритму зображена на рисунку 2.4.

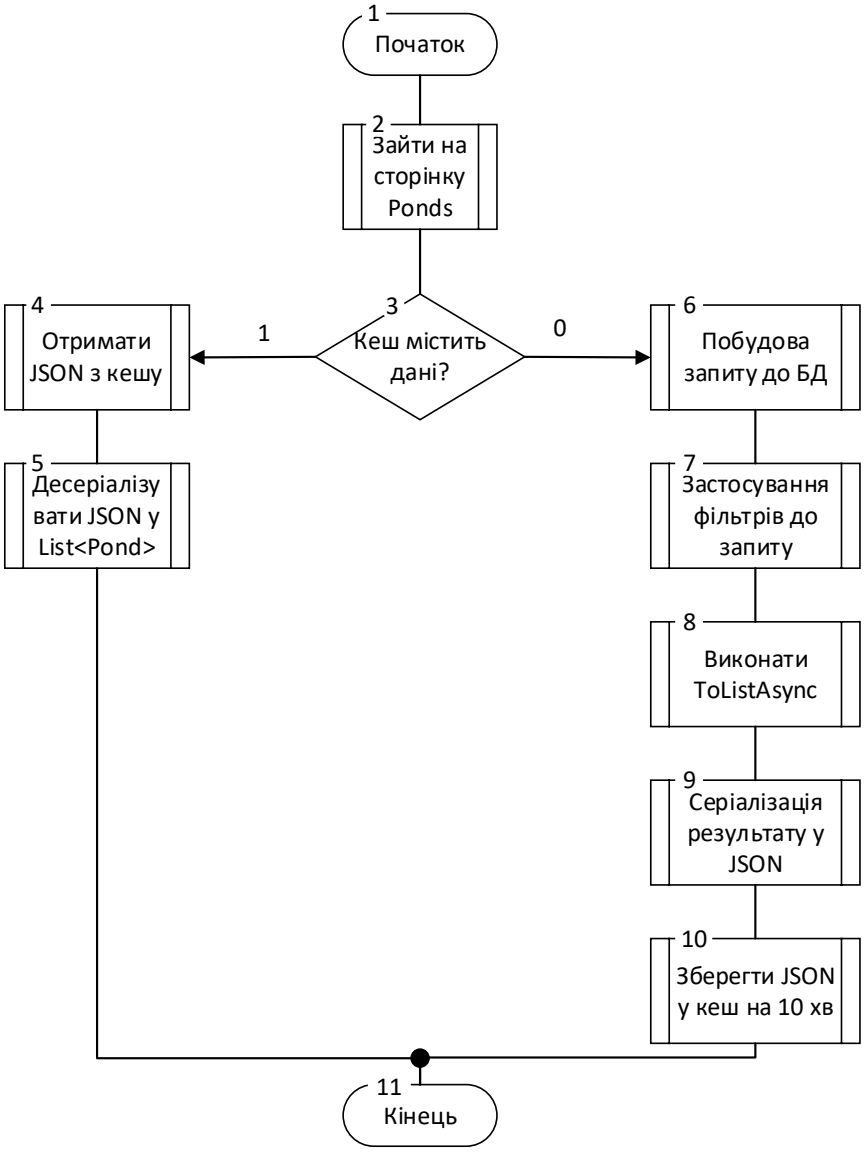


Рисунок 2.4 – Блок-схема алгоритму роботи методу кешування через Redis

Команди GET, SET і DEL в Redis реалізовані як операції з хеш-таблицею в RAM із часовою складністю O(1). Якщо заданий TTL, Redis автоматично видаляє прострочені ключі через фонові процеси або під час запитів, без додаткових дій з боку коду.

Перший запит із "cache miss" триває довше через звернення до бази та запис у Redis, але всі наступні запити з тими ж фільтрами виконуються за мілісекунди, оскільки дані беруться напряму з Redis. Навантаження на базу знижується до одного запиту на унікальну комбінацію фільтрів, що забезпечує високу швидкодію та масштабованість додатку.

2.5 Розробка методу формування системи нагадувань через електронну пошту

Для реалізації функціоналу автоматичного надсилання нагадувань про закінчення договорів оренди та дозволів на спецводокористування було використано фоновий сервіс у поєднанні з Gmail API та авторизацією через OAuth 2.0. Покроковий опис роботи цієї системи:

1. Застосунок під час запуску реєструє фоновий сервіс, який реалізує інтерфейс BackgroundService, і налаштовує його на запуск поза запитами користувача.

2. Служба обчислює кількість часу до першого числа наступного місяця о 9-й годині, використовуючи системний час сервера, і виконує асинхронне очікування через Task.Delay.

3. Коли настає час виконання, служба відкриває область залежностей через IServiceScopeFactory, щоб отримати екземпляр ApplicationDbContext із сконфігурованими налаштуваннями EF Core для доступу до бази даних.

4. З бази даних витягуються всі записи договорів оренди й дозволів, дата закінчення яких попадає в інтервал від сьогодні до 31 дня вперед. Дата закінчення обчислюється методом Date.AddYears(TermInYears) для договорів і StartDate.AddYears(TermInYears) для дозволів.

5. Якщо жодного договору або дозволу не знайдено, служба завершує поточне виконання без надсилання, повторивши очікування наступного місяця.

6. Для формування вмісту листа створюється HTML-фрагмент у StringBuilder, куди послідовно додаються заголовок із кінцевою датою та списки знайдених договорів і дозволів з їхніми номерами й датами.

7. Служба отримує шлях до файлу credentials.json із кореня проєкту та викликає GoogleWebAuthorizationBroker.AuthorizeAsync, передаючи область доступу GmailSend, щоб автентифікуватися через OAuth 2.0 і зберегти токени в папці token_store.

8. Після успішної авторизації створюється екземпляр GmailService з HttpClientInitializer, встановленим у щойно отриманий UserCredential, та задається ім’я програми "PondManager".

9. Для складання листа використовується бібліотека MimeKit: створюється об’єкт MimeMessage, задаються поля From, To, Subject та тіло в форматі TextPart("html") із раніше побудованим HTML.

10. Підготовлене Mime-повідомлення записується у пам’яті у MemoryStream, після чого його байти перетворюються в base64-url, замінюючи символи + на - і / на _, та обрізаючи символи = наприкінці.

11. Формується об’єкт Gmail API Message з властивістю Raw, куди присвоюється отриманий base64-url рядок.

12. Відправка виконується викликом gmail.Users.Messages.Send(message, "me").ExecuteAsync(), де "me" означає авторизованого користувача, на якого видано токен.

13. Після успішної відправки служба фіксує результат у логах або внутрішніх властивостях, щоб у разі помилки можна було побачити текст виключення та повторити спробу за наступного циклу.

14. Служба знову переходить в режим очікування до наступного запуску, зберігаючи автоматичне виконання процедури щомісяця без взаємодії користувача.

Блок-схема роботи методу зображена на рисунку 2.5.

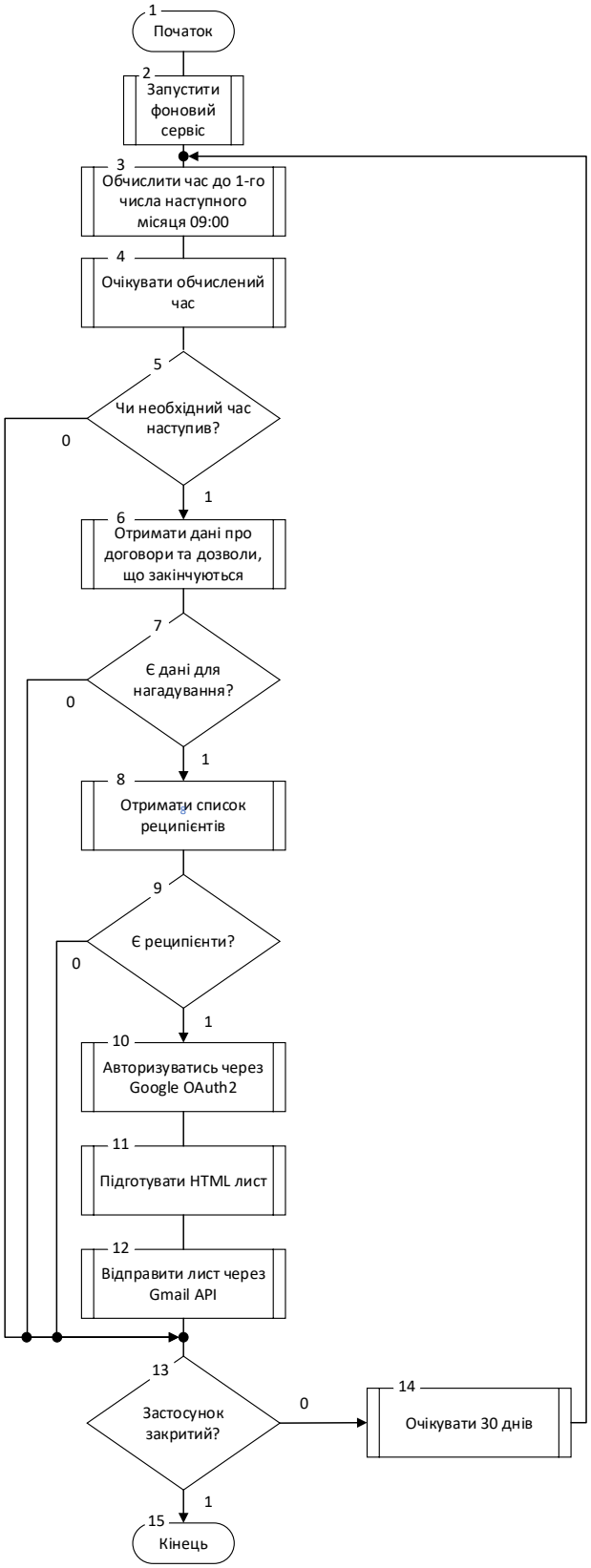


Рисунок 2.5 – Блок-схема алгоритму роботи методу

Отже, процес надсилання електронних сповіщень було повністю автоматизовано, що забезпечило своєчасне інформування користувачів про наближення закінчення термінів дії договорів та дозволів. Завдяки інтеграції з Gmail API та використанню сучасних методів авторизації, реалізоване рішення відповідає вимогам безпеки Google та є надійним для довготривалого використання.

2.6 Висновки

У другому розділі було детально проаналізовано вхідні та вихідні дані програмного забезпечення, а також проаналізовано розробку графічного інтерфейсу застосунку. Також створено базовий алгоритм для роботи програмних засобів, алгоритм імпорту ставків з Excel таблиці, алгоритм роботи методу кешування даних у системі через Redis та алгоритм роботи методу формування системи нагадувань через електронну пошту. Також було розроблено метод кешування даних у системі за допомогою Redis, метод формування системи нагадувань через електронну пошту та модель функціонування продукту за допомогою UML-діаграм.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ВЕБСИСТЕМИ ОБЛІКУ СТАВКІВ ВІННИЦЬКОЇ ОБЛАСТІ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Розробка вебсистеми для управління ставками з використанням сучасних технологій, вимагає обрання найкращих інструменти для кожного з компонентів системи. У випадку модулів вебсистеми обліку ставків вінницької області, мета полягає в досягненні високої продуктивності серверної частини, гнучкості моделі даних, зручності у створенні інтерфейсу користувача, а також інтеграції геопросторових сервісів і формуванні звітів у форматі Excel. Нижче наведено аналіз технологій, який було використано.

ASP.NET Core – це сучасний крос-платформний фреймворк від Microsoft, призначений для створення високопродуктивних вебдодатків і API [17]. Використання ASP.NET Core дозволяє зменшити споживання ресурсів завдяки оптимізованому конвеєру обробки HTTP-запитів (middleware pipeline), вбудованій підтримці асинхронного програмування та інтегрованій системі впровадження залежностей (DI). Модульна архітектура дозволяє підключати лише необхідні пакети, що зменшує розмір розгортання та підвищує безпеку. Завдяки вбудованому середовищу хостингу у вигляді Kestrel та можливості працювати як проксі-сервер для IIS, nginx чи Apache, ASP.NET Core забезпечує гнучкість у розгортанні.

Entity Framework Core – це ORM-бібліотека для .NET, яка дозволяє працювати з базою даних на рівні об’єктної моделі [18]. EF Core реалізує патерни Repository та Unit of Work, що автоматизує відстеження змін у сутностях і формування SQL-запитів. Завдяки підтримці LINQ, ви можете формулювати декларативні запити до бази даних без необхідності писати SQL вручну, а механізм міграцій спрощує еволюцію схеми даних протягом життєвого циклу проєкту. Entity Framework Core також забезпечує сумісність з різними СУБД (такими як SQL

Server, PostgreSQL, SQLite тощо), що дозволяє легко змінювати бекенд-базу без потреби переписувати логіку доступу до даних.

Razor Pages – це сторінково-орієнтована модель для створення інтерфейсу користувача в ASP.NET Core [14]. Вона поєднує шаблон розмітки (Razor) з кодом обробки запитів в одному файлі (.cshtml + .cshtml.cs). Razor Pages роблять структурування проєкту простішим, адже за принципом «одна сторінка – один файл моделі» зникає потреба в окремих контролерах і виглядах (views) для простих CRUD-сценаріїв. Завдяки вбудованому механізму прив’язки даних (model binding) та валідації за допомогою атрибутів, розробка форм для введення та обробки даних стає швидкою і зручною.

Leaflet – це легка JavaScript-бібліотека, яка допомагає створювати інтерактивні картографічні інтерфейси [19]. Leaflet розроблена з урахуванням мобільних і десктопних браузерів, і вона пропонує API для відображення маркерів, ліній та полігонів, з підтримкою різних подій, таких як навігація, масштабування та взаємодія з об’єктами. Використання Leaflet у PondManager дозволяє візуалізувати розташування водойм, налаштовувати індивідуальні стилі шарів і динамічно додавати геодані від користувачів, що забезпечує зручне управління просторовими даними.

ClosedXML – .NET-бібліотека для операцій з файлами Excel формату .xlsx, що спрощує створення, читання та зміни робочих листів без прямої взаємодії з OpenXML SDK [20]. ClosedXML надає зручні методи для форматування клітинок, об’єднання діапазонів, вставки формул та налаштування стилів, мінімізуючи обсяг коду. Завдяки цьому оперативна генерація звітів з даними про господарські операції на водоймах є можливими.

iText – це потужна бібліотека з відкритим вихідним кодом, яка дозволяє створювати та обробляти PDF-документи в середовищах Java та .NET (через порт iTextSharp) [21]. Вона надає розробникам зручний API для динамічної генерації PDF-файлів будь-якої складності, від простих текстових звітів до багатосторінкових форм з складним розміщенням графіки та таблиць.

Головна перевага iText полягає в підтримці стандартів ISO 32000 (PDF 1.7) та PDF/A для архівації, що забезпечує коректне відтворення документів на різних платформах і в різних програмах. Бібліотека також включає модулі для шифрування вмісту, створення цифрових підписів і захисту від несанкціонованого редагування. Підтримка спільного редагування (concatenation) дозволяє об'єднувати кілька джерел в один PDF-файл, а можливості розбору (parsing) дають змогу витягувати текст, зображення та метадані з існуючих документів.

Redis – це високопродуктивне сховище даних у пам’яті, що використовується для кешування, завдяки чому значно зменшується час відповіді системи на повторювані запити [22]. Його основна перевага полягає в тому, що дані зберігаються в оперативній пам’яті, що дозволяє миттєво отримувати доступ до інформації без звернення до бази даних. Redis підтримує різноманітні структури даних, що дає змогу адаптувати його під специфіку різних задач – від простого кешу до черг обробки подій. Завдяки можливості задавати час життя кожного ключа, Redis ефективно очищає застарілі записи, а вбудовані механізми масштабування й реплікації дозволяють використовувати його у високонавантажених додатках.

OAuth 2.0 – це сучасний протокол авторизації, який дає змогу додаткам отримувати обмежений доступ до ресурсів користувача без необхідності запитувати його пароль [23]. Користувач підтверджує доступ лише один раз, після чого система отримує спеціальний токен, який надалі використовується для доступу до захищених API. Такий підхід забезпечує високий рівень безпеки, оскільки облікові дані користувача не зберігаються в сторонніх системах. Крім того, OAuth 2.0 дозволяє оновлювати доступ автоматично за допомогою спеціального токена без повторного втручання користувача, що робить його зручним для інтеграції з зовнішніми сервісами.

Gmail API – це інтерфейс програмної взаємодії з поштовою службою Gmail, який дає змогу програмно надсилати листи, читати вхідні повідомлення, додавати вкладення та керувати ярликами [24]. Він повністю інтегрується з системою

авторизації Google через OAuth 2.0, що дозволяє безпечно працювати з поштовими скриньками користувачів. Завдяки цьому API розробники можуть реалізовувати автоматизовану систему розсилки нагадувань або повідомлень без прямого втручання користувача, зберігаючи при цьому високий рівень безпеки й контролю.

MimeKit – це .NET-бібліотека для створення, обробки та аналізу електронних листів у форматі MIME [25]. Вона розроблена для заміни стандартного .NET System.Net.Mail, забезпечуючи більшу гнучкість, продуктивність і підтримку сучасних форматів повідомлень. MimeKit підтримує складні MIME-структури, цифрові підписи, шифрування, кодування, вкладення, а також дозволяє читати і створювати листи в різних форматах, таких як HTML, текст чи змішані. Бібліотека активно використовується в додатках, які мають потребу в надійному обробленні пошти, включаючи поштові клієнти, сервери та служби

Отже, поєднання ASP.NET Core як надійного серверного шару, Entity Framework Core для зручного доступу до даних, Razor Pages для створення інтуїтивно зрозумілого веб-інтерфейсу, Leaflet для інтерактивної картографії, ClosedXML для генерації та обробки Excel-звітів, iText для створення захищених PDF-документів формує технологічний стек застосунку, Redis для кешування а також Gmail API, OAuth 2.0 та MimeKit для нагадувань про договори оренди та дозволи на спецводокористування, що скоро закінчаться через електронну пошту забезпечує високу продуктивність, масштабованість та гнучкість у формуванні звітності, що є критично важливими для ефективного управління ставками.

3.2 Розробка модуля імпорту ставків

Розробка модуля імпорту ставків вимагає врахування ряд технічних аспектів. Спершу необхідно встановити пакет OfficeOpenXml для роботи з файлами Excel. Після, треба відкрити потік читання файлу та отримати Excel-контейнер, з якого отримується інформація про аркуші з таблицями.

З отриманого Excel-контейнеру для кожного аркуша береться його назва як назва територіальної громади, до якої відноситься ставок. Потім, для кожного

ставка, що починаються у файлі з 4-го рядка починається прямий перенос (маппінг) інформації про ставки, такої як поселення, ім’я, річка, на якій він знаходиться, площа водного плеса, об’єм та інші. Якщо у файлі присутня інформація про договори оренди – вони також будуть імпортовані у відповідні сутності у базі даних.

Після маппінгу, ставок треба додати до бази даних, а після імпорту усіх ставків – зберегти інформацію, цього вимагає Entity Framework.

Блок-схема алгоритму роботи модуля наведена у розділі 2, на рисунку 2.3.

3.3 Розробка модуля експорту ставків у PDF та Excel

Для експорту інформації у файл формату PDF необхідно встановити пакет iText. Потім, так як експортуються тільки ставки, що були відфільтровані за введеними користувачем значеннями, необхідно додати самі фільтри до моделі сторінки з ставками.

Блок-схема алгоритму експорту у PDF зображена на рисунку 3.1.

Усі фільтри:

- IdFilter;
- NameFilter;
- DistrictFilter;
- TerritorialCommunityFilter;
- SettlementFilter;
- MinXFilter;
- MaxXFilter;
- MinYFilter;
- MaxYFilter;
- LengthFilter;
- WidthFilter;
- DepthFilter;
- WaterLevelFilter;

- VolumeFilter;
- LeasedAreaFilter;
- CadastralNumberFilter;
- WaterSurfaceAreaFilter;
- IsDrainableFilter;
- HasHydraulicStructureFilter;
- HydraulicStructureOwnerFilter;
- LesseeIdentificationCodeFilter;
- LeaseAgreementNumberFilter;
- WaterUsagePermitNumberFilter;
- RiverFilter;
- BasinFilter;
- StatusFilter;
- IssuesFilter;
- NotesFilter;
- ImposedFinesFilter;
- ImposedDamagesFilter;
- CollectedFinesFilter;
- CollectedDamagesFilter.

У методі експорту, для початку процесу, потрібно отримати усі відфільтровані ставки, відкрити pdf-записувач, створити документ, у який буде записуватись інформація.

Після створення документу, у який буде імпортуватись таблиця, потрібно встановити шрифт, що буде підтримувати інформацію, що написана українською мовою, відповідно, кирилицею. Для отримання автономної версії файлу з шрифтом, необхідно, щоб було завантажено шрифт Halvetica з директорії wwwroot/fonts. Якщо, за невідомої причини, файлу шрифта не буде у директорії, або він буде пошкоджений чи відсутній – буде завантажений шрифт системи за замовчуванням.

Далі, встановлюється ширина для тридцяти колонок, у які будуть експортовані дані а також будуть додані заголовки для кожної з цих колонок.

Після того, як буде підготовлено каркас таблиці для PDF файлу, починається додавання інформації для кожної властивості кожного ставка у відповідну для нього клітинку таблиці.

Як тільки усі клітинки будуть успішно заповнені, таблиця буде додана у документ. Після чого необхідно закрити документ та pdf-записувач та повернути файл.

Після повернення файлу з методу, у браузері буде завантажено створений файл.

Модуль експорту в Excel відповідає за створення Excel-файлу з переліком ставків і повернення його користувачеві у відповідь на запит (рис. 3.2). Він є асинхронним і запускається під час надсилання POST-запиту. На початку методу відбувається отримання даних викликається метод GetFilteredPonds(), який застосовує необхідні фільтри до колекції ставків. Після цього дані перетворюються в список для подальшої обробки.

Далі створюється нова Excel-книга за допомогою бібліотеки ClosedXML. У цій книзі додається новий аркуш з назвою "Ставки". У перший рядок аркуша записуються заголовки колонок, які відповідають різним властивостям ставка наприклад, назва, район, координати, площа, статус, наявність гідроспоруд, орендар, штрафи тощо.

Після цього метод переходить до заповнення аркуша. Для кожного ставка з підготовленого списку створюється окремий рядок. У цьому рядку по черзі записуються значення всіх потрібних властивостей. У тому числі обробляються значення типу bool, які відображаються як "Так" або "Ні", а також об’єкти, які можуть бути порожніми (наприклад, орендар чи дозвіл на спецводокористування) у таких випадках використовується умовна перевірка перед записом.

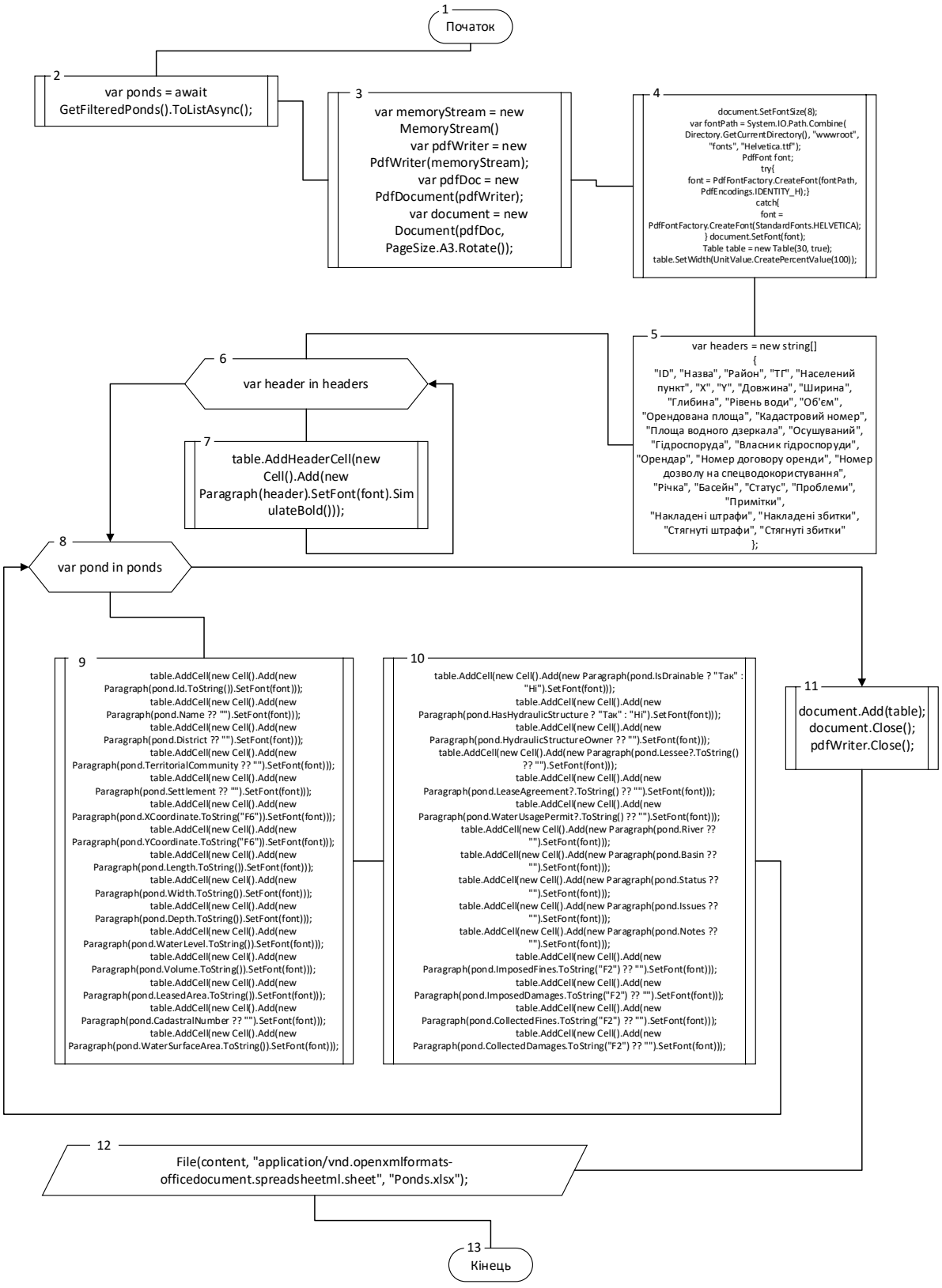


Рисунок 3.1 – Блок-схема експорту ставків в PDF

Після завершення заповнення таблиці Excel-файл зберігається у тимчасовий потік пам’яті. Вміст цього потоку перетворюється у масив байтів, який

повертається користувачеві у вигляді файлу з правильним MIME-типом (application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) та ім’ям "Ponds.xlsx".

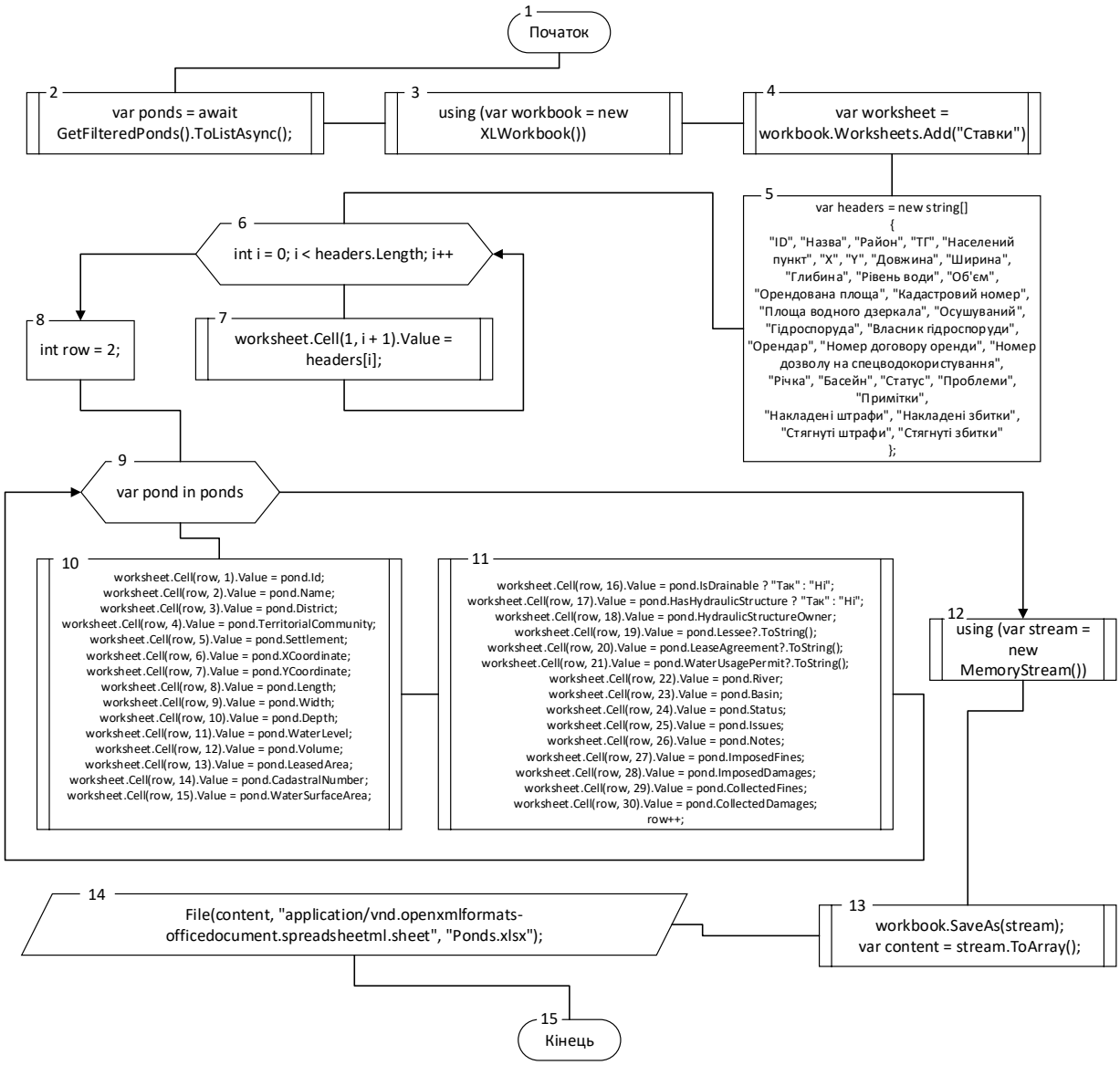


Рисунок 3.2 – Блок-схема експорту ставків в Excel

У результаті користувач отримує готовий файл Excel зі структурованою таблицею даних про ставки, яку можна відкривати, переглядати та обробляти в табличному редакторі.

3.4 Розробка модуля кешування через Redis

Під час розробки застосунку було розгорнуто зовнішній сервіс Redis в контейнері Docker для організації розподіленого кешування даних у веб-додатку ASP .NET Razor Pages «PondManager». Було підключено реалізацію інтерфейсу IDistributedCache, що надало абстракцію над провайдером кешу та забезпечило можливість у майбутньому легко замінювати сховище.

Для унікальної ідентифікації кешованих запитів було визначено формат ключів, що складався з префіксу, який позначав тип операції (наприклад, експорт в Excel або PDF), та серії пар «назва_фільтра=значення». Відповідно, кожна комбінація вхідних даних отримала власний простір у кеші, що запобігло колізіям і дало змогу одночасно зберігати різні результати.

Під час виконання експорту до Redis було виконано спробу отримання готових даних за поточним ключем. У разі успіху дані були десеріалізовані та передані до генератора файлу без звернення до бази даних. Якщо кеш виявився порожнім, було здійснено запит до СУБД, отримані сутності було серіалізовано у формат JSON і збережено в кеш із терміном життя (TTL) кілька хвилин. Така реалізація знизила час обробки повторних запитів із понад секунди до приблизно 200 мс та скоротила навантаження на базу даних до одного запиту на кожну унікальну комбінацію фільтрів.

Для підтримання консистентності даних було впроваджено механізм інвалідизації кешу: після операцій створення, редагування або видалення записи ключі, пов’язані з відповідними фільтрами, видалялися з Redis. Це гарантувало, що користувачі отримуватимуть лише актуальну інформацію.

Блок-схема алгоритму наведена у розділі 2 на рисунку 2.4.

3.5 Розробка модуля статистики

Розробка модуля статистики ставок була виконана у кілька етапів і завершена комплексною інтеграцією у веб-додаток. Спочатку було проведено аналіз вимог: визначено ключові показники, які необхідно відображати, серед яких загальна

кількість ставків, середні показники глибини, об’єму та орендованої площі, розподіл за районами та басейнами, а також співвідношення за вибором спуску води і наявністю гідроспоруд. Після затвердження переліку метрик було розроблено структуру даних і метод для їх отримання з бази за допомогою EF Core.

Блок-схема роботи модуля зображена на рисунку 3.3.

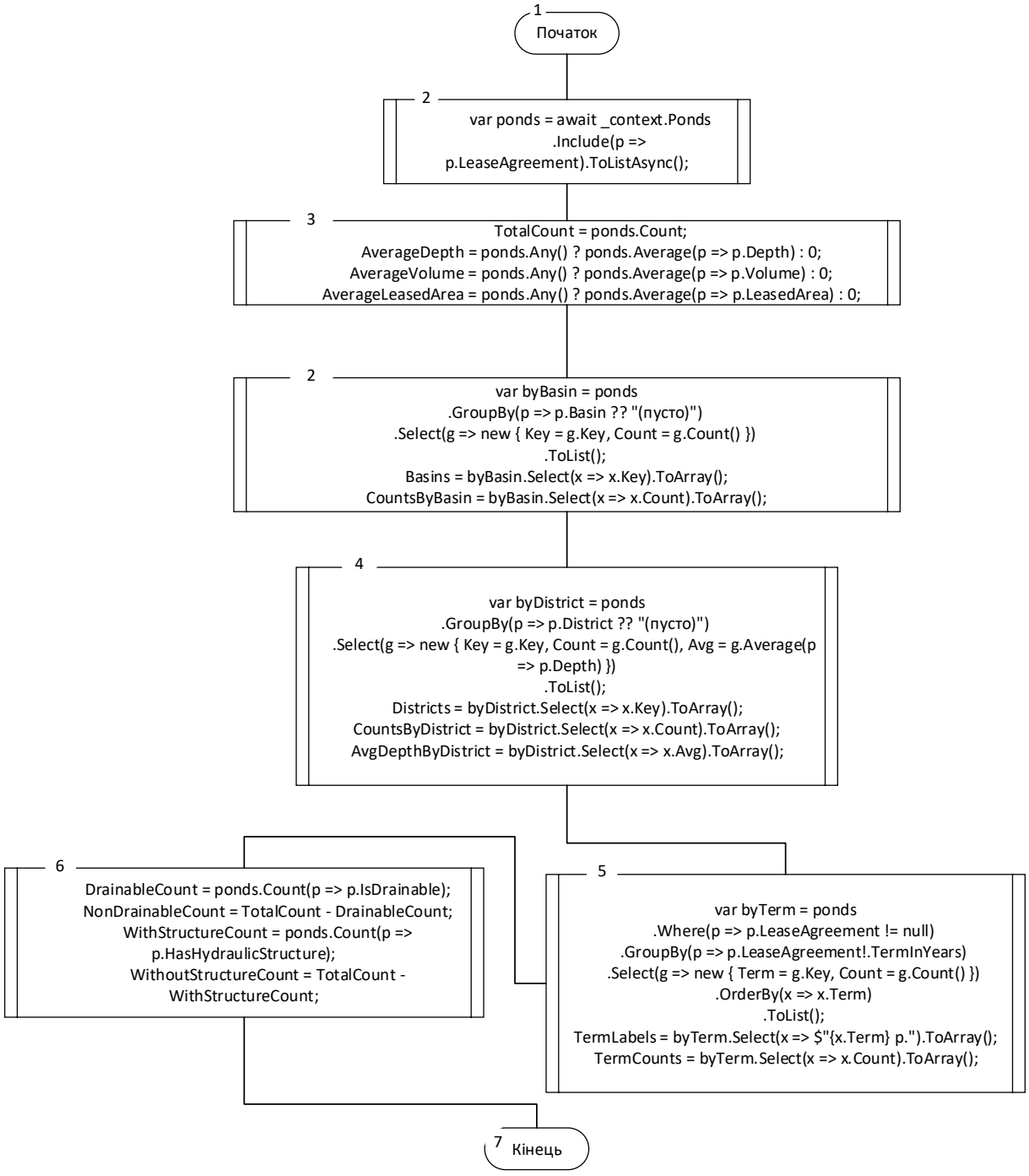


Рисунок 3.3 – Блок-схема завантаження даних модуля статистики

На наступному етапі була реалізована логіка обчислення: у методі OnGetAsync моделі сторінки було зібрано повний набір ставків, виконано групування за обраними категоріями та обчислено агреговані значення. Середньоглибинні показники й середній об’єм було пораховано за допомогою стандартних LINQ-функцій, а кількість спускних і неспускних ставків та наявність гідроспоруд – шляхом фільтрації списку. Результати було записано у властивості моделі у вигляді масивів і одиночних значень для подальшої обробки у представленні.

Після того, як серверна логіка була перевірена і протестована, було розгорнуто клієнтську частину з використанням Chart.js. Було додано шість різнопланових діаграм: дві стовпчастих діаграми для відображення кількості ставок за районами та басейнами, дві діаграми для середньої глибини та розподілу термінів договорів, а також декілька кругових діаграми для наглядного порівняння спускних/неспускних ставків і наявності/відсутності гідроспоруд. Кожен графік отримував дані через JSON-серіалізацію властивостей моделі, і налаштування підказок та заголовків було виконано так, щоб користувач бачив лише цілі значення та зрозумілі підписи.

Візуальна частина була впроваджена в Razor-сторінку Statistics.cshtml: для кожного графіка створено окремий <canvas>, а скрипти ініціалізували діаграми за даними моделі. Додатково було оформлено картки з ключовими метриками зверху сторінки, де значення форматувалося без десяткових крапок.

Відповідна стилізація з Bootstrap дозволила адаптивно розташувати елементи у кілька рядків і зробити інтерфейс зручним.

3.6 Розробка модуля нагадувань через електронну пошту

Розробка механізму автоматичної розсилки нагадувань електронною поштою розпочалася з формулювання функціональних вимог: необхідно було щомісяця інформувати відповідальних осіб про договори оренди й дозволи на спеціальне водокористування, термін дії яких спливатиме протягом наступного місяця. Після

затвердження вимог було обрано архітектурний підхід із фонового сервісу, який працюватиме поза запитами користувача і запускатиметься один раз на місяць у визначений день і час.

У ході реалізації було інтегровано механізм OAuth 2.0 та Gmail API замість традиційного SMTP, що забезпечило вищий рівень безпеки облікового запису. Для цього було створено облікові дані в консолі Google Cloud, налаштовано сховище токенів і побудовано процес автентифікації користувача перший запуск вимагав підтвердження доступу до пошти через браузер. Паралельно з цим здійснено адаптацію структури даних додатку для збереження списку одержувачів, їхнього управління через CRUD-інтерфейс на стороні користувача і довантаження у сервіс у момент надсилання.

Фоновий сервіс отримує з бази даних усі договори та дозволи з датами завершення в проміжку від поточної дати до дати через 31 день, формує тіло HTML-листа з переліченими номерами і датами закінчення, а потім через Gmail API відправляє повідомлення на всі адреси зі списку. Завдяки використанню MimeKit було забезпечено коректне складання MIME-повідомлень із HTML-контентом та закодованим у форматі base64url тілом листа.

Також було реалізовано ручний інтерфейс для негайної розсилки через сторінку управління списком одержувачів. Це дозволяє адміністраторам за потреби відправляти пробні повідомлення без очікування фонового запуску.

Блок-схема роботи модуля зображена у розділі 2 на рисунку 2.5.

3.7 Розробка інтерфейсу програмного додатку

При використанні програмного продукту важливо, щоб він був зручний та зрозумілий для користувача.

Основним кольором інтерфейсу застосунку було обрано синій, так як він асоціюється з водою та водними об’єктами. Додатковими кольорами було обрано чорний та білий через їхнє контрастування між собою.

При відкритті вебзастосунку, користувача зустрічає головна сторінка, що зображена на рисунку 3.4.

Так як кінцевий продукт – це вебсистема, прийнято розділяти її на сторінки, тому на сторінках доступне меню навігації (рис. 3.5), що включає в себе декілька меню з вкладеними посиланнями (рис. 3.6 та 3.7).

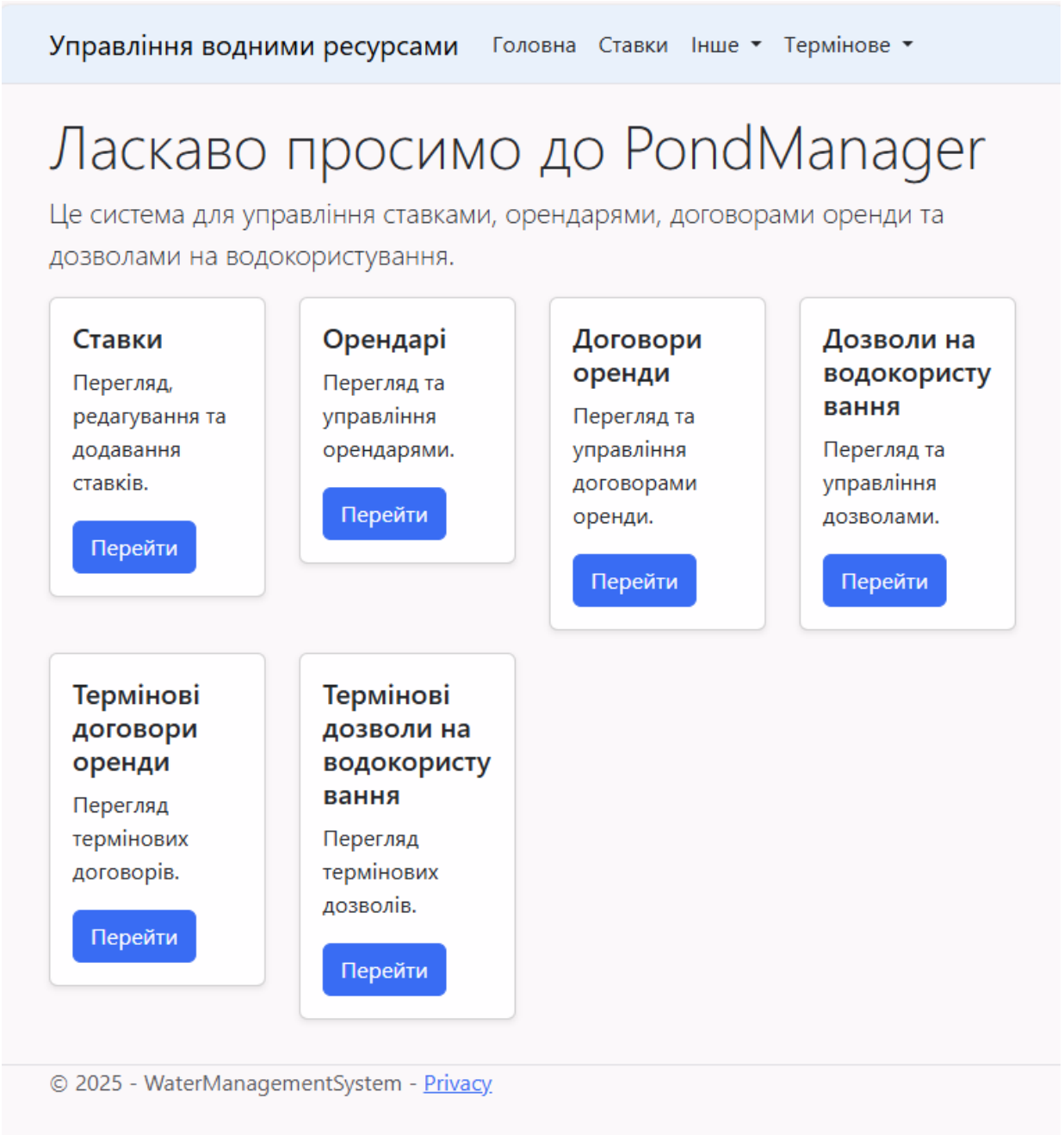


Рисунок 3.4 – Головна сторінка вебсистеми

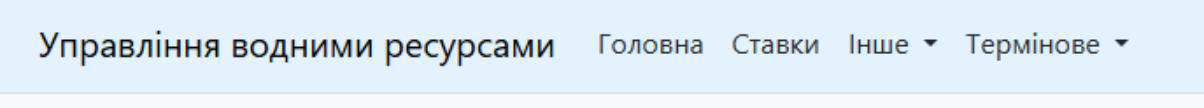


Рисунок 3.5 – Меню навігації

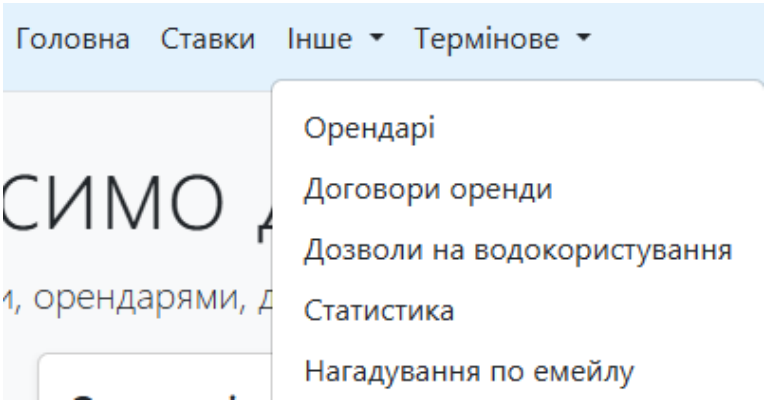


Рисунок 3.6 – Вкладене меню «Інше»

Кожен з пунктів вкладеного меню навігації «Інше» та вкладеного меню навігації «Термінове» є гіперпосиланням на відповідну сторінку. Доступ до вкладених меню є з будь-якої сторінки вебзастосунку.

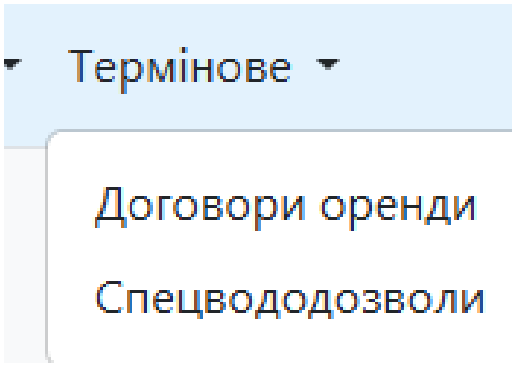


Рисунок 3.7 – Вкладене меню «Термінове»

Перша, і основна для роботи сторінка – «Ставки». На ній присутній список ставків у вигляді таблиці, яку можна гортати, так як кількість полів настільки велика, що всі не вміщаються на одному екрані (рис. 3.8). Також є кнопка операції з ставками, що відкриває меню для операцій додавання, імпорту та експорту ставків, що зображено на рисунку 3.9. Присутні також кнопки «Фільтрування» та

«Очистити фільтри». Перша відкриває випадаюче меню з фільтрми, на основі яких відображаються ставки. По замовчуванню фільтри відображають усі ставки, що зображено на рисунку 3.10.

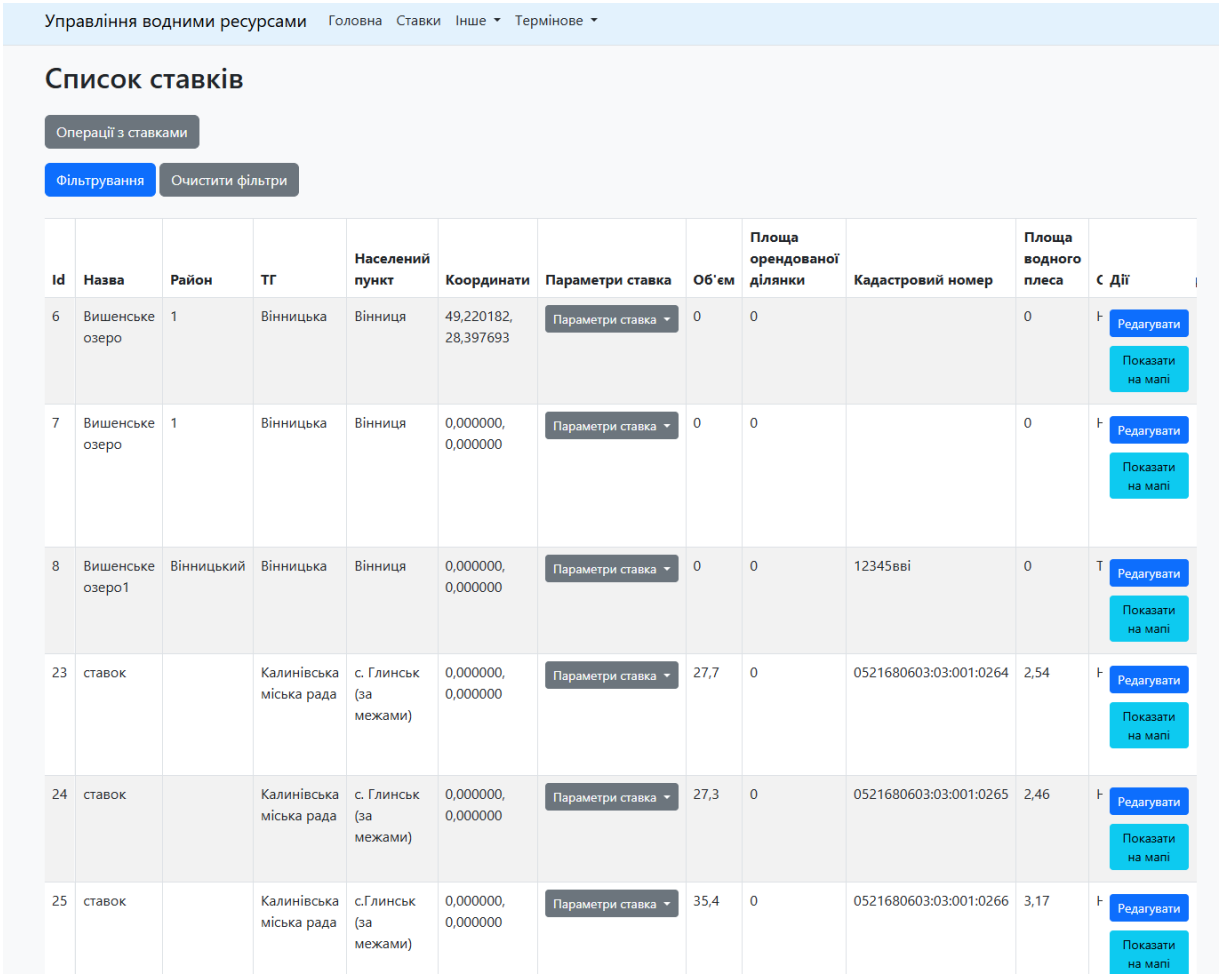


Рисунок 3.8 – Таблиця зі списком ставків

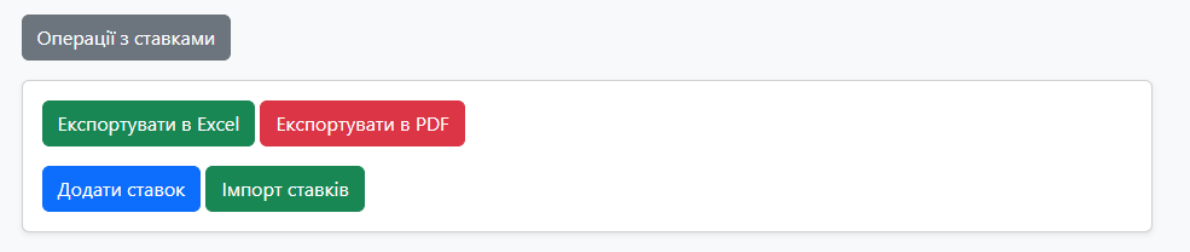


Рисунок 3.9 – Випадаюче меню з операціями над ставками

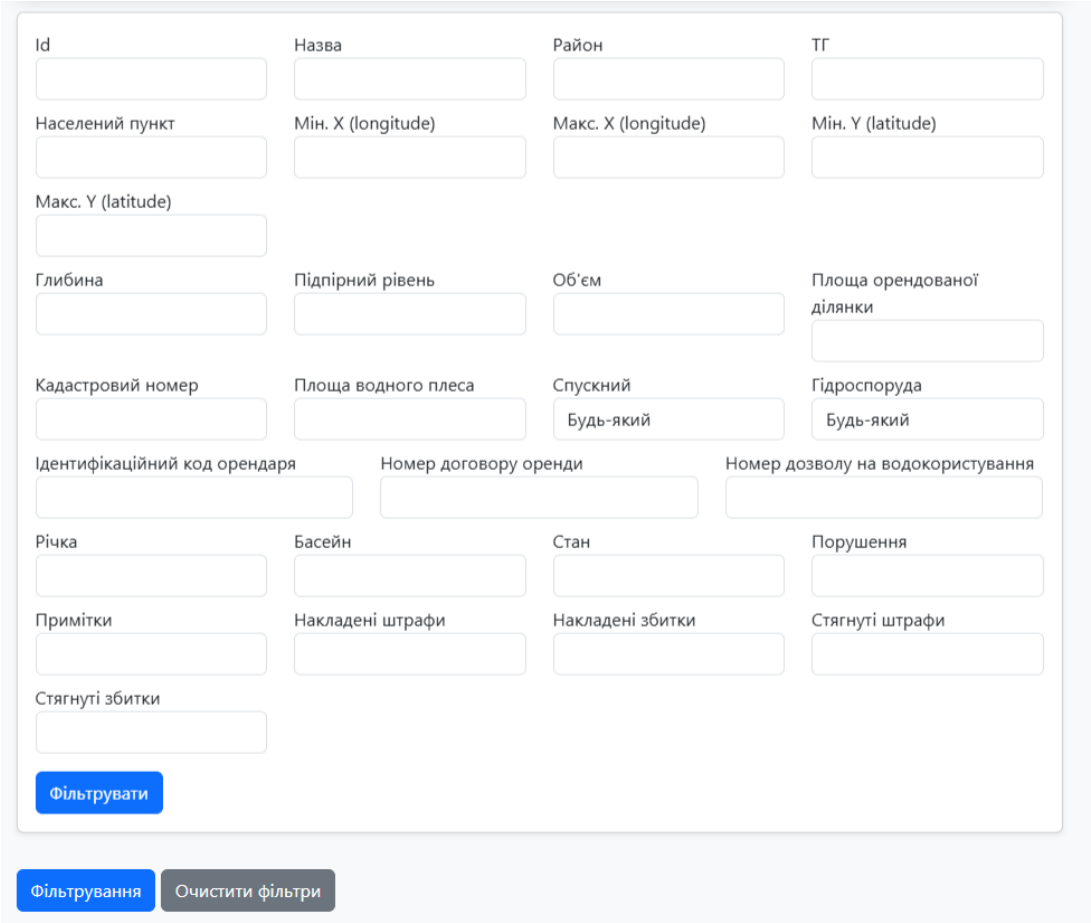


Рисунок 3.10 – Випадаюче меню з фільтрами для ставків

Також, на сторінці з ставками присутня карта, на якій відображаються відфільтровані ставки з таблиці (рис. 3.11).

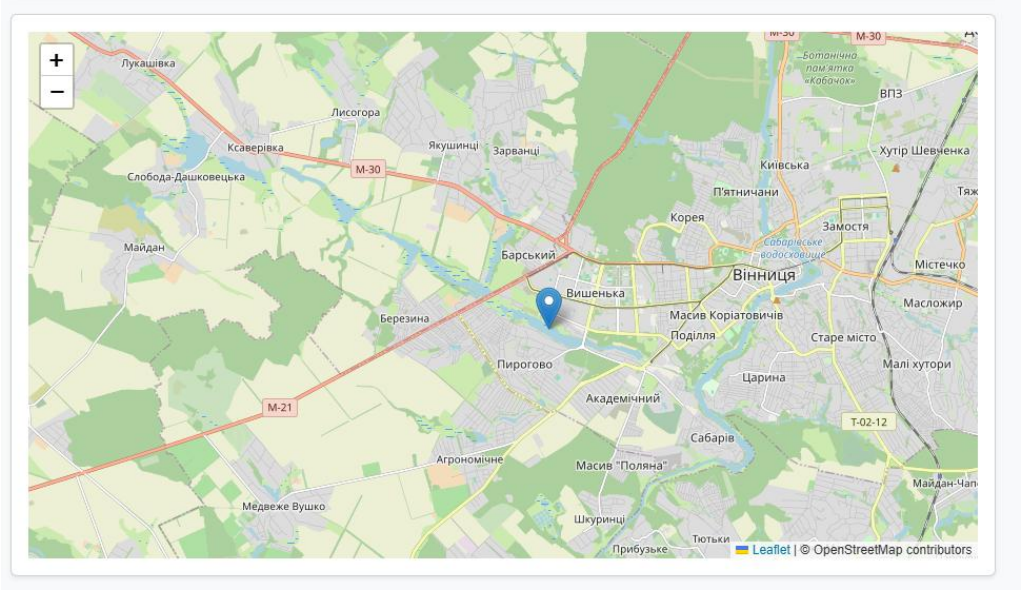


Рисунок 3.11 – Мапа з відфільтрованими ставками

Кожен окремий ставок можна редагувати, натиснувши відповідну кнопку у колонці «Дії». Після чого відкриється сторінка редагування ставка з усіма полями ставка (рис. 3.12).

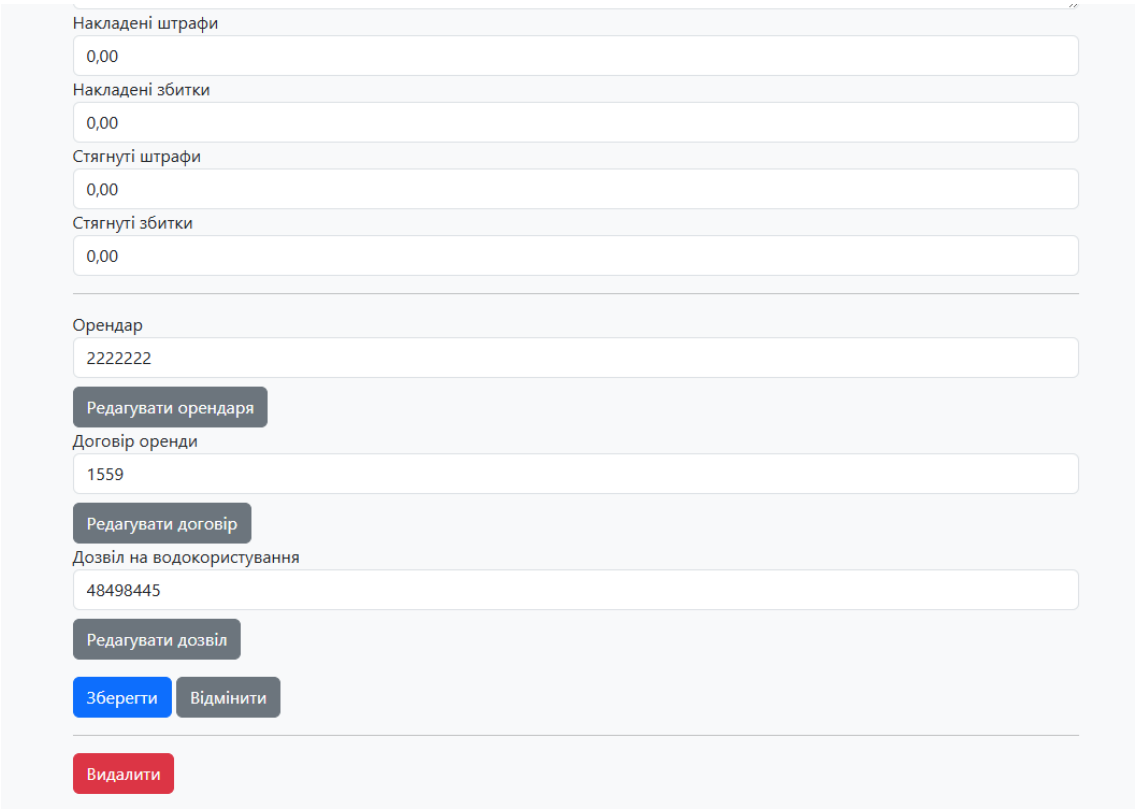


Рисунок 3.12 – Кінець сторінки редагування ставка

У всіх сторінок у вкладці «Інше» однакова структура: таблиця з інформацією та можливість редагувати чи видалити об’єкт (рис. 3.13). Сторінка редагування кожного об’єкта дозволяє змінити будь-яке його поле (рис. 3.14).

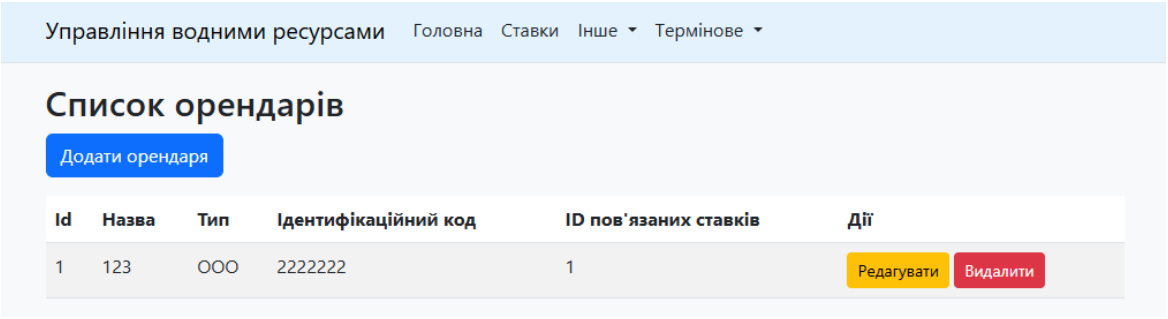


Рисунок 3.13 – Сторінка «Список орендарів»

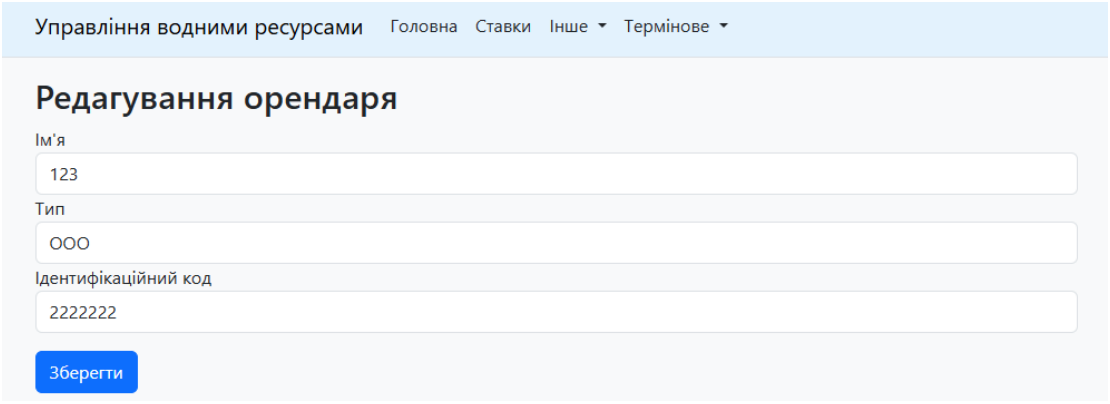


Рисунок 3.14 – Сторінка редагування орендаря

У вкладці «Термінове» є сторінки, що дозволяють подивитись договори оренди та спецвододозволи, що закінчуються через вказану кількість днів чи місяців (рис. 3.15).

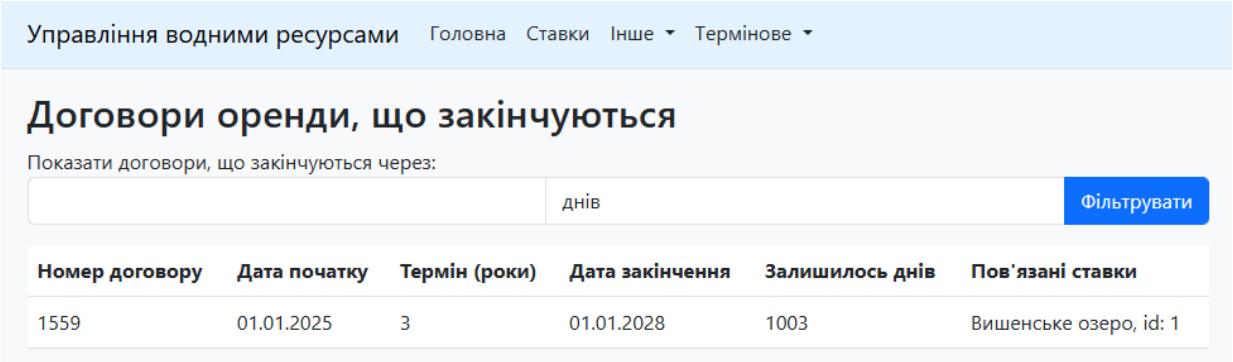


Рисунок 3.15 – Сторінка «Договори оренди, що закінчуються»

На сторінці статистики (рис. 3.16 та 3.17) є декілька видів репрезентації даних:

- загальні блоки з числовим значенням:
  - всього ставків;
  - середня глибина (м);
  - середній об’єм ($м^3$);
  - середня площа оренди ($м^2$);

- стовпчасті діаграми:
    - кількість ставків за районами;
    - кількість ставків за басейнами;
    - середня глибина ставків за районами;
- кругові діаграми:
    - розподіл термінів договорів;
    - спускні/неспускні ставки;
    - наявність гідроспоруд.

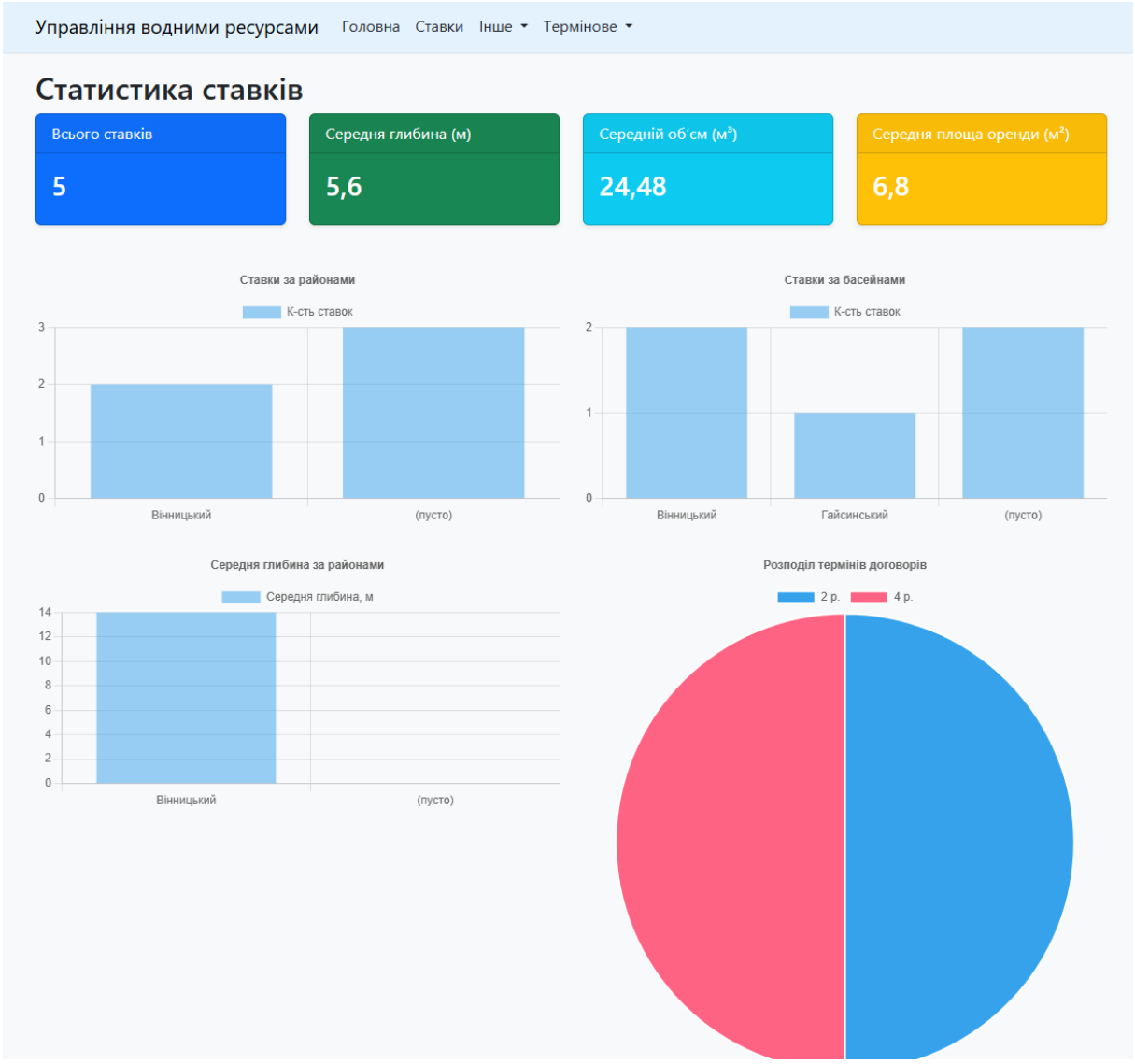


Рисунок 3.16 – Сторінка «Статистика ставків» з загальними блоками з числовими значеннями, стовпчастими діаграмами та круговою діаграмою

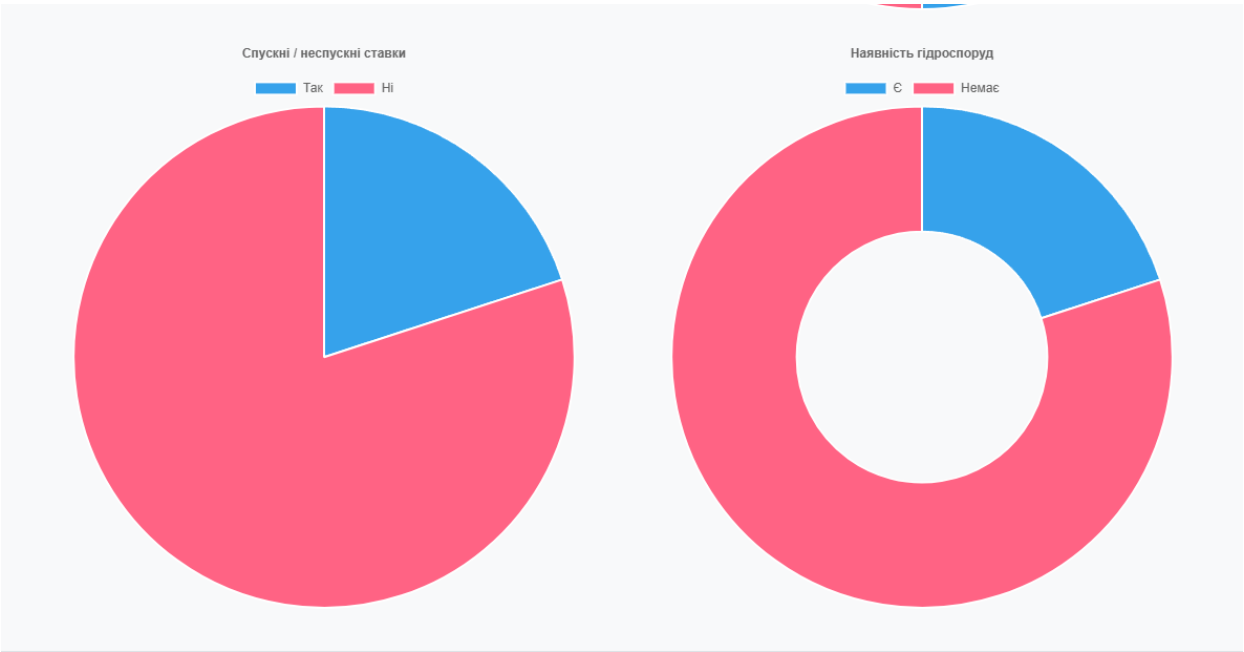


Рисунок 3.17 – Сторінка «Статистика ставків» з загальними блоками з числовими значеннями, стовпчастими діаграмами та круговою діаграмою

Розробка графічного інтерфейсу відбувалась за допомогою HTML, CSS, JavaScript та Razor Pages та мови програмування C# у середовищі розробки Visual Studio.

3.8 Висновки

Отже, у третьому розділі обґрунтовано вибір інструментів та технологій, що були використані при розробці програмних модулів. Проаналізувавши програмний продукт та його задачі, було обрано .NET та інструменти, які разом з ним використовуються. Для користувацького інтерфейсу використовувались Razor Pages та Leaflet. Для бекенду використовувались ASP.NET Core та Entity Framework. Для функціоналу імпорту та експорту використовувались ClosedXMLта iText відповідно. Також у розділі детально описано розробку модулів для імпорту з Excel файлу, експорту у файли типу PDF і Excel. Також описано роботу кешування в Redis та надсилання сповіщень про договори оренди та дозволів на водокористування, що скоро закінчаться, на електронну пошту.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Тестування програмного забезпечення – це складний процес, який передбачає перевірку створеного продукту з метою виявлення помилок, дефектів та невідповідностей вимогам замовника і стандартам якості. Основна мета цього етапу розробки – забезпечити надійність, стабільність і відповідність програмного забезпечення очікуванням користувачів перед його виходом на ринок.

Перед тим, як розпочати тестування, потрібно підготувати тестове середовище: встановити програму на відповідні робочі станції (Windows, macOS або інші платформи), налаштувати необхідні компоненти та сервіси, а також забезпечити доступ до потрібних даних. Важливо мати повний пакет документації – від специфікацій вимог до детального плану тестування.

Основний етап тестування зосереджується на функціональній перевірці, яка має на меті підтвердити, що кожен модуль працює правильно. Під час функціонального тестування вводять як коректні, так і некоректні дані, перевіряють, як система обробляє граничні значення, реагує на неконсистентні параметри та чи відповідає реалізована логіка бізнес-процесам. Головне завдання цього етапу – впевнитися, що всі функції виконують свої призначені дії і можуть обробляти виняткові ситуації без аварійного завершення.

Тестування відмов полягає в штучному створенні екстремальних умов – некоректних вхідних даних, раптових відключень зв’язку, нестачі ресурсів або одночасного виконання кількох складних операцій. Мета цього процесу – оцінити, наскільки система здатна виявляти збої, коректно їх обробляти та відновлювати роботу без втрати даних навіть у випадку відсутності зв’язку з іншими компонентами системи.

Для оцінки продуктивності використовують тести швидкодії, навантажувальні та стрес-тести. Тут вимірюють час відповіді на різні запити, пропускну здатність системи під піковим навантаженням, а також рівень

використання процесора, пам’яті й мережевих ресурсів. Результати цих тестів допомагають виявити вузькі місця в системі.

Тестування сумісності перевіряє, як програмне забезпечення взаємодіє з різними операційними системами, версіями сторонніх бібліотек, апаратними платформами та іншими програмами. Це дозволяє впевнитися, що продукт працює стабільно в різних умовах і не викликає конфліктів із вже встановленим ПЗ чи драйверами.

Крім того, важливо проводити регресійне тестування після внесення змін у код, щоб переконатися, що виправлення або нові функції не призвели до появи нових помилок. У сучасних проєктах все більше уваги приділяється автоматизації тестів, що дозволяє регулярно перевіряти критичні шляхи застосунку та прискорює процес випуску програмного забезпечення.

Під час основного етапу перевіряється реакція вебсистеми на неправильно введені дані. Система має повідомити, якщо при заповненні форми будуть невалідні дані чи при вивантаженні файлу буде вивантажено невірний файл. Наприклад, якщо у форму імпортування ставків вставити файл, якщо матиме формат не xls чи xlsx, то після натискання кнопки «Імпортувати» система виведе відповідне повідомлення про помилку, що зображено на рисунку 4.1.

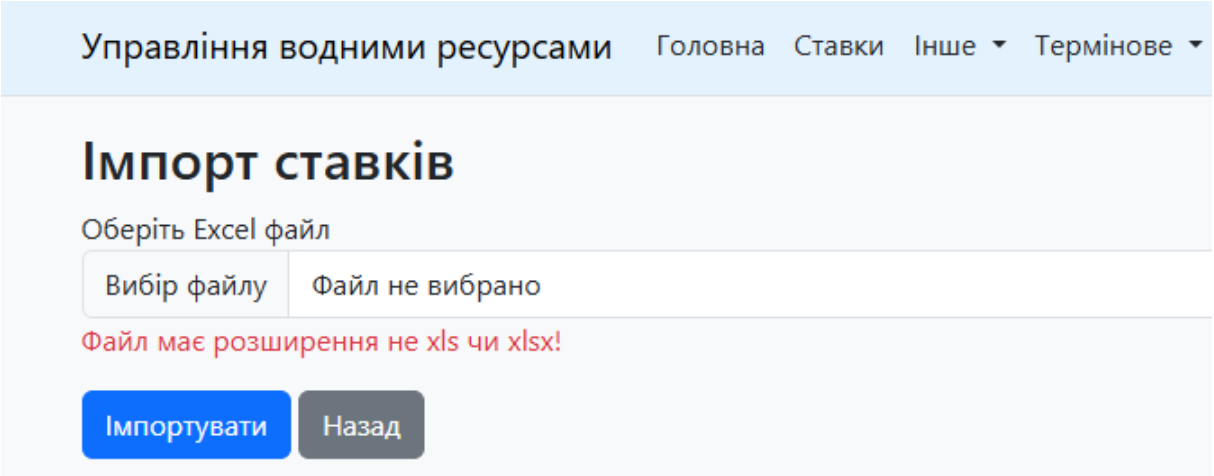


Рисунок 4.1 – Повідомлення про помилку вивантаження файлу з невідповідним розширенням

Якщо у форму «Редагування ставка» не вписати X координату, то при натисканні кнопки збереження, система поверне вікно на початок сторінки та виділить місце з невалідними даними. На рисунку 4.2 зображено результат цього сценарію виконання.

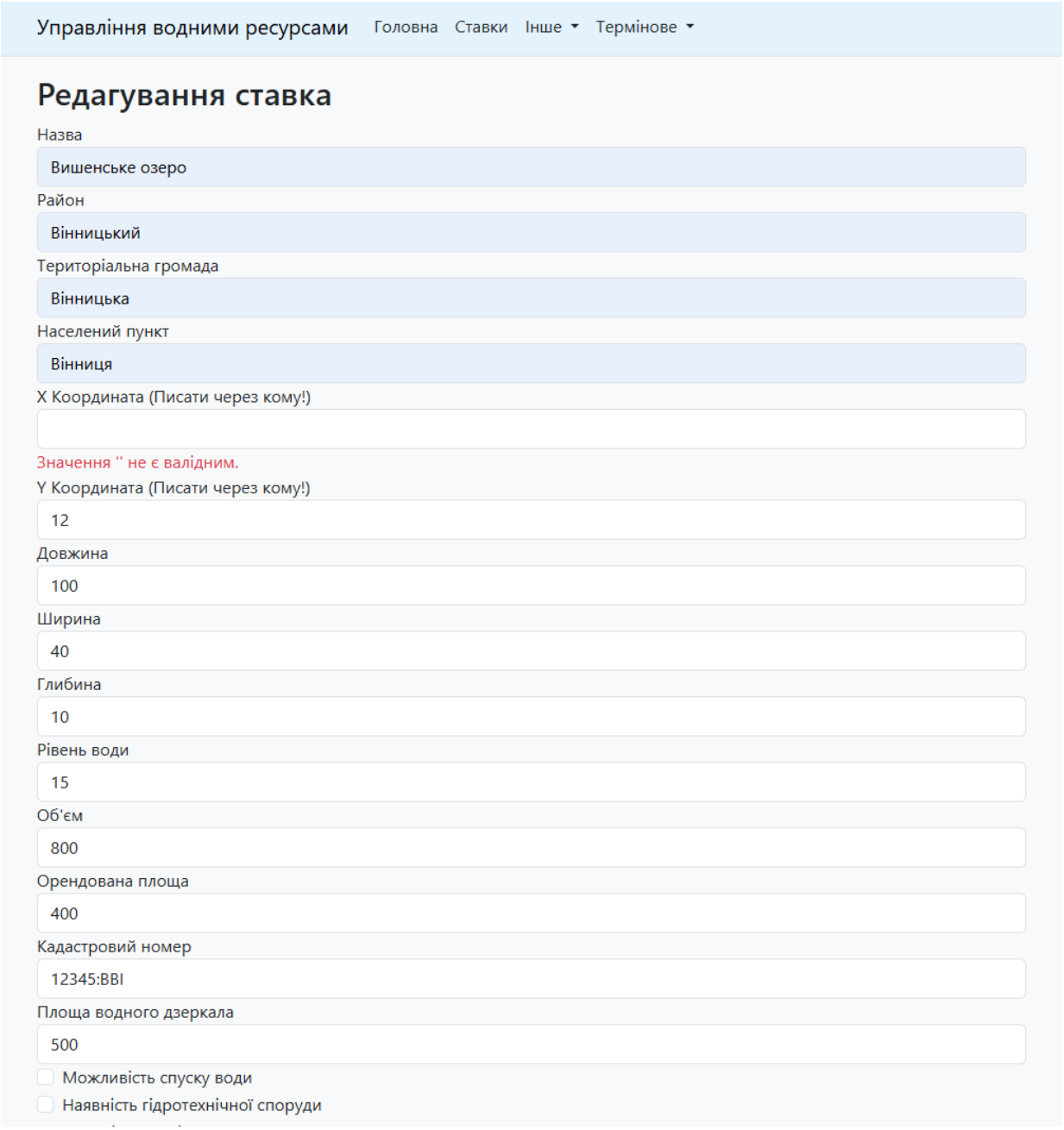


Рисунок 4.2 – Частина форми редагування ставка з невалідно введеною X координатою

Наступним етапом тестування, після перевірки на валідність даних, що вводяться, є перевірка функціональності системи на імпорт ставків з Excel файлу. Для тестування використовується файл, що містить в собі декілька ставків з Калинівської міської ради Хмільницького району, що зображений на рисунку 4.3.

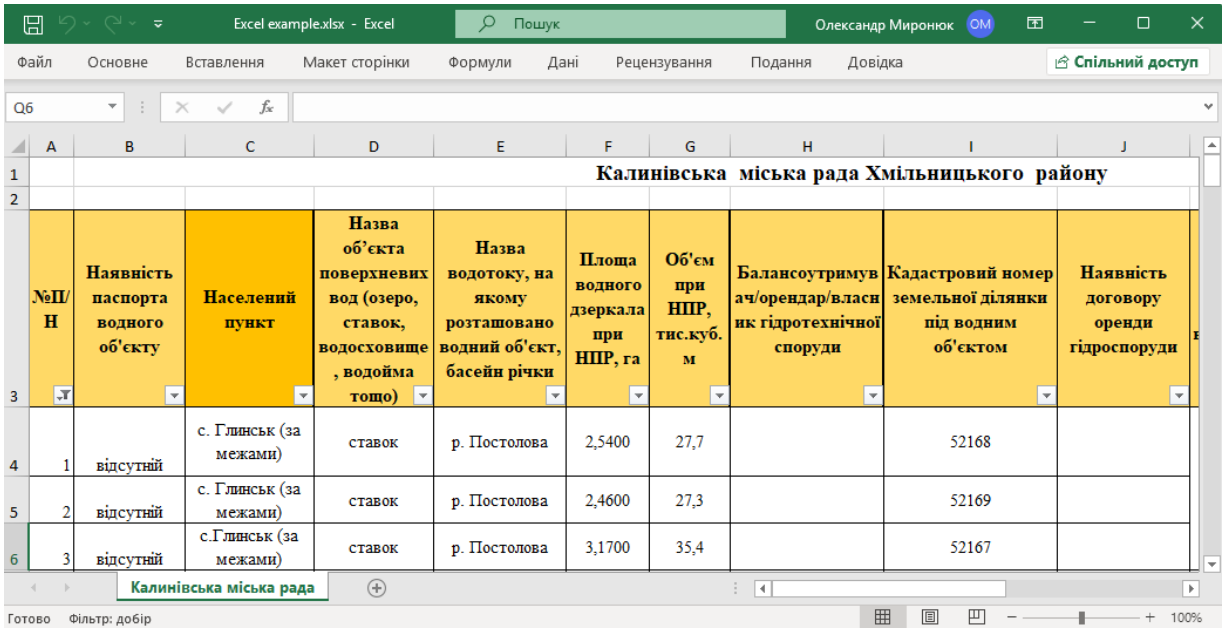


Рисунок 4.3 – Частина Excel файлу, що було імпортовано з трьома ставками Калинівської міської ради Хмільницького району

Після вивантаження файлу «Excel example» через сторінку імпорту ставків, на сторінці з усіма ставками з’являються ті ставки, що були у файлі для імпорту. Нові ставки у таблиці з усіма зображено на рисунку 4.4.

Тестування функціоналу додавання нового ставка вимагає заповнення усіх полів форми створення ставка валідними даними, після чого ставок з’явиться у таблиці з усіма ставками. Форма, що заповнена валідними даними зображена на рисунку 4.5. Результат виконання – створений ставок з валідними даними з форми зображено на рисунку 4.6.

Фільтрування ставків, що відображаються у таблиці є важливим функціоналом вебсистеми, тому для тестування було обрано відобразити ті ставки, що знаходяться на річці Постолова (тобто, ставки, що мають ID від 3 до 5 включно).

Введені фільтри зображені на рисунку 4.7. Після натиснення кнопки «Фільтрувати», у таблиці залишаться лише ті ставки, що знаходяться на річці Постолова (рис. 4.8).

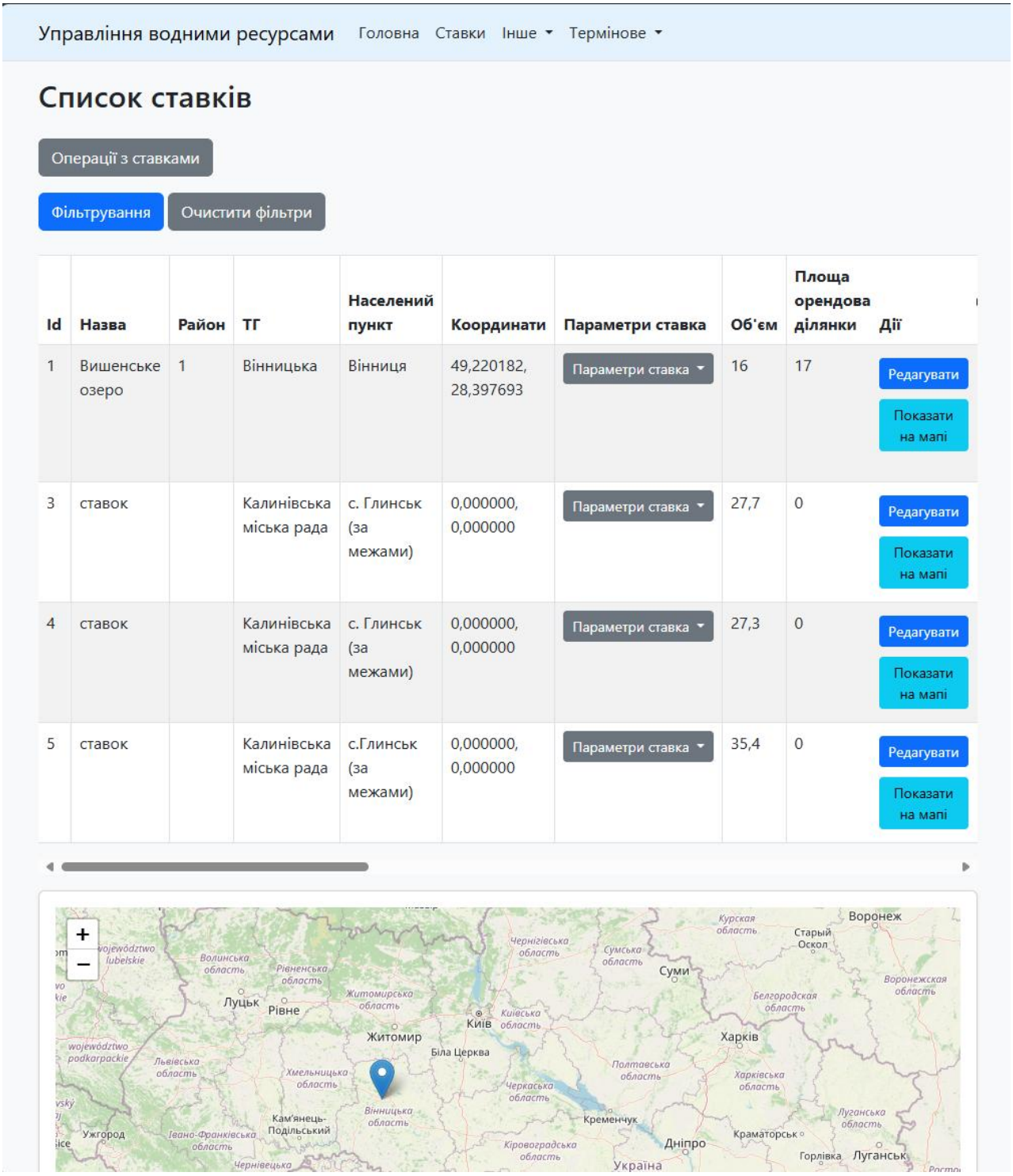


Рисунок 4.4 – Усі ставки, включно з імпортованими ставками з файлу «Excel example.xlsx»

Рисунок 4.5 – Частина валідно заповненої форми «Додати ставок»

Ставки, що відображаються у таблиці також відображаються на мапі. Відповідно, так як у ставків, що знаходяться на річці Постолова не введені координати – вони не зображаються на мапі, що зображено на рисунку 4.9. Проте, якщо очистити фільтри, і до списку повернуться ставки, що мають координати – вони будуть зображені на мапі (рис. 4.10).

У кожного ставка є дія «Показати на мапі». При натискані на цю кнопку – фокус перейде на мапу, а в самій мапі – фокус зміститься до обраного ставка, що зображено на рисунку 4.11.

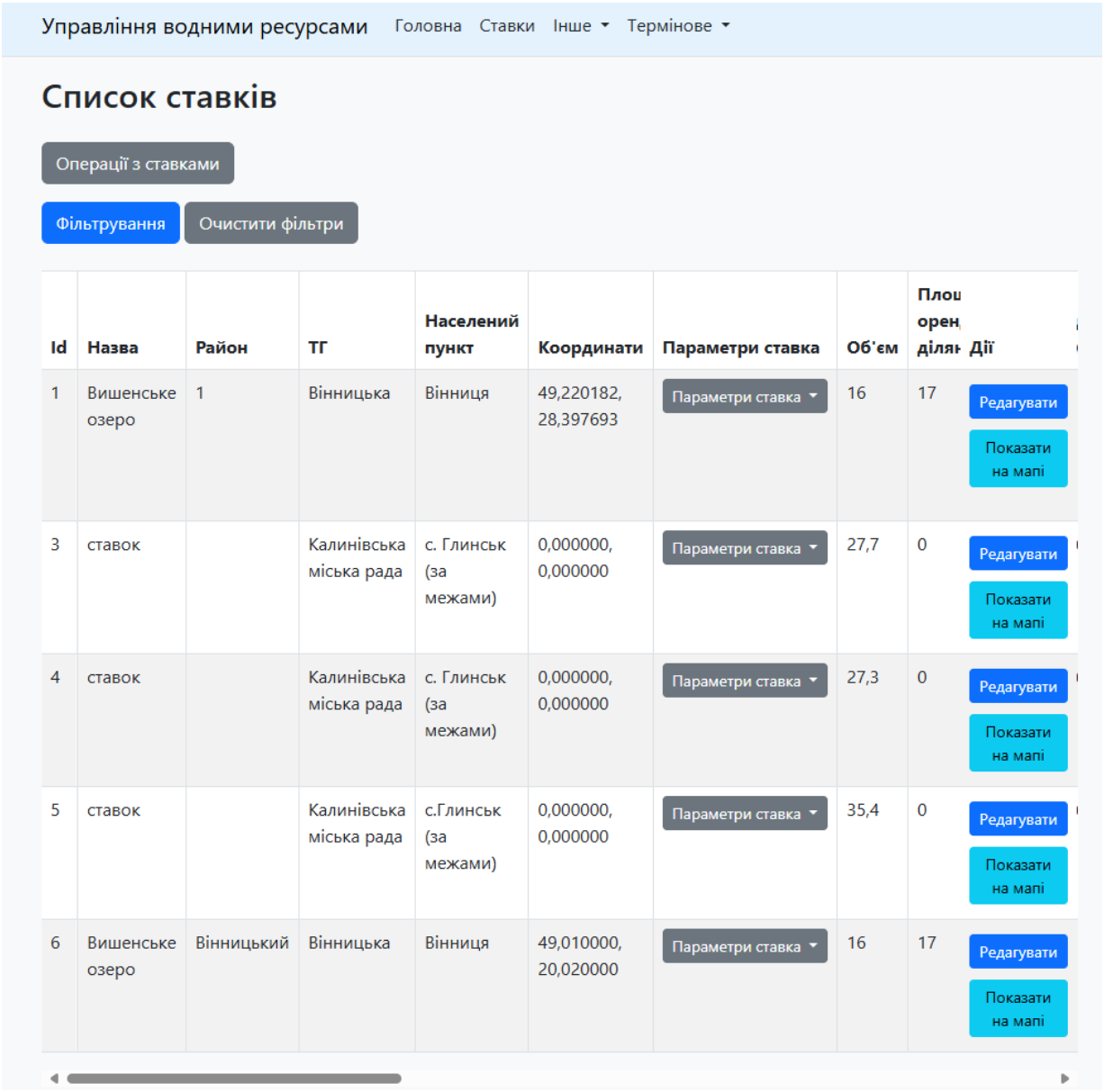


Рисунок 4.6 – Список усіх ставків з новоствореним ставком (ID ставка: 6)

Функціонал редагування ставка дозволяє змінити будь-яку властивість об’єкта, тому, якщо для ставка Вишенське озеро з ID: 1 натиснути кнопку редагувати, та район змінити з «1» на «Вінницький» – у списку ставків з’являться зміни для відповідного об’єкту (рис. 4.12).

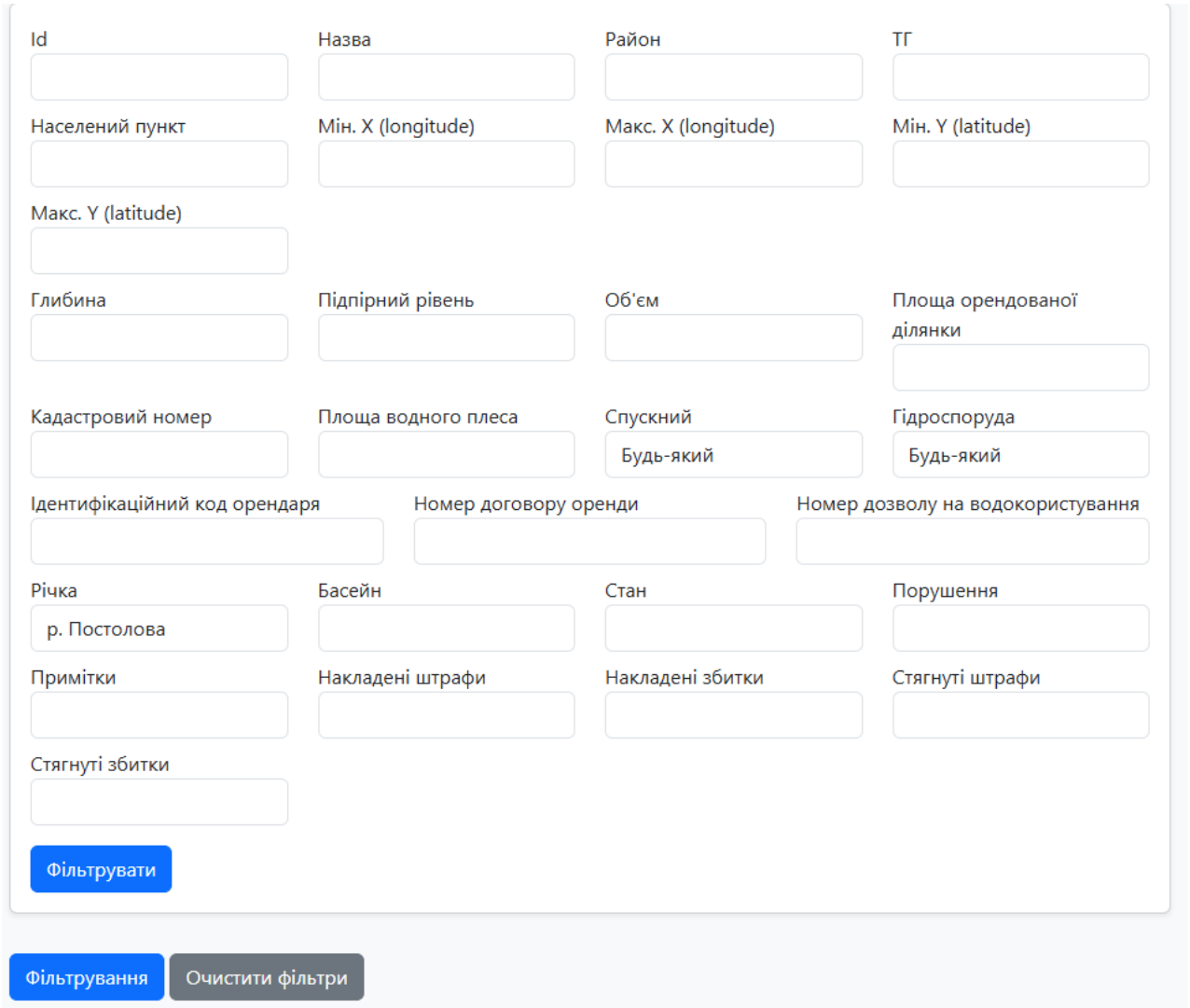


Рисунок 4.7 – Фільтрування за ставками, що знаходяться на річці Постолова

Експортуються лише інформація про ставки, що зараз знаходяться у списку. Після натискання клавіші «Експортувати в Excel» чи «Експортувати в PDF» буде сформовано та завантажено файл відповідного типу з інформацією про відфільтровані ставки. Наприклад, щоб експортувати тільки ті ставки, чия X координата знаходиться у проміжку від 45 до 50, потрібно ввести відповідні фільтри у меню фільтрів (рис. 4.13) та натиснути кнопку для експортування у відповідний формат. Результат експорту у Excel зображено на рисунку 4.14. Результат експорту у PDF зображено на рисунку 4.15.

Редагування орендарів, договорів оренди та дозволів на водовикористання виконується схожим чином. Потрібно перейти на відповідну сторінку, обрати

відповідного орендаря, договір оренди чи дозвіл на водовикористання та у колонці «Дії» обрати пункт «Редагувати». Наприклад, для орендаря з ID: 1 необхідно змінити тип з «ТОВ» на «ФОП». Список орендарів до змін та після змін наведено на рисунках 4.16 та 4.17 відповідно.

У договорі оренди з ID: 1 необхідно змінити термін з трьох до чотирьох років. Список договорів оренди до та після змін наведено на рисунках 4.18 та 4.19 відповідно.

Для дозволу на водокористування з ID: 1 необхідно змінити термін з трьох до шести років. Список дозволів на водокористування до та після змін наведено на рисунках 4.20 та 4.21 відповідно.

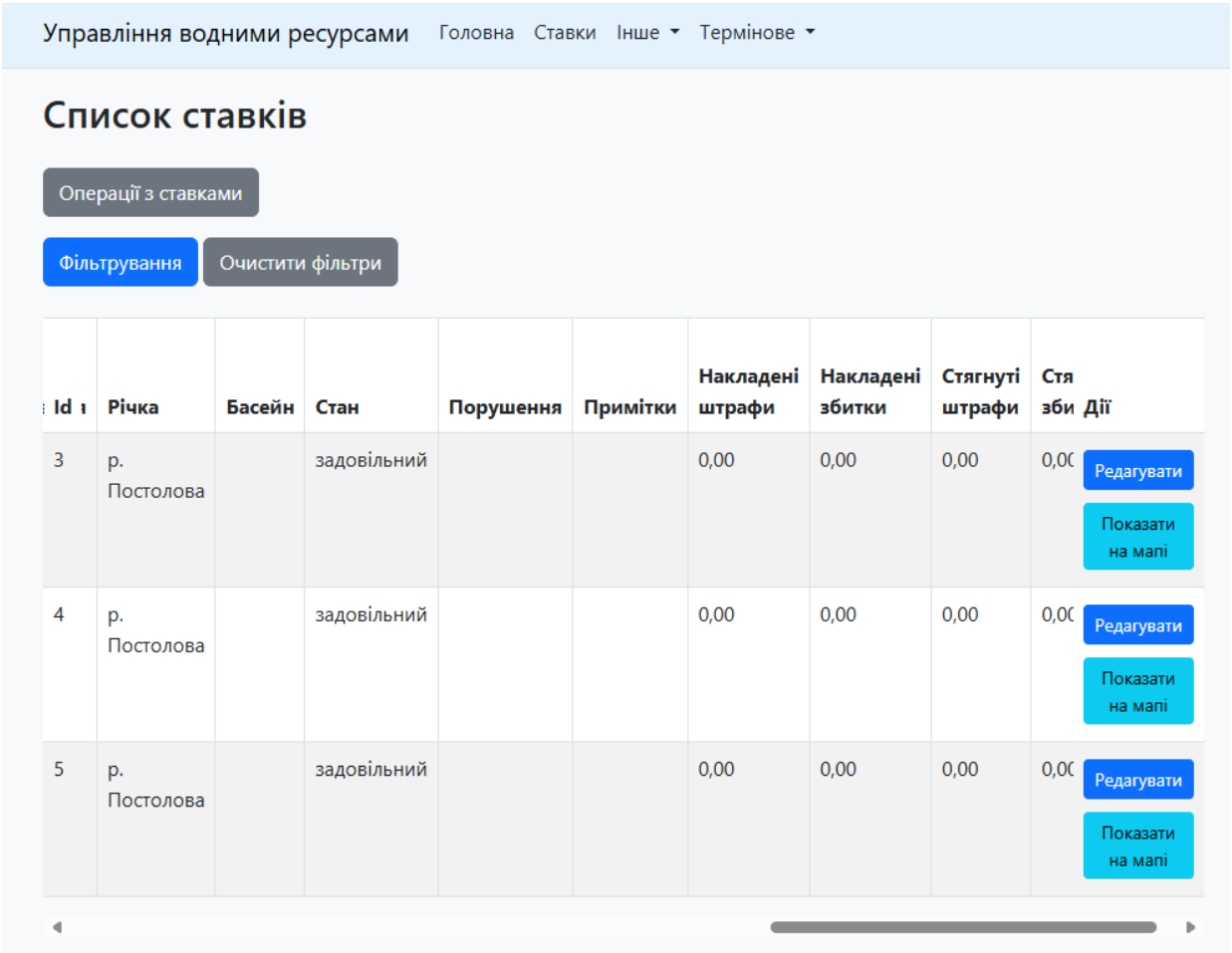


Рисунок 4.8 – Відфільтровані ставки за річкою Постолова

Id	Назва	Район	ТГ	Населений пункт	Координати	Параметри ставка	Об'єм	Площа орендованої ділянки	Дії
3	ставок		Калинівська міська рада	с. Глинськ (за межами)	0,000000, 0,000000	Параметри ставка	27,7	0	Редагувати / Показати на мапі
4	ставок		Калинівська міська рада	с. Глинськ (за межами)	0,000000, 0,000000	Параметри ставка	27,3	0	Редагувати / Показати на мапі
5	ставок		Калинівська міська рада	с.Глинськ (за межами)	0,000000, 0,000000	Параметри ставка	35,4	0	Редагувати / Показати на мапі

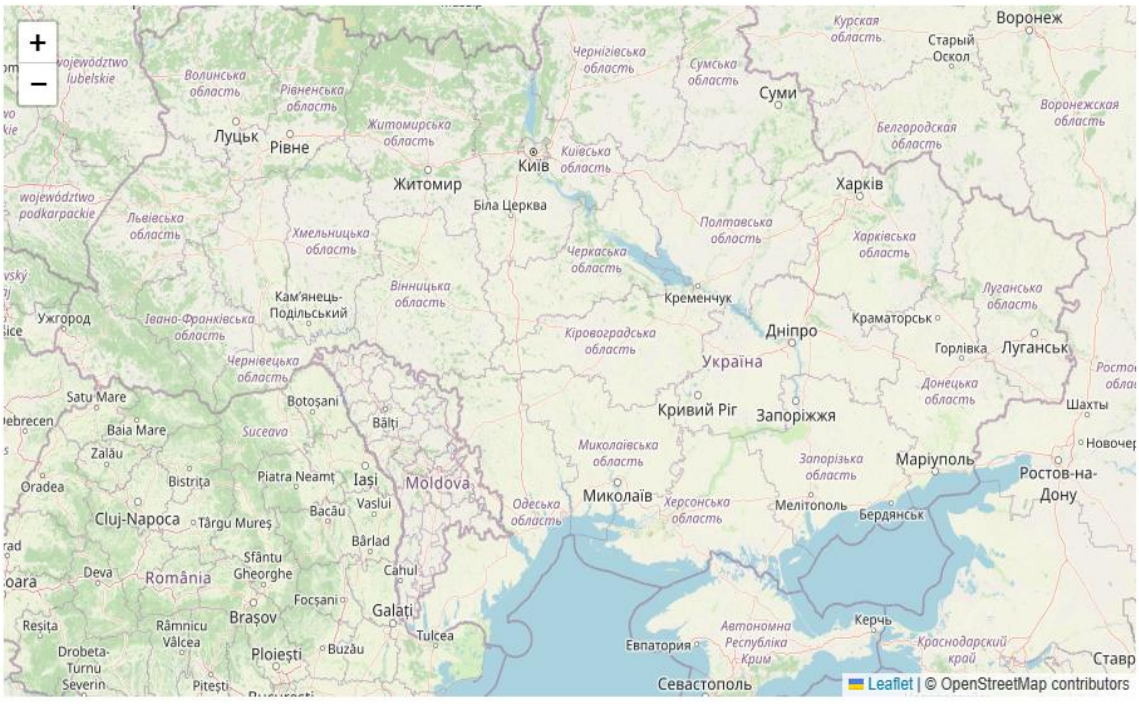


Рисунок 4.9 – Ставки без координат та пуста мапа

Якщо у ставка X координата має значення 0.00000 та Y координата має значення 0.00000, то на мапі взагалі не буде позначки, навіть враховуючи, що вона мала б бути у південній частині Атлантичного океану.

Id	Назва	Район	ТГ	Населений пункт	Координати	Параметри ставка	Об'єм	Площа орен ділян	Дії
1	Вишенське озеро	1	Вінницька	Вінниця	49,220182, 28,397693	Параметри ставка	16	17	Редагувати Показати на мапі
3	ставок		Калинівська міська рада	с. Глинськ (за межами)	0,000000, 0,000000	Параметри ставка	27,7	0	Редагувати Показати на мапі
4	ставок		Калинівська міська рада	с. Глинськ (за межами)	0,000000, 0,000000	Параметри ставка	27,3	0	Редагувати Показати на мапі
5	ставок		Калинівська міська рада	с.Глинськ (за межами)	0,000000, 0,000000	Параметри ставка	35,4	0	Редагувати Показати на мапі
6	Вишенське озеро	Вінницький	Вінницька	Вінниця	49,010000, 28,020000	Параметри ставка	16	17	Редагувати Показати на мапі

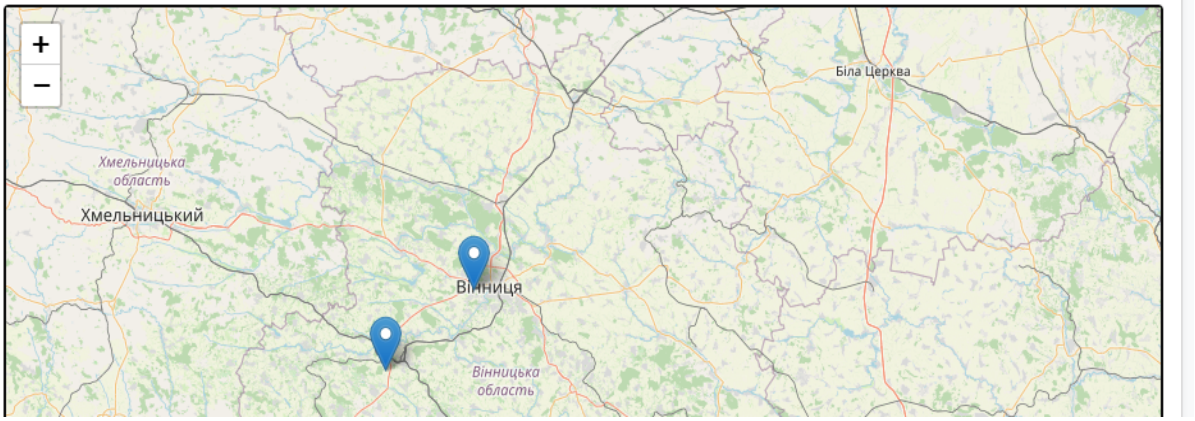


Рисунок 4.10 – Ставки з координатами, що зображені на мапі

При переході на сторінку з ставками, мапа буде встановлена на усю Україну з центральними координатами 48,3794 по X та 31,1656 по Y. Приклад зображено на рисунку 4.9.

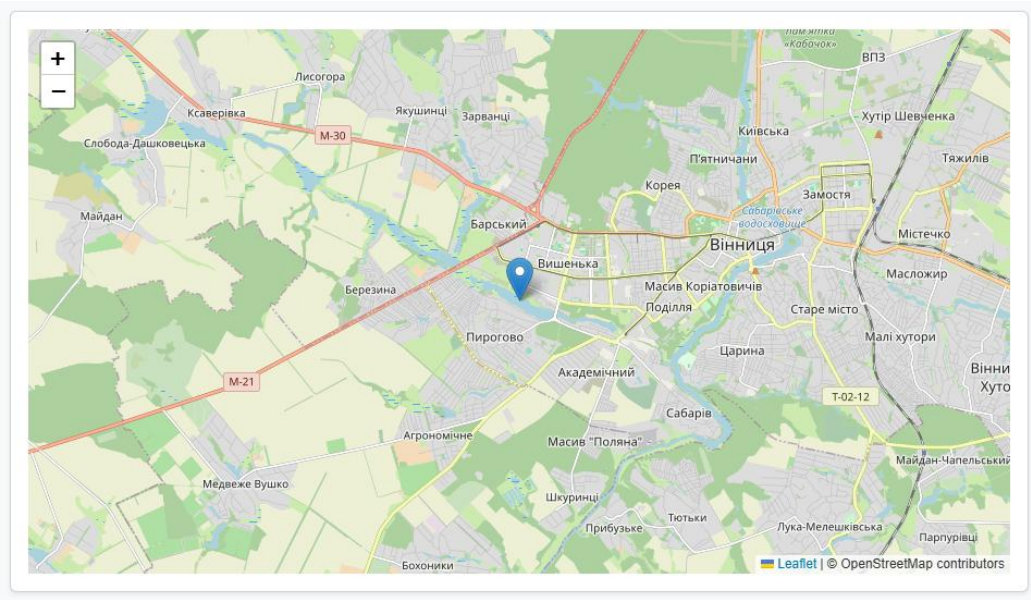


Рисунок 4.11 – Вигляд мапи, після натиску кнопки «Показати на мапі» для Вишенського озера з ID: 1

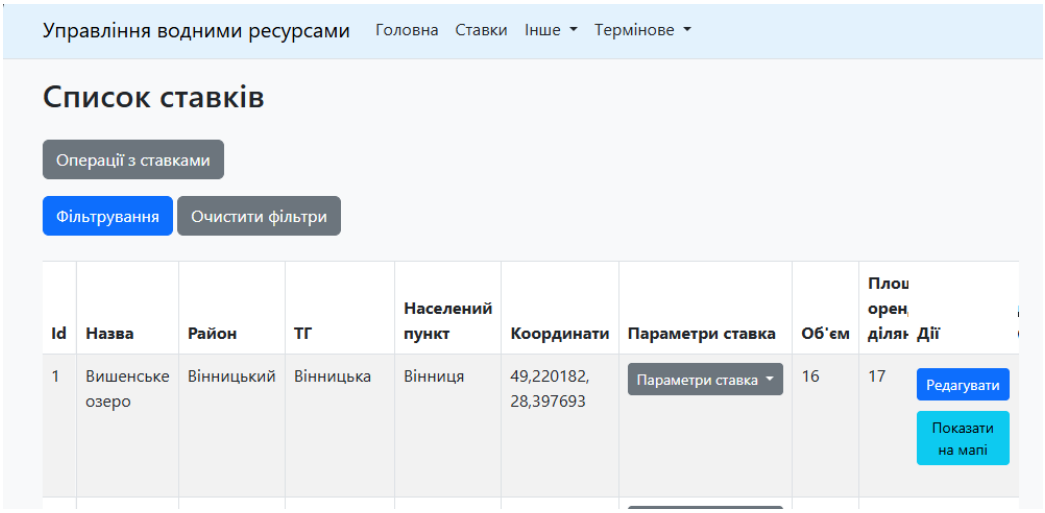


Рисунок 4.12 – Змінений ставок з ID: 1

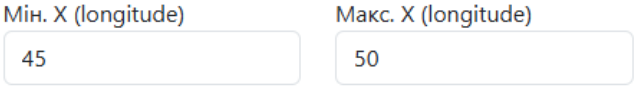


Рисунок 4.13 – Фільтрування ставків за X координатою у проміжку від 45 до 50

Координати мають бути від -180 до 180 для X та від -90 до 90 для Y.

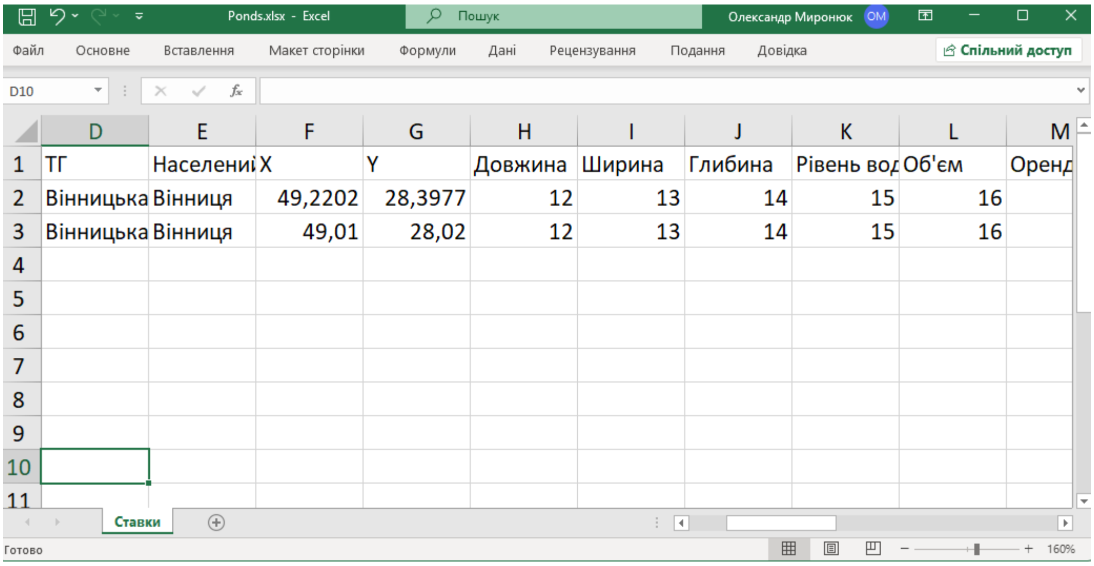


Рисунок 4.14 – Результат експорту в Excel

На рисунку 4.14 зображено тільки ту частину стовпців, що помістились у вікні застосунку Excel.

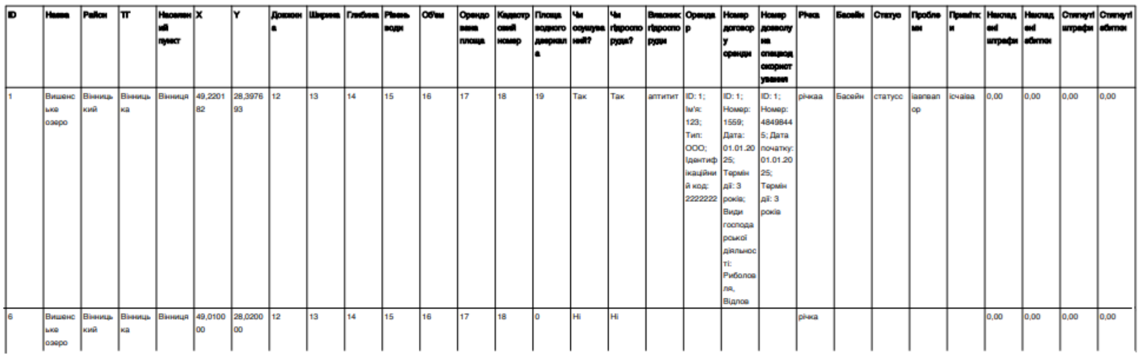


Рисунок 4.15 – Результат експорту в PDF

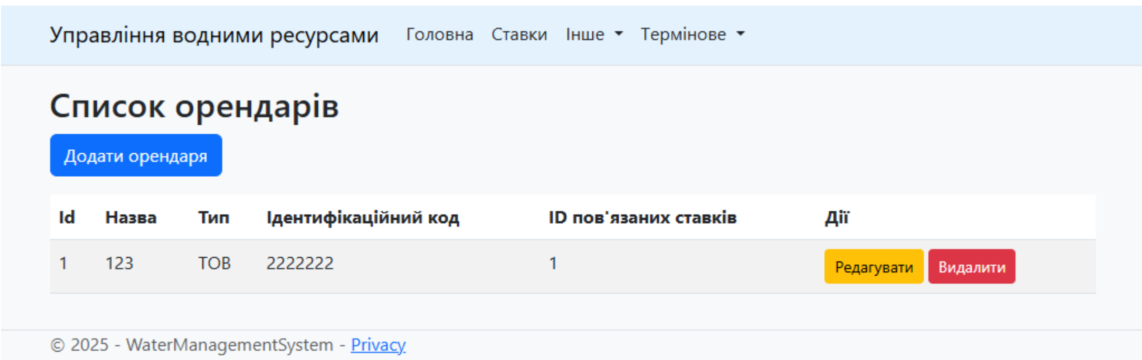


Рисунок 4.16 – Список орендарів до зміни орендаря з ID: 1

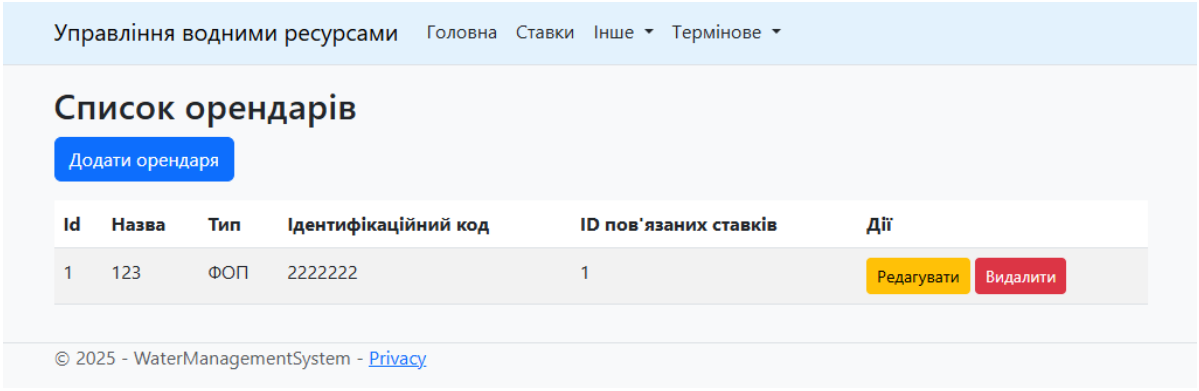


Рисунок 4.17 – Список орендарів після зміни орендаря з ID: 1

У прикладі на рисунках 4.16 та 4.17 поле «Назва» відповідає за ім’я орендаря, якщо це ФОП, чи назву, якщо це ТОВ.

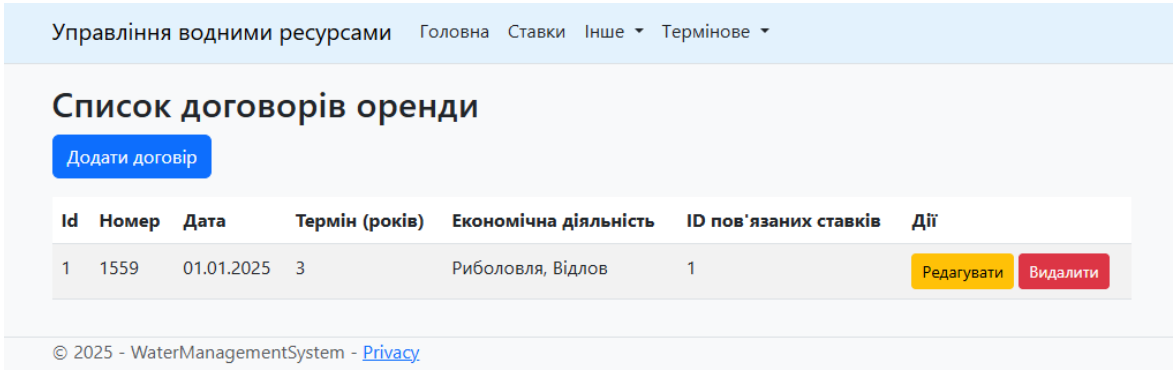


Рисунок 4.18 – Список договорів оренди до зміни договору з ID: 1

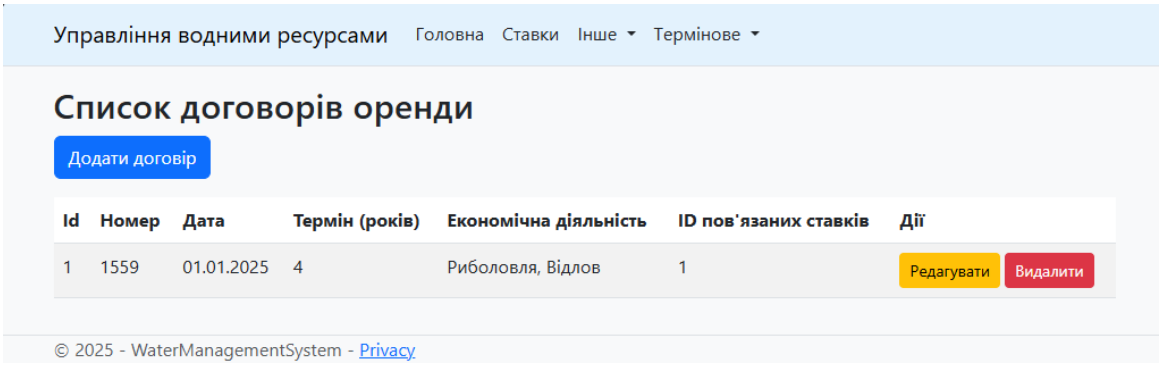


Рисунок 4.19 – Список договорів оренди після зміни договору з ID: 1

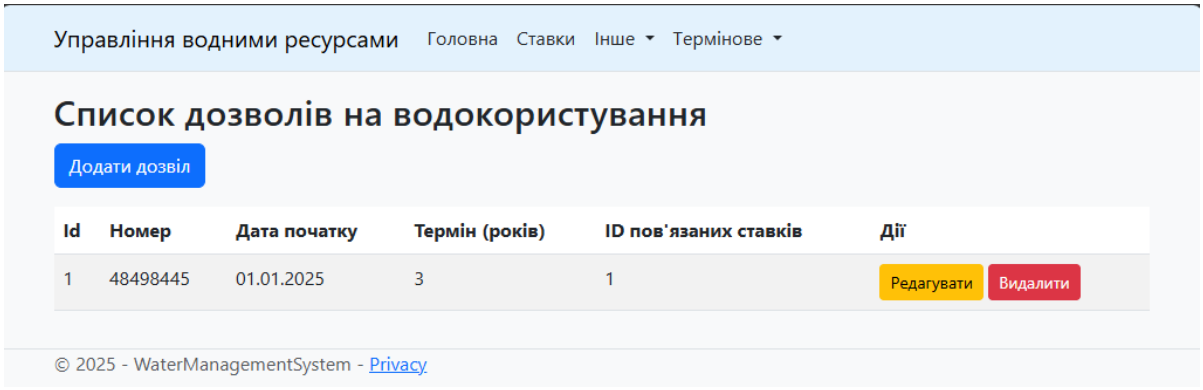


Рисунок 4.20 – Список дозволів на водокористування до зміни дозволу з ID: 1

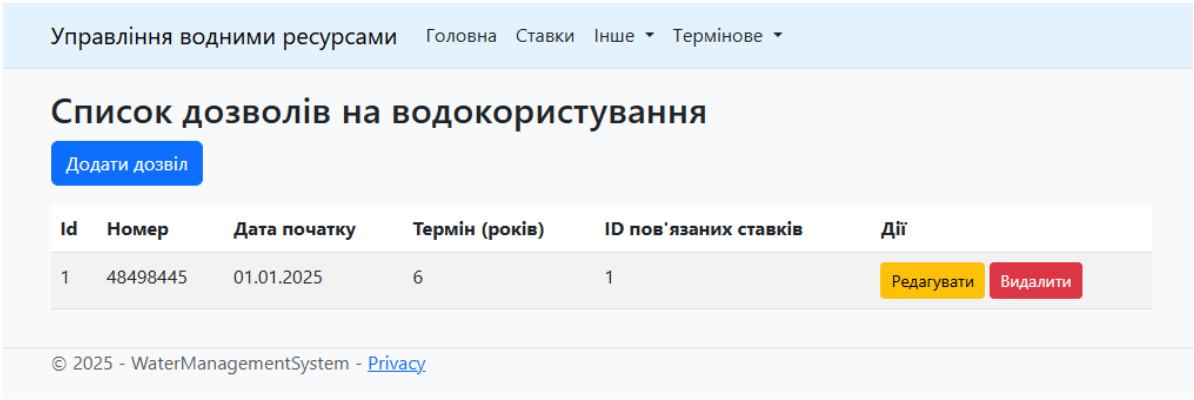


Рисунок 4.21 – Список дозволів на водокористування після зміни дозволу з ID: 1

Функціонал відправки нагадувань через електронну пошту можливо протестувати через сторінку реціпієнтів. На сторінці є таблиця з емейлами реціпієнтів а також кнопка «Надіслати нагадування зараз». При натисненні на кнопку перевіряється наявність тих договорів оренди та дозволів на водокористування, які закінчаться протягом 31 дня.

Якщо такі договори чи нагадування відсутні – система видасть відповідне повідомлення (рис. 4.22).

У іншому випадку, коли є такі договори чи дозволи, що закінчаться протягом 31 дня, система видасть повідомлення про успішне відправлення (рис. 4.23).

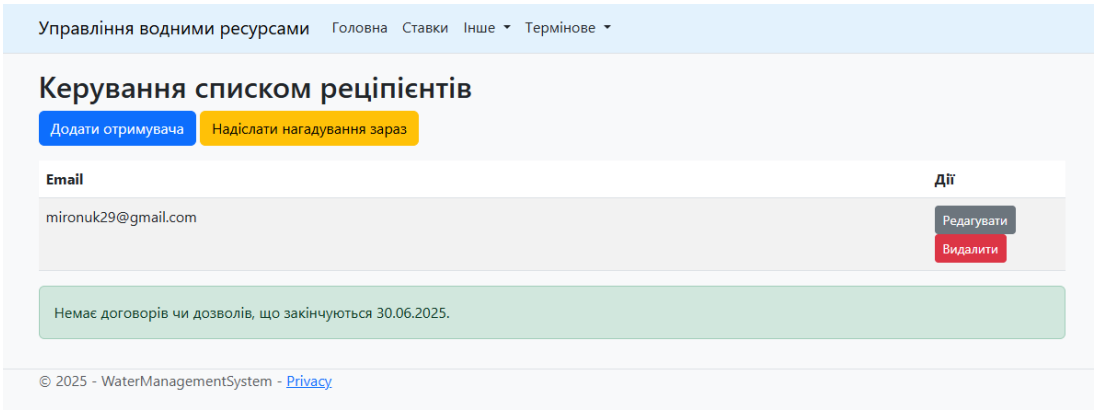


Рисунок 4.22 –Повідомлення про відсутність дозволів та договорів, що можна надіслати

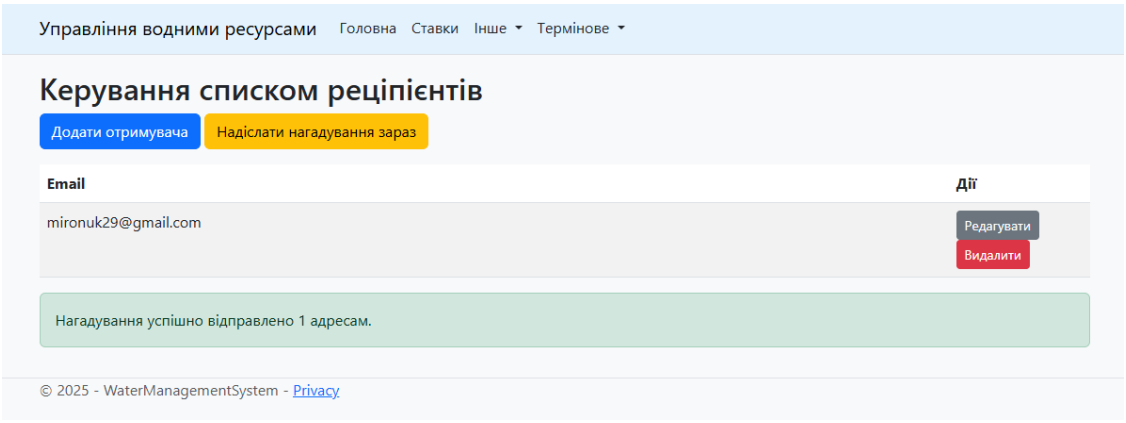


Рисунок 4.23 –Повідомлення про успішну відправку

Відправлений лист з нагадуванням зображено на рисунку 4.24.

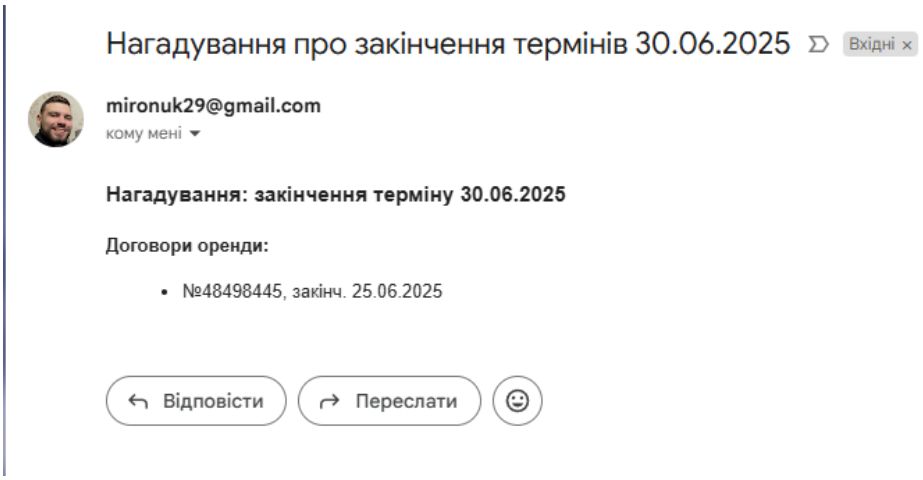
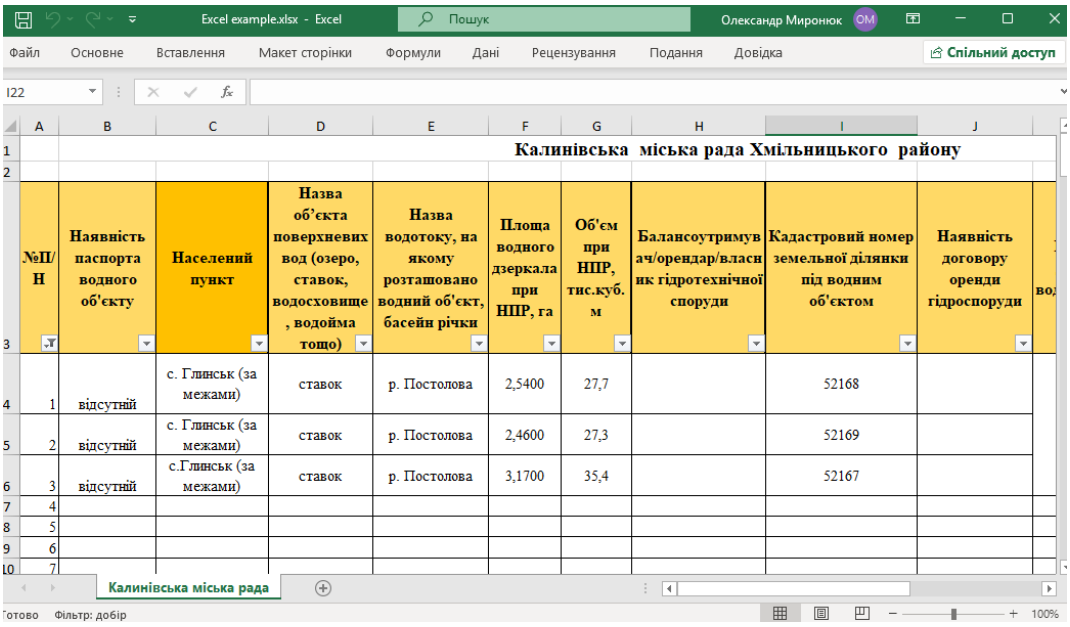


Рисунок 4.24 – Електронний лист з нагадуванням

Функціонал імпорту вимагає наявність файлу Excel, який має відповідну структуру з відповідними колонками (рис. 4.25). Результат імпорту зображений на риснуку 4.26.



Excel example.xlsx - Excel

Пошук

Олександр Миронюк

Файл Основне Вставлення Макет сторінки Формули Дані Рецензування Подання Довідка

Спільний доступ

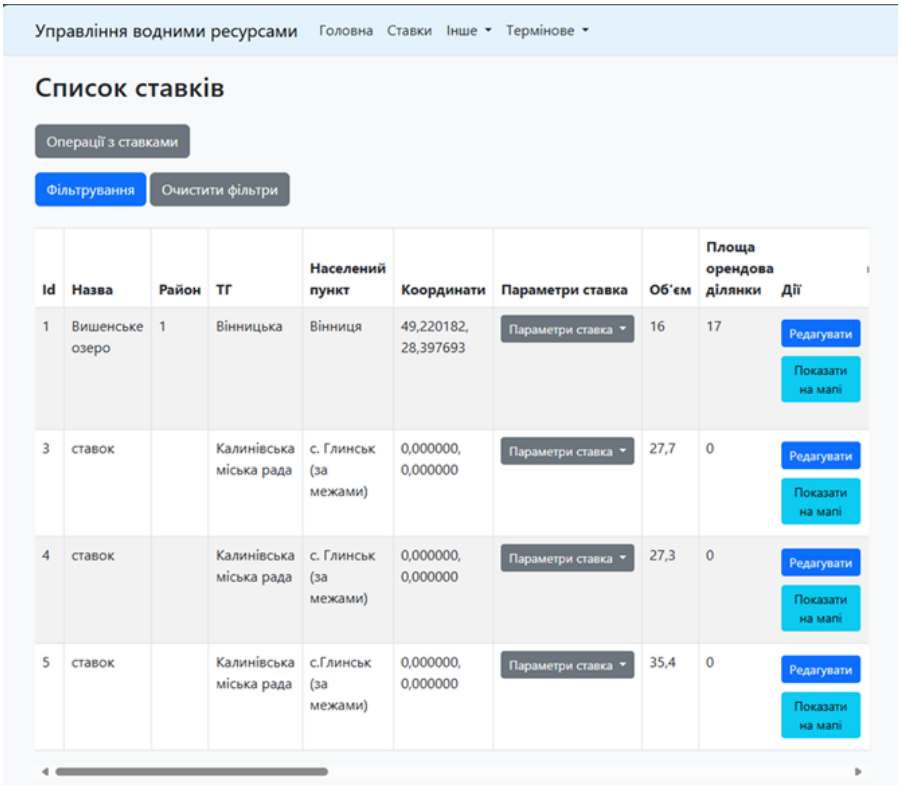
	A	B	C	D	E	F	G	H	I	J
1				Калинівська міська рада Хмільницького району						
2										
3	№П/Н	Наявність паспорта водного об'єкту	Населений пункт	Назва об'єкта поверхневих вод (озеро, ставок, водосховище, водойма тощо)	Назва водотоку, на якому розташовано водний об'єкт, басейн річки	Площа водного дзеркала при НПР, га	Об'єм при НПР, тис.куб. м	Балансоутримувач/орендар/власник гідротехнічної споруди	Кадастровий номер земельної ділянки під водним об'єктом	Наявність договору оренди гідроспоруди
4	1	відсутній	с. Глинськ (за межами)	ставок	р. Постолова	2,5400	27,7		52168	
5	2	відсутній	с. Глинськ (за межами)	ставок	р. Постолова	2,4600	27,3		52169	
6	3	відсутній	с.Глинськ (за межами)	ставок	р. Постолова	3,1700	35,4		52167	
7	4									
8	5									
9	6									
10	7									

Калинівська міська рада

Готово Фільтр: добір

100%

Рисунок 4.25 – Приклад файлу для імпорту



Управління водними ресурсами Головна Ставки Інше Термінове

Список ставків

Операції з ставками

Фільтрування Очистити фільтри

Id	Назва	Район	ТГ	Населений пункт	Координати	Параметри ставка	Об'єм	Площа орендова ділянки	Дії
1	Вишенське озеро	1	Вінницька	Вінниця	49,220182, 28,397693	Параметри ставка	16	17	Редагувати Показати на мапі
3	ставок		Калинівська міська рада	с. Глинськ (за межами)	0,000000, 0,000000	Параметри ставка	27,7	0	Редагувати Показати на мапі
4	ставок		Калинівська міська рада	с. Глинськ (за межами)	0,000000, 0,000000	Параметри ставка	27,3	0	Редагувати Показати на мапі
5	ставок		Калинівська міська рада	с.Глинськ (за межами)	0,000000, 0,000000	Параметри ставка	35,4	0	Редагувати Показати на мапі

Рисунок 4.26 – Результат імпорту

Окрім тестування функціональності, також було проведено тестування швидкості відгуку системи для дій користувача. Завантаження сторінки з ставками займає від 2 до 10 секунд, в залежності від кількості записів, що є в рамках норми для вебсторінок з великими обсягами даних. Інші сторінки завнтажуються з часом до двох секунд.

4.2 Розробка інструкції користувача

Інструкція користувача містить технічні вимоги для запуску програмного продукту. Інформацію про мінімальну та рекомендовану конфігурацію наведено в таблиці 4.1.

Таблиця 4.1 – Вимоги до апаратного забезпечення

Вимоги до конфігурації	Рекомендовано	Мінімально
Процесор	Intel Core i7	Intel Core i3
Оперативна пам'ять	16 ГБ RAM	4 ГБ RAM
Вільний простір на диску	10 ГБ	5 ГБ
Відеокарта	NVIDIA GeForce GTX 1050 або аналогічна	NVIDIA GeForce GTX 780 або аналогічна
Операційна система	Windows 11	Windows 10
Монітор	Роздільна здатність не менше 1920x1080 пікселів, діагональ 21" або більше	Роздільна здатність не менше 1280x720 пікселів, діагональ 15" або більше

Після інсталяції та запуску програмного забезпечення, користувача зустрічає головна сторінка з посиланнями на усі інші сторінки (Ставки, орендарі, договори оренди, дозволи на водокористування, термінові договори оренди, термінові дозволи на водокористування). Відкривши сторінку з ставками, у користувача є можливість додати новий ставок, заповнивши форму з його даними, імпортувати інформацію про ставки з Excel-файлу, відфільтрувати ставки за певними критеріями, експортувати інформацію про відфільтровані ставки у Excel чи PDF. Також на сторінці з таблицею ставків для кожного ставка можна показати його на мапі чи редагувати інформацію про нього. На мапі, що на сторінці з таблицею ставків є можливість натиснути на позначку ставка, і фокус вікна зміститься на цей ставок у таблиці.

На сторінках зі списком орендарів, договорів оренди, дозволів на водокористування є можливість додати чи переглянути орендаря/договір/дозвіл або його відредагувати чи видалити.

На сторінках з терміновими договорами оренди, чи спецдозволами на водокористування можна переглянути таблицю з відповідними даними про кількість залишених днів для кожного договору/дозволу, а також можна фільтрувати їх за кількістю днів чи місяців, що залишились.

4.3 Висновки

У четвертому розділі здійснено тестування модулів програмного забезпечення для обліку ставків. Перевірено поведінку системи в разі помилкових ситуацій та при обробці різних вхідних даних. Крім того, розроблено інструкцію користувача для початкової експлуатації програмного продукту.

ВИСНОВКИ

У рамках бакалаврської кваліфікаційної роботи були розроблені модулі вебсистеми обліку ставків Вінницької області. При розробці було використано інтегроване середовище розробки Visual Studio.

Бакалаврська кваліфікаційна робота оформлена згідно методичних вказівок[16].

Під час виконання бакалаврської кваліфікаційної роботи здійснено аналіз актуального стану проблеми: досліджено головні аналоги програмного забезпечення, виявлено їхні недоліки та співставлено з розробленим власним рішенням. За результатами цього порівняння сформульовано ключові завдання бакалаврської кваліфікаційної роботи.

Під час аналізу технологій для розробки було проведено обґрунтування мови програмування C# разом з бібліотеками Leaflet, iText і mimeKit, та технологіями Razor Pages, Entity Framework, MSSQL, Redis, Gmail API і OAuth 2.0.

Відповідно до поставлених задач було:

- розроблено модель автоматизованої вебсистеми обліку ставків, яка міститиме основний функціонал системи;
- розроблено механізм імпорту даних з Excel;
- створено механізми експорту даних у форматах PDF/Excel;
- реалізовано інтерактивної карти з використанням бібліотеки Leaflet.js;
- реалізовано функціонал нагадування через електронну пошту;
- розроблено модуль статистики
- інтегровано модулі у єдину систему з інтуїтивним інтерфейсом;
- здійснено тестування вебсистеми відповідно до поставлених задач.

Було розроблено схеми загального алгоритму роботи вебсистеми та модуля імпорту ставків. Також розроблено модель вебсистеми з використанням UML діаграм.

Під час тестування було підтверджено повну працездатність програмного продукту та його відповідність вимогам технічного завдання. Додатково було розроблено інструкцію користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Результати інвентаризації водних об'єктів м. Львова [Електронний ресурс] – Режим доступу до ресурсу: https://geography.lnu.edu.ua/wp-content/uploads/2024/06/shushniak_savka_vergeles_2014.pdf (дата звернення: 21.05.2025)
2. Державна служба статистики України. Дані про водні ресурси [Електронний ресурс] – Режим доступу до ресурсу: https://sdg.ukrstat.gov.ua/uk/6/ (дата звернення: 21.05.2025)
3. Зацерковний В. І. Геоінформаційні системи і бази даних /Уклад. В.І.Зацерковний, В. Г. Бурачек, О. О. Железняк, А. О. Терещенко – Ніжин: НДУ ім. М. Гоголя, 2014 – 494 с.
4. Войтко В.В. Розробка автоматизованої інформаційної вебсистеми для обліку ставків /Уклад. В.В. Войтко, С.М. Бурбело, О.В. Миронюк // Матеріали LIV Всеукраїнської науково-технічної конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2025) : збірник доповідей [Електронний ресурс]. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/24102/19886
5. Миронюк О.В. Визначення високорівневих вимог для автоматизованої інформаційної вебсистеми для обліку ставків Вінницької області /Уклад. О.В. Миронюк, Л.Б. Ліщинська Миронюк // Матеріали LIV Всеукраїнської науково-технічної конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2025) : збірник доповідей [Електронний ресурс]. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/23779/19737
6. ArcGIS [Електронний ресурс] – Режим доступу до ресурсу: https://www.esri.com/en-us/arcgis/products/arcgis-pro/overview/ (дата звернення: 24.02.2025).

7. Моніторинг та екологічна оцінка водних ресурсів України [Електронний ресурс] – Режим доступу до ресурсу: http://monitoring.davr.gov.ua/EcoWaterMon/GDKMap/Index (дата звернення: 21.05.2025).
8. HydroMonitor [Електронний ресурс] – Режим доступу до ресурсу: https://hydro-monitor.com/ (дата звернення: 21.05.2025)
9. QGIS [Електронний ресурс] – Режим доступу до ресурсу: https://qgis.org/ (дата звернення: 21.05.2025)
10. AQUARIUS [Електронний ресурс] – Режим доступу до ресурсу: https://aquaticinformatics.com/products/aquarius-environmental-water-data-management/ (дата звернення: 21.05.2025)
11. C# [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/dotnet/csharp/ (дата звернення: 21.05.2025)
12. ASP.NET [Електронний ресурс] – Режим доступу до ресурсу: https://dotnet.microsoft.com/en-us/apps/aspnet (дата звернення: 21.05.2025)
13. Entity Framework [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/ef/ (дата звернення: 21.05.2025)
14. Brind M. ASP.NET Core Razor Pages in Action. – New York: Manning. 2022. 456p.
15. Draw.IO [Електронний ресурс] – Режим доступу до ресурсу: https://app.diagrams.net/ (дата звернення: 08.04.2025)
16. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.
17. Lock A. ASP.NET Core in Action. – New York: Manning. 2023. 984p.
18. Smith J. Entity Framework Core in Action. – New York: Manning. 2021. 587p.

19. Leaflet [Електронний ресурс] – Режим доступу до ресурсу: https://leafletjs.com/ (дата звернення: 21.04.2025)
20. ClosedXML [Електронний ресурс] – Режим доступу до ресурсу: https://docs.closedxml.io/en/latest/ (дата звернення: 21.04.2025)
21. iText [Електронний ресурс] – Режим доступу до ресурсу: https://itextpdf.com/ (дата звернення: 21.04.2025)
22. Redis [Електронний ресурс] – Режим доступу до ресурсу: https://redis.io/ (дата звернення: 21.04.2025)
23. OAuth 2.0 [Електронний ресурс] – Режим доступу до ресурсу: https://oauth.net/2/ (дата звернення: 21.04.2025)
24. Gmail API [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/workspace/gmail/api/guides (дата звернення: 21.04.2025)
25. MimeKit [Електронний ресурс] – Режим доступу до ресурсу: https://mimekit.net/ (дата звернення: 21.04.2025)

ДОДАТКИ

74

Додаток А – Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

УЗГОДЖЕНО
НАЧАЛЬНИК ДЕРЖАВНОЇ
ЕКОЛОГІЧНОЇ ІНСПЕКЦІЇ У
ВІННИЦЬКІЙ ОБЛАСТІ
Дубовий Ю.В.
« 21 » березня 2025 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
« 21 » березня 2025 р.

Технічне завдання

на бакалаврську кваліфікаційну роботу

«Розробка методу та засобів реалізації вебсистеми обліку ставків Вінницької області»

за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доц. Войтко В. В.
« 21 » березня 2025 р.

Виконав:
студент гр. 1ПІ-21б Миронюк О.В.
« 21 » березня 2025 р.

Вінниця – 2025 року

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка методу та засобів реалізації вебсистеми обліку ставків Вінницької області».

Галузь застосування – автоматизовані вебсистеми.

2. Підстава для розробки

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від 20 березня 2025 року про закріплення тем БКР.

3. Мета та призначення розробки.

Метою дослідження є покращення управління водними ресурсами шляхом розробки програмного забезпечення для обліку ставків, яке дозволить ефективно збирати, аналізувати та візуалізувати дані про водні об'єкти.

Призначення роботи – розробка програмного продукту для автоматизації обліку ставків.

4. Вихідні дані для проведення НКР

1. Державна служба статистики України. Дані про водні ресурси [Електронний ресурс] – Режим доступу до ресурсу: https://sdg.ukrstat.gov.ua/uk/6/.

2. Моніторинг та екологічна оцінка водних ресурсів України [Електронний ресурс] – Режим доступу до ресурсу: http://monitoring.davr.gov.ua/EcoWaterMon/GDKMap/Index .

3. Redis [Електронний ресурс] – Режим доступу до ресурсу: https://redis.io/

4. ASP.NET [Електронний ресурс] – Режим доступу до ресурсу: https://dotnet.microsoft.com/en-us/apps/aspnet.

5. Технічні вимоги

Серверний комп’ютер з 64-розрядним (x64) процесором та тактовою

частотою 2 ГГц. Об’єм оперативної пам’яті 4 ГБ, розмір жорсткого диску 128 ГБ. Операційна система Windows 10.

З клієнтського боку: ОС Windows та будь-який браузер.

6. Конструктивні вимоги.

Вебзастосунок має відповідати ергономічним та технічним вимогам. Текстова та графічна документація повинна відповідати стандартам України.

7. Перелік технічної документації, що пред’являється по закінченню робіт:

- пояснювальна записка до бакалаврської кваліфікаційної роботи;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз розвитку автоматизованих вебсистем обліку ставків та постановка задач дослідження	25.03.25 – 03.04.2025
2	Розробка методів, моделі та алгоритмів роботи вебсистеми	03.04.25 – 20.04.2025
3	Розробка програмних модулів вебсистеми обліку ставків Вінницької області	20.04.25 – 03.05.2025
4	Тестування програми	03.05.25 – 23.05.2025
5	Оформлення матеріалів до захисту БКР	23.05.2025 – 30.05.25

10. Порядок контролю та прийняття

Всі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

78

Додаток Б – Протокол перевірки бакалаврської кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: «Розробка методу та засобів реалізації вебсистеми обліку ставків Вінницької області»

Тип роботи: БКР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Войтко В. В.

Оригінальність	98 %
Схожість	2 %

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою StrikePlagiarism

Автор роботи _____ Миронюк О. В.

Керівник роботи _____ Войтко В. В.

79

Додаток В – Акт впровадження (Державна екологічна інспекція у Вінницькій області)

АКТ

про впровадження результатів бакалаврської кваліфікаційної роботи «Розробка методу та засобів реалізації вебсистеми обліку ставків Вінницької області»

студента ВНТУ Миронюка Олександра Володимировича

Результати бакалаврської кваліфікаційної роботи студента ВНТУ Миронюка О.В., що полягають в розробці методу та засобів реалізації вебсистеми обліку ставків Вінницької області, прийняті до використання у Державній екологічній інспекції у Вінницькій області.

Начальник Державної екологічної інспекції у Вінницькій області — Дубовий Ю.В.

Додаток Г – Лістинг програми

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Список ставків";
}
<link rel="stylesheet" href="css/leaflet.css" />

<style>
    /* Контейнер для таблиці з горизонтальним скролом */
    .table-container {
        overflow-x: auto;
        position: relative;
    }
    /* Фіксована перша колонка */
    .table-fixed-col-first {
        position: sticky;
        left: 0;
        background: #fff;
        z-index: 2;
    }
    /* Фіксована остання колонка */
    .table-fixed-col-last {
        position: sticky;
        right: 0;
        background: #fff;
        z-index: 2;
    }
    /* Для заголовків таблиці */
    th.table-fixed-col-first {
        position: sticky;
        left: 0;
        background: #fff;
        z-index: 3;
    }

    th.table-fixed-col-last {
        position: sticky;
        right: 0;
        background: #fff;
        z-index: 3;
    }

    .table-fixed-col-last a {
        margin: 5px 0;
    }
</style>

<h2>Список ставків</h2>

<!-- Кнопка для розгортання/згортання операцій з ставками -->
<button class="btn btn-secondary mt-3" type="button" data-bs-toggle="collapse" data-bs-
target="#operationsPanel" aria-expanded="false" aria-controls="operationsPanel">
    Операції з ставками
</button>

<!-- Випадаюче меню з операціями (експорт, імпорт, додавання) -->
<div class="collapse mt-3 mb-3" id="operationsPanel">
    <div class="card card-body">
        <form method="post">
            <button type="submit" asp-page-handler="ExportExcel" class="btn btn-success">Експортувати в
Excel</button>
            <button type="submit" asp-page-handler="ExportPDF" class="btn btn-danger">Експортувати в
PDF</button>
        </form>
        <div class="mt-3">
            <a class="btn btn-primary" asp-page="/Ponds/Create">Додати ставок</a>
            <a class="btn btn-success" asp-page="/Ponds/Import">Імпорт ставків</a>
        </div>
    </div>
</div>


<!-- Фільтрація (розміщена у випадаючому меню) -->
<div class="collapse" id="filterCollapse">
    <div class="card card-body mb-3">
```

```
        <form method="get">
            <!-- Ряд 1 -->
            <div class="row">
                <div class="col-md-3">
                    <label>Id</label>
                    <input type="number" name="idFilter" value="@Model.IdFilter" class="form-control"
/>
                </div>
                <div class="col-md-3">
                    <label>Назва</label>
                    <input type="text" name="nameFilter" value="@Model.NameFilter" class="form-control"
/>
                </div>
                <div class="col-md-3">
                    <label>Район</label>
                    <input type="text" name="districtFilter" value="@Model.DistrictFilter" class="form-
control" />
                </div>
                <div class="col-md-3">
                    <label>ТГ</label>
                    <input type="text" name="territorialCommunityFilter"
value="@Model.TerritorialCommunityFilter" class="form-control" />
                </div>
            </div>
            <!-- Ряд 2 -->
            <div class="row mt-2">
                <div class="col-md-3">
                    <label>Населений пункт</label>
                    <input type="text" name="settlementFilter" value="@Model.SettlementFilter"
class="form-control" />
                </div>
                <div class="col-md-3">
                    <label>Мін. X (longitude)</label>
                    <input asp-for="MinXFilter" class="form-control" step="0.000001" />
                </div>
                <div class="col-md-3">
                    <label>Макс. X (longitude)</label>
                    <input asp-for="MaxXFilter" class="form-control" step="0.000001" />
                </div>
                <div class="col-md-3">
                    <label>Мін. Y (latitude)</label>
                    <input asp-for="MinYFilter" class="form-control" step="0.000001" />
                </div>
                <div class="col-md-3 mt-2">
                    <label>Макс. Y (latitude)</label>
                    <input asp-for="MaxYFilter" class="form-control" step="0.000001" />
                </div>
            </div>
            <!-- Ряд 3 -->
            <div class="row mt-2">
                <div class="col-md-3">
                    <label>Глибина</label>
                    <input type="number" step="any" name="depthFilter" value="@Model.DepthFilter"
class="form-control" />
                </div>
                <div class="col-md-3">
                    <label>Підпірний рівень</label>
                    <input type="number" step="any" name="waterLevelFilter"
value="@Model.WaterLevelFilter" class="form-control" />
                </div>
                <div class="col-md-3">
                    <label>Об'єм</label>
                    <input type="number" step="any" name="volumeFilter" value="@Model.VolumeFilter"
class="form-control" />
                </div>
                <div class="col-md-3">
                    <label>Площа орендованої ділянки</label>
                    <input type="number" step="any" name="leasedAreaFilter"
value="@Model.LeasedAreaFilter" class="form-control" />
                </div>
            </div>
            <!-- Ряд 4 -->
            <div class="row mt-2">
                <div class="col-md-3">
                    <label>Кадастровий номер</label>
                    <input type="text" name="cadastralNumberFilter"
value="@Model.CadastralNumberFilter" class="form-control" />
```

```
                </div>
                <div class="col-md-3">
                    <label>Площа водного плеса</label>
                    <input type="number" step="any" name="waterSurfaceAreaFilter"
value="@Model.WaterSurfaceAreaFilter" class="form-control" />
                </div>
                <div class="col-md-3">
                    <label>Спускний</label>
                    <select name="isDrainableFilter" class="form-control">
                        <option value="">Будь-який</option>
                        @if (Model.IsDrainableFilter == true)
                        {
                            <option value="true" selected>Так</option>
                        }
                        else
                        {
                            <option value="true">Так</option>
                        }
                        @if (Model.IsDrainableFilter == false)
                        {
                            <option value="false" selected>Ні</option>
                        }
                        else
                        {
                            <option value="false">Ні</option>
                        }
                    </select>
                </div>
                <div class="col-md-3">
                    <label>Гідроспоруда</label>
                    <select name="hasHydraulicStructureFilter" class="form-control">
                        <option value="">Будь-який</option>
                        @if (Model.HasHydraulicStructureFilter == true)
                        {
                            <option value="true" selected>Так</option>
                        }
                        else
                        {
                            <option value="true">Так</option>
                        }
                        @if (Model.HasHydraulicStructureFilter == false)
                        {
                            <option value="false" selected>Ні</option>
                        }
                        else
                        {
                            <option value="false">Ні</option>
                        }
                    </select>
                </div>
            </div>
            <!-- Ряд 5: Фільтрація за пов'язаними сутностями -->
            <div class="row mt-2">
                <div class="col-md-4">
                    <label>Ідентифікаційний код орендаря</label>
                    <input type="text" name="lesseeIdentificationCodeFilter"
value="@Model.LesseeIdentificationCodeFilter" class="form-control" />
                </div>
                <div class="col-md-4">
                    <label>Номер договору оренди</label>
                    <input type="text" name="leaseAgreementNumberFilter"
value="@Model.LeaseAgreementNumberFilter" class="form-control" />
                </div>
                <div class="col-md-4">
                    <label>Номер дозволу на водокористування</label>
                    <input type="text" name="waterUsagePermitNumberFilter"
value="@Model.WaterUsagePermitNumberFilter" class="form-control" />
                </div>
            </div>
            <!-- Ряд 6 -->
            <div class="row mt-2">
                <div class="col-md-3">
                    <label>Річка</label>
                    <input type="text" name="riverFilter" value="@Model.RiverFilter" class="form-
control" />
                </div>
                <div class="col-md-3">
```

```
                    <label>Басейн</label>
                    <input type="text" name="basinFilter" value="@Model.BasinFilter" class="form-
control" />
                </div>
                <div class="col-md-3">
                    <label>Стан</label>
                    <input type="text" name="statusFilter" value="@Model.StatusFilter" class="form-
control" />
                </div>
                <div class="col-md-3">
                    <label>Порушення</label>
                    <input type="text" name="issuesFilter" value="@Model.IssuesFilter" class="form-
control" />
                </div>
            </div>
            <!-- Ряд 7 -->
            <div class="row mt-2">
                <div class="col-md-3">
                    <label>Примітки</label>
                    <input type="text" name="notesFilter" value="@Model.NotesFilter" class="form-
control" />
                </div>
                <div class="col-md-3">
                    <label>Накладені штрафи</label>
                    <input type="number" step="any" name="imposedFinesFilter"
value="@Model.ImposedFinesFilter" class="form-control" />
                </div>
                <div class="col-md-3">
                    <label>Накладені збитки</label>
                    <input type="number" step="any" name="imposedDamagesFilter"
value="@Model.ImposedDamagesFilter" class="form-control" />
                </div>
                <div class="col-md-3">
                    <label>Стягнуті штрафи</label>
                    <input type="number" step="any" name="collectedFinesFilter"
value="@Model.CollectedFinesFilter" class="form-control" />
                </div>
            </div>
            <!-- Ряд 8 -->
            <div class="row mt-2">
                <div class="col-md-3">
                    <label>Стягнуті збитки</label>
                    <input type="number" step="any" name="collectedDamagesFilter"
value="@Model.CollectedDamagesFilter" class="form-control" />
                </div>
            </div>
            <div class="mt-3">
                <button type="submit" class="btn btn-primary">Фільтрувати</button>
            </div>
        </form>
    </div>
</div>

<!-- Кнопка для відкриття/закриття фільтрації -->
<p>
    <button class="btn btn-primary mt-3" type="button" data-bs-toggle="collapse" data-bs-
target="#filterCollapse" aria-expanded="false" aria-controls="filterCollapse">
        Фільтрування
    </button>
    <a asp-page="./Index" class="btn btn-secondary mt-3">Очистити фільтри</a>
</p>



<!-- Таблиця ставків у своєму блоці -->
<div class="table-container mt-4">
    <table class="table table-striped table-bordered">
        <thead>
            <tr>
                <th class="table-fixed-col-first">Id</th>
                <th>Назва</th>
                <th>Район</th>
                <th>ТГ</th>
                <th>Населений пункт</th>
                <th>Координати</th>
                <th>Параметри ставка</th>
                <th>Об'єм</th>
```

```
                <th>Площа орендованої ділянки</th>
                <th>Кадастровий номер</th>
                <th>Площа водного плеса</th>
                <th>Спускний</th>
                <th>Гідроспоруда</th>
                <th>Балансоутримувач гідроспоруди</th>
                <th>Орендар</th>
                <th>Договір оренди</th>
                <th>Дозвіл на водокористування</th>
                <th>Річка</th>
                <th>Басейн</th>
                <th>Стан</th>
                <th>Порушення</th>
                <th>Примітки</th>
                <th>Накладені штрафи</th>
                <th>Накладені збитки</th>
                <th>Стягнуті штрафи</th>
                <th>Стягнуті збитки</th>
                <th class="table-fixed-col-last">Дії</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var pond in Model.Ponds)
            {
                <tr id="pond-@pond.Id">
                    <td class="table-fixed-col-first">@pond.Id</td>
                    <td>@pond.Name</td>
                    <td>@pond.District</td>
                    <td>@pond.TerritorialCommunity</td>
                    <td>@pond.Settlement</td>
                    <td>@($"{pond.XCoordinate:N6}, {pond.YCoordinate:N6}")</td>
                    <td>
                        <div class="dropdown">
                            <button class="btn btn-secondary btn-sm dropdown-toggle" type="button"
id="dropdownMenuButton_@pond.Id" data-bs-toggle="dropdown" aria-expanded="false">
                                Параметри ставка
                            </button>
                            <ul class="dropdown-menu" aria-labelledby="dropdownMenuButton_@pond.Id">
                                <li><span class="dropdown-item-text">Довжина: @pond.Length</span></li>
                                <li><span class="dropdown-item-text">Ширина: @pond.Width</span></li>
                                <li><span class="dropdown-item-text">Глибина: @pond.Depth</span></li>
                                <li><span class="dropdown-item-text">Підпірний рівень:
@pond.WaterLevel</span></li>
                            </ul>
                        </div>
                    </td>
                    <td>@pond.Volume</td>
                    <td>@pond.LeasedArea</td>
                    <td>@pond.CadastralNumber</td>
                    <td>@pond.WaterSurfaceArea</td>
                    <td>@if(pond.IsDrainable)
                        {
                            <span>Так</span>
                        }
                        else
                        {
                            <span>Ні</span>
                        }
                    </td>
                    <td>@if (pond.HasHydraulicStructure)
                        {
                            <span>Так</span>
                        }
                        else
                        {
                            <span>Ні</span>
                        }

                    </td>
                    <td>@pond.HydraulicStructureOwner</td>
                    <td>
                        @if (pond.Lessee != null)
                        {
                            <div><strong>@pond.Lessee.Name</strong></div>
                            <div>Тип: @pond.Lessee.Type</div>
                            <div>Код: @pond.Lessee.IdentificationCode</div>
                        }
```

```
                    </td>
                    <td>
                        @if (pond.LeaseAgreement != null)
                        {
                            <div><strong>@pond.LeaseAgreement.Number</strong></div>
                            <div>Дата: @pond.LeaseAgreement.Date.ToShortDateString()</div>
                            <div>Термін: @pond.LeaseAgreement.TermInYears років</div>
                        }
                    </td>
                    <td>
                        @if (pond.WaterUsagePermit != null)
                        {
                            <div><strong>@pond.WaterUsagePermit.Number</strong></div>
                            <div>Початок: @pond.WaterUsagePermit.StartDate.ToShortDateString()</div>
                            <div>Термін: @pond.WaterUsagePermit.TermInYears років</div>
                        }
                    </td>
                    <td>@pond.River</td>
                    <td>@pond.Basin</td>
                    <td>@pond.Status</td>
                    <td>@pond.Issues</td>
                    <td>@pond.Notes</td>
                    <td>@pond.ImposedFines</td>
                    <td>@pond.ImposedDamages</td>
                    <td>@pond.CollectedFines</td>
                    <td>@pond.CollectedDamages</td>
                    <td class="table-fixed-col-last">
                        <a asp-page="./Edit" asp-route-id="@pond.Id" class="btn btn-sm btn-
primary">Редагувати</a>
                        <a href="#" class="btn btn-sm btn-info" onclick="showOnMap(@pond.Id,
@(pond.XCoordinate.ToString(System.Globalization.CultureInfo.InvariantCulture)),
@(pond.YCoordinate.ToString(System.Globalization.CultureInfo.InvariantCulture))); return
false;">Показати на мапі</a>
                    </td>
                </tr>
            }
        </tbody>
    </table>
</div>

<!-- Карта, розміщена під таблицею -->
<div class="card card-body mt-3">
    <div id="map" style="height: 500px; width: 100%;"></div>
</div>

@section Scripts {
    <script src="js/leaflet.js"></script>
    <script>
            // Ініціалізація мапи (центр України)
            var map = L.map('map').setView([48.3794, 31.1656], 6);
            L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
                attribution: '&copy; OpenStreetMap contributors'
            }).addTo(map);

            var ponds = @Html.Raw(Json.Serialize(Model.Ponds));

            ponds.forEach(pond => {
                if (pond.xCoordinate && pond.yCoordinate) {
                    var marker = L.marker([pond.xCoordinate, pond.yCoordinate]).addTo(map)
                        // .bindPopup(`<b>${pond.name}</b><br>${pond.district}, ${pond.settlement}`);
                    marker.on('click', function(e) {
                        var row = document.getElementById('pond-' + pond.id);
                        if (row) {
                            row.scrollIntoView({ behavior: 'smooth', block: 'center' });
                        }
                    });
                }
            });

           function showOnMap(pondId, xCoordinate, yCoordinate) {
                map.setView([xCoordinate, yCoordinate],12);

                var mapContainer = document.getElementById('map');
                if (mapContainer) {
                    mapContainer.scrollIntoView({ behavior: 'smooth', block: 'start' });
                }
            }
```

```
    </script>
}

using ClosedXML.Excel;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.EntityFrameworkCore;
using PondManager.Data.Data;
using PondManager.Data.Models;
using iText.Kernel.Pdf;
using iText.Layout;
using iText.Layout.Element;
using iText.Layout.Properties;
using Paragraph = iText.Layout.Element.Paragraph;
using iText.IO.Font.Constants;
using iText.Kernel.Font;
using iText.IO.Font;
using iText.Kernel.Geom;
using Microsoft.Extensions.Caching.Distributed;
using Newtonsoft.Json;
using DocumentFormat.OpenXml.Drawing.Diagrams;
using System.Text.Json.Serialization;
using System.Text.Json;
using JsonSerializer = System.Text.Json.JsonSerializer;

namespace PondManager.Pages.Ponds;

public class IndexModel : PageModel
{
    private readonly ApplicationDbContext _context;
    private readonly IDistributedCache _cache;

    // Prefixes so Excel and PDF caches don't collide
    private const string ExcelCachePrefix = "Ponds:Export:Excel:";
    private const string PdfCachePrefix = "Ponds:Export:Pdf:";

    public IndexModel(ApplicationDbContext context, IDistributedCache cache)
    {
        _context = context;
        _cache = cache;
    }



    // Фільтри для усіх властивостей сутності Pond
    [BindProperty(SupportsGet = true)]
    public int? IdFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public string? NameFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public string? DistrictFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public string? TerritorialCommunityFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public string? SettlementFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public double? MinXFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public double? MaxXFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public double? MinYFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public double? MaxYFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public double? LengthFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public double? WidthFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public double? DepthFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public double? WaterLevelFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public double? VolumeFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public double? LeasedAreaFilter { get; set; }
    [BindProperty(SupportsGet = true)]
```

```
    public string? CadastralNumberFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public double? WaterSurfaceAreaFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public bool? IsDrainableFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public bool? HasHydraulicStructureFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public string? HydraulicStructureOwnerFilter { get; set; }
    // Замість числових Id для пов'язаних сутностей використаємо рядкові фільтри
    [BindProperty(SupportsGet = true)]
    public string? LesseeIdentificationCodeFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public string? LeaseAgreementNumberFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public string? WaterUsagePermitNumberFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public string? RiverFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public string? BasinFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public string? StatusFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public string? IssuesFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public string? NotesFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public decimal? ImposedFinesFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public decimal? ImposedDamagesFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public decimal? CollectedFinesFilter { get; set; }
    [BindProperty(SupportsGet = true)]
    public decimal? CollectedDamagesFilter { get; set; }

    public IList<Pond> Ponds { get; set; } = new List<Pond>();

    public async Task OnGetAsync()
    {
        var cacheKey = BuildCacheKey(); // Unique cache key from filters

        var cachedData = await _cache.GetStringAsync(cacheKey);

        if (cachedData != null)
        {
            Ponds = JsonConvert.DeserializeObject<List<Pond>>(cachedData)!;
            return;
        }

        // Query with includes
        IQueryable<Pond> query = _context.Ponds
            .Include(p => p.Lessee)
            .Include(p => p.LeaseAgreement)
            .Include(p => p.WaterUsagePermit);

        // Apply filters
        if (IdFilter.HasValue) query = query.Where(p => p.Id == IdFilter.Value);
        if (!string.IsNullOrEmpty(NameFilter)) query = query.Where(p => p.Name.Contains(NameFilter));
        if       (!string.IsNullOrEmpty(DistrictFilter))       query       =       query.Where(p       =>
p.District.Contains(DistrictFilter));
        if     (!string.IsNullOrEmpty(TerritorialCommunityFilter))     query     =     query.Where(p     =>
p.TerritorialCommunity.Contains(TerritorialCommunityFilter));
        if       (!string.IsNullOrEmpty(SettlementFilter))       query       =       query.Where(p       =>
p.Settlement.Contains(SettlementFilter));
        if (MinXFilter.HasValue) query = query.Where(p => p.XCoordinate >= MinXFilter.Value);
        if (MaxXFilter.HasValue) query = query.Where(p => p.XCoordinate <= MaxXFilter.Value);
        if (MinYFilter.HasValue) query = query.Where(p => p.YCoordinate >= MinYFilter.Value);
        if (MaxYFilter.HasValue) query = query.Where(p => p.YCoordinate <= MaxYFilter.Value);
        if (LengthFilter.HasValue) query = query.Where(p => p.Length == LengthFilter.Value);
        if (WidthFilter.HasValue) query = query.Where(p => p.Width == WidthFilter.Value);
        if (DepthFilter.HasValue) query = query.Where(p => p.Depth == DepthFilter.Value);
        if (WaterLevelFilter.HasValue) query = query.Where(p => p.WaterLevel == WaterLevelFilter.Value);
        if (VolumeFilter.HasValue) query = query.Where(p => p.Volume == VolumeFilter.Value);
        if (LeasedAreaFilter.HasValue) query = query.Where(p => p.LeasedArea == LeasedAreaFilter.Value);
        if      (!string.IsNullOrEmpty(CadastralNumberFilter))      query      =      query.Where(p      =>
p.CadastralNumber.Contains(CadastralNumberFilter));
```

```
        if (WaterSurfaceAreaFilter.HasValue) query = query.Where(p => p.WaterSurfaceArea ==
WaterSurfaceAreaFilter.Value);
        if (IsDrainableFilter.HasValue) query = query.Where(p => p.IsDrainable ==
IsDrainableFilter.Value);
        if (HasHydraulicStructureFilter.HasValue) query = query.Where(p => p.HasHydraulicStructure ==
HasHydraulicStructureFilter.Value);
        if (!string.IsNullOrEmpty(HydraulicStructureOwnerFilter)) query = query.Where(p =>
p.HydraulicStructureOwner.Contains(HydraulicStructureOwnerFilter));
        if (!string.IsNullOrEmpty(LesseeIdentificationCodeFilter)) query = query.Where(p => p.Lessee !=
null && p.Lessee.IdentificationCode.Contains(LesseeIdentificationCodeFilter));
        if (!string.IsNullOrEmpty(LeaseAgreementNumberFilter)) query = query.Where(p => p.LeaseAgreement
!= null && p.LeaseAgreement.Number.Contains(LeaseAgreementNumberFilter));
        if (!string.IsNullOrEmpty(WaterUsagePermitNumberFilter)) query = query.Where(p =>
p.WaterUsagePermit != null && p.WaterUsagePermit.Number.Contains(WaterUsagePermitNumberFilter));
        if (!string.IsNullOrEmpty(RiverFilter)) query = query.Where(p => p.River.Contains(RiverFilter));
        if (!string.IsNullOrEmpty(BasinFilter)) query = query.Where(p => p.Basin.Contains(BasinFilter));
        if (!string.IsNullOrEmpty(StatusFilter)) query = query.Where(p =>
p.Status.Contains(StatusFilter));
        if (!string.IsNullOrEmpty(IssuesFilter)) query = query.Where(p =>
p.Issues.Contains(IssuesFilter));
        if (!string.IsNullOrEmpty(NotesFilter)) query = query.Where(p => p.Notes.Contains(NotesFilter));
        if (ImposedFinesFilter.HasValue) query = query.Where(p => p.ImposedFines ==
ImposedFinesFilter.Value);
        if (ImposedDamagesFilter.HasValue) query = query.Where(p => p.ImposedDamages ==
ImposedDamagesFilter.Value);
        if (CollectedFinesFilter.HasValue) query = query.Where(p => p.CollectedFines ==
CollectedFinesFilter.Value);
        if (CollectedDamagesFilter.HasValue) query = query.Where(p => p.CollectedDamages ==
CollectedDamagesFilter.Value);

        Ponds = await query.ToListAsync();

        var jsonOptions = new JsonSerializerOptions
        {
            ReferenceHandler = ReferenceHandler.IgnoreCycles,
            WriteIndented = false
        };

        var serialized = JsonSerializer.Serialize(Ponds, jsonOptions);
        await _cache.SetStringAsync(cacheKey, serialized, new DistributedCacheEntryOptions
        {
            AbsoluteExpirationRelativeToNow = TimeSpan.FromMinutes(10)
        });
    }

    private string BuildCacheKey()
    {
        var parts = new List<string>
        {
            IdFilter?.ToString(),
            NameFilter,
            DistrictFilter,
            TerritorialCommunityFilter,
            SettlementFilter,
            MinXFilter?.ToString(),
            MaxXFilter?.ToString(),
            MinYFilter?.ToString(),
            MaxYFilter?.ToString(),
            LengthFilter?.ToString(),
            WidthFilter?.ToString(),
            DepthFilter?.ToString(),
            WaterLevelFilter?.ToString(),
            VolumeFilter?.ToString(),
            LeasedAreaFilter?.ToString(),
            CadastralNumberFilter,
            WaterSurfaceAreaFilter?.ToString(),
            IsDrainableFilter?.ToString(),
            HasHydraulicStructureFilter?.ToString(),
            HydraulicStructureOwnerFilter,
            LesseeIdentificationCodeFilter,
            LeaseAgreementNumberFilter,
            WaterUsagePermitNumberFilter,
            RiverFilter,
            BasinFilter,
            StatusFilter,
            IssuesFilter,
            NotesFilter,
```

```
            ImposedFinesFilter?.ToString(),
            ImposedDamagesFilter?.ToString(),
            CollectedFinesFilter?.ToString(),
            CollectedDamagesFilter?.ToString()
        };

        var key = string.Join(":", parts.Select(p => p ?? "null"));
        return $"PondQuery:{key}";
    }


    public async Task<IActionResult> OnPostExportExcelAsync()
    {
        var ponds = await GetFilteredPonds().ToListAsync();

        using (var workbook = new XLWorkbook())
        {
            var worksheet = workbook.Worksheets.Add("Ставки");

            // Заголовки колонок
            var headers = new string[]
            {
            "ID", "Назва", "Район", "ТГ", "Населений пункт", "X", "Y", "Довжина", "Ширина",
            "Глибина", "Рівень води", "Об'єм", "Орендована площа", "Кадастровий номер",
            "Площа водного дзеркала", "Осушуваний", "Гідроспоруда", "Власник гідроспоруди",
            "Орендар", "Номер договору оренди", "Номер дозволу на спецводокористування",
            "Річка", "Басейн", "Статус", "Проблеми", "Примітки",
            "Накладені штрафи", "Накладені збитки", "Стягнуті штрафи", "Стягнуті збитки"
            };

            for (int i = 0; i < headers.Length; i++)
            {
                worksheet.Cell(1, i + 1).Value = headers[i];
            }

            int row = 2;
            foreach (var pond in ponds)
            {
                worksheet.Cell(row, 1).Value = pond.Id;
                worksheet.Cell(row, 2).Value = pond.Name;
                worksheet.Cell(row, 3).Value = pond.District;
                worksheet.Cell(row, 4).Value = pond.TerritorialCommunity;
                worksheet.Cell(row, 5).Value = pond.Settlement;
                worksheet.Cell(row, 6).Value = pond.XCoordinate;
                worksheet.Cell(row, 7).Value = pond.YCoordinate;
                worksheet.Cell(row, 8).Value = pond.Length;
                worksheet.Cell(row, 9).Value = pond.Width;
                worksheet.Cell(row, 10).Value = pond.Depth;
                worksheet.Cell(row, 11).Value = pond.WaterLevel;
                worksheet.Cell(row, 12).Value = pond.Volume;
                worksheet.Cell(row, 13).Value = pond.LeasedArea;
                worksheet.Cell(row, 14).Value = pond.CadastralNumber;
                worksheet.Cell(row, 15).Value = pond.WaterSurfaceArea;
                worksheet.Cell(row, 16).Value = pond.IsDrainable ? "Так" : "Ні";
                worksheet.Cell(row, 17).Value = pond.HasHydraulicStructure ? "Так" : "Ні";
                worksheet.Cell(row, 18).Value = pond.HydraulicStructureOwner;
                worksheet.Cell(row, 19).Value = pond.Lessee?.ToString();
                worksheet.Cell(row, 20).Value = pond.LeaseAgreement?.ToString();
                worksheet.Cell(row, 21).Value = pond.WaterUsagePermit?.ToString();
                worksheet.Cell(row, 22).Value = pond.River;
                worksheet.Cell(row, 23).Value = pond.Basin;
                worksheet.Cell(row, 24).Value = pond.Status;
                worksheet.Cell(row, 25).Value = pond.Issues;
                worksheet.Cell(row, 26).Value = pond.Notes;
                worksheet.Cell(row, 27).Value = pond.ImposedFines;
                worksheet.Cell(row, 28).Value = pond.ImposedDamages;
                worksheet.Cell(row, 29).Value = pond.CollectedFines;
                worksheet.Cell(row, 30).Value = pond.CollectedDamages;
                row++;
            }

            using (var stream = new MemoryStream())
            {
                workbook.SaveAs(stream);
                var content = stream.ToArray();
                return                  File(content,                  "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet", "Ponds.xlsx");
```

```
            }
        }
    }

    public async Task<IActionResult> OnPostExportPDFAsync()
    {
        // Отримуємо відфільтровані ставки
        var ponds = await GetFilteredPonds().ToListAsync();

        using (var memoryStream = new MemoryStream())
        {
            // Використовуємо сторінковий розмір A3 в ландшафтній орієнтації
            var pdfWriter = new PdfWriter(memoryStream);
            var pdfDoc = new PdfDocument(pdfWriter);
            var document = new Document(pdfDoc, PageSize.A3.Rotate());

            // Завантажуємо шрифт, що підтримує кирилицю
            document.SetFontSize(8);
            var fontPath = System.IO.Path.Combine(
                Directory.GetCurrentDirectory(), "wwwroot", "fonts", "Helvetica.ttf");
            PdfFont font;
            try
            {
                font = PdfFontFactory.CreateFont(fontPath, PdfEncodings.IDENTITY_H);
            }
            catch
            {
                font = PdfFontFactory.CreateFont(StandardFonts.HELVETICA);
            }
            document.SetFont(font);

            // Визначаємо кількість колонок (30) і створюємо таблицю
            Table table = new Table(30, true);
            table.SetWidth(UnitValue.CreatePercentValue(100));

            string[] headers = {
            "ID", "Назва", "Район", "ТГ", "Населений пункт", "X", "Y", "Довжина", "Ширина",
            "Глибина", "Рівень води", "Об'єм", "Орендована площа", "Кадастровий номер",
            "Площа водного дзеркала", "Чи осушуваний?", "Чи гідроспоруда?", "Власник гідроспоруди",
            "Орендар", "Номер договору оренди", "Номер дозволу на спецводокористування",
            "Річка", "Басейн", "Статус", "Проблеми", "Примітки",
            "Накладені штрафи", "Накладені збитки", "Стягнуті штрафи", "Стягнуті збитки"
        };

            foreach (var header in headers)
            {
                table.AddHeaderCell(new                                                    Cell().Add(new
Paragraph(header).SetFont(font).SimulateBold()));
            }

            // Додаємо дані з кожного запису
            foreach (var pond in ponds)
            {
                table.AddCell(new Cell().Add(new Paragraph(pond.Id.ToString()).SetFont(font)));
                table.AddCell(new Cell().Add(new Paragraph(pond.Name ?? "").SetFont(font)));
                table.AddCell(new Cell().Add(new Paragraph(pond.District ?? "").SetFont(font)));
                table.AddCell(new      Cell().Add(new      Paragraph(pond.TerritorialCommunity      ??
"").SetFont(font)));
                table.AddCell(new Cell().Add(new Paragraph(pond.Settlement ?? "").SetFont(font)));
                table.AddCell(new                                                          Cell().Add(new
Paragraph(pond.XCoordinate.ToString("F6")).SetFont(font)));
                table.AddCell(new                                                          Cell().Add(new
Paragraph(pond.YCoordinate.ToString("F6")).SetFont(font)));
                table.AddCell(new Cell().Add(new Paragraph(pond.Length.ToString()).SetFont(font)));
                table.AddCell(new Cell().Add(new Paragraph(pond.Width.ToString()).SetFont(font)));
                table.AddCell(new Cell().Add(new Paragraph(pond.Depth.ToString()).SetFont(font)));
                table.AddCell(new Cell().Add(new Paragraph(pond.WaterLevel.ToString()).SetFont(font)));
                table.AddCell(new Cell().Add(new Paragraph(pond.Volume.ToString()).SetFont(font)));
                table.AddCell(new Cell().Add(new Paragraph(pond.LeasedArea.ToString()).SetFont(font)));
                table.AddCell(new Cell().Add(new Paragraph(pond.CadastralNumber ?? "").SetFont(font)));
                table.AddCell(new                                                          Cell().Add(new
Paragraph(pond.WaterSurfaceArea.ToString()).SetFont(font)));
                table.AddCell(new     Cell().Add(new     Paragraph(pond.IsDrainable     ?     "Так"     :
"Ні").SetFont(font)));
                table.AddCell(new   Cell().Add(new   Paragraph(pond.HasHydraulicStructure   ?   "Так"   :
"Ні").SetFont(font)));
```

```
                table.AddCell(new      Cell().Add(new      Paragraph(pond.HydraulicStructureOwner      ??
"").SetFont(font)));
                table.AddCell(new       Cell().Add(new       Paragraph(pond.Lessee?.ToString()       ??
"").SetFont(font)));
                table.AddCell(new     Cell().Add(new     Paragraph(pond.LeaseAgreement?.ToString()     ??
"").SetFont(font)));
                table.AddCell(new    Cell().Add(new    Paragraph(pond.WaterUsagePermit?.ToString()    ??
"").SetFont(font)));
                table.AddCell(new Cell().Add(new Paragraph(pond.River ?? "").SetFont(font)));
                table.AddCell(new Cell().Add(new Paragraph(pond.Basin ?? "").SetFont(font)));
                table.AddCell(new Cell().Add(new Paragraph(pond.Status ?? "").SetFont(font)));
                table.AddCell(new Cell().Add(new Paragraph(pond.Issues ?? "").SetFont(font)));
                table.AddCell(new Cell().Add(new Paragraph(pond.Notes ?? "").SetFont(font)));
                table.AddCell(new    Cell().Add(new    Paragraph(pond.ImposedFines.ToString("F2")    ??
"").SetFont(font)));
                table.AddCell(new    Cell().Add(new    Paragraph(pond.ImposedDamages.ToString("F2")    ??
"").SetFont(font)));
                table.AddCell(new    Cell().Add(new    Paragraph(pond.CollectedFines.ToString("F2")    ??
"").SetFont(font)));
                table.AddCell(new   Cell().Add(new   Paragraph(pond.CollectedDamages.ToString("F2")   ??
"").SetFont(font)));
            }

            document.Add(table);
            document.Close();
            pdfWriter.Close();

            return File(memoryStream.ToArray(), "application/pdf", "Ponds.pdf");
        }
    }


    private IQueryable<Pond> GetFilteredPonds()
    {
        var query = _context.Ponds
            .Include(p => p.Lessee)
            .Include(p => p.LeaseAgreement)
            .Include(p => p.WaterUsagePermit)
            .AsQueryable();

        if (IdFilter.HasValue)
            query = query.Where(p => p.Id == IdFilter.Value);
        if (!string.IsNullOrEmpty(NameFilter))
            query = query.Where(p => p.Name.Contains(NameFilter));
        if (!string.IsNullOrEmpty(DistrictFilter))
            query = query.Where(p => p.District.Contains(DistrictFilter));
        if (!string.IsNullOrEmpty(TerritorialCommunityFilter))
            query = query.Where(p => p.TerritorialCommunity.Contains(TerritorialCommunityFilter));
        if (!string.IsNullOrEmpty(SettlementFilter))
            query = query.Where(p => p.Settlement.Contains(SettlementFilter));

        if (MinXFilter.HasValue)
            query = query.Where(p => p.XCoordinate >= MinXFilter.Value);
        if (MaxXFilter.HasValue)
            query = query.Where(p => p.XCoordinate <= MaxXFilter.Value);
        if (MinYFilter.HasValue)
            query = query.Where(p => p.YCoordinate >= MinYFilter.Value);
        if (MaxYFilter.HasValue)
            query = query.Where(p => p.YCoordinate <= MaxYFilter.Value);

        if (LengthFilter.HasValue)
            query = query.Where(p => p.Length == LengthFilter.Value);
        if (WidthFilter.HasValue)
            query = query.Where(p => p.Width == WidthFilter.Value);
        if (DepthFilter.HasValue)
            query = query.Where(p => p.Depth == DepthFilter.Value);
        if (WaterLevelFilter.HasValue)
            query = query.Where(p => p.WaterLevel == WaterLevelFilter.Value);
        if (VolumeFilter.HasValue)
            query = query.Where(p => p.Volume == VolumeFilter.Value);

        if (LeasedAreaFilter.HasValue)
            query = query.Where(p => p.LeasedArea == LeasedAreaFilter.Value);
        if (!string.IsNullOrEmpty(CadastralNumberFilter))
            query = query.Where(p => p.CadastralNumber.Contains(CadastralNumberFilter));
        if (WaterSurfaceAreaFilter.HasValue)
            query = query.Where(p => p.WaterSurfaceArea == WaterSurfaceAreaFilter.Value);
```

```
        if (IsDrainableFilter.HasValue)
            query = query.Where(p => p.IsDrainable == IsDrainableFilter.Value);
        if (HasHydraulicStructureFilter.HasValue)
            query = query.Where(p => p.HasHydraulicStructure == HasHydraulicStructureFilter.Value);

        if (!string.IsNullOrEmpty(HydraulicStructureOwnerFilter))
            query = query.Where(p => p.HydraulicStructureOwner.Contains(HydraulicStructureOwnerFilter));
        if (!string.IsNullOrEmpty(LesseeIdentificationCodeFilter))
            query = query.Where(p =>
p.Lessee.IdentificationCode.Contains(LesseeIdentificationCodeFilter));
        if (!string.IsNullOrEmpty(LeaseAgreementNumberFilter))
            query = query.Where(p => p.LeaseAgreement.Number.Contains(LeaseAgreementNumberFilter));
        if (!string.IsNullOrEmpty(WaterUsagePermitNumberFilter))
            query = query.Where(p => p.WaterUsagePermit.Number.Contains(WaterUsagePermitNumberFilter));

        return query;
    }



}

@page
@model PondManager.Pages.Recipients.IndexModel
@{
    ViewData["Title"] = "Керування списком реціпієнтів";
}

<h2>@ViewData["Title"]</h2>

<div class="mb-3">
    <a asp-page="./Create" class="btn btn-primary">Додати отримувача</a>
    <!-- Оновлена кнопка для негайного надсилання нагадувань через Gmail API -->
    <form method="post" asp-page-handler="SendNotifications" class="d-inline">
        <button type="submit" class="btn btn-warning">
            Надіслати нагадування зараз
        </button>
    </form>
</div>

<table class="table table-striped">
    <thead>
    <tr>
        <th>Email</th>
        <th style="width:150px">Дії</th>
    </tr>
    </thead>
    <tbody>
    @foreach(var r in Model.Recipients)
    {
        <tr>
            <td>@r.Email</td>
            <td>
                <a asp-page="./Edit" asp-route-id="@r.Id" class="btn btn-sm btn-
secondary">Редагувати</a>
                <a asp-page="./Delete" asp-route-id="@r.Id" class="btn btn-sm btn-danger">Видалити</a>
            </td>
        </tr>
    }
    </tbody>
</table>

@if (Model.ShowNotification)
{
    <div class="alert @(Model.NotificationSuccess ? "alert-success" : "alert-danger") mt-3">
        @Model.NotificationMessage
    </div>
}

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using Google.Apis.Auth.OAuth2;
```

```
using Google.Apis.Gmail.v1;
using Google.Apis.Gmail.v1.Data;
using Google.Apis.Services;
using Google.Apis.Util.Store;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using MimeKit;
using PondManager.Data;
using PondManager.Data.Data;
using PondManager.Data.Models;


namespace PondManager.Pages.Recipients
{
    public class IndexModel : PageModel
    {
        private readonly ApplicationDbContext _context;
        private readonly IConfiguration _config;
        private readonly string[] _scopes = { GmailService.Scope.GmailSend };
        private const string CredentialFile = "credentials.json";
        private const string TokenFolder = "token_store";

        public IndexModel(ApplicationDbContext context, IConfiguration config)
        {
            _context = context;
            _config = config;
        }

        public IList<EmailRecipient> Recipients { get; set; } = new List<EmailRecipient>();

        // Для виводу повідомлення після спроби надсилання
        public bool ShowNotification { get; set; }
        public bool NotificationSuccess { get; set; }
        public string NotificationMessage { get; set; } = "";

        public async Task OnGetAsync()
        {
            Recipients = await _context.EmailRecipients.ToListAsync();
        }

        public async Task<IActionResult> OnPostSendNotificationsAsync()
        {
            Recipients = await _context.EmailRecipients.ToListAsync();
            if (!Recipients.Any())
            {
                ShowNotification = true;
                NotificationSuccess = false;
                NotificationMessage = "Список отримувачів порожній.";
                return Page();
            }

            try
            {
                // 1. Отримати OAuth2-credential
                var credential = await GetGoogleCredentialAsync();

                // 2. Ініціалізувати GmailService
                var gmail = new GmailService(new BaseClientService.Initializer
                {
                    HttpClientInitializer = credential,
                    ApplicationName = "PondManager"
                });

                // 3. Зчитати дату нагадування (сьогодні +31 день)
                var targetDate = DateTime.Today.AddDays(31);

                // 4. Зібрати договори і дозволи, що закінчуються в targetDate
                var expiringLeases = await _context.LeaseAgreements
                    .Where(a => a.Date.AddYears(a.TermInYears) <= targetDate)
                    .Select(a => new { a.Number, Expiration = a.Date.AddYears(a.TermInYears) })
                    .ToListAsync();

                var expiringPermits = await _context.WaterUsagePermits
                    .Where(p => p.StartDate.AddYears(p.TermInYears) <= targetDate)
                    .Select(p => new { p.Number, Expiration = p.StartDate.AddYears(p.TermInYears) })
```

```
                    .ToListAsync();

                if (!expiringLeases.Any() && !expiringPermits.Any())
                {
                    ShowNotification = true;
                    NotificationSuccess = true;
                    NotificationMessage = $"Немає договорів чи дозволів, що закінчуються
{targetDate:dd.MM.yyyy}.";
                    return Page();
                }

                // 5. Формуємо HTML-тіло листа
                var sb = new System.Text.StringBuilder();
                sb.AppendLine($"<h3>Нагадування: закінчення терміну {targetDate:dd.MM.yyyy}</h3>");
                if (expiringLeases.Any())
                {
                    sb.AppendLine("<b>Договори оренди:</b><ul>");
                    expiringLeases.ForEach(l =>
                        sb.AppendLine($"<li>№{l.Number}, закінч. {l.Expiration:dd.MM.yyyy}</li>"));
                    sb.AppendLine("</ul>");
                }
                if (expiringPermits.Any())
                {
                    sb.AppendLine("<b>Дозволи на спецводокористування:</b><ul>");
                    expiringPermits.ForEach(p =>
                        sb.AppendLine($"<li>№{p.Number}, закінч. {p.Expiration:dd.MM.yyyy}</li>"));
                    sb.AppendLine("</ul>");
                }

                // 6. Зчитати FROM з конфігурації
                var fromAddress = _config["EmailReminder:Smtp:From"] ?? "noreply@example.com";

                // 7. Створити MimeMessage
                var mime = new MimeMessage();
                mime.From.Add(new MailboxAddress("PondManager", fromAddress));
                Recipients.ToList().ForEach(r => mime.To.Add(MailboxAddress.Parse(r.Email)));
                mime.Subject = $"Нагадування про закінчення термінів {targetDate:dd.MM.yyyy}";
                mime.Body = new TextPart("html") { Text = sb.ToString() };

                // 8. Конвертувати в base64url
                using var ms = new MemoryStream();
                mime.WriteTo(ms);
                var raw = Convert.ToBase64String(ms.ToArray())
                                .Replace('+', '-').Replace('/', '_').TrimEnd('=');

                var gmailMsg = new Message { Raw = raw };

                // 9. Надіслати
                await gmail.Users.Messages.Send(gmailMsg, "me").ExecuteAsync();

                ShowNotification = true;
                NotificationSuccess = true;
                NotificationMessage = $"Нагадування успішно відправлено {Recipients.Count} адресам.";
            }
            catch (Exception ex)
            {
                ShowNotification = true;
                NotificationSuccess = false;
                NotificationMessage = $"Помилка при відправці: {ex.Message}";
            }

            return Page();
        }

        private async Task<UserCredential> GetGoogleCredentialAsync()
        {
            using var stream = new FileStream(CredentialFile, FileMode.Open, FileAccess.Read);
            var credPath = Path.Combine(Environment.CurrentDirectory, TokenFolder);

            return await GoogleWebAuthorizationBroker.AuthorizeAsync(
                GoogleClientSecrets.FromStream(stream).Secrets,
                _scopes,
                "user",
                CancellationToken.None,
                new FileDataStore(credPath, true)
            );
        }
```

```
    }
}


<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - WaterManagementSystem</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom
box-shadow mb-3">
            <div class="container">
                <a class="navbar-brand" asp-page="/Index">Управління водними ресурсами</a>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target=".navbar-collapse">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-page="/Index">Головна</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-page="/Ponds/Index">Ставки</a>
                        </li>
                        <li class="nav-item dropdown">
                            <a class="nav-link dropdown-toggle text-dark" href="#" role="button" data-
bs-toggle="dropdown">
                                Інше
                            </a>
                            <ul class="dropdown-menu">
                                <li class="nav-item">
                                    <a class="dropdown-item" asp-page="/Lessees/Index">Орендарі</a>
                                </li>
                                <li class="nav-item">
                                    <a class="dropdown-item" asp-page="/LeaseAgreements/Index">Договори
оренди</a>
                                </li>
                                <li class="nav-item">
                                    <a class="dropdown-item" asp-
page="/WaterUsagePermits/Index">Дозволи на водокористування</a>
                                </li>
                            </ul>
                        </li>
                        <li class="nav-item dropdown">
                            <a class="nav-link dropdown-toggle text-dark" href="#" role="button" data-
bs-toggle="dropdown">
                                Термінове
                            </a>
                            <ul class="dropdown-menu">
                                <li><a class="dropdown-item" asp-
page="/Expiring/LeaseAgreements">Договори оренди</a></li>
                                <li><a class="dropdown-item" asp-
page="/Expiring/WaterUsagePermits">Спецвододозволи</a></li>
                            </ul>
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>

    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container">
```

```
            &copy; 2025 - WaterManagementSystem - <a asp-page="/Privacy">Privacy</a>
        </div>
    </footer>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>

/* Please see documentation at https://learn.microsoft.com/aspnet/core/client-side/bundling-and-
minification
for details on configuring this project to bundle and minify static web assets. */

a.navbar-brand {
  white-space: normal;
  text-align: center;
  word-break: break-all;
}

a {
  color: #0077cc;
}

.btn-primary {
  color: #fff;
  background-color: #1b6ec2;
  border-color: #1861ac;
}

.nav-pills .nav-link.active, .nav-pills .show > .nav-link {
  color: #fff;
  background-color: #1b6ec2;
  border-color: #1861ac;
}

.border-top {
  border-top: 1px solid #e5e5e5;
}
.border-bottom {
  border-bottom: 1px solid #e5e5e5;
}

.box-shadow {
  box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);
}

button.accept-policy {
  font-size: 1rem;
  line-height: inherit;
}

.footer {
  position: absolute;
  bottom: 0;
  width: 100%;
  white-space: nowrap;
  line-height: 60px;
}

<script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>
<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"></script>

@page
@model WaterUsagePermits.CreateModel
@{
    ViewData["Title"] = "Додати дозвіл на водокористування";
}

<h2>Додати дозвіл на водокористування</h2>

<form method="post">
    <div class="form-group">
        <label asp-for="WaterUsagePermit.Number">Номер</label>
        <input asp-for="WaterUsagePermit.Number" class="form-control" />
    </div>
```

```
    <div class="form-group">
        <label asp-for="WaterUsagePermit.StartDate">Дата</label>
        <input asp-for="WaterUsagePermit.StartDate" type="date" class="form-control" />
    </div>
    <div class="form-group">
        <label asp-for="WaterUsagePermit.TermInYears">Термін у роках</label>
        <input asp-for="WaterUsagePermit.TermInYears" class="form-control" />
    </div>
    <button type="submit" class="btn btn-primary mt-3">Зберегти</button>
    <a asp-page="Index" class="btn btn-secondary mt-3">Скасувати</a>
</form>
```

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using PondManager.Data.Data;
using PondManager.Data.Models;

namespace PondManager.Pages.WaterUsagePermits
{
    public class CreateModel : PageModel
    {
        private readonly ApplicationDbContext _context;
        public CreateModel(ApplicationDbContext context)
        {
            _context = context;
        }

        [BindProperty]
        public WaterUsagePermit WaterUsagePermit { get; set; } = new WaterUsagePermit();

        public IActionResult OnGet() => Page();

        public async Task<IActionResult> OnPostAsync()
        {
            if (!ModelState.IsValid)
                return Page();

            _context.WaterUsagePermits.Add(WaterUsagePermit);
            await _context.SaveChangesAsync();
            return RedirectToPage("Index");
        }
    }
}
```

```
@page "{id:int}"
@model EditWaterUsagePermitModel
@{
    ViewData["Title"] = "Редагування дозволу на водокористування";
}

<h2>Редагування дозволу на водокористування</h2>

<form method="post">
    <input type="hidden" asp-for="WaterUsagePermit.Id" />
    <div class="form-group">
        <label asp-for="WaterUsagePermit.Number">Номер</label>
        <input asp-for="WaterUsagePermit.Number" class="form-control"/>
    </div>
    <div class="form-group">
        <label asp-for="WaterUsagePermit.StartDate">Дата</label>
        <input asp-for="WaterUsagePermit.StartDate" class="form-control" type="date"/>
    </div>
    <div class="form-group">
        <label asp-for="WaterUsagePermit.TermInYears">Термін у роках</label>
        <input asp-for="WaterUsagePermit.TermInYears" class="form-control"/>
    </div>
    <button type="submit" class="btn btn-primary mt-3">Зберегти</button>
</form>
```

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.EntityFrameworkCore;
using PondManager.Data.Data;
using PondManager.Data.Models;

public class EditWaterUsagePermitModel : PageModel
{
```

```
    private readonly ApplicationDbContext _context;

    public EditWaterUsagePermitModel(ApplicationDbContext context)
    {
        _context = context;
    }

    [BindProperty]
    public WaterUsagePermit WaterUsagePermit { get; set; } = default!;

    public async Task<IActionResult> OnGetAsync(int id)
    {
        WaterUsagePermit = await _context.WaterUsagePermits.FindAsync(id);
        if (WaterUsagePermit == null)
        {
            return NotFound();
        }
        return Page();
    }

    public async Task<IActionResult> OnPostAsync()
    {

         context.Attach(WaterUsagePermit).State = EntityState.Modified;

        try
        {
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!_context.WaterUsagePermits.Any(e => e.Id == WaterUsagePermit.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }

        return RedirectToPage("./Index");
    }
}

@page
@model WaterUsagePermits.IndexModel
@{
    ViewData["Title"] = "Список дозволів на водокористування";
}

<h2>Список дозволів на водокористування</h2>
<a asp-page="Create" class="btn btn-primary mb-3">Додати дозвіл</a>

<table class="table table-striped">
    <thead>
    <tr>
        <th>Id</th>
        <th>Номер</th>
        <th>Дата початку</th>
        <th>Термін (років)</th>
        <th>ID пов'язаних ставків</th>
        <th>Дії</th>
    </tr>
    </thead>
    <tbody>
    @foreach (var permit in Model.WaterUsagePermits)
    {
        <tr>
            <td>@permit.Id</td>
            <td>@permit.Number</td>
            <td>@permit.StartDate.ToShortDateString()</td>
            <td>@permit.TermInYears</td>
            <td>
                @if(permit.Ponds is not null)
                    @string.Join(", ", permit.Ponds.Select(p => p.Id))
            </td>
```

```
            <td>
                <a asp-page="Edit" asp-route-id="@permit.Id" class="btn btn-sm btn-
warning">Редагувати</a>
                <form method="post" asp-page-handler="Delete" asp-route-id="@permit.Id"
style="display:inline;">
                    <button type="submit" class="btn btn-sm btn-danger" onclick="return confirm('Ви
впевнені, що хочете видалити цей дозвіл?');">Видалити</button>
                </form>
            </td>
        </tr>
    }
    </tbody>
</table>

using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.EntityFrameworkCore;
using PondManager.Data.Data;
using PondManager.Data.Models;

namespace PondManager.Pages.WaterUsagePermits
{
    public class IndexModel : PageModel
    {
        private readonly ApplicationDbContext _context;

        public IndexModel(ApplicationDbContext context)
        {
            _context = context;
        }

        public IList<WaterUsagePermit> WaterUsagePermits { get; set; }

        public async Task<IActionResult> OnGetAsync()
        {
            WaterUsagePermits = await  context.WaterUsagePermits.ToListAsync();

            foreach (var waterUsagePermit in WaterUsagePermits)
            {
                waterUsagePermit.Ponds = _context.Ponds.Where(p => p.WaterUsagePermitId ==
waterUsagePermit.Id).ToList();
            }

            return Page();
        }

        public async Task<IActionResult> OnPostDeleteAsync(int id)
        {
            var permit = await _context.WaterUsagePermits.FindAsync(id);

            if (permit == null)
            {
                return NotFound();
            }


            permit.Ponds = await _context.Ponds.Where(p => p.WaterUsagePermitId == id).ToListAsync();

            _context.WaterUsagePermits.Remove(permit);
            await _context.SaveChangesAsync();

            return RedirectToPage();
        }
    }

}

@using PondManager
@namespace PondManager.Pages
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers

@{
    Layout = "_Layout";
}

{
  "Logging": {
```

```
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection":
"Server=.;Database=PondManager.DB;Trusted_Connection=True;TrustServerCertificate=True;"
  },
  "Redis": {
    "Configuration": "localhost:6379",
    "InstanceName": "PondManager:"
  }
}
```

```
using Microsoft.EntityFrameworkCore;
using PondManager.Business.Interfaces;
using PondManager.Business.Services;
using PondManager.Data.Data;

var builder = WebApplication.CreateBuilder(args);


var redisSection = builder.Configuration.GetSection("Redis");
builder.Services.AddStackExchangeRedisCache(options =>
{
    options.Configuration = redisSection["Configuration"];
    options.InstanceName = redisSection["InstanceName"];
});

builder.Services.AddRazorPages();

builder.Services.AddScoped<IExcelImportService, ExcelImportService>();

builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

var app = builder.Build();

// Налаштування middleware
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.UseAuthorization();
app.MapRazorPages();

app.Run();
```

Додаток Д – Ілюстративна частина

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ
КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Бакалаврська кваліфікаційна робота на тему:
«Розробка методу та засобів реалізації вебсистеми обліку ставків Вінницької області»

Автор: ст. гр. 1ПІ-21б
Миронюк Олександр

Науковий керівник: к.т.н.,
доц. каф. ПЗ Войтко В. В.

Рисунок Д.1 – Титульний слайд

АКТУАЛЬНІСТЬ РОЗРОБКИ (Ч.1)

- Управління ставками досі переважно здійснюється вручну, що спричиняє помилки та знижує ефективність.
- Відсутність автоматизованих систем обмежує можливості точного обліку географічних, екологічних і економічних параметрів.
- Геоінформаційні системи (ГІС) дозволяють репрезентувати ставки у вигляді просторових даних для швидшої ідентифікації місцязнаходження

Рисунок Д.2 – Актуальність розробки ч.1

АКТУАЛЬНІСТЬ РОЗРОБКИ (Ч.2)

- Дані про ставки включають багато параметрів, зібраних у одному місці: координати, глибину, площу, заростання, орендарів, історію порушень тощо.
- Автоматизація процесів обліку значно економить час і підвищує точність у порівнянні з ручними методами.
- Точні дані необхідні для стратегічного управління ресурсами – від профілактики замулення до контролю за умовами оренди.
- Існуючі рішення часто недоступні для місцевої влади через відсутність ГІС-інтеграції та низький рівень деталізації.

Рисунок Д.3 – Актуальність розробки ч.2

МЕТА, ПРЕДМЕТ ТА ОБ’ЄКТ ДОСЛІДЖЕННЯ

Метою дослідження є покращення управління водними ресурсами шляхом розробки програмного забезпечення для обліку ставків, яке дозволить ефективно збирати, аналізувати та візуалізувати дані про водні об'єкти.

Об’єкт дослідження – процес розробки вебсистеми для управління водними ресурсами на прикладі ставків Вінницької області.

Предмет дослідження – методи та засоби автоматизації обліку та моніторингу ставків.

Рисунок Д.4 – Мета, предмет та об’єкт дослідження

ПОСТАНОВКА ЗАДАЧІ

- розробити модель автоматизованої вебсистеми обліку ставків, яка міститиме основний функціонал системи;
- розробка механізму імпорту даних з Excel;
- створення механізмів експорту даних у форматах PDF/Excel;
- реалізація інтерактивної карти з використанням бібліотеки Leaflet.js;
- реалізувати функціонал нагадування через електронну пошту;
- розробити модуль статистики;
- інтеграція модулів у єдину систему з інтуїтивним інтерфейсом;
- здійснити тестування вебсистеми відповідно до поставлених задач.

Рисунок Д.5 – Постановка задачі

НАУКОВА НОВИЗНА

1. Подальшого розвитку набув метод кешування даних у системі за допомогою Redis, який, на відміну від існуючих рішень, базується на високопродуктивному зберіганні в пам'яті та асинхронному TTL-контролі, що значно зменшує час відповіді на запити, які часто зустрічаються, забезпечує зниження навантаження на базу даних і підвищує масштабованість додатка.

2. Подальшого розвитку набув метод формування системи нагадувань через електронну пошту, що реалізована з використанням протоколу OAuth 2.0 та Gmail API і, на відміну від традиційних SMTP-рішень, забезпечує безпечну авторизацію без зберігання паролів, автоматичну обробку черги повідомлень і можливість персоналізації листів відповідно до налаштувань користувача.

Рисунок Д.6 – Наукова новизна

ПРАКТИЧНА ЦІННІСТЬ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Практична цінність отриманих результатів полягає у можливості використання розробленої вебсистеми для автоматизації процесів обліку ставків, зокрема, дозволяє користувачам отримати всю необхідну інформацію про ставки у вигляді інтерактивної мапи з позначеними ставками з використанням Redis для кешування, можливістю експортувати цю інформацію у Excel/PDF формати, отримати різноманітну статистику про водні ресурси у системі та отримувати сповіщення на емейл про договори оренди і дозволи на водокористування, що скоро закінчаться.

Рисунок Д.7 – Практична цінність отриманих результатів

АНАЛІЗ АНАЛОГІВ

Критерії	ArcGIS	Держводагентство	HydroMonitor	QGIS	AQUARIUS	Власна розробка
Деталізовані дані водойми	+	+	+	-	+	+
Експорт/імпорт	+	-	+	+	+	+
Нагадування	-	-	-	-	-	+
Кадастрові дані	+	+	-	+	+	+
Відстеження порушень	-	+	-	-	-	+
Загальна оцінка	60%	60%	40%	40%	60%	100%

Рисунок Д.8 – Аналіз аналогів

Рисунок Д.9 – Модель роботи системи

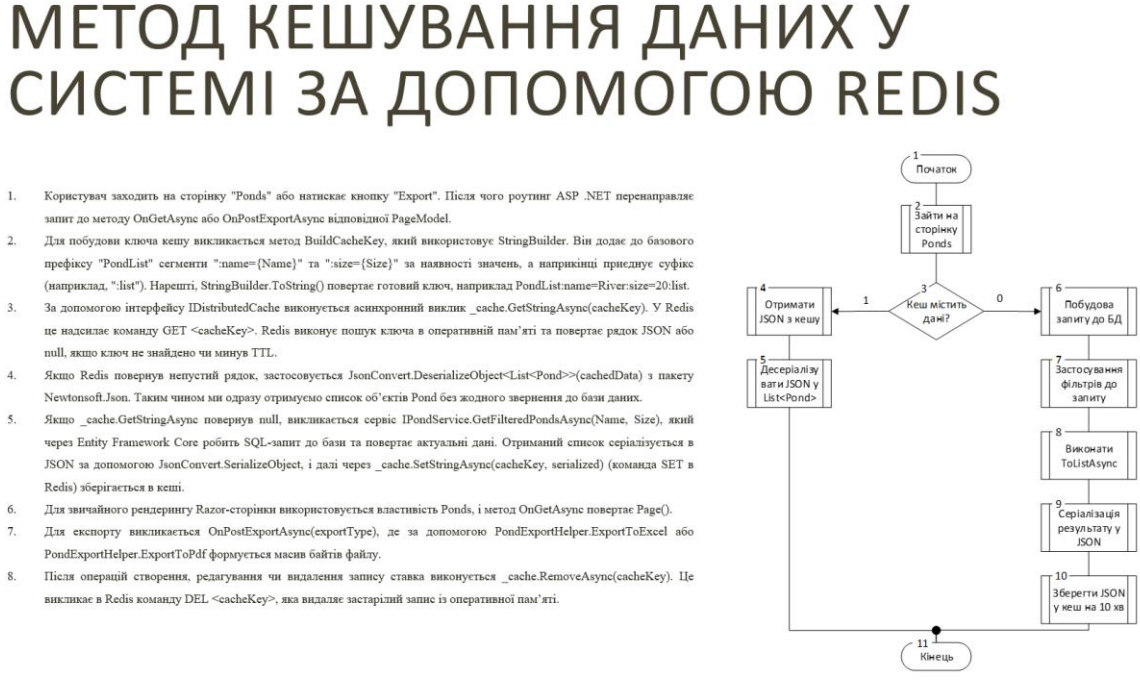


Рисунок Д.10 – Метод кешування даних у системі за допомогою Redis

МЕТОД ФОРМУВАННЯ СИСТЕМИ НАГАДУВАНЬ ЧЕРЕЗ ЕЛЕКТРОННУ ПОШТУ

1. Застосунок під час запуску реєструє фоновий сервіс, який реалізує інтерфейс BackgroundService, і налаштовує його на запуск поза запитами користувача.
2. Служба обчислює кількість часу до першого числа наступного місяця о 9-й годині, використовуючи системний час сервера, і виконує асинхронне очікування через Task.Delay.
3. Коли настає час виконання, служба відкриває область залежностей через IServiceScopeFactory, щоб отримати екземпляр ApplicationDbContext із сконфігурованими налаштуваннями EF Core для доступу до бази даних.
4. З бази даних витягуються всі записи договорів оренди й дозволів, дата закінчення яких попадає в інтервал від сьогодні до 31 дня вперед. Дата закінчення обчислюється методом Date.AddYears(TermInYears) для договорів і StartDate.AddYears(TermInYears) для дозволів.
5. Якщо жодного договору або дозволу не знайдено, служба завершує поточне виконання без надсилання, повторивши очікування наступного місяця.
6. Для формування вмісту листа створюється HTML-фрагмент у StringBuilder, куди послідовно додаються заголовок із кінцевою датою та списки знайдених договорів і дозволів з їхніми номерами й датами.
7. Служба отримує шлях до файлу credentials.json із кореня проєкту та викликає GoogleWebAuthorizationBroker.AuthorizeAsync, передаючи область доступу GmailSend, щоб автентифікуватися через OAuth 2.0 і зберегти токени в папці token_store.
8. Після успішної авторизації створюється екземпляр GmailService з HttpClientInitializer, встановленим у щойно отриманий UserCredential, та задається ім'я програми "PondManager".
9. Для складання листа використовується бібліотека MimeKit: створюється об'єкт MimeMessage, задаються поля From, To, Subject та тіло в форматі TextPart("html") із раніше побудованим HTML.
10. Підготовлене Mime-повідомлення записується у пам'яті у MemoryStream, після чого його байти перетворюються в base64-url, замінюючи символи + на - і / на _, та обрізаючи символи = наприкінці.
11. Формується об'єкт Gmail API Message з властивістю Raw, куди присвоюється отриманий base64-url рядок.
12. Відправка виконується викликом gmail.Users.Messages.Send(message, "me").ExecuteAsync(), де "me" означає авторизованого користувача, на якого видано токен.
13. Після успішної відправки служба фіксує результат у логах або внутрішніх властивостях, щоб у разі помилки можна було побачити текст виключення та повторити спробу за наступного циклу.
14. Служба знову переходить в режим очікування до наступного запуску, зберігаючи автоматичне виконання процедури щомісяця без взаємодії користувача.

Рисунок Д.11 – Метод формування системи нагадувань через електронну пошту

БЛОК-СХЕМА АЛГОРИТМУ РОБОТИ ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ СТОРІНОК «СТАВКИ» ТА «ОРЕНДАРІ»

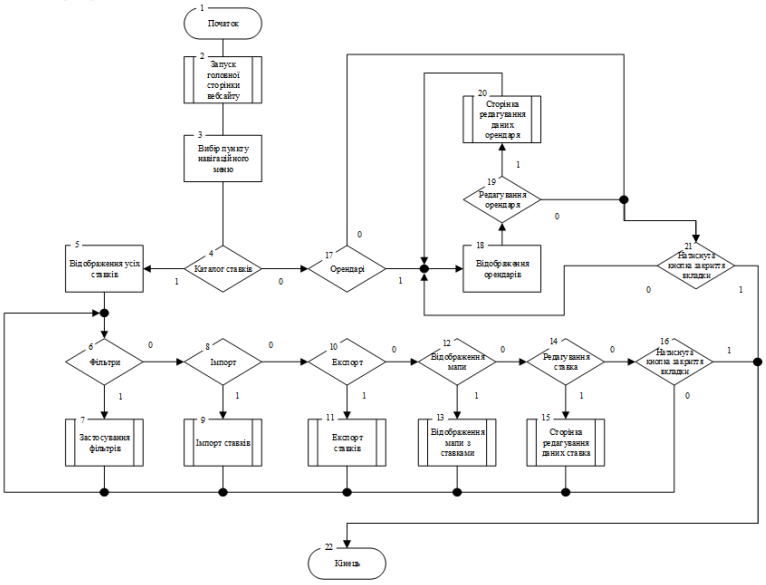


Рисунок Д.12 – Блок схема алгоритму роботи програмного застосунку для сторінок «Ставки» та «Орендарі»

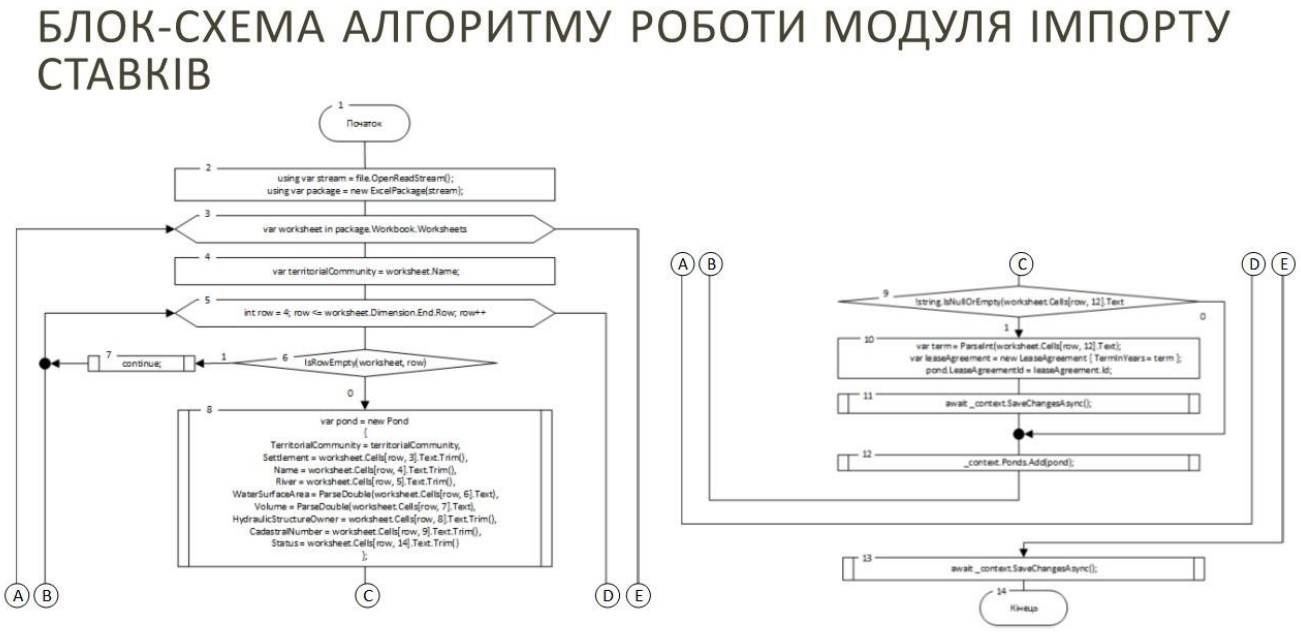


Рисунок Д.13 – Блок-схема алгоритму роботи модуля імпорту ставків

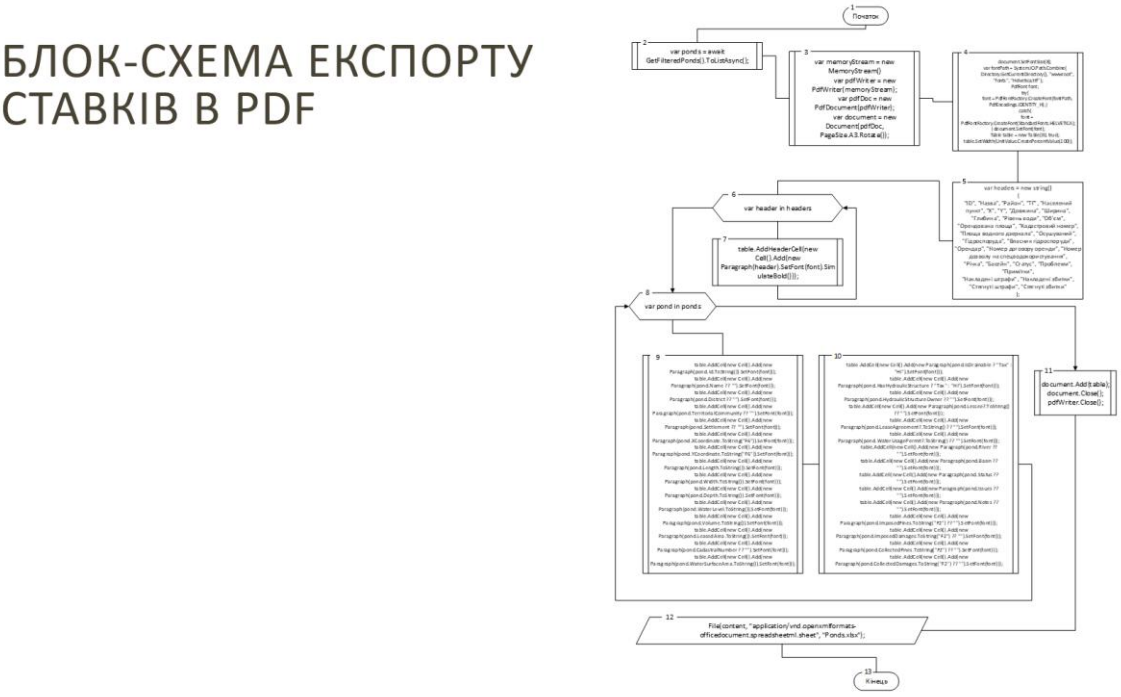


Рисунок Д.14 – Блок-схема експорту ставків в PDF

БЛОК-СХЕМА ЕКСПОРТУ СТАВКІВ В EXCEL

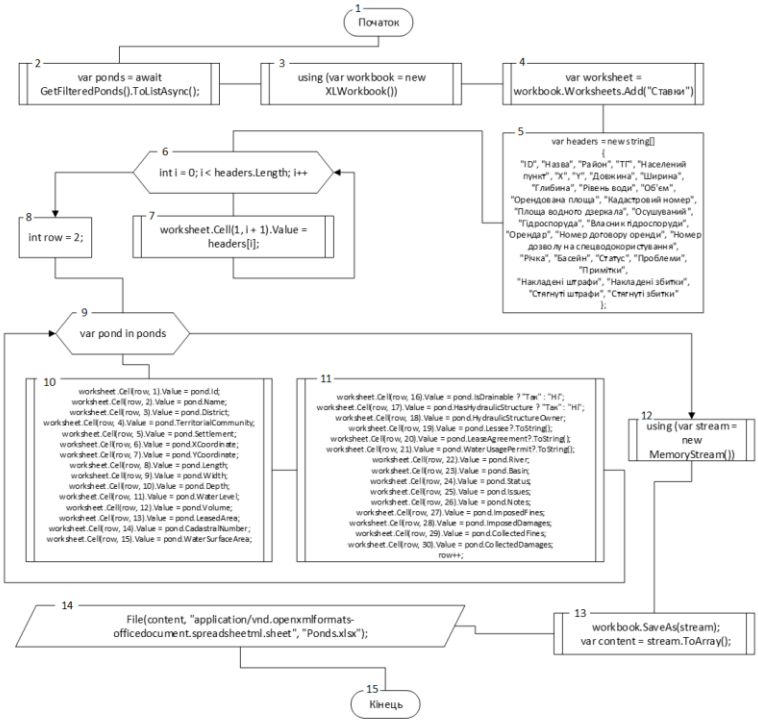


Рисунок Д.15 – Блок-схема експорту ставків в Excel

БЛОК-СХЕМА ЗАВАНТАЖЕННЯ ДАНИХ МОДУЛЯ СТАТИСТИКИ

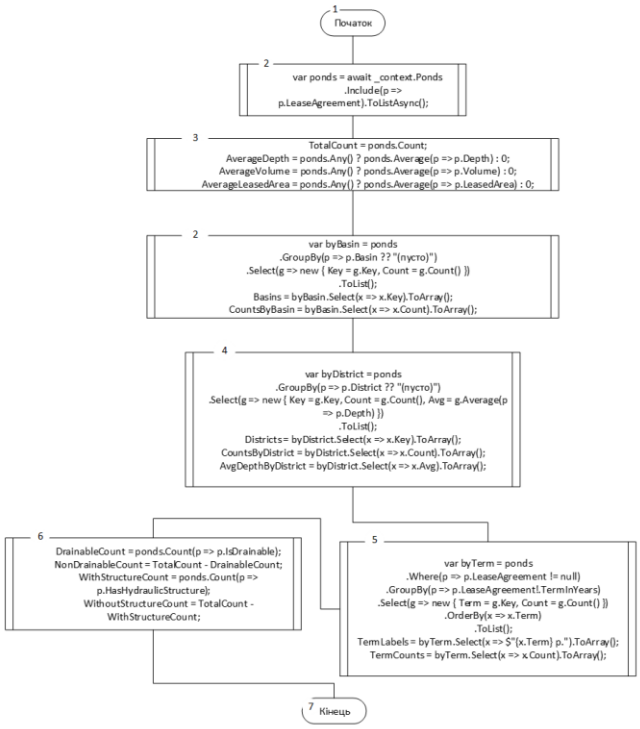


Рисунок Д.16 – Блок-схема завантаження даних модуля статистики

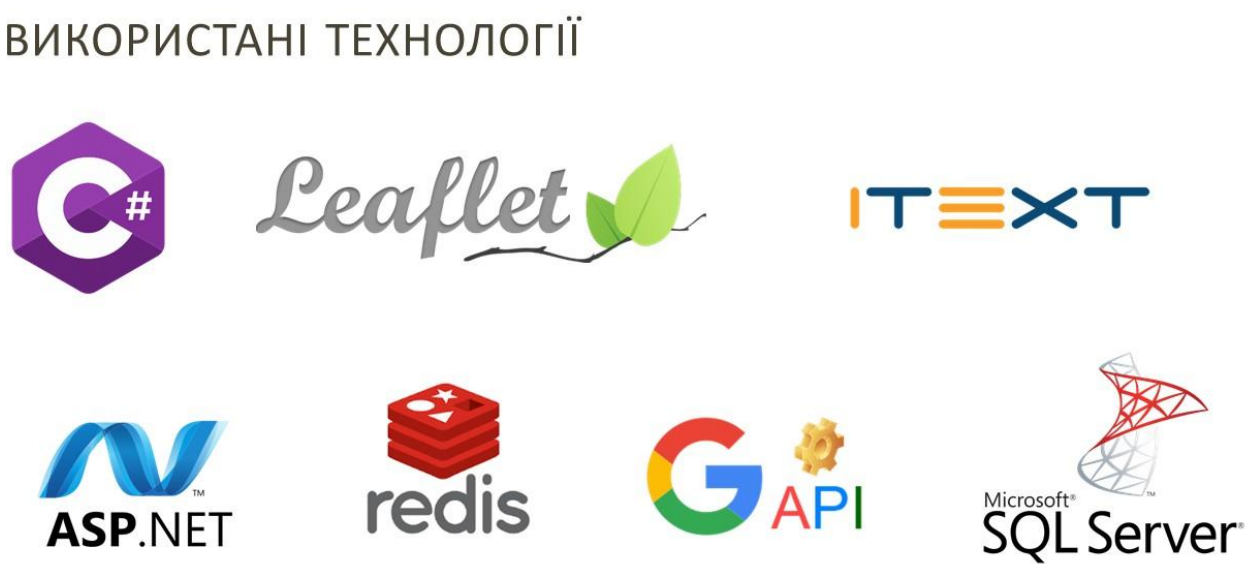


Рисунок Д.17 – Використані технології

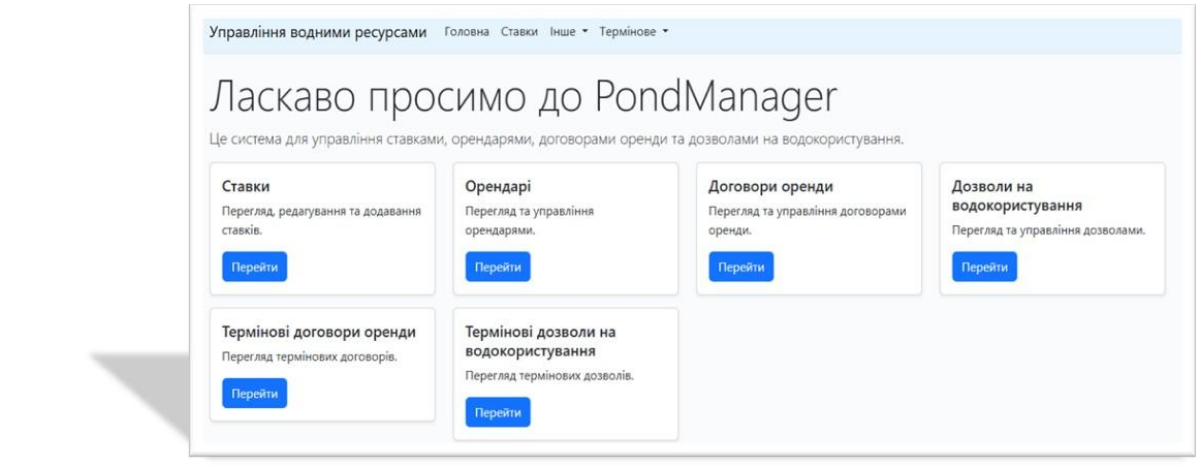


Рисунок Д.18 – Функціонал вебсистеми

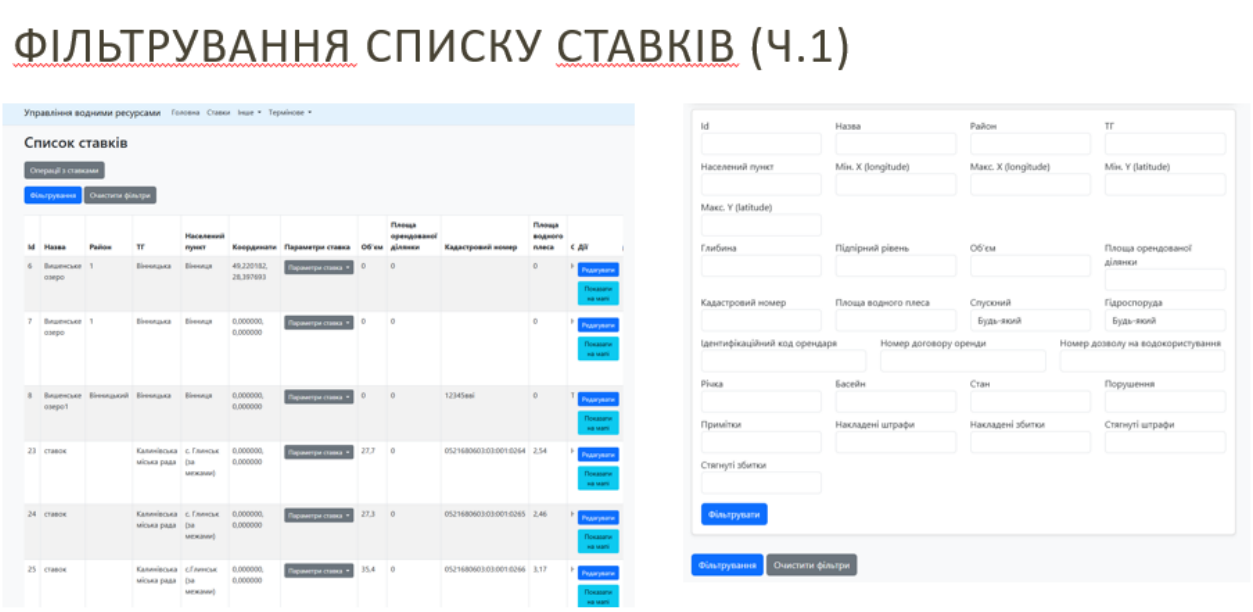


Рисунок Д.19 – Фільтрування списку ставків ч.1

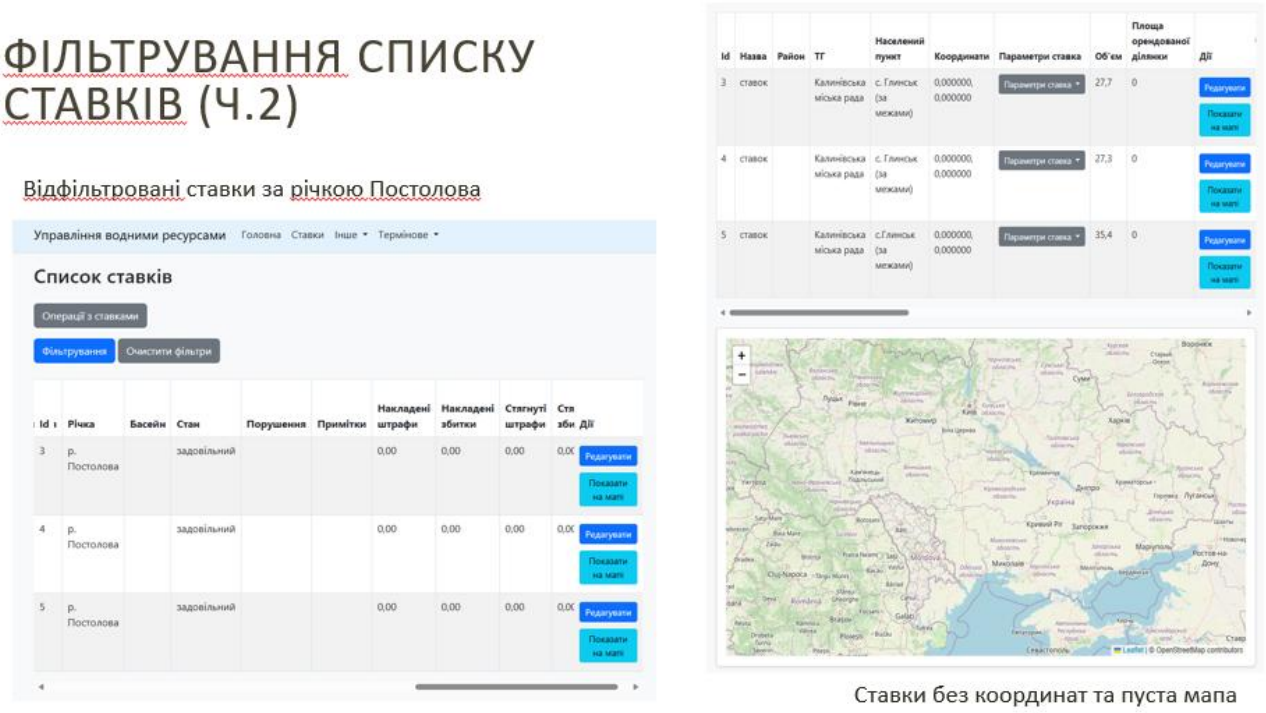


Рисунок Д.20 – Фільтрування списку ставків ч.2

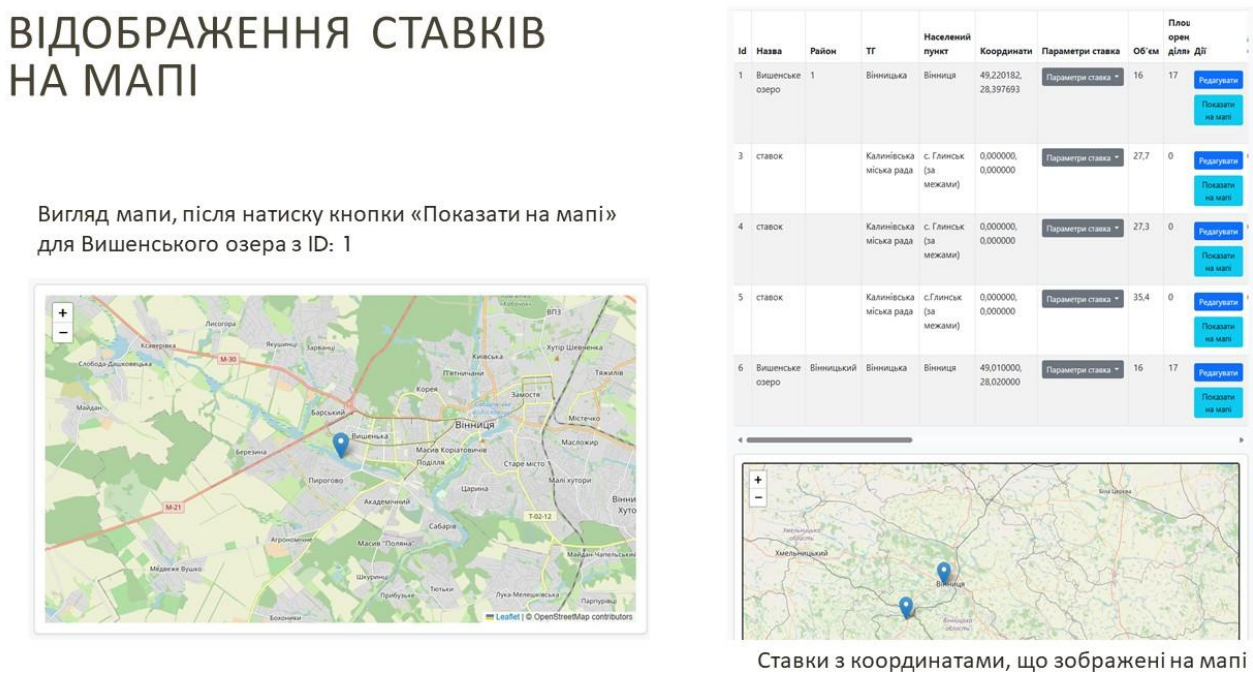


Рисунок Д.21 – Відображення ставків на мапі

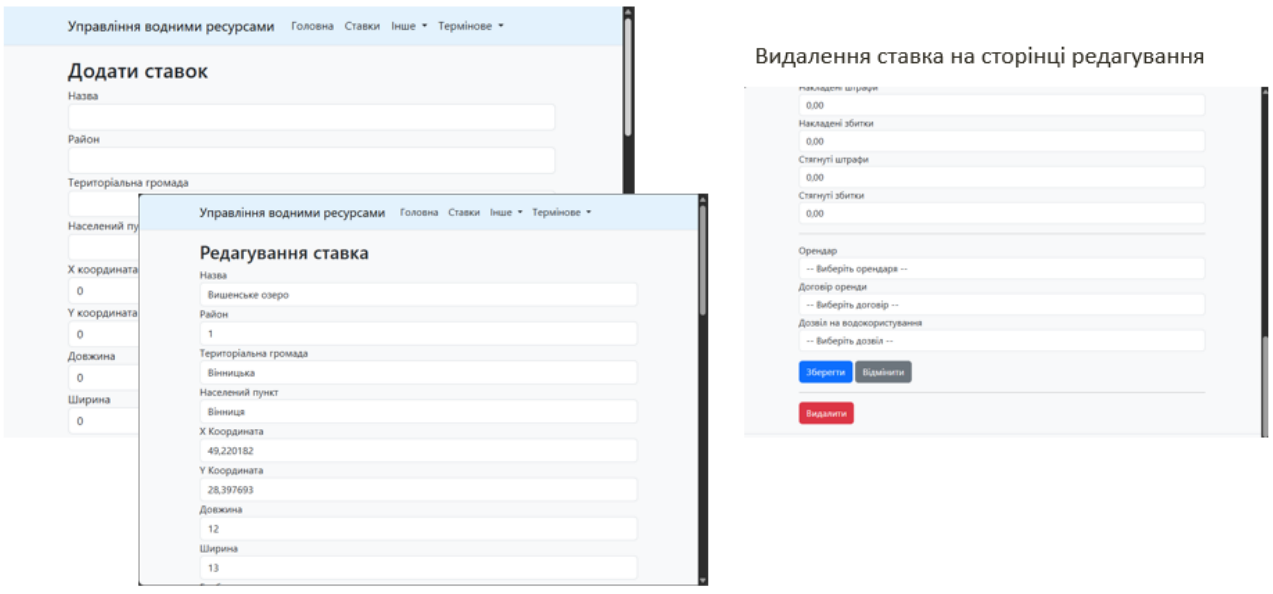


Рисунок Д.22 – CRUD операції ч.1

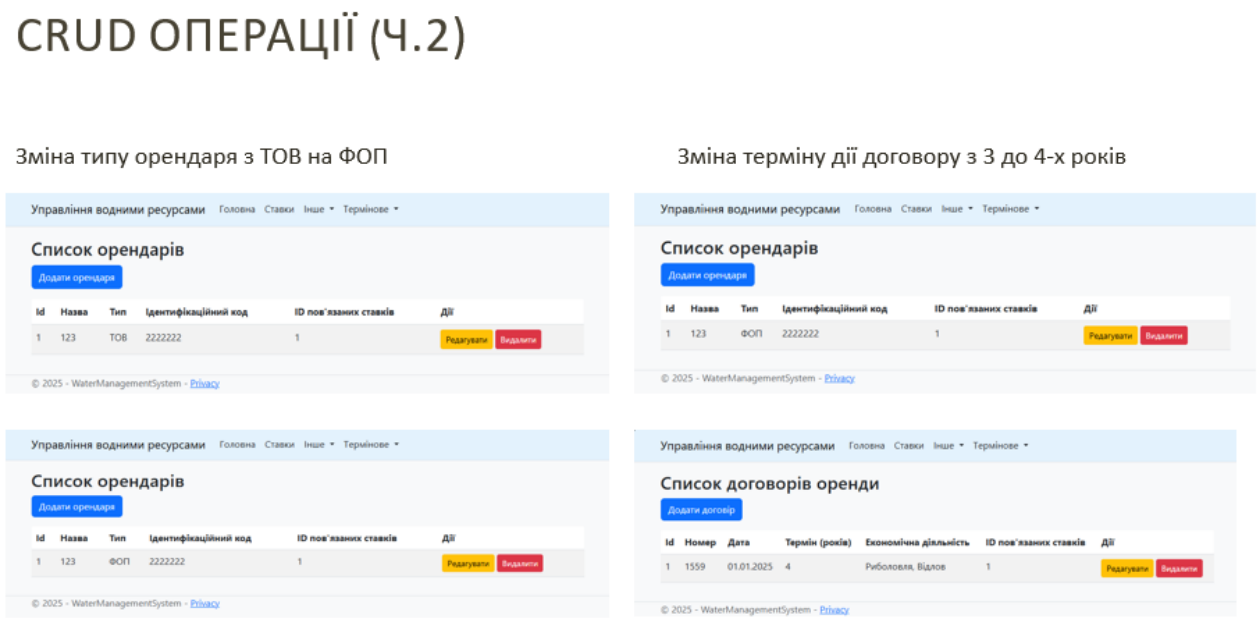


Рисунок Д.23 – CRUD операції ч.2

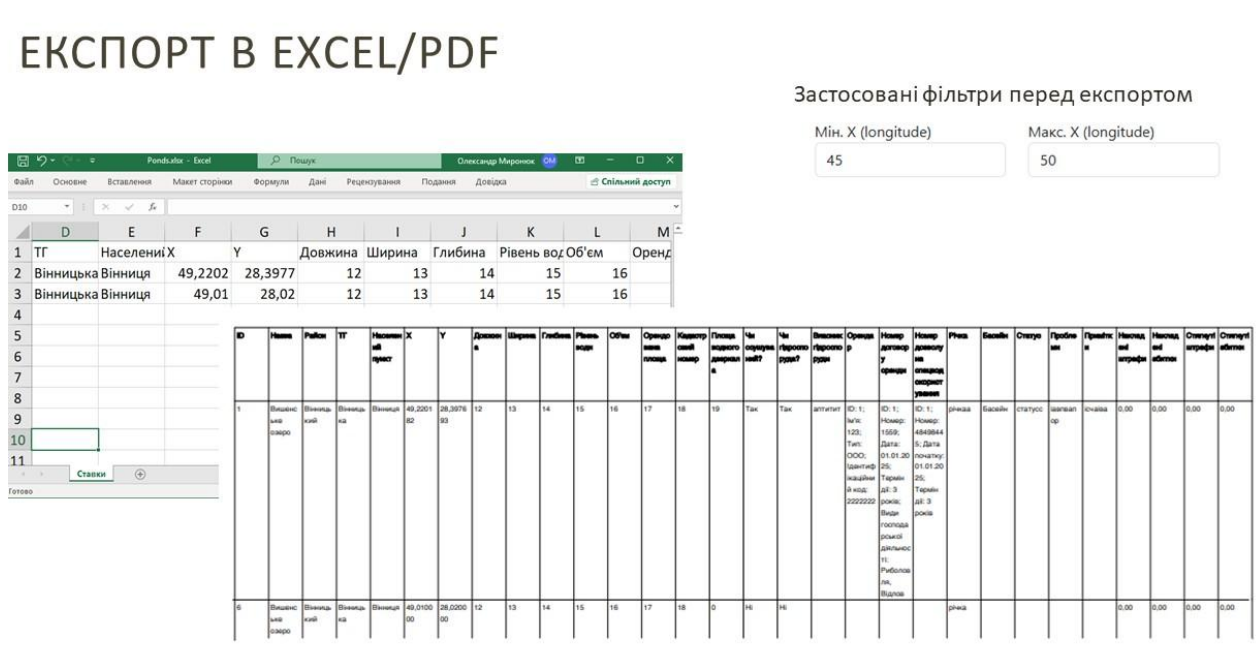


Рисунок Д.24 – Дкспорт в EXCEL/PDF

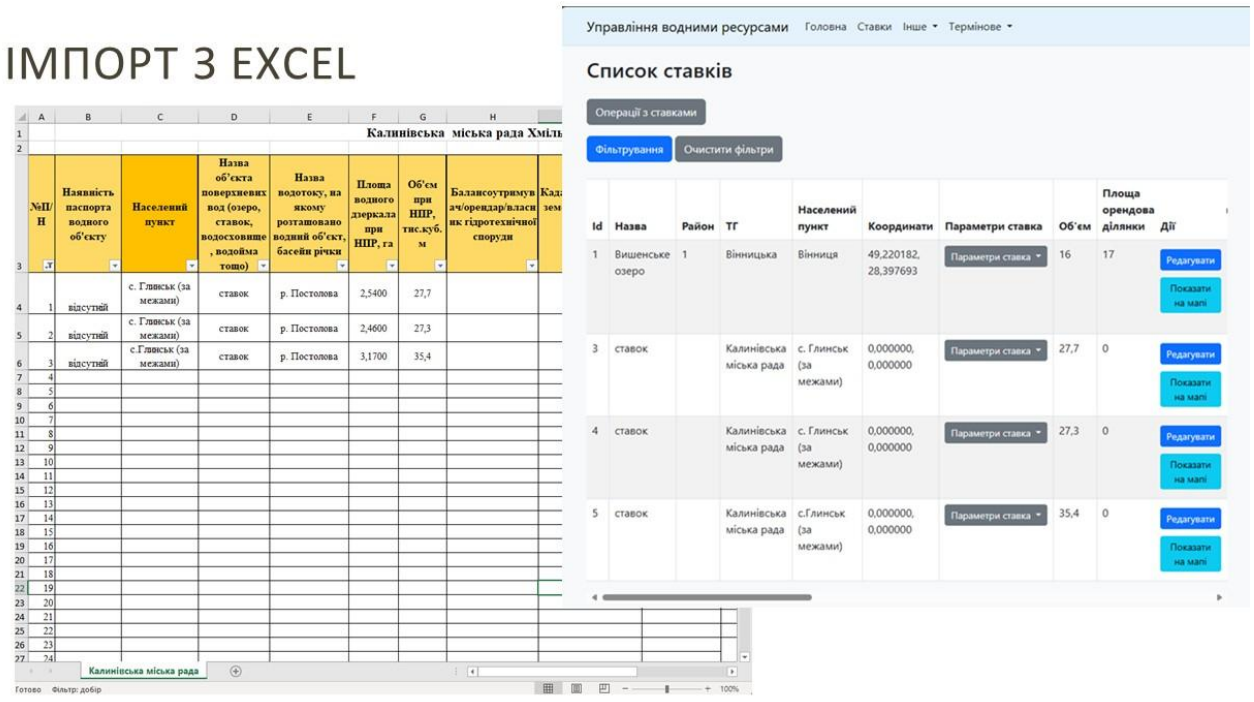


Рисунок Д.25 – Імпорт з EXCEL

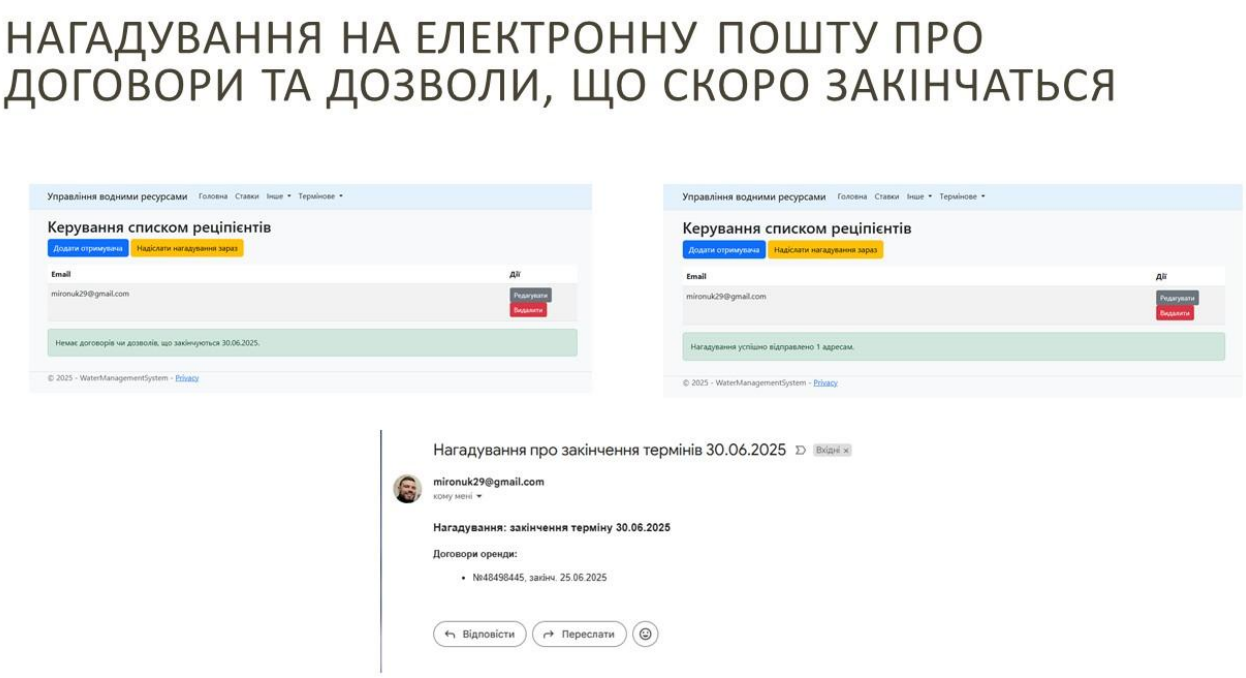


Рисунок Д.26 – Нагадування на електронну пошту про договори та дозволи, що скоро закінчаться

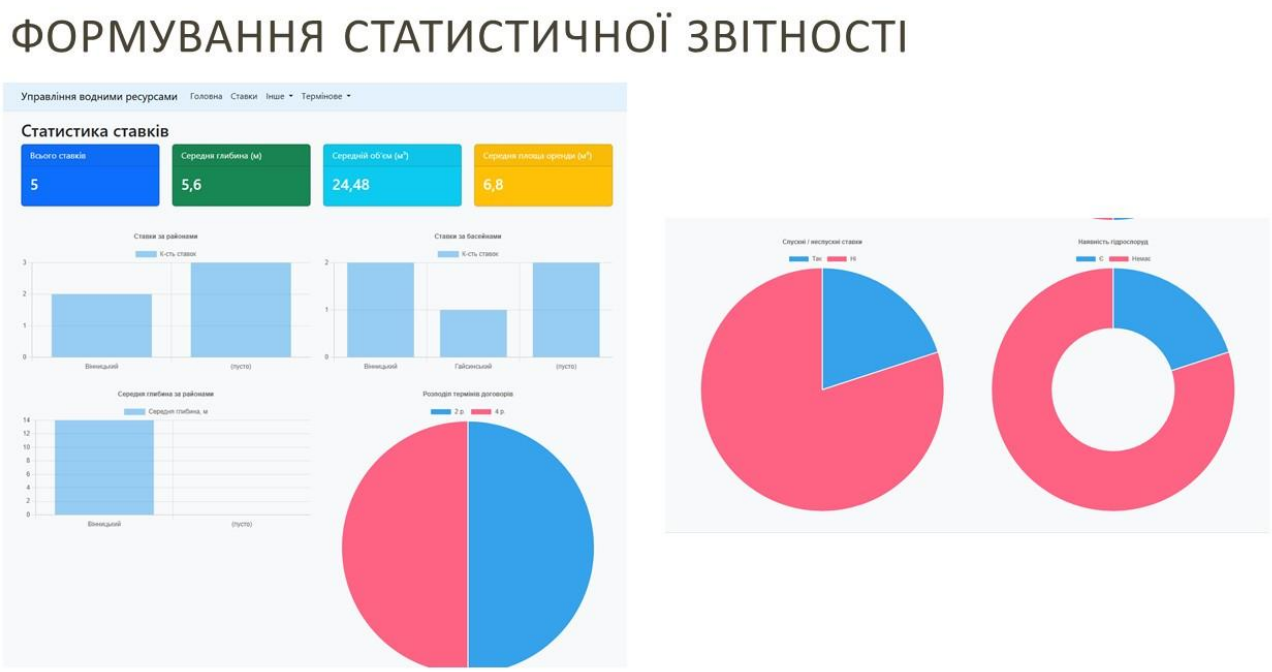


Рисунок Д.27 – Формування статистичної звітності

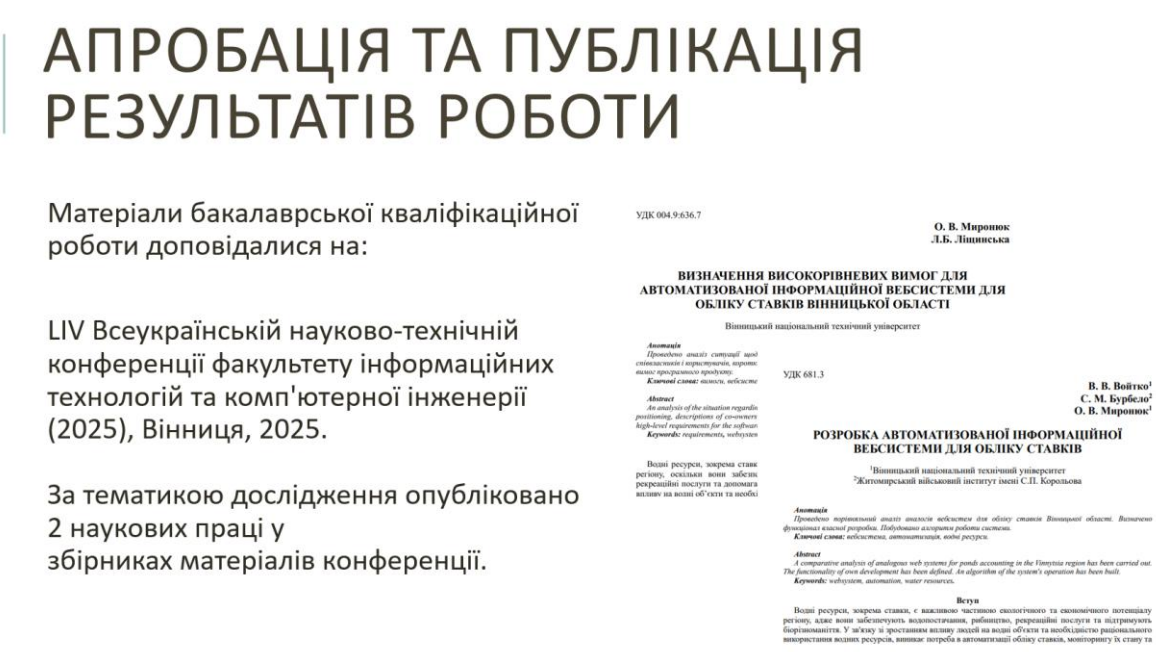


Рисунок Д.28 – Апробація та публікація результатів роботи

ВИСНОВКИ

- розроблено модель автоматизованої вебсистеми обліку ставків, яка міститиме основний функціонал системи;
- розроблено алгоритми для автоматизації збору та аналізу даних про ставки;
- створено механізми експорту даних у форматах PDF/Excel;
- реалізовано інтерактивної карти з використанням бібліотеки Leaflet.js;
- реалізовано функціонал нагадування через електронну пошту;
- розроблено модуль статистики
- інтегровано модулі у єдину систему з інтуїтивним інтерфейсом;
- здійснено тестування вебсистеми відповідно до поставлених задач.

Рисунок Д.29 – Висновки