504:002,004.45:[504.064(477):349.6(4-672 ЄС)]

КП 87.01.29

№ держреєстрації 0122U201790

Інв. №

МІНІСТЕРСТВО ЗАХИСТУ ДОВКІЛЛЯ ТА ПРИРОДНИХ РЕСУРСІВ УКРАЇНИ
НДУ "УКРАЇНСЬКИЙ НАУКОВИЙ ЦЕНТР ЕКОЛОГІЇ МОРЯ"
(УКРНЦЕМ)
65009, м. Одеса, Французький бульвар, 89. тел. (094) 9468721
e-mail: accem@te.net.ua, www.sea.gov.ua

ЗАТВЕРДЖУЮ
Виконуючий обов'язки директора УкрНЦЕМ, заступник директора з науки, канд. геогр. наук, старш. наук. співроб.
_____ Віктор КОМОРІН
«20» січня 2025 року

# ЗВІТ

## ПРО НАУКОВО-ДОСЛІДНУ РОБОТУ

## РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ ВИКОНАННЯ ЗАВДАНЬ МОРСЬКОЇ СТРАТЕГІЇ УКРАЇНИ У 2022-2024 РР.

Науковий керівник НДР
Виконуючий обов'язки директора
УкрНЦЕМ, заступник директора з
науки, канд. геогр. наук, с. н. с. _____ Віктор КОМОРІН

2024

Рукопис закінчено 26 грудня 2024 р.

Результати роботи розглянуто Вченою Радою УкрНЦЕМ,
протокол від 27 грудня 2024 № 9

# СПИСОК АВТОРІВ

| | | |
|---|---|---|
| Науковий керівник НДР<br>Виконуючий обов'язки директора<br>Українського наукового центру<br>екології моря, заступник директора з<br>наукової роботи,<br>канд. геогр. наук | "___" _____2024 | В. М. Коморін<br>(вступ;<br>висновки;) |
| Начальник відділу<br>управління екологічними даними | "26" 12 2024 | О. В. М'яснікова<br>(розділ 6) |
| Завідувач сектором<br>розробки інформаційних систем<br>відділу інформаційних систем | "26" 12 2024 | А. М. Круглов<br>(розділи 1, 2) |
| Завідувач сектором<br>геоінформаційного аналізу<br>інформаційних систем | "26" 12 2024 | О. В. Лепьошкін<br>(розділи 3, 4, 5) |
| Науковий співробітник<br>сектору геоінформаційного аналізу<br>відділу інформаційних систем | "26" 12 2024 | О. С. Братченко<br>(розділ 3) |

# РЕФЕРАТ

Звіт з НДР: 98 с., 3 табл., 7 рис., 5 джерел.

БАЗА ДАНИХ, ВЕБ-САЙТ, МОРСЬКА СТРАТЕГІЯ, ЕКОЛОГО-ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ, ІНТЕРАКТИВНЕ КАРТОГРАФУВАННЯ ІНТЕГРАЛЬНИХ ОЦІНОК СТАНУ МОРСЬКИХ ЕКОСИСТЕМ.

Актуальність науково-дослідної роботи обумовлюється, вперш за все необхідністю створення інформаційного забезпечення державного екологічного моніторингу морів України в межах створення та реалізації Морської стратегії України.

Метою НДР є удосконалення інформаційного забезпечення системи морського екологічного моніторингу в межах реалізації Морської стратегії України.

Основними завданнями НДР є:

1) забезпечення функціонування апаратно-програмного комплексу для виконання науково-практичних завдань;

2) розробка інтерфейсів баз даних (каталоги, аналітичні інструменти тощо);

3) оновлення та забезпечення функціонування інтерактивної картографічної системи для візуалізації та аналізу інтегральних оцінок стану морських екосистем відповідно до вимог Рамкової Директиви ЄС про морську стратегію;

4) забезпечення функціонування інформаційного веб-сайту УкрНЦЕМ;

5) поточна робота в межах діяльності вузла OBIS;

6) внесення даних в базу даних державного моніторингу морських вод Чорного та Азовського морів у 2022-2024 роках.

Об'єкт дослідження – інформаційне забезпечення екологічного моніторингу стану морського середовища Чорного моря в межах Морської стратегії України відповідно до вимог Директиви 2008/56/ЄС щодо морської стратегії.

ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних;

ВЕЗ – виключна морська економічна зони України;

ВРД - Водна Рамкова Директива;

ГІС – геоінформаційна система;

ДЕС – добрий екологічний стан

ДЕММ – державний екологічний морський моніторинг

МОК – міжурядова океанографічна комісія;

МООДІ – Міжнародний Обмін океанографічними Даними та інформацією;

НДР – науково-дослідна робота;

ПЗЧМ – північно-західна частина Чорного моря;

ПАВ – поліциклічні ароматичні вуглеводні;

ПХБ – поліхлорбіфеніли;

РДМС – Рамкова директива про морську стратегію;

ХОП – хлорорганічні пестициди ;

УкрНЦЕМ – Український науковий центр екології морів;

DCT – Data Collection Templates;

IODE – International Oceanographic Data and Information Exchange Program of IOC;

IOC – Intergovernmental Oceanographic Commission of UNESCO;

HEAT – HELCOM Eutrophication Assessment Tool;

HELCOM – Балтійська комісія з охорони морського навколишнього середовища;

MSFD – Рамкова Директива про морську стратегію;

IZI – Інтегральний індекс зоопланктону;

OBIS – Ocean Biogeographic Information System;

# ВСТУП

Україні, як морській державі, потрібно створити систему управління якістю морського середовища на сучасному міжнародному рівні. В ЄС діють інструменти, перевірені багатолітнім досвідом, такі як: Рамкова директива про морську стратегію (РДМС) - Директива 2008/56/ЄС Європейського парламенту та Ради Європи «Про встановлення рамок діяльності Співтовариства у сфері екологічної політки щодо морського середовища» від 17 червня 2008 року та Водна Рамкова Директива (ВРД) - Директива 2000/60/ЄС Європейського парламенту та Ради «Про встановлення рамок діяльності Співтовариства в галузі водної політики» від 23 жовтня 2000 року.

Відповідно до Угоди про асоціацію між Україною та ЄС Мінекоенерго з метою імплементації Директиви ЄС з морської стратегії необхідно здійснити заходи для визначення базового екологічного стану та статусу екосистем Чорного та Азовського морів в межах виключної морської економічної зони України (ВЕЗ), визначити та затвердити критерії доброго екологічного стану (ДЕС) для екосистем Чорного та Азовського морів в межах територіальних вод та ВЕЗ України, визначити природоохоронні цілі та індикатори, досягнення яких має забезпечити наближення екологічного стану та статусу екосистем Чорного та Азовського морів в цих межах до ДЕС [1], [2]. Все це повинно увійти до Морської стратегії України.

Українській науковий центр екології моря (УкрНЦЕМ), відповідно до ст. 11 РДМС [3], та на основі базової оцінки, здійсненої відповідно до частини 1 ст. 8, розробив програму екологічного моніторингу для постійної оцінки екологічного стану морських вод, базуючись на переліках характеристик, видів джерел та наслідків впливу, зазначених у Додатках III і V РДМС [4].

Програма державного екологічного моніторингу морів України, розроблялась з урахуванням орієнтирів розвитку України як морської держави і пов’язаного з цим процесу інтеграції до ЄС, що потребує поступового

впровадження загальноєвропейських стандартів і зокрема директив у сфері водної політики.

Програма державного екологічного моніторингу створена відповідно до Порядку здійснення державного моніторингу вод, який затверджено Постановою Кабінету Міністрів України від 19 вересня 2018 р. № 758 [5] на виконання вимог РДМС та МРД, які Україна зобов'язалася імплементувати в межах виконання Угоди про асоціацію між Україною та ЄС.

Актуальність науково-дослідної роботи (НДР) обумовлюється, перш за все необхідністю створення інформаційного забезпечення державного екологічного моніторингу морів України в межах створення та реалізації Морської стратегії України.

Метою НДР є удосконалення інформаційного забезпечення системи морського екологічного моніторингу в межах реалізації Морської стратегії України.

Основні завдання НДР:

- забезпечення функціонування апаратно-програмного комплексу для виконання науково-практичних завдань;
- розробка інтерфейсів баз даних;
- забезпечення функціонування вузла Біогеографічній Інформаційної Системи Океану (OBIS - Ocean Biogeographic Information System);
- оновлення та забезпечення функціонування інтерактивної картографічної системи для візуалізації та аналізу інтегральних оцінок стану морських екосистем відповідно до вимог рамкової директиви ЄС про морську стратегію;
- забезпечення функціонування інформаційного веб-сайту УкрНЦЕМ;
- внесення даних в базу даних державного моніторингу морських вод Чорного та Азовського морів у 2022-2024 роках.

Наукова робота здійснюється на базі попередніх розробок УкрНЦЕМ в межах національних та міжнародних програм. Дані, які були отримані

протягом попередніх років є функціональною основою для наступних розробок та модифікацій.

Робота буде використовуватися в інформаційній діяльності Морського інформаційно-аналітичного центру УкрНЦЕМ для забезпечення державних організацій щодо прийняття оперативних і стратегічних управлінських рішень у сфері охорони морського середовища і прибережної смуги України та регулювання морського і прибережного природокористування, імплементації РДМС до основних напрямків досліджень, а також для забезпечення потреб широкого кола наукових співробітників і громадськості в екологічній інформації. Термін виконання НДР: 2022-2024 рр.

# 1. ЗАБЕЗПЕЧЕННЯ ФУНКЦІОНУВАННЯ АПАРАТНО-ПРОГРАМНОГО КОМПЛЕКСУ ДЛЯ ВИКОНАННЯ НАУКОВО-ПРАКТИЧНИХ ЗАВДАНЬ

Для забезпечення функціонування апаратного програмного комплексу УкрНЦЕМ було проведено наступний перелік робіт:

1) оновлення програмного забезпечення для задовільнення вимог безпеки даних та інформаційного простору УкрНЦЕМ;

2) вдосконалення нової локальної мережі для більш надійного доступу до мережевих ресурсів співробітниками УкрНЦЕМ.

## 1.1 Оновлення програмного забезпечення для задовільнення вимог безпеки даних та інформаційного простору УкрНЦЕМ

Для удосконалення захисту інформаційного простору УкрНЦЕМ, як від зовнішніх так і від внутрішніх загроз, було оновлено всі сервери на операційній системі Ubuntu Server 22.04 LTS до Ubuntu Server 24.04 LTS.

Ця система привнесла багато виправлень помилок та вдосконалень, зокрема:

- ядро Linux 6.8 з функціями низької затримки, увімкненими за замовчуванням;
- вказівники фреймів увімкнені за замовчуванням для більшості пакетів на архітектурах x86;
- Rust 1.75, .NET 8 та OpenJDK 21 із сертифікацією TCK, а також інші оновлення інструментів;
- 64-бітне значення time_t за замовчуванням для armhf для вирішення проблеми 2038 року;

- обмеження непривілейованих просторів імен користувачів із примусовим використанням AppArmor за замовчуванням.

Крім великих оновлень програмного забезпечення проводилися роботи з оптимізації та покращення роботи інших серверів УкрНЦЕМ. Тобто, було модернізовано систему резервного копіювання “на ходу” - при внесенні будь-яких змін у головні бази даних УкрНЦЕМ, ці зміни в автоматичному режимі копіюються на сервер резервних копій, не чекаючи часу копіювання за розкладом. Крім того, було оновлено ПЗ всіх основних систем на серверному майданчику УкрНЦЕМ. Також було встановлено нове програмне забезпечення Map Server.

## 1.2 Вдосконалення нової локальної мережі для більш надійного доступу до мережевих ресурсів співробітниками УкрНЦЕМ

Протягом року проводилось розширення локальної мережі УкрНЦЕМ.

Користуючись новими можливостями у оновленому програмному забезпеченні, як головного керуючого маршрутизатора, так і другорядних маршрутизаторів, було вдосконалено доступ до бездротової мережі УкрНЦЕМ (Wi-Fi мережа), покращення стабільності роботи. Крім того, були проведені роботи для автоматичної зміни джерела живлення всієї серверної та мережевої інфраструктури УкрНЦЕМ – це дало змогу працювати із локальною мережею та серверами у разі відключень єлектроєнергії.

Також проводилося регулярне оновлення програмного забезпечення маршрутизаторів та роутерів в локальній мережі.

# 2. РОЗРОБКА ІНТЕРФЕЙСІВ БАЗ ДАНИХ (КАТАЛОГИ, АНАЛІТИЧНІ ІНСТРУМЕНТИ ТОЩО)

## 2.1 Оптимізація бази даних УкрНЦЕМ

Для подальшого розвитку та оптимізації даних, які зберігаються у базі данних УкрНЦЕМ був проведений ретельний аналіз можливих покращень. Першим покращенням була спроба змінити концепцію роботи самої бази даних, для чого була проведена робота по переведенню ключових таблиць у базі даних у формат In-Memory OLTP. Відмінності у роботі бази даних наведено у таблиці 1.

Таблиця 1 - Відмінності від звичайних дискових таблиць

| Характеристика | In-Memory таблиці | Звичайні таблиці |
|---|---|---|
| Зберігання даних | В оперативній пам'яті | На диску |
| Індекси | NONCLUSTERED HASH, NONCLUSTERED | CLUSTERED, NONCLUSTERED |
| Блокування | Немає блокувань (MVCC) | Блокування та очікування |
| Продуктивність | Висока для транзакцій | Залежить від дискових операцій |
| Обмеження | Немає FOREIGN KEY | Повна підтримка |
| Стійкість даних | SCHEMA_AND_DATA або SCHEMA_ONLY | Повна стійкість |
| Підтримка функцій | Обмежена | Повна |
| Масштабованість | Обмежена пам'яттю сервера | Обмежена дисковим простором |
| Типи індексів | Хеш та некластерні | Кластерні та некластерні |

In-Memory OLTP (Online Transaction Processing) — це технологія, яка дозволяє зберігати та обробляти таблиці безпосередньо в оперативній пам'яті для підвищення продуктивності транзакцій та аналітичних запитів. На відміну від традиційних дискових таблиць, In-Memory таблиці зберігаються та обробляються без звернення до дискової системи під час виконання операцій.

Основні особливості In-Memory таблиць:

- зберігання даних в оперативній пам'яті;
- два типи стійкості даних: SCHEMA_AND_DATA та SCHEMA_ONLY;
- спеціальні типи індексів (NONCLUSTERED HASH та NONCLUSTERED);
- відсутність блокувань і очікувань завдяки багатоверсійності (MVCC);
- обмежена підтримка тригерів та обмежень (без FOREIGN KEY);
- слабка сумісність із деякими функціями SQL Server.

Особливості всієї бази даних при використанні In-Memory OLTP:

- файлові групи для зберігання даних In-Memory;
- висока продуктивність за рахунок виключення операцій із диском;
- обмежена масштабованість, яка залежить від оперативної пам'яті;
- оптимізовані транзакції з використанням ATOMIC WITH;
- особливості резервного копіювання та відновлення.

Крім того, для підвищення продуктивності роботи бази з великими обсягами інформації та збільшення швидкості відповіді під час аналізу та виведення даних з пристроїв CTD або з інших, структура бази даних була змінена таким чином, щоб з'явилась можливість розподілу даних за типами — звичайні дані та дані CTD. Для цього була проведена модифікація структури всіх таблиць із метаінформацією для наявних дескрипторів. Усі новостворені дескриптори (такі як шум, птахи тощо) вже враховуватимуть ці зміни.

Крім цього, з урахуванням зберігання інформації за одним дескриптором у п'ятнадцяти таблицях у середньому (таблиці з даними, метаданими та таблиці-довідники), проведений аналіз даних у цих таблицях та визначено, які таблиці особливо сильно впливають на час виконання запитів. Після цього

інформацію з найбільш «повільних таблиць» об'єднано в одну супертаблицю, що значно прискорило вибірку даних (вибір даних із однієї таблиці здійснюється швидше, але такими даними складніше управляти).

Також були написани різноманітні тригери для каскадного автоматичного оновлення всіх даних, що знаходяться у «повільних таблицях» та супертаблиці, у разі їх зміни. Для прискорення занесення та обробки даних із CTD переписані всі наявні інструменти для внесення інформації у базу даних (парсери) з урахуванням нової структури бази даних («повільні таблиці» та супертаблиця) для кожного дескриптора.

Такі зміни дозволили  зберігати в базі даних великі обсяги інформації (наприклад, CTD, тощо) і не відчувати уповільнення роботи бази під час вибору та формування великих файлів із даними.

## 2.2 Розробка аналітичних інструментів для бази даних УкрНЦЕМ

Для зберігання та аналізу даних Південного океану, зібраних за допомогою різних приладів у вигляді трансект, виникла задача опрацювати та зберегти понад 10 мільйонів записів у базі даних.

Одночасне внесення такого великого обсягу даних стало справжнім викликом. Виявилося, що традиційні інструменти обробки інформації не справляються із завданням. Лінійні парсери, які були використані на ранніх етапах, демонстрували серйозні проблеми продуктивності. Процес обробки та завантаження одного великого масиву даних міг займати кілька годин, а іноді навіть призводив до збоїв через перевантаження серверів.

Щоб подолати ці труднощі, було розглянуто можливість використання технологій Big Data, зокрема таких рішень, як розподілена обробка даних і паралельні процеси завантаження. Застосування таких підходів дозволило розділити великі масиви даних на менші блоки та обробляти їх одночасно на

кількох вузлах обчислювальної системи. Це значно прискорило процес занесення інформації до бази даних і забезпечило стабільність роботи системи навіть під час пікових навантажень.

Особливості впроваджених технологій наступні:

− розподілена обробка (використання паралельних потоків для одночасного внесення даних із різних джерел);

− масштабованість (можливість додавання нових обчислювальних вузлів для збільшення продуктивності системи);

− оптимізація процесів парсингу (перехід від лінійних алгоритмів до багаторівневих підходів із попередньою перевіркою даних на коректність перед внесенням);

− забезпечення надійності (механізми автоматичного відновлення операцій у разі помилок або втрати з'єднання);

Після успішного впровадження нової архітектури з'явилися нові труднощі, пов’язані з аналізом такого великого обсягу інформації «на льоту». Запити до бази даних вимагали значних обчислювальних ресурсів, а час виконання запитів у деяких випадках був неприйнятно довгим. Це призводило до затримок у формуванні звітів та обробці запитів користувачів.

Для вирішення цієї проблеми було проведено масштабну оптимізацію структури бази даних. Замість традиційної лінійної архітектури таблиць була запропонована нова структура з використанням супертаблиць, що об'єднували найбільш критичні набори даних для швидкого доступу. Внесено правки у вже існуючі таблиці, де модифікували зв'язки між об'єктами, метаданими та довідниками. Розроблено тригери для автоматичного оновлення пов'язаних записів у разі змін даних у супертаблиці. Впроваджено індексацію даних та використання матеріалізованих подань для пришвидшення обробки складних аналітичних запитів.

Розроблений інтерфейс для роботи з базою даних перебуває на стадії активного тестування. Крім того, надано доступ до бази міжнародним партнерам, який був реалізований за допомогою додаткових механізмів

безпеки й авторизації, а також забезпечена стабільна робота під час високих навантажень.

Зрештою, проведені зміни дозволили підвищити продуктивність системи, забезпечити можливість роботи з великими обсягами інформації та створити платформу для подальшого розширення аналітичних можливостей.

# 3. ОНОВЛЕННЯ ТА ЗАБЕЗПЕЧЕННЯ ФУНКЦІОНУВАННЯ ІНТЕРАКТИВНОЇ КАРТОГРАФІЧНОЇ СИСТЕМИ ДЛЯ ВІЗУАЛІЗАЦІЇ ТА АНАЛІЗУ ІНТЕГРАЛЬНИХ ОЦІНОК СТАНУ МОРСЬКИХ ЕКОСИСТЕМ ВІДПОВІДНО ДО ВИМОГ РАМКОВОЇ ДИРЕКТИВИ ЄС ПРО МОРСЬКУ СТРАТЕГІЮ

Веб-карти є невід'ємною частиною сучасних веб-додатків, оскільки вони надають користувачу потужні інструменти для аналізу та взаємодії з географічними даними. Найбільш актуальні карти, що відображають дані в реальному часі, дозволяючи відслідковувати зміни та отримувати інформацію про різні географічні об'єкти. У рамках цієї НДР було поставлено завдання створити інтерактивну карту, що використовує бібліотеку OpenLayers для відображення даних, які надходять з сервера даних через протокол WFS (Web Feature Service). Використання WFS дозволяє підключати карти, які динамічно завантажують географічні дані у вигляді точкових, лінійних та полігональних обєктів з віддаленого сервера. Це дасть змогу працювати з актуальними даними без необхідності їх попереднього збереження в браузері користувача.

## 3.1 Мета створення та основні технології розробки картографічної системи

Кінцевою метою роботи було створення високоякісного інструменту для відображення та аналізу географічних даних. Це не просто карта, а інтерактивний веб-додаток, який дозволяє користувачеві працювати з різними шарами даних та отримувати детальну інформацію про географічні об'єкти. Важливим аспектом є можливість динамічного оновлення карти, що дозволяє користувачеві отримувати актуальну інформацію без необхідності оновлення всієї сторінки.

Ця робота включає кілька важливих аспектів. По-перше, необхідно реалізувати можливість відображення різних *шарів, згрупованих по темам, але користувачи можуть компонувати власний набір шарів , згідно з інтересами і потребами.* Ці шари можуть бути активовані або вимкнені залежно від уподобань користувача. Користувач має взаємодіяти з картою: обирати об'єкти та отримувати інформацію про них у спливаючих вікнах. Крім того, необхідно організувати динамічне оновлення легенди, яка відображатиме лише ті шари, які в даний момент видимі на карті. Процес розробки карти також включає створення зручного інтерфейсу для керування шарами. Користувач може вмикати та вимикати відображення кожного шару за допомогою простих елементів управління, таких як перемикачі (чекбокси) . Крім того, при вимкненні всіх шарів легенда повинна автоматично ховатися, щоб користувач не бачив порожню панель. Таким чином, інтерфейс залишатиметься зрозумілим та зручним, забезпечуючи максимальну наочність та функціональність.

Крім того, картографічна система спрямована на покращення користувацького досвіду через інтуїтивно зрозумілий інтерфейс та зручну взаємодію з картою та має містити кілька ключових функцій. По-перше, користувачі повинні мати можливість керувати видимістю шарів карти за допомогою простих елементів управління, таких як прапорці. Це дозволяє зробити інтерфейс карти гнучким та зручним. По-друге, повинна бути реалізована можливість відображення властивостей об'єктів карти при їх виборі. Для цього використовуються спливаючі вікна, які з'являються при клацанні на об'єкті. Ці вікна міститимуть інформацію про вибрані об'єкти, наприклад, їхню назву, опис та інші характеристики. По-третє, карта повинна містити динамічну легенду, яка відображатиме лише активні шари, що забезпечить користувачеві зручність у сприйнятті інформації. В рамках реалізації цієї задачі необхідно також врахувати оптимізацію роботи з картою. Це включає використання таких технологій, як кешування та завантаження даних за запитом (lazy loading), щоб не перевантажувати карту зайвими

даними та забезпечити швидке реагування на дії користувача. Все це має бути реалізовано з урахуванням зручності роботи на мобільних пристроях, що передбачає адаптивність інтерфейсу. Таким чином, ця інтерактивна система має на меті створення не лише функціональної, а й зручної карти, яка використовуватиме актуальні дані та забезпечуватиме комфортну взаємодію з користувачем.

Для реалізації цієї задачі використано кілька ключових технологій, кожна з яких виконує важливу роль у забезпеченні функціональності та взаємодії з картою. Однією з основних технологій є бібліотека OpenLayers, яка надає широкі можливості для роботи з картами. OpenLayers дозволяє створювати карти, відображати різні шари, а також інтегрувати дані з різних джерел. Ця бібліотека підтримує роботу з форматом GeoJSON, що дозволяє легко працювати з географічними даними, такими як точки, полігони та лінії. Ще однією важливою технологією є WFS (Web Feature Service), протокол, який надає доступ до географічних даних через інтернет. WFS дозволяє не тільки запитувати дані про географічні об'єкти, але й працювати з ними в реальному часі. У цьому випадку WFS використовується для отримання даних про різні шари карти, такі як море, суша та точки. Дані передаються у форматі GeoJSON, який є популярним форматом для роботи з географічними даними в JSON-структурі. Він дозволяє зручно представляти різні геометричні об'єкти, такі як точки, лінії та полігони, та використовувати ці дані для відображення на карті. Для створення інтерфейсу карти та взаємодії з користувачем використовуються стандартні веб-технології: HTML, CSS та JavaScript. HTML та CSS відповідають за структуру та зовнішній вигляд сторінки, а JavaScript — за логіку роботи з картою, обробку подій та взаємодію з даними. JavaScript дозволяє динамічно оновлювати вміст сторінки, наприклад, змінювати видимість шарів карти або відображати спливаючі вікна з інформацією про вибрані об'єкти. Також за допомогою JavaScript реалізовано елементи керування картою, такі як прапорці для включення та вимкнення шарів, а також кнопки для закриття спливаючих вікон.

## 3.2 Технологічний стек і архітектура системи

Система використовує кілька технологій для реалізації функціоналу веб-карти. Як основа для відображення карти обрана бібліотека OpenLayers, яка є однією з найбільш популярних і потужних бібліотек для роботи з картами у веб-додатках. Ця бібліотека підтримує різні типи карт і географічні дані, включаючи WFS, WMS, GeoJSON та інші формати. Використання OpenLayers дозволяє ефективно взаємодіяти з картографічними даними, а також надає широкі можливості для кастомізації та налаштування відображення шарів і об'єктів. Для роботи з географічними даними використовується формат GeoJSON, який є стандартом для обміну географічною інформацією у форматі JSON. Це дозволяє гнучко працювати з даними в JavaScript і легко інтегрувати їх в карту, отримуючи можливість відображати точки, лінії і полігони. Протокол WFS (Web Feature Service) застосовується для динамічного завантаження даних з сервера. За допомогою WFS можна запитати потрібні географічні об'єкти з віддаленого сервера в реальному часі, що дає змогу відображати актуальні дані без необхідності перезавантажувати карту чи оновлювати сторінку.

Сервер, на якому розміщено географічні дані, використовує технологію MapServer, яка дозволяє обслуговувати запити через WFS і інтегрувати дані в карти. MapServer — це потужна і гнучка система для керування картографічними даними і відображення карт, яка підтримує різні формати даних, включаючи Shapefiles, PostGIS і GeoTIFF. MapServer взаємодіє з OpenLayers через WFS-запити, надаючи географічні об'єкти для відображення на карті. Також у системі використовуються стандартні веб-технології для створення інтерфейсу: HTML, CSS і JavaScript. HTML і CSS забезпечують структуру і стилізацію сторінки, тоді як JavaScript відповідає за динамічне оновлення карти, взаємодію з користувацьким інтерфейсом і роботу з картографічними шарами.

Архітектура системи побудована з урахуванням модульності та розподілу обов'язків. Основні компоненти архітектури включають клієнтську частину (frontend) та серверну частину (backend). Клієнтська частина: Клієнтська частина відповідає за взаємодію з користувачем і відображення карти. Усі візуальні елементи (карта, шари, легенда, кнопки керування) керуються за допомогою JavaScript, а бібліотека OpenLayers використовується для відображення карти і роботи з картографічними даними. Компоненти інтерфейсу, такі як панелі керування для включення та вимкнення шарів, спливаючі вікна для відображення інформації про вибрані об'єкти, а також динамічна легенда, створюються з використанням стандартних веб-технологій. Основною задачею клієнтської частини є створення інтерактивної карти, яка буде реагувати на дії користувача, такі як кліки по об'єктах карти, перемикання видимості шарів і відображення інформації. Для цього карта інтегрується з сервером, який обробляє запити через WFS і надсилає географічні дані у форматі GeoJSON. Серверна частина: Серверна частина складається з MapServer, який обслуговує запити від клієнта. Він працює з картографічними даними, отриманими з різних джерел, таких як бази даних, файли формату Shapefile або інші географічні формати. MapServer обробляє запити WFS і повертає дані про географічні об'єкти у форматі GeoJSON. Крім того, серверна частина керує логікою роботи з картографічними шарами і надає динамічну інформацію про шари, такі як їх назви, стилі і атрибути, які будуть відображатися в легенді. Для цього MapServer взаємодіє з базою даних, що містить інформацію про різні шари карти (наприклад, шари моря, суші, точки), і передає цю інформацію клієнту у форматі, який можна обробити в OpenLayers. Взаємодія між клієнтом і сервером відбувається через запити WFS, які дозволяють запитувати дані про географічні об'єкти за запитом користувача. Сервер передає дані у форматі GeoJSON, які потім обробляються OpenLayers на клієнтській стороні і відображаються на карті.

## 3.3 Принцип роботи з картою

Основний принцип роботи з картою полягає в тому, щоб надати користувачеві інтуїтивно зрозумілий інтерфейс для взаємодії з картографічними даними. Карта відображає кілька шарів, які можуть бути включені або вимкнені за бажанням користувача. Кожен шар відображає певні географічні об'єкти, і може бути стилізований по-різному. Крім того, карта дозволяє користувачу отримувати інформацію про вибрані об'єкти. Для цього при кліку на об'єкт з'являється спливаюче вікно, в якому відображаються властивості цього об'єкта. Це може бути назва, опис, координати або інші дані, які можуть бути корисними для аналізу. Для зручності сприйняття карти на ній також відображається легенда, яка динамічно оновлюється залежно від того, які шари відображаються на карті. Якщо всі шари вимкнені, легенда ховається, а якщо хоча б один шар видимий, вона відображається з актуальною інформацією про включені шари. Таким чином, архітектура додатка та вибір технологій забезпечують ефективну взаємодію між клієнтом і сервером, а також дозволяють користувачу отримувати актуальну інформацію про географічні об'єкти в реальному часі.

HTML-структура сторінки є основою для організації та візуалізації всіх компонентів веб-додатка. У цій системі структура розроблена таким чином, щоб користувачі могли ефективно працювати з картою, перемикатися між шарами та отримувати інформацію про об'єкти. Ця продумана структура робить інтерфейс зручним, інтуїтивно зрозумілим і функціональним. У цьому розділі докладно описано всі складові елементи сторінки, їх функції та взаємозв'язок.

### 3.3.1 Основні розділи сторінки

Сторінка складається з кількох ключових компонентів, які відіграють важливу роль у функціональності додатка. Вони розділені на логічні блоки для покращення зручності роботи користувача.

Блок керування картою - цей блок розташовується в верхній частині сторінки і є панеллю, через яку користувачі можуть керувати відображенням різних шарів карти. Він містить перемикачі (чекбокси), кожен з яких пов'язаний з конкретним шаром (Рисунок 3.1):


Рисунок 3.1 – Блок керування картою

Кожен шар представлений на карті унікальним візуальним стилем, що допомагає користувачу їх розрізняти. Наприклад, шар моря може відображатися у синіх тонах, шар суші – у зелених, а точкові об′єкти – у вигляді червоних маркерів. Блок керування розроблений так, щоб користувач міг легко вмикати чи вимикати шари, не відволікаючись від основної роботи з картою. Особливу увагу приділено розташуванню елементів усередині блоку керування. Вони впорядковані вертикально, щоб мінімізувати займане місце, але при цьому залишатися легкодоступними. Фон панелі виконано в нейтральних відтінках, щоб вона не відволікала увагу від карти.

Основний блок карти - карта є центральним елементом сторінки і займає велику частину простору. Цей блок розроблений так, щоб користувачі могли зосередитися на аналізі географічної інформації. На карті відображаються всі шари, активовані користувачем через блок керування (Рисунок 3.2).

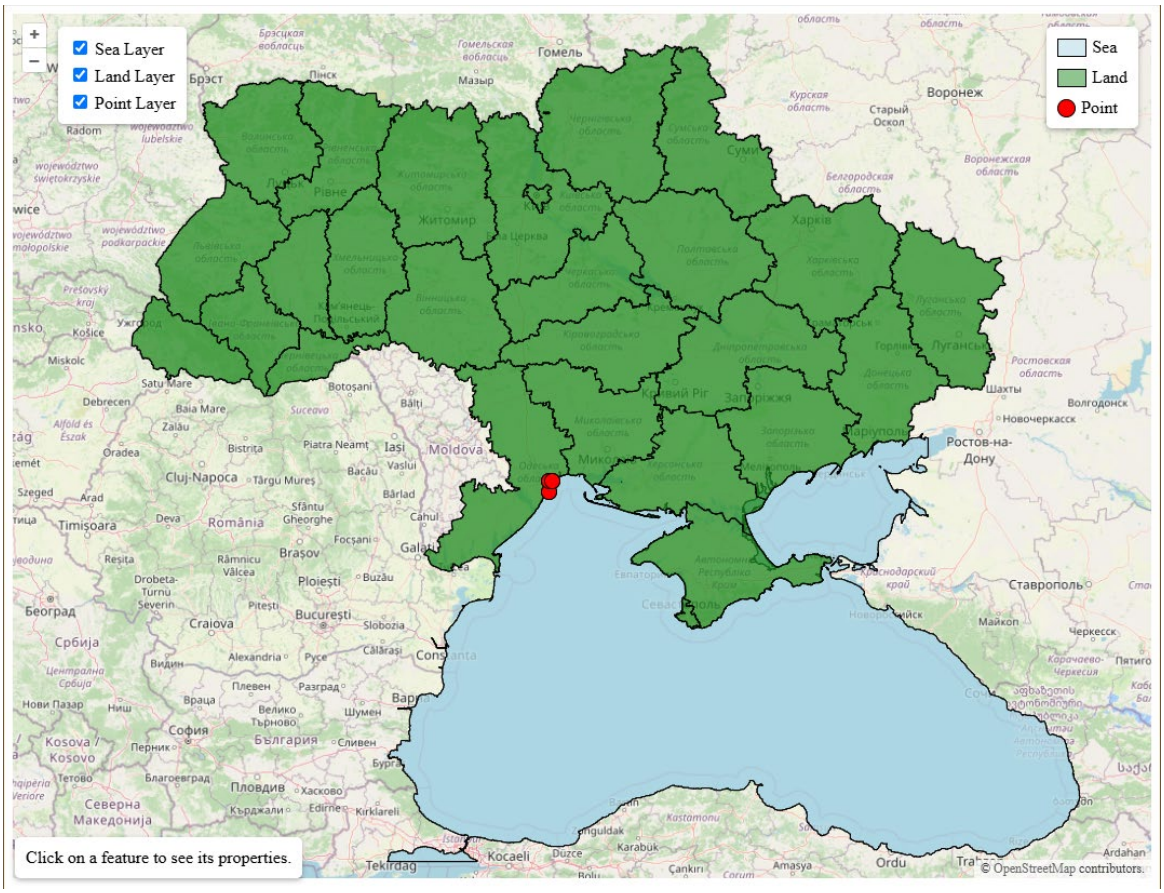


Рисунок 3.2 – Основний блок карти

Карта підтримує таки взаємодії, як масштабування, переміщення по карті та вибір об'єктів. Це дозволяє користувачам детально розглядати їхні області інтересу. Вона автоматично адаптується до розміру екрану, забезпечуючи оптимальне відображення як на десктопах, так і на мобільних пристроях.

Інформаційний блок - це область, де відображаються дані про об'єкти, виділені на карті. Наприклад, при кліку на точку або полігон, користувач може побачити назву об'єкта, його опис, координати та інші атрибути. Для зручності читання інформаційний блок виконано в світлих відтінках з м'якими тінями, що допомагає виділити його на фоні карти. Текст автоматично оновлюється при взаємодії з об'єктами на карті, надаючи користувачу актуальну інформацію (Рисунок 3.3).

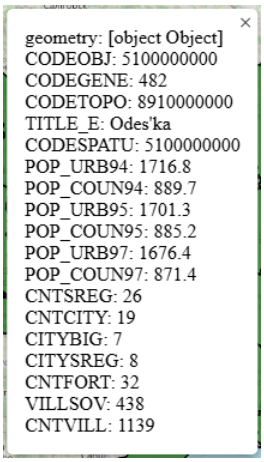


Рисунок 3.3 – Інформаційний блок

Легенда карти - це елемент, який допомагає користувачам зрозуміти, що означають різні символи та кольори на карті. Вона розташована в правому верхньому кутку сторінки та містить інформацію про шари, які активні в даний час. Легенда динамічно оновлюється: якщо користувач вимикає якийсь шар, він одразу зникає зі списку в легенді. Це спрощує сприйняття даних, оскільки користувач бачить тільки ту інформацію, яка йому справді потрібна. Кожен шар в легенді супроводжується візуальним маркером – кольоровим прямокутником (для полігонів), символом (для точкових об'єктів) або лінією (для лінійних об'єктів), який відповідає стилю відображення цього шару на карті (Рисунок 3.4).

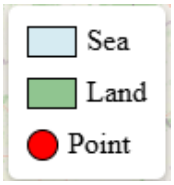


Рисунок 3.4 – Легенда карти

### 3.3.2 Організація  та стилізація елементів сторінки

Структура сторінки продумана так, щоб кожен елемент знаходився на своєму логічно обґрунтованому місці. Блок керування розташований в верхній

частині, оскільки він містить часто використовувані елементи інтерфейсу. Легенда розміщена в правому верхньому кутку, щоб бути видимою, але не заважати роботі з картою. Основний принцип організації – мінімізація перетину функціональних елементів. Це дозволяє користувачу комфортно взаємодіяти з картою, не відволікаючись на непотрібні деталі. Елементи інтерфейсу взаємопов'язані між собою: при зміні стану одного з них (наприклад, включення нового шару) автоматично оновлюються пов'язані компоненти, такі як легенда чи вміст карти.

Для забезпечення естетичного та функціонального інтерфейсу використовуються сучасні підходи до стилізації. Кожен елемент сторінки оформлений так, щоб виділятися візуально, але не відволікати від основного контенту – карти.

Карта займає центральне місце і відображається на нейтральному фоні, щоб усі елементи на ній були чітко видимі.

Блок керування виконаний у мінімалістичному стилі з використанням контрастних кольорів для тексту та фону, що полегшує взаємодію.

Легенда та інформаційний блок оформлені в одному стилі з блоком керування, що створює єдине візуальне сприйняття інтерфейсу.

Спливаюче вікно виділяється завдяки тіням і акуратному оформленню кнопки закриття.

Завдяки чіткому поділу функцій кожного елемента, інтерфейс додатка інтуїтивно зрозумілий і зручний для роботи. Усі блоки інтегровані в єдину систему, яка забезпечує динамічне оновлення даних і комфортну взаємодію з картою. Такий підхід дозволяє користувачам зосередитися на аналізі даних, не відволікаючись на технічні деталі інтерфейсу. Це особливо важливо при роботі з великими обсягами географічної інформації.

## 3.4 Етапи впровадження системи

Ініціалізація карти OpenLayers проходить кілька послідовних етапів, кожен з яких вносить свій вклад у створення повноцінного географічного інтерфейсу.

Етап 1 - Створення об'єкта карти.

Створення об'єкта карти - це перший крок, який визначає, де карта буде відображатися, які шари вона буде містити, і які параметри відображення будуть використовуватися.

Розміщення карти в інтерфейсі - карта відображається в певному HTML-елементі, який виступає як контейнер. Цей елемент зазвичай визначається за унікальним ідентифікатором, що дозволяє карті зайняти саме те місце на сторінці, де це передбачено дизайном. Наприклад, контейнер з ідентифікатором map. HTML-код сторінки має бути налаштований таким чином, щоб контейнер карти мав фіксовані розміри, що дозволить карті коректно відобразитися. Без коректної настройки розмірів контейнера карта може відображатися неправильно або зовсім не завантажуватися.

Додавання шарів карти - карта може містити кілька шарів, кожен з яких виконує свою функцію. Це можуть бути базові шари, такі як карти OpenStreetMap, а також векторні шари, які відображають об'єкти, такі як точки, лінії та полігони. Шари додаються в певному порядку, що впливає на їх видимість: спочатку додаються фонові шари, потім шари з даними. Це дозволяє відображати користувацькі дані поверх базового шару.

Визначення параметрів відображення - параметри відображення включають початковий центр карти і рівень масштабу. Ці параметри задаються з урахуванням області, яку користувач буде аналізувати, і необхідної деталізації. Центр карти вказується в географічних координатах (довгота і широта), які потім перетворюються в проекцію карти. Рівень

масштабу вибирається таким чином, щоб карта охоплювала потрібну область повністю, але при цьому зберігалась достатня деталізація.

Етап 2 - Підключення джерел даних.

Джерела даних є основою для відображення інформації на карті. Вони визначають, звідки карта бере дані про географічні об'єкти.

Типи джерел даних: в OpenLayers можна використовувати різні типи даних, включаючи векторні (точки, лінії, полігони) і растрові (наприклад, супутникові знімки). Для векторних даних часто застосовується формат GeoJSON, який дозволяє зберігати і передавати географічну інформацію в зручному текстовому форматі. Наприклад, GeoJSON-джерело може використовуватися для відображення об'єктів, таких як річки, будівлі або дороги, у вигляді точок, ліній чи полігонів.

Способи підключення даних: джерела даних можуть бути завантажені з локальних файлів або отримані через веб-сервіси. Наприклад, для динамічного оновлення карти використовуються запити до Web Feature Service (WFS), який надає дані в режимі реального часу. Підключення до WFS дозволяє працювати з великими наборами даних, завантажуючи лише ті елементи, які необхідні для поточного масштабу карти. Це зменшує обсяг переданої інформації та покращує продуктивність карти.

Кешування даних: для оптимізації роботи карти джерела даних можуть використовувати стратегії кешування, такі як завантаження даних за областю видимості (bounding box). Це означає, що дані завантажуються лише для тієї частини карти, яка відображається в даний момент, і оновлюються при переміщенні або зміні масштабу карти.

Етап 3. Налаштування шарів.

Після підключення джерел даних необхідно налаштувати шари карти.

Функції шарів: Кожен шар відповідає за відображення певного типу об'єктів. Наприклад, один шар може представляти водні об'єкти, інший – земельні ділянки, а третій – точки інтересу. Шари поділяються на базові (фонові) і користувацькі. Базові шари зазвичай використовуються для надання

контексту, такого як карта місцевості або супутникові знімки, тоді як користувацькі шари відображають дані, специфічні для додатку.

Стилізація шарів: Для кожного шару можна задати індивідуальний стиль. Наприклад, шар моря можна виділити заливкою блакитного кольору, сушу – зеленим, а точки інтересу – червоними маркерами. Стиль шару включає параметри заливки, обводки та форми об'єктів. Наприклад, точки можуть бути відображені у вигляді кіл заданого розміру, лінії – з певною товщиною та кольором, а полігони – з напівпрозорою заливкою і кордонами.

Логіка відображення шарів: Шари можуть бути незалежними один від одного або взаємодіяти. Наприклад, користувач може вмикати чи вимикати окремі шари для спрощення аналізу карти. Також можна налаштувати умовне відображення об'єктів, коли певні елементи відображаються тільки при виконанні заданих умов, таких як масштаб карти або значення атрибута об'єкта.

Етап 4. Налаштування вигляду карти

Відображення карти визначається її виглядом, який задає початкову область перегляду і рівень деталізації.

Центр карти: для визначення центру карти використовуються географічні координати (широта і довгота). Ці координати перетворюються в стандартну проекцію карти (EPSG:3857), щоб забезпечити коректне відображення. Центр карти вибирається таким чином, щоб охопити ключові географічні об'єкти, які користувач буде вивчати в першу чергу. Наприклад, якщо карта відображає певний регіон, центр задається на основі його географічних меж.

Масштаб карти: масштаб визначає, наскільки великим або малим буде зображення карти при її завантаженні. Рівень масштабування вибирається в залежності від завдань користувача – для вивчення великої області масштаб буде менший, а для детального аналізу – більший. Масштаб також впливає на видимість об'єктів: дрібніші деталі зазвичай ховаються на низьких рівнях масштабу, щоб карта залишалася читабельною.

Етап 5. Додавання додаткових елементів.

На завершальному етапі ініціалізації можна додати інтерактивні елементи, які зроблять роботу з картою зручнішою.

Такими є вспливаючі вікна та обробники подій.

Вспливаючі вікна використовуються для відображення додаткової інформації про об'єкти карти. При кліку на точку користувач може побачити її назву, опис і координати. Вспливаючі вікна прив'язуються до певних об'єктів на карті і автоматично змінюють своє положення при переміщенні карти.

За допомогою обробників подій до карти можна прив'язати події, такі як кліки, переміщення або зміни масштабу. Це дозволяє створювати інтерактивні функції, наприклад, відображення інформації про вибрані об'єкти, зміну шарів при певних умовах або запуск додаткових операцій, таких як побудова маршрутів.

## 3.5 Особливості налаштування джерел даних

Кожне джерело даних є основою для відображення об'єктів на карті.

Підтримувані формати даних: серед найбільш популярних форматів даних – GeoJSON, KML і WMS. Ці формати зручні для передачі та обробки географічної інформації. GeoJSON використовується для векторних даних, таких як точки або полігони, тоді як WMS підходить для растрових даних, таких як карти рельєфу або супутникові знімки.

Завантаження даних з сервера: джерела даних можуть бути підключені до зовнішніх серверів, де вони оновлюються в реальному часі. Це особливо корисно для динамічних карт, які відображають поточну інформацію, наприклад, дані про погоду або транспортні маршрути. Для стабільної роботи з серверами необхідно враховувати можливі обмеження на кількість запитів і проблеми сумісності, пов'язані з політикою CORS.

Обробка помилок: при роботі з зовнішніми джерелами даних можуть виникати помилки, наприклад, пов'язані з доступом або форматом даних. Тому важливо передбачити обробку таких ситуацій, щоб карта залишалася функціональною. Можна вивести повідомлення користувачу, якщо дані недоступні, або завантажити альтернативне джерело.

Функціональність карти: в результаті налаштування карта надає користувачеві зручний інтерфейс для роботи з географічними даними. Вона включає базовий шар для орієнтації, користувацькі шари для відображення специфічної інформації, а також інтерактивні елементи, такі як вспливаючі вікна і обробники подій. Користувач може змінювати масштаб, переміщатися по карті та взаємодіяти з об'єктами.

Гнучкість налаштування: завдяки модульній структурі OpenLayers, карта може бути доповнена новими функціями або змінена в залежності від вимог. Наприклад, можна додати додаткові шари, змінити стилі об'єктів, підключити нові джерела даних або інтегрувати карту з іншими системами, такими як бази даних або API.

Продуктивність і масштабованість: використання стратегій завантаження даних і оптимізація рендерингу дозволяють карті ефективно працювати навіть при великих обсягах інформації. Це особливо важливо для веб-додатків, де карта повинна залишатися responsive незалежно від пристрою користувача або швидкості з'єднання.

## 3.6 Інструменти та панелі управління

Карта може бути доповнена різними інструментами, які роблять роботу користувача зручнішою:

– панель управління шарами (дозволяє користувачеві вмикати та вимикати шари, змінювати їх порядок або налаштовувати параметри

відображення. Це особливо корисно, якщо карта містить багато шарів, і не всі з них мають бути відображені одночасно);

− інструменти вимірювання (Такі інструменти, як лінійка і площа, дозволяють вимірювати відстані між об'єктами і площу вибраних ділянок. Це особливо корисно для карт, що використовуються в аналітиці або плануванні);

− пошук об'єктів (Для зручності роботи можна додати рядок пошуку, який дозволяє знаходити об'єкти на карті за їх атрибутами. Наприклад, користувач може вводити назву об'єкта, і карта автоматично переміститься до його місцезнаходження).

## 3.7 Робота з WFS-джерелами

Робота з WFS-джерелами є важливим аспектом при інтеграції географічних даних у веб-додатки. На відміну від WMS (Web Map Service), який надає лише зображення карт, WFS (Web Feature Service) надає доступ безпосередньо до географічних об'єктів, таких як точки, лінії, полігони та інші геометричні форми. Це дає користувачеві можливість працювати з даними на більш глибокому рівні: аналізувати, редагувати та візуалізувати дані в реальному часі. Веб-сервіси WFS дозволяють інтегрувати не лише картографічну інформацію, а й фактичні дані про розташування об'єктів, що є невід'ємною частиною географічних інформаційних систем (ГІС). OpenLayers — це потужна бібліотека JavaScript, призначена для відображення карт і роботи з різними джерелами географічних даних, включаючи WFS. У цій статті розглядається використання WFS для динамічного отримання та відображення даних з сервера MapServer у форматі GeoJSON. Цей формат є особливо зручним для роботи в JavaScript і може бути легко інтегрований у веб-додатки. WFS-запити дозволяють отримати лише ті географічні дані, які

необхідні, і відображати їх на карті без зайвого завантаження даних, що підвищує продуктивність додатку.

WFS-запити представляють собою HTTP-запити, відправлені на сервер для отримання географічних даних у вигляді об'єктів. Ці об'єкти можуть бути представлені в різних форматах, і в даному прикладі використовується формат GeoJSON. Запити можуть містити різні параметри, які вказують серверу, які дані потрібно надати. Наприклад: • SERVICE=WFS — вказує на тип запиту, що виконується (у даному випадку запит WFS). • VERSION=1.1.0 — версія протоколу WFS, яка визначає структуру та можливості запиту. • REQUEST=GetFeature — тип запиту, який повідомляє серверу, що необхідно повернути географічні об'єкти. • TYPENAME=Sea — ім'я шару даних, який необхідно запитати (у цьому випадку шар даних про море). • CRSNAME=EPSG:3857 — система координат, у якій будуть повернуті дані (у цьому випадку використовується проекція Web Mercator, що підходить для веб-карт). • OUTPUTFORMAT=application/json — вказує, що дані повинні бути повернуті в форматі JSON, що зручно для подальшої обробки в JavaScript. Кожен параметр у запиті відіграє важливу роль у отриманні потрібних даних. Наприклад, параметр TYPENAME визначає, які саме дані будуть повернуті: це може бути шар про море, сушу, точки інтересу тощо.

В OpenLayers для роботи з WFS-джерелами використовується об'єкт ol.source.Vector. Цей об'єкт представляє собою джерело даних, до якого можна підключатися для отримання географічних об'єктів. При створенні такого джерела важливо вказати URL для запиту даних, а також інші параметри, які можуть бути використані для налаштування джерела. Одним із ключових параметрів є формат даних. У даному прикладі для роботи з GeoJSON використовується об'єкт ol.format.GeoJSON, який автоматично обробляє відповідь сервера та перетворює її в географічні об'єкти, такі як точки, лінії або полігони. Крім того, важливо вказати стратегію завантаження даних. В OpenLayers існує кілька варіантів завантаження даних:

− завантаження за областю видимості (bounding box) — це найбільш поширена стратегія. Вона дозволяє завантажувати дані лише для тієї частини карти, яка видна на екрані. Це ефективно працює при роботі з великими даними, оскільки сервер завантажує лише ті об'єкти, що знаходяться в межах видимої області карти;

− завантаження всіх даних — при цій стратегії завантажуються всі об'єкти шару, незалежно від того, видні вони на екрані чи ні. Це може бути корисно для невеликих наборів даних, але може негативно вплинути на продуктивність, якщо даних занадто багато;

− завантаження даних по тайлам — дані завантажуються по квадратним ділянкам карти, що може бути корисно при роботі з картами, розбитими на частини;

Крім того, при роботі з зовнішніми серверами важливо налаштувати параметр crossOrigin, щоб уникнути проблем з політикою CORS (Cross-Origin Resource Sharing), особливо якщо сервер і клієнт знаходяться на різних доменах.

Після того як джерело даних створено, його необхідно інтегрувати в шар карти. В OpenLayers для відображення векторних даних використовується об'єкт ol.layer.Vector, який дозволяє відображати дані, отримані з сервера, у вигляді об'єктів на карті. Кожен шар може бути налаштований з певним стилем для візуалізації даних. Стилі можуть бути різними і залежать від типу даних, що відображаються. Наприклад, для шару моря може бути застосований стиль з напівпрозорим заповненням і чорною межею, а для точок може бути використаний стиль з круглими символами, що будуть відображати важливі об'єкти на карті. Таким чином, можна налаштовувати зовнішній вигляд кожного шару залежно від уподобань користувача.

Стратегія завантаження даних відіграє важливу роль у роботі з WFS-джерелами. Стратегія завантаження за областю видимості (bounding box) є найбільш ефективною, оскільки мінімізує кількість даних, що передаються від сервера до клієнта. Це дозволяє пришвидшити завантаження карти та

забезпечити більш плавну взаємодію з користувачем. Однак для деяких випадків можуть бути корисні інші стратегії, такі як завантаження всіх даних або по тайлам. Важливо вибирати стратегію залежно від розміру та структури даних, щоб забезпечити найкращу продуктивність.

В деяких випадках потрібно працювати з кількома WFS-джерелами одночасно. Наприклад, якщо проєкт відображає кілька типів об'єктів (сушу, море та точки інтересу), кожен тип даних може бути отриманий через окремий WFS-запит. Для кожного запиту створюється окреме джерело даних, яке потім додається до свого шару. В OpenLayers можна легко працювати з кількома шарами, додаючи їх на карту та налаштовуючи їх стилі. Важливо, щоб кожен шар використовував своє власне джерело даних, щоб уникнути конфліктів та забезпечити коректне відображення всіх об'єктів.

Отримані дані можуть бути оброблені перед відображенням на карті. В OpenLayers є методи для роботи з векторними об'єктами. Наприклад, за допомогою методу getFeatures() можна отримати всі об'єкти, що знаходяться в джерелі даних, а за допомогою getFeatureById() можна витягнути конкретний об'єкт за його ідентифікатором. Ці об'єкти можна фільтрувати за різними параметрами, аналізувати їх властивості або використовувати для відображення додаткової інформації. Обробка даних дозволяє зробити карту більш інтерактивною, наприклад, при додаванні можливості фільтрувати дані, змінювати їх відображення або додавати анотації та інші елементи. Інтеграція WFS-джерел у веб-додаток за допомогою OpenLayers відкриває безліч можливостей для роботи з географічними даними. Користувачі можуть взаємодіяти з картою, переглядати та аналізувати дані, фільтрувати їх, а також отримувати доступ до даних в реальному часі. Це забезпечує гнучкість і динамічність картографічних додатків, дозволяючи працювати з великими обсягами даних та забезпечувати високу продуктивність при мінімальному часі завантаження. Використовуючи OpenLayers, можна ефективно інтегрувати та відображати дані з зовнішніх WFS-серверів, а також

застосовувати різні стилі та стратегії завантаження, щоб покращити користувацький досвід і продуктивність додатку.

## 3.8 Робота з векторними шарами та стилями в OpenLayers

Векторні шари є невід'ємною частиною відображення географічних об'єктів на інтерактивних картах. Вони дозволяють візуалізувати різні геометричні форми, такі як точки, лінії та полігони, використовуючи координати та інші географічні дані. У OpenLayers векторні шари грають ключову роль при відображенні даних, отриманих, наприклад, через WFS (Web Feature Service) або інші джерела, що робить їх гнучкими та потужними інструментами для розробки картографічних додатків.

Векторний шар в OpenLayers — це шар, який використовується для відображення векторних даних, таких як географічні об'єкти у вигляді точок, ліній і полігонів. Ці об'єкти можуть бути завантажені з різних джерел даних, таких як GeoJSON, WFS, KML та інші. Векторні шари не тільки забезпечують відображення даних, але й дають можливість взаємодіяти з картою — наприклад, додавати нові об'єкти або змінювати існуючі. Щоб створити векторний шар, необхідно використовувати конструктор ol.layer.Vector, який приймає параметри, такі як джерело даних та стилі відображення об'єктів. Ці параметри визначають, як саме буде відображатися інформація на карті. Важливим аспектом є здатність векторних шарів працювати з різними джерелами даних і адаптувати їх відображення залежно від обраного стилю.

Для того, щоб векторний шар відображав географічні дані, необхідно підключити джерело даних. В OpenLayers джерелами даних для векторних шарів можуть бути різні формати, включаючи GeoJSON, WFS та інші. Одним з поширених джерел є ol.source.Vector, який дозволяє завантажувати дані у форматі GeoJSON. Ці дані можуть бути отримані з віддаленого сервера,

наприклад, через WFS-сервіс. Коли дані завантажуються з сервера, в OpenLayers можна налаштувати параметри завантаження, такі як стратегія завантаження даних і крос-оригінальні налаштування для роботи з CORS (Cross-Origin Resource Sharing). Важливо зазначити, що OpenLayers підтримує стратегію завантаження даних по області видимості, що дозволяє завантажувати дані тільки для тієї частини карти, яка видна користувачеві. Це підвищує продуктивність і зменшує навантаження на сервер, особливо при роботі з великими обсягами даних.

У OpenLayers стиль об'єкта визначає, як виглядатиме географічний об'єкт на карті. Кожен об'єкт на векторному шарі може мати свій власний стиль, який включає такі параметри, як колір заливки, колір і товщина обвідки, форма символів і інші характеристики. Це дозволяє детально налаштовувати зовнішній вигляд карти та виділяти важливі об'єкти. Стилі можуть бути різними залежно від типу об'єкта. Наприклад, для полігонів використовується заливка і межа, для ліній — лише обвідка, а для точок — символи (наприклад, кола або зображення). В OpenLayers існують вбудовані класи для створення стилів: ol.style.Fill для заливки, ol.style.Stroke для обвідки та ol.style.Image для відображення точок.

Однією з сильних сторін OpenLayers є можливість задавати індивідуальні стилі для різних типів геометрій. Наприклад, для точок можна використовувати прості форми, такі як кола або квадрати, а для полігонів — більш складні форми з різними параметрами заливки та обвідки. Для точок, наприклад, можна використовувати стиль з колом, де задається радіус, колір заливки та колір обвідки. Для полігонів можна задати стиль з напівпрозорою заливкою і чіткою межею. Важливо, що для кожного типу геометрії можна налаштувати окремий стиль, що дозволяє створювати більш наочні та інформативні карти.

Після того, як стиль для об'єктів на векторному шарі визначено, його можна застосувати до всього шару. Це дозволяє уніфіковано керувати відображенням всіх об'єктів на цьому шарі, що спрощує налаштування карти.

Наприклад, для шару, що відображає дані про море, можна задати стиль, який буде застосовуватися до всіх об'єктів на цьому шарі, будь то різні моря, озера чи інші водойми. Це особливо корисно в ситуаціях, коли необхідно застосувати загальний стиль до всіх об'єктів, наприклад, зробити їх заливку напівпрозорою, а межу — чорною. У такому випадку можна заздалегідь підготувати стиль і застосувати його до всіх об'єктів на шарі.

OpenLayers надає можливість динамічно змінювати стилі об'єктів на шарі в залежності від різних умов. Наприклад, можна змінювати стиль об'єктів залежно від їх атрибутів, таких як тип об'єкта, категорія чи інші дані, пов'язані з географічним об'єктом. Для динамічного зміни стилю використовується функція styleFunction, яка дозволяє задати умови для зміни стилю кожного об'єкта. Це дає гнучкість при відображенні даних, особливо якщо потрібно виділити об'єкти з певними характеристиками, наприклад, змінити колір об'єкта залежно від його категорії або значень атрибутів.

Робота з векторними шарами та стилями в OpenLayers дає потужні інструменти для створення інтерактивних карт з налаштуванням візуалізації даних. Векторні шари дозволяють відображати географічні об'єкти різних типів, таких як точки, лінії та полігони, і дають можливість маніпулювати їх відображенням за допомогою гнучких стилів. Крім того, OpenLayers підтримує динамічне зміна стилів, що робить картографічні додатки ще більш адаптивними та інтерактивними. Використання стилів для різних типів об'єктів дозволяє створювати карти, які точно відображають дані, виділяють важливі елементи та роблять інформацію більш зрозумілою і доступною для користувачів. Ця можливість налаштування зовнішнього вигляду об'єктів і їх взаємодії з користувачем є ключовим елементом у побудові картографічних додатків з OpenLayers.

## 3.9 Легенда та управління шарами

Легенда на карті та управління шарами — це два важливі аспекти, які значно підвищують зручність та ефективність використання веб-карт. Легенда допомагає користувачу зрозуміти, які дані відображаються на карті, а також розшифровує символи, кольори та інші візуальні елементи, що використовуються для представлення об'єктів. В свою чергу, управління шарами дозволяє користувачу обирати, які шари карти будуть видимі, а які приховані, що дає гнучкість у відображенні різних даних.

Легенда карти виконує ключову роль у тлумаченні відображуваної інформації. Вона служить орієнтиром для користувача, який може не завжди розуміти, що означає той чи інший колір, символ або форма на карті. Наприклад, за допомогою легенди можна пояснити, що зелений колір на карті позначає ліси, а синій — водойми. Також у легенді можуть бути вказані типи об'єктів, такі як міста, дороги чи адміністративні межі, і відповідні їм символи та кольори. Веб-карти, що працюють з багатьма шарами даних, повинні забезпечувати динамічне оновлення легенди в залежності від того, які шари активовані або приховані. Динамічна легенда актуалізується в реальному часі, тобто кожного разу, коли користувач змінює видимість шару на карті, легенда автоматично оновлюється, показуючи лише ті елементи, що стосуються активних шарів. Наприклад, якщо користувач включає шар з морем, у легенді з'являється елемент, пов'язаний з цим шаром, наприклад, синій колір та опис "Морський шар". Якщо користувач вимикає цей шар, елемент зникає з легенди. Такий підхід робить інтерфейс карти більш зрозумілим і адаптованим до дій користувача. Легенда не лише відображає дані, а й взаємодіє з картою, оновлюючись залежно від того, які шари відображаються.

Управління шарами — це одна з основних функцій картографічних додатків, особливо у тих випадках, коли карта містить велику кількість різних типів даних. В OpenLayers управління шарами зазвичай реалізується через

елементи управління, такі як чекбокси або перемикачі, що дозволяють користувачу вмикати та вимикати видимість окремих шарів. Кожен шар може представляти різні типи даних, такі як географічні об'єкти, карти, зображення, а також спеціалізовані дані, наприклад, кліматичні карти, дані про землетруси або дорожню ситуацію. Чекбокси, наприклад, дають користувачу простий спосіб вибрати, які дані він хоче бачити на карті. Увімкнувши або вимкнувши певний чекбокс, користувач керує тим, який шар буде відображатися в даний момент. Це особливо корисно, коли потрібно контролювати відображення великого обсягу даних, як у випадку з багатошаровими картами, на яких є різні елементи — від топографічних карт до шарів з аналітичною інформацією. Коли користувач змінює стан *чекбокса* (наприклад, увімкнувши або вимкнувши шар), це автоматично відображається на видимості шару на карті. Крім того, часто зі зміною видимості шарів оновлюється й легенда, яка стає актуальною залежно від активних шарів. Такий механізм дозволяє користувачу завжди бачити лише ті дані, які йому потрібні, без зайвого перевантаження інформацією.

### 3.10 Вплив на відображення карти

Однією з ключових переваг управління шарами є можливість контролювати, які дані відображаються в кожен момент часу. Це має не лише практичне значення для користувача, а й сприяє поліпшенню продуктивності карти. У ситуаціях, коли на карті відображається велика кількість даних, можливість увімкнути або вимкнути шари дозволяє зосередитися лише на тих даних, які в даний момент цікавлять користувача. Наприклад, якщо карта відображає шари з даними про погодні умови, землетруси та завантаженість доріг, користувач може за своїм бажанням вибирати, який шар йому потрібен у даний момент, а який можна приховати. Це зменшує візуальне

перевантаження та спрощує сприйняття карти. Водночас приховування неактуальних шарів покращує продуктивність, оскільки карта не обробляє та не відображає зайві дані. Крім того, управління шарами важливе для користувачів, які працюють з різноманітною інформацією. Наприклад, у випадку з картами, призначеними для екологічного моніторингу, користувачі можуть бути зацікавлені лише в відображенні даних про рослинність та водойми, ігноруючи дорожню інфраструктуру та інші шари. Можливість вмикати та вимикати шари дозволяє точніше підлаштувати відображення карти під конкретні завдання, покращуючи як сприйняття даних, так і функціональність карти в цілому. Легенда та управління шарами відіграють важливу роль в організації взаємодії користувачів з картою. Легенда допомагає тлумачити символи, кольори та об'єкти, що відображаються на карті, роблячи інформацію більш зрозумілою. Управління шарами дозволяє користувачу гнучко налаштовувати відображувані дані, вмикаючи або приховуючи шари, що полегшує сприйняття карти та підвищує її функціональність. В OpenLayers ці можливості легко реалізуються за допомогою динамічних елементів управління, таких як перемикачі, що робить карту не тільки більш інформативною, але й зручною для використання в реальних додатках.

## 3.11 Обробка видимості та проблема управління видимістю шарів

Однією з центральних та найбільш корисних функцій картографічних веб-додатків є управління видимістю шарів. Це дає користувачу можливість контролювати, які дані або об'єкти будуть відображатися на карті в кожен момент часу, що значно збільшує гнучкість роботи з картою. Важливо розуміти, що карти, що містять велику кількість шарів і даних, можуть стати перевантаженими, якщо не передбачити ефективне управління видимістю.

Наприклад, на карті, що відображає інформацію про різноманітні природні об'єкти, будівлі, інфраструктуру, маршрути та інші елементи, можливість вмикати та вимикати шари дозволяє користувачу зосередитися тільки на тих даних, які актуальні для його аналізу чи завдання. Це робить карту більш зручною та ефективною для використання, а також підвищує користувацький досвід через спрощення візуалізації.

Складність роботи з картами, що містять велику кількість шарів, полягає в тому, що користувач може зіткнутися з труднощами при спробі одночасно сприймати занадто велику кількість інформації. Наприклад, якщо на карті відображаються всі доступні дані — географічні об'єкти, території, кордони, інфраструктура, транспортні маршрути та точки інтересу — карта може стати надмірно насиченою. Це ускладнює сприйняття та аналіз даних. У таких випадках управління видимістю шарів надає величезні можливості для зручної взаємодії. Користувачі можуть вмикати та вимикати ті чи інші шари в залежності від того, яку інформацію вони хочуть бачити на карті в даний момент. Це особливо важливо, коли необхідно працювати з картами, що містять величезні обсяги даних. Карти, на яких відображається сотні тисяч точок або полігонів, можуть стати дуже повільними і важкими для обробки, якщо всі шари активні одночасно. Вмикання та вимикання шарів дозволяє контролювати, які дані будуть відображатися, тим самим покращуючи продуктивність карти та пришвидшуючи її роботу. Вимкнення непотрібних шарів не тільки покращує сприйняття, але й знижує навантаження на ресурси пристрою, що призводить до пришвидшення обробки та рендерингу карти.

В OpenLayers управління видимістю шарів реалізується через метод setVisible(), який дозволяє змінювати стан видимості кожного шару карти. Коли цей метод викликається з параметром true, шар стає видимим на карті, і навпаки — якщо параметр дорівнює false, шар приховується. Таким чином, цей метод дає можливість контролювати, які дані відображаються на карті в кожен момент часу. Однак важливим аспектом є не лише можливість управляти видимістю шарів, але й забезпечити користувачу зручний та

інтуїтивно зрозумілий спосіб взаємодії з картою. Для покращення взаємодії часто використовуються елементи управління, такі як чекбокси, перемикачі або кнопки, з допомогою яких користувач може динамічно змінювати видимість шарів. Наприклад, користувач може увімкнути шар, що відображає географічні об'єкти, і вимкнути шар з маршрутами, щоб зосередитися на певній області карти. Таке управління видимістю дозволяє значно спростити сприйняття карти, роблячи інтерфейс більш зручним і гнучким для використання.

Одним з найважливіших елементів управління видимістю шарів є необхідність синхронізації легенди карти зі змінами в відображуваних даних. Легенда служить орієнтиром для користувачів, допомагаючи їм розуміти, які шари активні і які дані вони в даний момент бачать на карті. При зміні видимості шарів важливо, щоб легенда автоматично оновлювалась, відображаючи тільки ті елементи, які відповідають включеним шарам. Наприклад, якщо користувач увімкнув шар з природними зонами, то легенда повинна відобразити цей шар, використовуючи відповідні кольори та символи. Якщо шар з природними зонами був вимкнений, елемент в легенді повинен зникнути. Також варто звернути увагу на той факт, що якщо всі шари на карті приховані, легенда повинна автоматично приховуватись, щоб не створювати зайвого інформаційного шуму.

Ця динамічна синхронізація легенди з станом шарів покращує інтерфейс карти, роблячи його більш зрозумілим і інформативним для користувачів. Важливим моментом є те, що оновлення легенди має відбуватися в реальному часі, при кожній зміні видимості шару, без необхідності оновлення всієї сторінки. Це дозволяє користувачам миттєво бачити актуальний стан карти та оперативно оцінювати, які дані вони зараз бачать.

Щоб управління видимістю шарів було справді ефективним, важливо, щоб воно інтегрувалось з іншими елементами інтерфейсу та реакціями системи. В OpenLayers це можна реалізувати за допомогою обробників подій. Наприклад, щоразу, коли користувач змінює стан видимості певного шару

(наприклад, увімкнувши чи вимкнувши шар через чекбокс), цей процес автоматично ініціює виклик функції, яка змінить видимість шару на карті, а також оновить легенду та інші елементи інтерфейсу. Обробники подій дозволяють створити динамічне взаємодія з картою, коли зміни відбуваються без затримок і одразу відображаються на карті. Це може бути корисно в різних сценаріях: від простого вмикання/вимикання шарів до складніших взаємодій, коли видимість шарів змінюється в залежності від певних умов (наприклад, фільтри за типами об'єктів, територією чи часом).

Динамічне управління видимістю шарів має кілька ключових переваг. По-перше, воно дозволяє користувачам з легкістю контролювати, які дані відображаються на карті, що значно покращує сприйняття карти. Користувачі можуть в будь-який момент вмикати або вимикати шари, щоб адаптувати карту під свої потреби та цілі аналізу. Це створює більш зручний та персоналізований досвід роботи з картою. По-друге, управління видимістю шарів сприяє значному покращенню продуктивності картографічного додатку. Вимкнення непотрібних шарів зменшує обсяг даних, які потрібно обробити та відобразити на карті. Це особливо важливо, коли йдеться про карти з великим обсягом даних, такі як карти з мільйонами об'єктів. Зменшення числа відображених шарів дозволяє покращити час відгуку карти та скоротити час рендерингу. По-третє, динамічне управління видимістю шарів підвищує адаптивність інтерфейсу. Легенда та інші елементи управління можуть змінюватися в залежності від стану шарів, що дозволяє зробити карту більш гнучкою та інформативною. Це дає змогу користувачу швидко адаптувати карту під свої потреби, використовуючи лише ті шари, які важливі в даний момент часу. Такий підхід робить інтерфейс карти більш інтерактивним та інтуїтивно зрозумілим. Обробка видимості шарів є невід'ємною частиною роботи з картами в OpenLayers, оскільки вона значно покращує взаємодію користувачів з картою та підвищує її ефективність. Можливість динамічно управляти видимістю шарів дозволяє користувачам адаптувати карту під свої потреби, покращуючи сприйняття даних та

підвищуючи продуктивність карти. Синхронізація змін видимості з іншими елементами інтерфейсу, такими як легенда, допомагає створити більш зручний та гнучкий інтерфейс, що робить взаємодію з картою більш інформативною та зрозумілою. Цей підхід є важливим кроком у створенні високоякісних картографічних додатків, які можуть ефективно обробляти великі обсяги даних та надавати користувачеві повний контроль над відображенням інформації. В кінцевому підсумку, обробка видимості шарів допомагає зробити карти не тільки зручними для користувачів, але й більш продуктивними, що особливо важливо для карт, що містять складну або багаторівневу інформацію.

## 3.12 Проектування інтерфейсу користувача

Проектування інтерфейсу користувача (UI) для картографічних веб-застосунків є важливим завданням, оскільки він безпосередньо впливає на користувацький досвід (UX) і ефективність взаємодії з картою. У випадку з OpenLayers та іншими картографічними бібліотеками інтерфейс має бути не лише естетично привабливим, але й функціональним, зручним для користувача, забезпечуючи доступ до всіх необхідних інструментів для роботи з картою. Процес проєктування інтерфейсу для картографічного застосунку охоплює багато аспектів: від основного макету сторінки до більш детальних компонентів управління картою та відображуваними даними. Розглянемо ключові принципи і елементи, які повинні бути враховані при розробці інтерфейсу для картографічних застосунків. Проєктування інтерфейсу для картографічного веб-застосунку має кілька цілей, кожна з яких важлива для покращення взаємодії з користувачем:

1. Зручність використання: Інтерфейс має бути максимально інтуїтивно зрозумілим. Незалежно від рівня підготовки користувача,

інтерфейс має дозволяти легко орієнтуватися в функціоналі застосунку, керувати картою, вмикати та вимикати шари і взаємодіяти з об'єктами на карті.

2. Ефективність: Час відгуку та швидкість роботи застосунку є важливими аспектами для ефективної роботи з картою. Інструменти управління шарами та інші елементи інтерфейсу повинні бути швидко доступними та працювати без затримок.

3. Інтуїтивність: Елементи інтерфейсу мають бути розміщені так, щоб їх використання було природним. Розміщення елементів управління, таких як кнопки та перемикачі, повинно бути логічним і відповідати звичним очікуванням користувача.

4. Адаптивність: Важливо, щоб інтерфейс коректно відображався на різних пристроях, включаючи мобільні телефони, планшети та настільні комп'ютери. Для цього необхідно використовувати адаптивний дизайн, що дозволяє елементам інтерфейсу змінювати розміри та розташування залежно від розміру екрану.

### 3.12.1 Оптимізація продуктивності завантаження та оптимізація рендерингу карти

Оптимізація картографічного веб-застосунку є важливим етапом розробки, який спрямований на покращення продуктивності, зменшення часу завантаження та підвищення зручності використання. У випадку з картами, особливо при використанні таких бібліотек, як OpenLayers, оптимізація стосується як серверної частини (наприклад, запитів до даних і їх завантаження), так і клієнтської (обробка карти, рендеринг об'єктів та взаємодія з користувачем). У цьому розділі розглядаються методи оптимізації та покращення функціоналу веб-карт, які допомагають покращити користувацький досвід і прискорити роботу застосунку.

Одним з важливих аспектів при роботі з картографічними застосунками є швидкість завантаження та рендерингу карти. Складні карти з великою кількістю шарів або даних можуть значно уповільнити роботу застосунку. Для вирішення цієї проблеми можна використовувати кілька підходів:

− Використання кешування: кешування даних, отриманих з сервера, допомагає уникнути багатократних запитів за тими ж самими даними. Це може суттєво прискорити час відгуку, оскільки повторні запити можуть бути обслуговані з кешу, а не завантажуватися з сервера. У випадку з WFS-даними, можна налаштувати кешування на сервері або в браузері, щоб зменшити кількість запитів.

− Підвантаження даних по мірі необхідності (Lazy Loading): замість того, щоб завантажувати всі дані одразу, можна завантажувати їх по мірі необхідності, залежно від того, яка область карти видна користувачеві. Наприклад, дані про точки, полігони або інші об'єкти завантажуються тільки для видимої області карти. Це допомагає зменшити обсяг даних, що завантажуються за один раз, і прискорює роботу карти. В OpenLayers для цього можна використовувати стратегію підвантаження за межами екрану (ol.loadingstrategy.bbox), яка автоматично запитує дані тільки для видимої області карти.

− Використання Tile-сетів: замість завантаження всіх даних у вигляді одного великого шару (наприклад, GeoJSON) можна використовувати тайли (плитки), які є розділеними на маленькі частини зображення або векторні дані. Це дозволяє ефективніше завантажувати карту, оскільки кожен тайл завантажується по мірі необхідності. OpenLayers підтримує роботу з тайлами, включаючи підтримку різних типів тайлів (наприклад, растрових або векторних).

− Оптимізація WFS-запитів: коли працюєш з WFS, можна оптимізувати запити до сервера, щоб не запитувати зайві дані. Наприклад, можна вказати в запиті обмеження по географічним координатам або фільтри

для отримання тільки необхідних об'єктів. Це допоможе зменшити навантаження на сервер і прискорить отримання даних.

Процес рендерингу карти та всіх об'єктів на ній може бути досить ресурсоємним, особливо при великій кількості даних. Для покращення продуктивності рендерингу можна використовувати кілька методів:

1. Використання спрощення геометрії (Simplification): Для великих геометричних об'єктів, таких як полігони або лінії, можна використовувати методи спрощення геометрії. Це зменшить кількість точок в об'єкті, що знизить навантаження на рендеринг. OpenLayers підтримує різні способи спрощення, включаючи використання спеціалізованих бібліотек для роботи з геометріями.

2. Використання кластеризації об'єктів: Коли на карті відображається велика кількість об'єктів (наприклад, точок), можна використовувати кластеризацію для групування точок в кластери при великому масштабі. Це допоможе зменшити кількість відображуваних елементів, покращуючи продуктивність карти. В OpenLayers для цього використовується клас ol.layer.Vector з кластеризацією.

3. Оптимізація стилів об'єктів: Стилі об'єктів, такі як точки, лінії та полігони, можуть також впливати на продуктивність. Використання занадто складних стилів (наприклад, з великою кількістю ефектів, прозорості або градієнтів) може сповільнити рендеринг. Варто уникати надмірно складних стилів і використовувати простіші та швидші рішення для стилізації об'єктів.

4. Використання WebGL для рендерингу: OpenLayers підтримує рендеринг з використанням WebGL, що значно прискорює роботу з картою, особливо при відображенні великої кількості об'єктів. WebGL дозволяє рендерити графіку через апаратне прискорення, що пришвидшує процес рендерингу і підвищує продуктивність, особливо на мобільних пристроях.

### 3.12.2 Усунення проблем з продуктивністю

Проблеми з продуктивністю можуть виникати через надмірне використання ресурсів браузера або через помилки в коді. Щоб усунути такі проблеми, слід:

1. Профілювати продуктивність: для виявлення проблем з продуктивністю можна використовувати інструменти профілювання, такі як вбудовані інструменти розробника в браузері (наприклад, Chrome DevTools). З їх допомогою можна відстежувати час завантаження сторінки, час рендерингу та інші параметри, що стосуються продуктивності;
2. Мінімізувати рендеринг об'єктів - коли користувач не взаємодіє з картою або не дивиться на певну область, не слід рендерити об'єкти на цій області. OpenLayers надає механізми для динамічного оновлення видимих об'єктів, що дозволяє знижувати навантаження на рендеринг в моменти, коли дані не видимі;
3. Використовувати віртуалізацію у випадках, коли відображається дуже велика кількість даних (наприклад, безліч точок або полігонів). Віртуалізація даних дозволяє рендерити лише ті об'єкти, які знаходяться в області видимості, і завантажувати нові дані по мірі прокручування карти.

### 3.12.3 Покращення користувацького інтерфейсу

Окрім оптимізації продуктивності карти, важливим аспектом є покращення взаємодії користувача з картою, а також зручності роботи з інтерфейсом. Для цього можна використовувати такі підходи:

− Адаптивний дизайн: Інтерфейс карти повинен бути адаптивним, щоб коректно відображатися на різних пристроях. Особливо важливо, щоб

елементи керування картою (наприклад, кнопки збільшення/зменшення масштабу, панель керування шарами) були зручні для використання як на мобільних пристроях, так і на настільних ПК.

− Додавання інструментів для аналізу даних: Карти можуть бути не лише інструментами для візуалізації, а й для аналізу даних. Можна додати додаткові інструменти, такі як вимірювання відстаней, площ, створення позначок та інших інтерактивних елементів. Ці функції допоможуть користувачам ефективно працювати з картою.

− Оптимізація спливаючих вікон: Спливаючі вікна повинні бути інформативними та не перевантажувати інтерфейс. Можна додати функції для покращення роботи з ними, наприклад, можливість закріплення вікна, зміна його розміру або можливість приховування вікна, коли воно не потрібне.

− Зворотний зв'язок від користувача: Важливо включити механізми отримання зворотного зв'язку від користувачів, щоб покращити інтерфейс і функціональність карти. Це може бути через форми зворотного зв'язку, анкети або можливість користувацького вводу (наприклад, додавання нових об'єктів на карту).

Оптимізація та покращення картографічного веб-застосунку — це процес, який включає як технічні аспекти, так і покращення взаємодії з користувачем. Використання правильних методів оптимізації даних, рендерингу та взаємодії з картою допомагає створити більш швидкі та зручні застосунки. Покращення інтерфейсу, адаптивність та зворотний зв'язок з користувачем також відіграють ключову роль у успішній розробці картографічних рішень.

## 3.13 Підсумки

Розробка картографічних веб-застосунків з використанням бібліотеки OpenLayers охоплює важливі аспекти роботи з картами, включаючи інтеграцію з WFS-сервісами, управління векторними шарами та створення інтуїтивно зрозумілих інтерфейсів. Застосування OpenLayers дозволяє створювати потужні та гнучкі рішення для різних сфер, від географічного аналізу до динамічних карт для користувачів.

Один з ключових аспектів успішної реалізації — це робота з даними в реальному часі. В роботі використано інтеграцію з WFS-сервісами, що дозволяє завантажувати та відображати геопросторові дані з віддалених серверів. Це підвищує гнучкість застосунку, дозволяючи користувачеві отримувати актуальні дані за запитом.

Робота з векторними шарами та стилями в OpenLayers дає можливість детально налаштувати зовнішній вигляд карти, виокремлюючи важливі об'єкти та створюючи інтуїтивно зрозумілі візуальні представлення даних. Векторні шари дозволяють відображати різні типи географічних об'єктів, що розширює функціональність карти та робить представлення даних більш різноманітним.

Інтерактивність карти — важлива складова сучасного картографічного застосунку. Можливість взаємодіяти з картою, отримувати інформацію про об'єкти в реальному часі через спливаючі вікна, покращує користувацький досвід і робить інтерфейс більш персоналізованим.

Управління видимістю шарів допомагає користувачеві контролювати відображувані дані, що особливо важливо при роботі з великими обсягами інформації. Зручні перемикачі шарів в інтерфейсі сприяють покращенню взаємодії з картою.

Оптимізація карти — це не тільки підвищення продуктивності, але й покращення зручності використання. Методи, такі як підвантаження даних за

потребою, кешування та спрощення геометрій, сприяють підвищенню швидкості завантаження та чутливості карти, особливо при роботі з великими обсягами даних.

На завершення, створення картографічних застосунків з використанням OpenLayers вимагає комплексного підходу, що включає роботу з даними, створення візуальних елементів, забезпечення інтерактивності та оптимізацію продуктивності. Успішна реалізація цих аспектів дозволяє створювати ефективні та зручні карти для різних сфер діяльності, від географічного аналізу до інформаційних систем для користувачів.

# 4. ЗАБЕЗПЕЧЕННЯ ФУНКЦІОНУВАННЯ ІНФОРМАЦІЙНОГО ВЕБ-САЙТУ УКРНЦЕМ

Метою виконаної роботи є підтримка веб-сайту УкрНЦЕМ після його введення в експлуатацію, а також забезпечення інформаційної безпеки та зручності користування для кінцевого користувача. Новий веб-сайт має гідно представляти нашу організацію як на національному рівні, так і в міжнародній спільноті.

## 4.1 Забезпечення безпеки оновленого веб-сайту

Одним з пріоритетних напрямків під час розробки сайту є забезпечення інформаційної та технічної безпеки. Виконану роботу в цьому напрямку можна поділити на наступні категорії:

− оновлення системи керування вмістом (CMS) WordPress останньої актуальної версії 6.5.2, шаблону Avada Design Tool версії 6.4, допоміжних модулів. Підтримка актуальних версій усіх модулів веб-сайту дозволяє забезпечити захист від програмних вразливостей, що виявляють самі розробники та користувачі. Ці дії необхідні, оскільки через програмні вразливості здійснюється більша частина усіх хакерських атак;

− забезпечення резервного копіювання файлів веб-сайту, бази даних та веб-сервісів. Резервне копіювання проходить в автоматичному режимі, усі дані копіюються на 2 різних сервери (основний та сервер резервних копій) 2-3 рази на тиждень, в залежності від типу даних. Базове резервне копіювання відбувається на основний сервер, що знаходиться під управлінням операційної системи (ОС) Microsoft Windows Server, після чого дублюється на спеціальний сервер резервних копій, на якому встановлено ОС на базі Linux». Крім того,

усі резервні копії зберігаються у декількох версіях, з різною датою створення. Таким чином, навіть якщо до резервної копії потраплять данні, що вже піддалися зараженню зловмисниками, все одно, зберігається більш стара, але не заражена копія даних. Ці заходи дозволяють відновити роботу системи у випадку її пошкодження або взламу;

− забезпечення програмного захисту на стороні серверу. Заходи, що включають у себе налаштування політики безпеки в залежності від користувача та його групи, закриття доступу до системних файлів із зовнішньої мережі – спрямовані на закриття можливих шляхів несанкціонованого доступу до веб-сайту (файлів та БД).

## 4.2 Удосконалення елементів дизайну та сторонніх модулів

За основу для побудови дизайну використовується набір модулів, шаблонів та інструментів під назвою «Avada Design Tool», що розробляється та підтримується компанією «Theme Fusion». Модулі, що входять до складу цього набору, дозволяють швидко та зручно організувати структуру та дизайн веб-сайту. Головна сторінка нового веб-сайту УкрНЦЕМ відображена на рисунку 4.1.

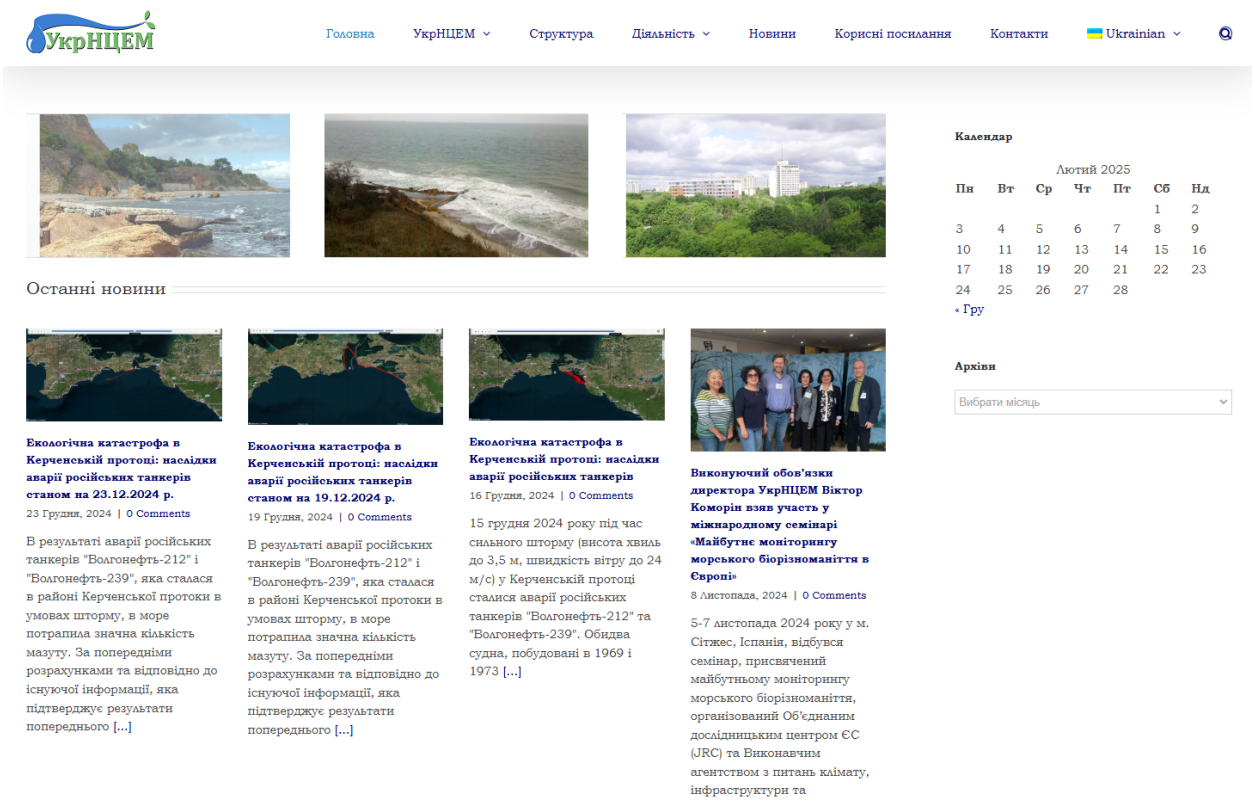


Рисунок 4.1 – Головна сторінка нового веб-сайту УкрНЦЕМ

## 4.3 Наповнення веб-сайту інформацією

Веб-сайт та його функціонал постійно оновлюється, проведено його наповнення новим або актуалізованим контентом, а саме:

- додано новини починаючи з початку 2024 року (29 новин);
- оновлено розділ «Документи»;
- додано інформацію в розділ «Бази даних»;
- додано інформацію в розділ «ГІС»;
- додано інформацію в розділ «Проекти»;
- додано фотографії до галереї, що відображається на головній сторінці веб-сайту.

# 5. ПОТОЧНА РОБОТА В МЕЖАХ ДІЯЛЬНОСТІ ВУЗЛА OBIS

За звітний період виконані наступні задачі та проведені заходи:

Проведено поточне оновлення до останньої актуальної версії програмного забезпечення обміну даних Integrated Publishing Toolkit (IPT - http://gp.sea.gov.ua:8082/ipt/);

Проведено оновленя існуючих наборів даних з урахукванням останніх вимог до зберігання даних у Integrated Publishing Toolkit (IPT) УкрНЦЕМ з подальшою новою публікацією цих наборів даних в мережах OBIS та EMODNET.

У 2024 році УкрНЦЕМ продовжував виконнаня своїх обов’зяків у фазі №5 проекту.

# 6. ВНЕСЕННЯ ДАНИХ В БАЗУ ДАНИХ ДЕРЖАВНОГО МОНІТОРИНГУ МОРСЬКИХ ВОД ЧОРНОГО ТА АЗОВСЬКОГО МОРІВ У 2022-2024 РОКАХ

Метою роботи та основним завданням є розробка системи збору, поповнення, критконтролю і завантаження до БД «SeaBase» даних за програмою морського прибережного моніторингу.

## 6.1 Роботи, проведені з даними у 2022 році

Протягом звітного періоду були виконані наступні роботи:

1. Аналіз отриманих УкрНЦЕМ даних за програмою моніторингу у 2021 році для подальшого внесення у БД «SeaBase»:
   - перевірка координат станцій та створення карт рейсів;
   - перевірка горизонтів та глибин станцій;
   - корегування одиниць вимірювання для забруднюючих речовин у воді, донних відкладах та біоті;
   - перевірка дат станцій моніторингу.
2. Таблиця Platform доповнена даними з дванадцяти станцій, отриманих в двох рейсах, що виконувалися в травні та жовтні 2021 року. Рейси виконувалися в рамках НДР «Контрольні моніторингові спостереження в процесі експлуатації глибоководного суднового ходу Дунай-Чорне море (морська частина)» спільно з Інститутом морської біології НАН України;
3. Внесено 124 станції Прибережного моніторингу 2021 року у таблицю Stations бази даних УкрНЦЕМ «SeaBase».

4. Внесенні дані екологічного моніторингу Чорного моря УкрНЦЕМ для 17 параметрів гідрології та гідрохімії за 2021 р. в таблицю Samples БД «SeaBase»:

- 40 станцій «Пляж "Аркадия"»;
- 40 станцій «мис Малий Фонтан»;
- 12 станцій Дунай травень;
- 12 станцій Дунай жовтень;
- 18 станцій прибережного моніторингу.

5. Внесенні дані екологічного моніторингу Чорного моря УкрНЦЕМ для 77 параметрів забруднюючих речовин у воді за 2021 р. в таблицю Samples БД «SeaBase»:

- 2 станції «Пляж "Аркадия"»;
- 2 станції «мис Малий Фонтан»;
- 12 станцій Дунай травень;
- 12 станцій Дунай жовтень;
- 18 станцій прибережного моніторингу.

6. Внесенні дані екологічного моніторингу Чорного моря УкрНЦЕМ для 82 параметрів забруднюючих речовин у донних відкладеннях за 2021 р. в таблицю Samples БД «SeaBase»:

- 12 станцій Дунай травень;
- 12 станцій Дунай жовтень.

7. Внесенні дані для п’яти параметрів фотосинтетичних пігментів у таблицю Samples БД «SeaBase» за 2021 рік:

- 38 станцій «Пляж "Аркадия"»;
- 37 станцій «мис Малий Фонтан»;
- 16 станцій прибережного моніторингу.

## 6.2. Роботи, проведені з даними у 2023 році

Протягом звітного періоду було виконано наступні роботи:

1. Аналіз отриманих УкрНЦЕМ даних за програмою моніторингу у 2022 році для подальшого внесення у БД «SeaBase»:
   - перевірка координат станцій та створення карт рейсів;
   - перевірка горизонтів та глибин станцій;
   - корегування одиниць вимірювання для забруднюючих речовин у воді, донних відкладах та біоті;
   - перевірка дат станцій моніторингу.
2. Таблиця Platform оновлена новою береговою станцією «Чорноморський яхт-клуб», яку почали аналізувати внаслідок воєнних дій;
3. Внесено 33 станція Прибережного моніторингу 2022 року у таблицю Stations бази даних УкрНЦЕМ «SeaBase».
4. Внесенні дані екологічного моніторингу Чорного моря УкрНЦЕМ для 15 параметрів гідрології та гідрохімії за 2022 р. в таблицю Samples БД «SeaBase»:
   - 4 станції «Пляж "Аркадия"»;
   - 4 станції «мис Малий Фонтан»;
   - 23 станції «Чорноморський яхт-клуб».
5. Внесенні дані екологічного моніторингу Чорного моря УкрНЦЕМ для 77 параметрів забруднюючих речовин у воді за 2022 р. в таблицю Samples БД «SeaBase» - 8 станцій «Чорноморський яхт-клуб».
6. Внесенні дані екологічного моніторингу Чорного моря УкрНЦЕМ для 82 параметрів забруднюючих речовин у донних відкладеннях за 2022 р. в таблицю Samples БД «SeaBase» - 2 станції «Чорноморський яхт-клуб».
7. Внесенні дані для п’яти параметрів фотосинтетичних пігментів у таблицю Samples БД «SeaBase» за 2022 рік:
   - 4 станції «мис Малий Фонтан»;

- 20 станцій «Чорноморський яхт-клуб».

8. Внесені дані для дванадцяти параметрів геології у таблицю Samples БД «SeaBase» за 2009-2021 роки:

- 9 станцій «Параллель 03.2009»;
- 11 станцій «Параллель 11.2009»;
- 16 станцій «Параллель 05.2010»;
- 20 станцій «Параллель 07.2010»;
- 12 станцій «Параллель 10.2010»;
- 13 станцій «Экоконтроль 1.2011»;
- 20 станцій «Экоконтроль 2.2011»;
- 36 станцій «Нефтегаз-68»;
- 12 станцій «Гермес 11.2012»;
- 15 станцій «Гермес 08.2013»;
- 14 станцій «Гермес 10.2013»;
- 10 станцій «Дунай 09.2014»;
- 11 станцій «Дунай 09.2015»;
- 12 станцій «Дунай 10.2015»;
- 15 станцій «NPMS-UA 2016»;
- 12 станцій «Дунай 08.2016»;
- 11 станцій «Дунай 10-11.2016»;
- 6 станцій «NPMS-UA Phyllophora 04.2017»;
- 4 станцій «NPMS-UA Phyllophora 07.2017»;
- 4 станцій «NPMS-UA Phyllophora 08.2017»;
- 7 станцій «NPMS-UA August 2017»;
- 12 станцій «Дунай 08.2017»;
- 12 станцій «Дунай 11.2017»;
- 12 станцій «Дунай 08.2018»;
- 12 станцій «Дунай 11.2018»;
- 12 станцій «Дунай 05.2019»;
- 12 станцій «Дунай 11.2019»;

- 15 станцій «JBSS GE-UA 2019»;
- 12 станцій «Дунай 08.2020»;
- 12 станцій «Дунай 05.2021»;
- 12 станцій «Дунай 10.2021».

9. Внесені дані екологічного моніторингу УкрНЦЕМ проведеного після руйнування Каховської ГЕС на 22 станціях, приведених на рисунку 6.1:

- для 14 станцій 9 параметрів гідрології та гідрохімії;
- для 13 станцій 74 параметрів забруднюючих речовин у воді.

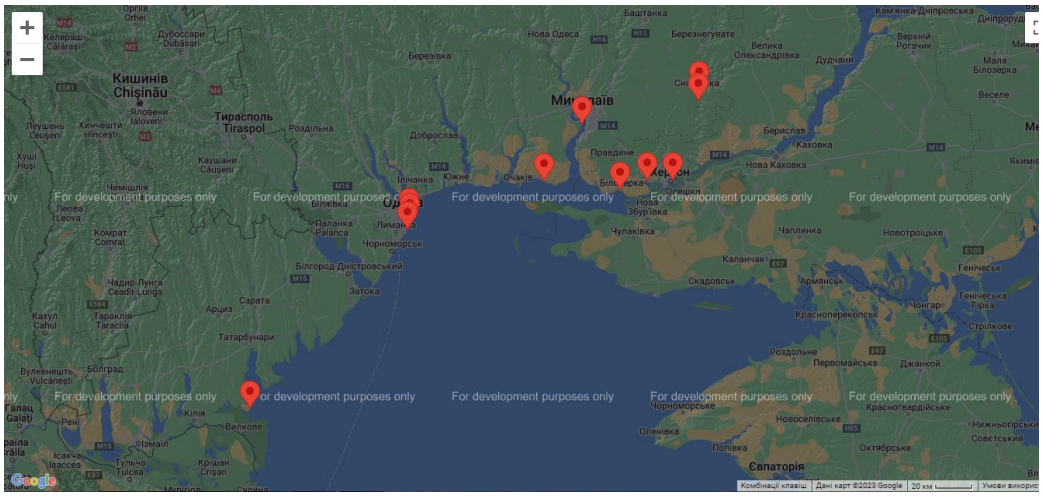


Рисунок 6.1 – Станції моніторингу УкрНЦЕМ після руйнування Каховської ГЕС

## 6.3 Роботи, проведені з даними у 2024 році

Протягом звітного періоду було виконано наступні роботи:

1. Аналіз отриманих УкрНЦЕМ даних за програмою моніторингу у 2023 році для подальшого внесення у БД «SeaBase»:

- перевірка координат станцій та створення карт рейсів;
- перевірка горизонтів та глибин станцій;
- корегування одиниць вимірювання для забруднюючих речовин у воді, донних відкладах та біоті;

- перевірка дат станцій моніторингу.

2. Таблиця Platform оновлена станцією «13 станція Великого Фонтану», яку почали аналізувати внаслідок воєнних дій і підриву Каховської ГЕС;

3. Внесено 67 станцій Прибережного моніторингу 2023 року у таблицю Stations бази даних УкрНЦЕМ «SeaBase».

4. Внесенні дані екологічного моніторингу Чорного моря УкрНЦЕМ для 15 параметрів гідрології та гідрохімії за 2022 р. в таблицю Samples БД «SeaBase»:

- 4 станції «Пляж "Аркадия"»;
- 18 станцій «мис Малий Фонтан»;
- 4 станції «13 ст.Великого Фонтану»;
- 41 станція «Чорноморський яхт-клуб».

5. Внесенні дані екологічного моніторингу Чорного моря УкрНЦЕМ для 55 параметрів забруднюючих речовин у воді за 2023 р. в таблицю Samples БД «SeaBase»:

- 1 станція «Чорноморський яхт-клуб»
- 3 станції «13 ст.Великого Фонтану»;
- 2 станції «мис Малий Фонтан».

6. Внесенні дані екологічного моніторингу Чорного моря УкрНЦЕМ для 55 параметрів забруднюючих речовин у донних відкладеннях за 2023 р. в таблицю Samples БД «SeaBase»:

- 7 станцій після підриву Каховської ГЕС у червні 2023р.;
- 8 станцій після підриву Каховської ГЕС у липні 2023р.;
- 8 станцій після підриву Каховської ГЕС у листопаді 2023р.

7. Внесенні дані екологічного моніторингу Чорного моря УкрНЦЕМ для 74 параметрів забруднюючих речовин у біоті за 2023 р. в таблицю Samples БД «SeaBase» - 7 проб біоти на 7 станціях ПЗЧ ЧМ.

8. Внесенні дані для п’яти параметрів фотосинтетичних пігментів у таблицю Samples БД «SeaBase» за 2023 рік:

- 6 станцій «мис Малий Фонтан»;
- 4 станції «13 ст.Великого Фонтану»;
- 40 станцій «Чорноморський яхт-клуб».

За три роки в БД «SeaBase» була оновлена загальною кількістю станцій - 260, та загальною кількістью зразків – 16 828. У таблиці 6.1 наведена статистика станцій та проб по роках.

Таблиця 6.1 – Загальна кількість станцій та зразків.

| Рік | Кількість станцій | Кількість зразків |
|---|---|---|
| 2022 | 122 | 9 065 |
| 2023 | 33 | 4 220 |
| 2024 | 105 | 6 580 |
| Всього | 260 | 19 865 |

У таблиці 6.2. та на рисунку 6.2 наведена статистика кількості зразків за групами параметрів і за роками. Найбільша група спостережень – це забруднюючі речовини. Вони складаються з наступних груп параметрів: ПАВ, ПХБ, пестициди, нафтові вуглеводні, феноли, токсичні метали. Також значний вклад у 2023 році зробили зразки групи геології, які були зібрані за за 2009-2021 роки.

Таблиця 6.2 – Статистика кількості зразків за видами спостережень і за роками.

| Група параметрів | Рік | | |
|---|---|---|---|
| | **2022** | **2023** | **2024** |
| Гідрологія -гідрохімія | 2268 | 453 | 1040 |
| Забруднюючі речовини у воді | 4596 | 586 | 3895 |
| Забруднюючі речовини у донних відкладеннях | 1756 | 24 | 1395 |
| Фотосинтетичні пігменти | 445 | 120 | 250 |
| Геологія | | 3037 | |
| Всього | 9065 | 4220 | 6580 |

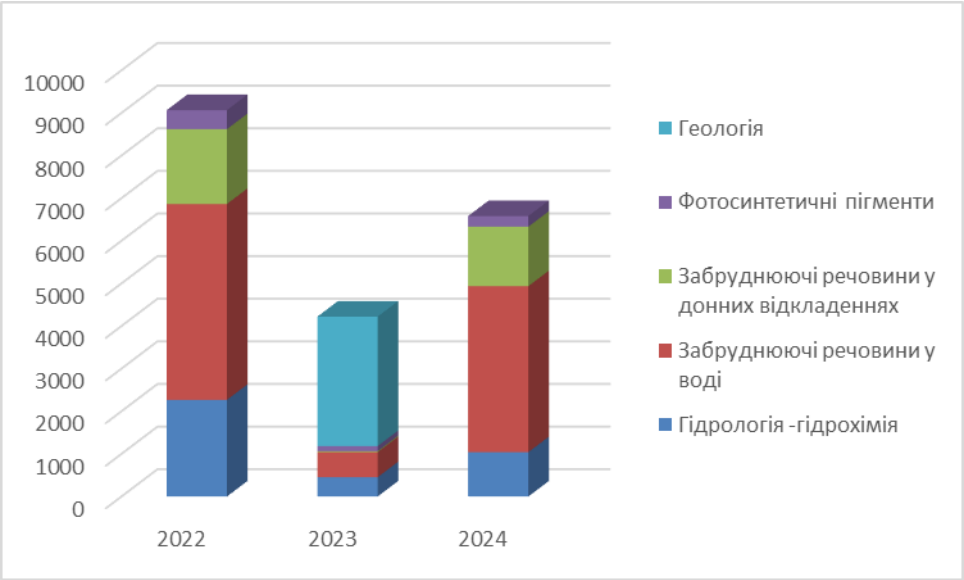


Рисунок 6.2 – Статистика кількості зразків.

# ВИСНОВКИ

В ході НДР був виконаний основний комплекс дій, який містив удосконалення локальної мережі та апаратного забезпечення серверного майданчику. Локальна мережа УкрНЦЕМ отримала більше протоколів захисту, які захищатимуть дані від несанкціонованого доступу ззовні та зсередини. Крім того, були виконані основні вимоги до налаштувань безпеки даних, такі як: конфіденційність, цілісність, доступність, достовірність, відмовостійкістью. Все виконано на апаратному рівні та у повному обсязі. Всі веб-ресурси УкрНЦЕМ зараз захищено асинхронним шифруванням.

В ході НДР було проведено суттєвву оптимізацію всіх інтерфейсів доступу до бази даних УкрНЦЕМ шляхом зміни вихідного коду та запитів до бази даних.

За звітній період було розроблено нову картографічну систему за допомогою таких програмних засобів як OpenLayers та MapServer. Використання OpenLayers для створення картографічних веб-застосунків дозволяє ефективно працювати з геопросторовими даними, забезпечуючи інтеграцію з WFS-сервісами, налаштування векторних шарів та інтуїтивний інтерфейс. MapServer забезпечує ефективне обслуговування географічних даних через WFS, інтегруючись з OpenLayers для відображення картографічних об'єктів. Його підтримка різних форматів даних, таких як Shapefiles, PostGIS і GeoTIFF, робить систему гнучкою для різних завдань. Поєднання MapServer з HTML, CSS і JavaScript дозволяє створювати зручний і динамічний картографічний інтерфейс. JavaScript забезпечує взаємодію користувача з картою, оновлення даних у реальному часі та керування шарами, що робить систему ефективною для візуалізації геопросторової інформації.

Протягом звітного періоду на постійній основі оновлювався сайт УкрНЦЕМ та його функціональні можливості й продуктивність. Стрічка

новин сайту оновлюється на постійній основі з надходженням нової інформації.

За три роки БД «SeaBase» була оновлена загальною кількістю станцій - 260, та загальною кількістю зразків – 16 828. Згідно зі статистикою, у 2022 та 2023 роках простежується зменшення кількості моніторингових спостережень, як по станціях так і по кількості проб. Це пов’язане з широкомасштабною збройною агресією російської федерації проти України та тимчасовою окупацією частини території України разом з морськими акваторіями, тому можливості здійснення повноцінного моніторингу морського середовища значно обмежені.

У 2024 році УкрНЦЕМ продовжив успішне виконання обов’язків вузла «OBIS Black Sea» в межах програми МООДІ під егідою МОК/ЮНЕСКО. За звітний період було оновлено існуючі набори даних за допомогою Integrated Publishing Toolkit на сервер УкрНЦЕМ. Продовжено співпрацю в межах проекту EMODNET Biology у новій фазі під номером 5. Взято участь в усіх заходах програми, переважна більшість яких, проводилась онлайн.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Звіт про науково-дослідну роботу «Базова оцінка та визначення Доброго екологічного стану (ДЕС) морського середовища Чорного моря в межах виключної морської економічної зони України» у 6-ті томах [Текст] / Науковий керівник В.В. Український // Одеса, УкрНЦЕМ, 2018. – 636 с. Державна реєстрація № 0118U006641.

2. Звіт про науково-дослідну роботу «Базова оцінка та визначення Доброго екологічного стану біоценозів і біорізноманіття Чорного моря в межах виключної морської економічної зони України» [Текст] / Науковий керівник С.П. Ковалишина // Одеса, УкрНЦЕМ, 2018. – 138 с. Державна реєстрація № 0118U006642.

3. Directive 2008/56/EC of the European Parliament and of the Council of 17 June 2008, establishing a framework for community action in the field of marine environmental policy (Marine Strategy Framework Directive) [Text] // Official Journal of the European Union, 25.6.2008. P. 19-40.

4. Звіт про науково-дослідну роботу «Розроблення Програми державного екологічного моніторингу морів України на 2019-2025 рр. відповідно до вимог Директив ЄС 2008/56/ЄС, 2008/105/ЄС» [Текст] / Науковий керівник В. М. Коморін// Одеса, УкрНЦЕМ, 2019. – 387 с. Державна реєстрація №0118U006644.

5. Кабінет Міністрів України, Постанова від 19 вересня 2018 р. № 758, Київ «Про затвердження Порядку державного моніторингу вод» [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/758-2018-п – Назва з екрану.

# Додаток А

## Оптимізовані запити до бази даних

```
SELECT Stations.IDst as &#39;ID&#39;, DataSource.Name as &#39;Cruise&#39;, Stations.Name as &#39;Station&#39;,
Organization.ShortName as &#39;Organization&#39;, Laboratory.Name as &#39;Laboratory&#39;, Stations.LonDecimalStart as
&#39;Longitude&#39;, Stations.LatDecimalStart as &#39;Latitude&#39;, CONVERT(varchar, (CONVERT (date,(Convert
(varchar,MetaPolWater.SamplingDay)+&#39;.&#39;+Convert (varchar,MetaPolWater.SamplingMonth)+&#39;.&#39;+Convert
(varchar,MetaPolWater.SamplingYear)),104)), 104) as &#39;Date&#39;, StationsPrecisionCoord.Name as &#39;Precision of
coordinates&#39;, DataSourceType.Name as &#39;Data source type&#39;, MonitoringType.Name as &#39;Monitoring type&#39;,
DataSource.ProjectTitle as &#39;Project&#39;, SamplesFraction.Name as &#39;Fraction&#39;, Stations.SeaName as &#39;Sea&#39;,
SamplesProxyPressures.Name as &#39;Proxy pressure&#39;, TypeOfDepth.Name as &#39;Depth type&#39;, Stations.Depth as &#39;Depth&#39;,
MetaPolWater.SamplingDepth as &#39;Sampling Depth&#39;, SamplesConcentration.Name as &#39;Concentration&#39;,
ParameterName.Name as &#39;Parameter&#39;, SamplesWater.Value as &#39;Value, [ng/l]&#39;, ParameterName.MoleculFormula as
&#39;Molecular formula&#39;, ParameterName.MonoisotopicMass as &#39;Monoisotopic mass&#39;, ParameterName.InChI as &#39;InChI&#39;,
ParameterName.Smile as &#39;SMILES&#39;, ParameterName.CAScode as &#39;CAS code&#39;, Method.LoD as &#39;LoD&#39;, Method.LoQ as
```

'LoQ', Method.Uncertainty as 'Uncertainty', Method.CoverageFactor as 'Coverage factor',Method.StandAMName

as 'Standardised analytical method',Method.StandAMNumb as 'Analytical method number',

Method.ProtocolValidated as 'Validated according to protocol', Method.ExtractionRecovery as 'Extraction

recovery', Method.BlankChecked as 'Blank checked', Method.ISO17025 as 'ISO

17025'Method.LaboratoryAccredited as 'Laboratory Accredited', Method.InterlaboratoryStudies as

'Interlaboratory Studies', Method.SumPer as 'Summary of performance', Method.ControlCharts as 'Control charts',

Method.IntStORIsLabComp as 'Internal standards or isotopically labelled compounds', Method.DataControlledBy

as 'Data controlled by', Method.SamPrepMethod as 'Sample preparation', Method.AnalyticalMethod as 'Analytical

Method', MethodAnalyticalLCPart.InstrManufacturer as '(LC) Instrument manufacturer',

MethodAnalyticalLCPart.InstrModel as '(LC) Model', MethodAnalyticalLCPart.AnalyticalColumn as '(LC) Analytical

method', MethodAnalyticalLCPart.DimensionsColumn as '(LC) Dimension column', MethodAnalyticalLCPart.Solvent

as '(LC) Solvent', MethodAnalyticalLCPart.TemperatureColumn as '(LC) Temperature',

MethodAnalyticalLCPart.MobilePhaseCompos as '(LC) Mobile phase composition',

MethodAnalyticalLCPart.MobilePhaseProgr as '(LC) Gradient programme',

MethodAnalyticalLCPart.MobilePhaseFlow as '(LC) Flow rate [ml/min]', MethodAnalyticalLCPart.InjectVolume as

'(LC) Injection volume [ml]', MethodAnalyticalLCPart.Deriv as '(LC) Derivatization',

MethodAnalyticalLCPart.DerivAgent as '(LC) Derivatization agent', MethodAnalyticalGCPart.InstrManufacturer as

'(GC) Instrument manufacturer', MethodAnalyticalGCPart.InstrModel as '(GC) Model',

MethodAnalyticalGCPart.AnalyticalColumn as '(GC) Analytical method',

MethodAnalyticalGCPart.DimensionsColumn as '(GC) Dimension column', MethodAnalyticalGCPart.Solvent as '(GC)

Solvent', MethodAnalyticalGCPart.InjectionMode as '(GC) Injection mode', MethodAnalyticalGCPart.InjVolume as

'(GC) Injection volume', MethodAnalyticalGCPart.CarrierGas as '(GC) Carrier gas', MethodAnalyticalGCPart.GasFlow

as '(GC) Flow rate'MethodAnalyticalGCPart.Oven as '(GC) Oven programme', MethodAnalyticalGCPart.Deriv as

'(GC) Derivatization', MethodAnalyticalGCPart.DerivAgent as '(GC) Derivatization agent',

MethodAnalyticalMasSpect.InstrManufacturer as '(MS) Instrument manufacturer',

MethodAnalyticalMasSpect.InstrModel as '(MS) Instrument model', MethodAnalyticalMasSpect.MasAnalyzer as

'(MS) Mas analyzer', MethodAnalyticalMasSpect.IonMode as '(MS) Ionization mode',

MethodAnalyticalMasSpect.MasMode as '(MS) Mass mode', MethodAnalyticalMasSpect.Precursor as '(MS)

Precursor [m/z]', MethodAnalyticalMasSpect.PrecIonType as '(MS) Precursor ion type',

MethodAnalyticalMasSpect.Products as '(MS) Products, SIMs', MethodAnalyticalMasSpect.ColEnergy as '(MS)

Collision energy', MethodAnalyticalMasSpect.ResolvPower as '(MS) Resolving power',

MethodAnalyticalMasSpect.AccurMS as '(MS) Accuracy MS [mDa]', MethodAnalyticalMasSpect.AccurMSMS as

'(MS) Accuracy MS/MS [mDa]', MethodAnalyticalMasSpect.SoftwareTarget as '(MS) Software for target',

MethodAnalyticalMasSpect.ProcedureTarget as '(MS) Procedure for target',

MethodAnalyticalMasSpect.SofwareSusp as '(MS) Software for suspect',

MethodAnalyticalMasSpect.ProcedureSusp as '(MS) Procedure for suspect',

MethodAnalyticalSpectroscopy.InstrManufacturer as '(S) Instrument manufacturer',

MethodAnalyticalSpectroscopy.InstrModel as '(S) Instrument model', MethodAnalyticalSpectroscopy.SpecTechn as

'(S) Spectrometric techniques', MethodAnalyticalSpectroscopy.SamplePre as '(S) Sample pretreatment',

MethodAnalyticalSpectroscopy.Isotope as '(S) Isotope', MethodAnalyticalSpectroscopy.InternalStand as '(S)

Internal standart', MethodAnalyticalSpectroscopy.SpecificChar as '(S) Specific characterization',

NormanQuality.Name as 'NORMAN Quality flag'

FROM DataSource, Stations, Organization, Laboratory, MetaPolWater, SamplesWater, Method,

```
     SamplesConcentration,   ParameterName,   Parameter,   Unit,
DataSourceType, DataSourcePlatform,
     StationsPrecisionCoord,   MonitoringType,   SamplesFraction,
SamplesProxyPressures, NormanQuality, TypeOfDepth,
     MethodAnalyticalLCPart,              MethodAnalyticalGCpart,
MethodAnalyticalMasSpect, MethodAnalyticalSpectroscopy
     WHERE MetaPolWater.IDOrgOwn = Organization.IDorg
     AND MetaPolWater.IDStationName = Stations.IDst
     AND Stations.IDdataSource = DataSource.IDdataSource
     AND SamplesWater.IDlab = Laboratory.IDlab
     AND              SamplesWater.IDMetaPolWater              =
MetaPolWater.IDMetaPolWater
     AND SamplesWater.IDmethod = Method.IDmethod
     AND SamplesConcentration.IDconcent = SamplesWater.IDconcent
     AND SamplesWater.IDpar = Parameter.IDpar
     AND Parameter.IDparName = ParameterName.IDparName
     AND Parameter.IDunit = Unit.IDunit
     AND              DataSource.IDDataSourceType              =
DataSourceType.IDDataSourceType
     AND DataSourcePlatform.IDPlatform = DataSource.IDPlatform
     AND StationsPrecisionCoord.IDcoorPrec = Stations.IDcoorPrec
     AND MonitoringType.IDMonType = Stations.IDMonType
     AND SamplesFraction.IDfraction = MetaPolWater.IDFraction
     AND          SamplesProxyPressures.IDproxyPress          =
Stations.IDProxyPres
     AND TypeOfDepth.IDtod = MetaPolWater.IDtod
     AND Method.IDLC = MethodAnalyticalLCPart.IDLCpart
     AND Method.IDGC = MethodAnalyticalGCpart.IDGCpart
     AND MethodAnalyticalMasSpect.IDMasSpect = Method.IDMasSpect
     AND      MethodAnalyticalSpectroscopy.IDSpectroscopy      =
Method.IDSpectroscopy
     AND Method.IDquality = NormanQuality.IDNQ
     AND                  Stations.IDcoutry                  in
(&quot;.$_POST[&#39;country&#39;].&quot;)
```

```
    and                MetaPolWater.IDOrgOwn                in
(&quot;.$_POST[&#39;org&#39;].&quot;)
    and               DataSource.IDdataSource               in
(&quot;.$_POST[&#39;cruise&#39;].&quot;)
    AND                    Stations.IDst                    in
(&quot;.$_POST[&#39;station&#39;].&quot;) AND Parameter.IDpar in
(&quot;.$_POST[&#39;par&#39;].&quot;)
    AND            SamplesConcentration.IDconcent           in
(&quot;.$_POST[&#39;concent&#39;].&quot;)
    AND                 Stations.MonthStart                 in
(&quot;.$_POST[&#39;months&#39;].&quot;)$str_date$str_coords and
    MetaPolWater.SamplingDepth $str_depth $filter
    ORDER BY DataSource.Name, Stations.Name

    INNER JOIN MetaPolWater ON MetaPolWater.IDOrgOwn =
Organization.IDorg
    INNER JOIN Stations ON MetaPolWater.IDStationName =
Stations.IDst
    INNER JOIN DataSource ON Stations.IDdataSource =
DataSource.IDdataSource
    INNER JOIN Laboratory ON SamplesWater.IDlab =
Laboratory.IDlab
    INNER JOIN SamplesWater ON SamplesWater.IDMetaPolWater =
MetaPolWater.IDMetaPolWater
    INNER JOIN Method ON SamplesWater.IDmethod = Method.IDmethod
    INNER JOIN SamplesConcentration ON
SamplesConcentration.IDconcent = SamplesWater.IDconcent
    INNER JOIN Parameter ON SamplesWater.IDpar = Parameter.IDpar
    INNER JOIN ParameterName ON Parameter.IDparName =
ParameterName.IDparName
    INNER JOIN Unit ON Parameter.IDunit = Unit.IDunit
    INNER JOIN DataSourceType ON DataSource.IDDataSourceType =
DataSourceType.IDDataSourceType
    INNER JOIN DataSourcePlatform ON
DataSourcePlatform.IDPlatform = DataSource.IDPlatform
```

```
INNER JOIN StationsPrecisionCoord ON
StationsPrecisionCoord.IDcoorPrec = Stations.IDcoorPrec
INNER JOIN MonitoringType ON MonitoringType.IDMonType =
Stations.IDMonType
INNER JOIN SamplesFraction ON SamplesFraction.IDfraction =
MetaPolWater.IDFraction
INNER JOIN SamplesProxyPressures ON
SamplesProxyPressures.IDproxyPress = Stations.IDProxyPres
INNER JOIN TypeOfDepth ON TypeOfDepth.IDtod =
MetaPolWater.IDtod
INNER JOIN MethodAnalyticalLCPart ON Method.IDLC =
MethodAnalyticalLCPart.IDLCpart
INNER JOIN MethodAnalyticalGCpart ON Method.IDGC =
MethodAnalyticalGCpart.IDGCpart
INNER JOIN MethodAnalyticalMasSpect ON
MethodAnalyticalMasSpect.IDMasSpect =
Method.IDMasSpect
INNER JOIN MethodAnalyticalSpectroscopy ON
MethodAnalyticalSpectroscopy.IDSpectroscopy =
Method.IDSpectroscopy
INNER JOIN NormanQuality ON Method.IDquality =
NormanQuality.IDNQ

Where Stations.IDcoutry in
(&quot;.$_POST[&#39;country&#39;].&quot;)
and MetaPolWater.IDOrgOwn in
(&quot;.$_POST[&#39;org&#39;].&quot;)
and DataSource.IDdataSource in
(&quot;.$_POST[&#39;cruise&#39;].&quot;)
AND Stations.IDst in
(&quot;.$_POST[&#39;station&#39;].&quot;) AND Parameter.IDpar in
(&quot;.$_POST[&#39;par&#39;].&quot;)
AND SamplesConcentration.IDconcent in
(&quot;.$_POST[&#39;concent&#39;].&quot;)
```

```
    AND                Stations.MonthStart                in
(&quot;.$_POST[&#39;months&#39;].&quot;)$str_date$str_coords  and
MetaPolWater.SamplingDepth
    $str_depth $filter

    SELECT
    Stations.IDst as ID, DataSource.Name as &#39;Cruise&#39;,
Stations.Name as &#39;Station&#39;,
    Stations.LonDecimalStart                                as
&#39;Longitude&#39;,Stations.LatDecimalStart                as
&#39;Latitude&#39;,
    CONVERT(varchar, (CONVERT (date,(Convert
    (varchar,PhytoplanktonMeta.SamplingDay)+&#39;.&#39;+Convert
    (varchar,PhytoplanktonMeta.SamplingMonth)+&#39;.&#39;+Conver
t
    (varchar,PhytoplanktonMeta.SamplingYear)),104)),   104)   as
&#39;Date&#39;, Stations.Depth as &#39;Station
    depth&#39;,PhytoplanktonMeta.MinSamplingDepth  as  &#39;Min
Sampling depth [m]&#39;,
    PhytoplanktonMeta.MaxSamplingDepth as &#39;Max Sampling depth
[m]&#39;, LayerType.Name as
    &#39;Layer  Type&#39;,  PhytoplanktonMethodSampling.Name  as
&#39;Method of sampling&#39;,
    PhytoplanktonMeta.SampleVolume as &#39;Volume of the sample

    [L]&#39;,PhytoplanktonMethodPreservation.Name as &#39;Method
of preservation&#39;,
    PhytoplanktonMethodPretreatment.Name as &#39;Method of sample
pretreatment&#39;,
    PhytoplanktonMicroscope.Name    as    &#39;Microscope&#39;,
PhytoplanktonMeta.Magnification as
    &#39;Magnification of microscope&#39;, People.Name + &#39;
&#39; + People.Surname as &#39;Taking
    Person&#39;,PhytoplanktonKingdom.Name as &#39;Kingdom&#39; ,
PhytoplanktonP.Name as &#39;Phylum&#39;,
```

PhytoplanktonC.Name as 'Class', PhytoplanktonO.Name as 'Order', PhytoplanktonF.Name as

'Family', PhytoplanktonG.Name as 'Genus', PhytoplanktonS.Name as 'Species',

PhytoplanktonS.WORMS as 'WORMS', PhytoplanktonFormula.Name as

'Formula',PhytoplanktonGeometricShape.Name as 'Shape', PhytoplanktonSamples.Length1 as

'Length (l1) [µm]', PhytoplanktonSamples.Length2 as 'Length (l2)

[µm]',PhytoplanktonSamples.Width as 'Width (w) [µm]', PhytoplanktonSamples.Height as

'Height (h) [µm]', PhytoplanktonSamples.Diameter1 as 'Diameter (d1) [µm]',

PhytoplanktonSamples.Diameter2 as 'Diameter (d2)

[µm]',PhytoplanktonSamples.CountedNumber as 'Counted number',

PhytoplanktonCountingUnit.Name as 'Counting Unit', PhytoplanktonStage.Name as

'STAGE',PhytoplanktonSamples.MicroscopeScale as 'Microscope scale value [µm]',

PhytoplanktonSamples.SampleVolume as 'Volume of the sample

[ml]',PhytoplanktonSamples.AfterConcVolume as 'Volume after concentration [ml]',

PhytoplanktonSamples.SubsamplesNumber as 'Number of

subsamples',PhytoplanktonSamples.SubsampleVolume as 'Volume of subsample for counting

[ml]', PhytoplanktonSamples.CoefToLiter as 'Coefficient to liter',SC.Name + ' ' + SC.Surname as

'Scientist',PhytoplanktonSamples.CellVolume as 'Volume of the cell [µm3]',

PhytoplanktonSamples.Abundance as 'Abundance [*1000cells/l]',

```
     PhytoplanktonSamples.Biomass as &#39;Biomass [mg/m3]&#39;,
PhytoplanktonSamples.CarbonBiomass
     as &#39;Carbon biomass&#39;
     FROM
     Stations,DataSource,PhytoplanktonMeta,PhytoplanktonSamples,P
hytoplanktonC,Phyto
     planktonF,PhytoplanktonG,PhytoplanktonO,PhytoplanktonP,Phyto
planktonS, LayerType,
     PhytoplanktonMethodSampling,
PhytoplanktonMethodPreservation,
     PhytoplanktonMethodPretreatment,PhytoplanktonMicroscope,
People,
     PhytoplanktonKingdom,          PhytoplanktonGeometricShape,
PhytoplanktonFormula,
     PhytoplanktonCountingUnit, PhytoplanktonStage, People SC

     WHERE
     LayerType.IDDepthType = PhytoplanktonMeta.IDLayerType
     AND              PhytoplanktonStage.IDStage             =
PhytoplanktonSamples.IDStage
     AND          PhytoplanktonSamples.IDCountingUnit        =
PhytoplanktonCountingUnit.IDcu
     AND          PhytoplanktonGeometricShape.IDShape        =
PhytoplanktonFormula.IDShape
     AND PhytoplanktonFormula.IDFormula = PhytoplanktonS.IDForm
     AND            PhytoplanktonKingdom.IDKingdom           =
PhytoplanktonP.IDKingdom
     AND People.IDPeople = PhytoplanktonMeta.IDPerson
     AND PhytoplanktonMicroscope.IDPhytoMicroscope =
     PhytoplanktonMeta.IDPhytoMicroscope
     AND PhytoplanktonMethodPretreatment.IDPretreatmentMethod =
     PhytoplanktonMeta.IDPretreatmentMethod
     AND PhytoplanktonMethodPreservation.IDPreservMethod =
     PhytoplanktonMeta.IDPreservMethod
     AND PhytoplanktonMethodSampling.IDSampMethod =
```

```
     PhytoplanktonMeta.IDSampMethod
     AND Stations.IDst = PhytoplanktonMeta.IDStationName
     AND Stations.IDdataSource = DataSource.IDdataSource
     AND             PhytoplanktonMeta.IDPhytoMeta             =
PhytoplanktonSamples.IDMeta
     AND PhytoplanktonP.IDPhylum = PhytoplanktonC.IDPhylum
     AND PhytoplanktonC.IDClass = PhytoplanktonO.IDClass
     AND PhytoplanktonO.IDOrder = PhytoplanktonF.IDOrder
     AND PhytoplanktonF.IDFamily = PhytoplanktonG.IDFamily
     AND PhytoplanktonG.IDGenus = PhytoplanktonS.IDGenus
     AND             PhytoplanktonSamples.IDSpecies             =
PhytoplanktonS.IDSpecies
     AND SC.IDPeople = PhytoplanktonSamples.IDPeople
     AND                   Stations.IDcoutry                   in
(&quot;.$_POST[&#39;country&#39;].&quot;)                    AND
PhytoplanktonMeta.IDOrgOwn in
     (&quot;.$_POST[&#39;org&#39;].&quot;)                    AND
Stations.IDdataSource in (&quot;.$_POST[&#39;cruise&#39;].&quot;)
     AND            PhytoplanktonMeta.IDStationName            in
(&quot;.$_POST[&#39;station&#39;].&quot;) AND
     PhytoplanktonMeta.MinSamplingDepth $str_depth AND
     PhytoplanktonMeta.MaxSamplingDepth $str_depth
     AND            PhytoplanktonMeta.SamplingMonth            in
(&quot;.$_POST[&#39;months&#39;].&quot;)$str_date$str_coords
     AND $condition

     ORDER BY DataSource.Name,Stations.IDst, &#39;Min Sampling
depth [m]&#39;

     INNER   JOIN   LayerType   ON   LayerType.IDDepthType   =
PhytoplanktonMeta.IDLayerType
     INNER JOIN PhytoplanktonStage ON PhytoplanktonStage.IDStage
=
     PhytoplanktonSamples.IDStage
```

```
     INNER JOIN PhytoplanktonCountingUnit ON
PhytoplanktonSamples.IDCountingUnit =
     PhytoplanktonCountingUnit.IDcu
     INNER JOIN PhytoplanktonGeometricShape ON
PhytoplanktonGeometricShape.IDShape =
     PhytoplanktonFormula.IDShape
     INNER JOIN PhytoplanktonFormula ON
PhytoplanktonFormula.IDFormula =
     PhytoplanktonS.IDForm
     INNER JOIN PhytoplanktonKingdomPhytoplanktonKingdom
ON .IDKingdom =
     PhytoplanktonP.IDKingdom
     INNER JOIN People ON People.IDPeople =
PhytoplanktonMeta.IDPerson
     INNER JOIN PhytoplanktonMicroscope ON
PhytoplanktonMicroscope.IDPhytoMicroscope =
     PhytoplanktonMeta.IDPhytoMicroscope
     INNER JOIN PhytoplanktonMethodPretreatment ON
     PhytoplanktonMethodPretreatment.IDPretreatmentMethod =
     PhytoplanktonMeta.IDPretreatmentMethod
     INNER JOIN PhytoplanktonMethodPreservation ON
     PhytoplanktonMethodPreservation.IDPreservMethod = Phytoplan
ktonMeta.IDPreservMethod
     INNER JOIN PhytoplanktonMethodSampling ON
     PhytoplanktonMethodSampling.IDSampMethod =
PhytoplanktonMeta.IDSampMethod
     INNER JOIN Stations ON Stations.IDst =
PhytoplanktonMeta.IDStationName
     INNER JOIN DataSource ON Stations.IDdataSource =
DataSource.IDdataSource

     INNER JOIN PhytoplanktonMeta ON PhytoplanktonMeta.IDPhytoMeta
=
     PhytoplanktonSamples.IDMeta
```

```
INNER JOIN PhytoplanktonC ON PhytoplanktonP.IDPhylum =
PhytoplanktonC.IDPhylum
INNER JOIN PhytoplanktonO ON PhytoplanktonC.IDClass
= .IDClass
INNER JOIN PhytoplanktonF ON PhytoplanktonO.IDOrder =
PhytoplanktonF.IDOrder
INNER JOIN PhytoplanktonG ON PhytoplanktonF.IDFamily =
PhytoplanktonG.IDFamily
INNER JOIN PhytoplanktonS ON PhytoplanktonG.IDGenus =
PhytoplanktonS.IDGenus
INNER JOIN PhytoplanktonSamples ON
PhytoplanktonSamples.IDSpecies =
PhytoplanktonS.IDSpecies
INNER JOIN PhytoplanktonSamples ON SC.IDPeople =
PhytoplanktonSamples.IDPeople
WHERE
Stations.IDcoutry in
(&quot;.$_POST[&#39;country&#39;].&quot;) AND
PhytoplanktonMeta.IDOrgOwn in
(&quot;.$_POST[&#39;org&#39;].&quot;) AND
Stations.IDdataSource in (&quot;.$_POST[&#39;cruise&#39;].&quot;)
AND PhytoplanktonMeta.IDStationName in
(&quot;.$_POST[&#39;station&#39;].&quot;) AND
PhytoplanktonMeta.MinSamplingDepth $str_depth AND
PhytoplanktonMeta.MaxSamplingDepth $str_depth
AND PhytoplanktonMeta.SamplingMonth in
(&quot;.$_POST[&#39;months&#39;].&quot;)$str_date$str_coords AND
$condition
ORDER BY DataSource.Name,Stations.IDst, &#39;Min Sampling
depth [m]&#39;
```

# Додаток Б

## Програмний код інтерактивної картографічної системи

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>WFS with OpenLayers</title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/ol@v10.3.1/ol.css" />
  <script
src="https://cdn.jsdelivr.net/npm/ol@v10.3.1/dist/ol.js"></scrip
t>
  <style>
    #map {
      width: 100%;
      height: 800px;
    }
    #controls {
      position: absolute;
      top: 20px;
      left: 50px;
      background: white;
      padding: 10px;
      border-radius: 5px;
      box-shadow: 0px 2px 5px rgba(0, 0, 0, 0.3);
      z-index: 1000;
      display: flex;
      flex-direction: column;
      gap: 5px;
    }
    #info {
```

```
  position: absolute;
  bottom: 10px;
  left: 10px;
  background: white;
  padding: 10px;
  border-radius: 5px;
  box-shadow: 0px 2px 5px rgba(0, 0, 0, 0.3);
  max-height: 200px;
  overflow-y: auto;
  z-index: 1000;
}
#legend {
  position: absolute;
  top: 20px;
  right: 20px;
  background: white;
  padding: 10px;
  border-radius: 5px;
  box-shadow: 0px 2px 5px rgba(0, 0, 0, 0.3);
  z-index: 1000;
  max-width: 200px;
  display: flex;
  flex-direction: column;
  gap: 10px;
}
.legend-item {
  display: flex;
  align-items: center;
  gap: 5px;
}
.color-circle {
  width: 15px;
  height: 15px;
  border-radius: 50%;
  border: 1px solid #000;
```

```
        }
        .color-rect {
          width: 25px;
          height: 15px;
          border: 1px solid #000;
        }
        .ol-popup {
          background-color: white;
          border-radius: 5px;
          box-shadow: 0px 2px 10px rgba(0, 0, 0, 0.5);
          padding: 15px;
          max-width: 200px;
          position: relative;
          margin-bottom: 20px;
        }

        .close-popup {
          position: absolute;
          top: 0px;
          right: 5px;
          font-size: 20px;
          color: #000;
          cursor: pointer;
          z-index: 2000;
        }
      </style>
    </head>
    <body>
      <div id="controls">
        <label>
          <input type="checkbox" id="seaLayer" checked> Sea Layer
        </label>
        <label>
          <input  type="checkbox"  id="landLayer"  checked>  Land
Layer
```

```
        </label>
        <label>
          <input type="checkbox" id="pointLayer" checked> Point
Layer
        </label>
      </div>
      <div id="map"></div>
      <div   id="info">Click   on   a   feature   to   see   its
properties.</div>
      <div id="legend">
        <!-- Легенда буде оновлюватися динамічно -->
      </div>

      <script>
        // ================================
        // Коментарі на українській мові
        // ================================
        //  Ініціалізація джерел даних GeoJSON через WFS для
різних шарів
        const seaSource = new ol.source.Vector({
          format: new ol.format.GeoJSON(),
          url:                              'http://localhost/cgi-
bin/mapserv.exe?map=/ms4w/apps/blacksea/blacksea.map&SERVICE=WFS
&VERSION=1.1.0&REQUEST=GetFeature&TYPENAME=Sea&CRSNAME=EPSG:3857
&OUTPUTFORMAT=application/json',
          strategy: ol.loadingstrategy.bbox,
          crossOrigin: 'anonymous'
        });

        const landSource = new ol.source.Vector({
          format: new ol.format.GeoJSON(),
          url:                              'http://localhost/cgi-
bin/mapserv.exe?map=/ms4w/apps/blacksea/blacksea.map&SERVICE=WFS
&VERSION=1.1.0&REQUEST=GetFeature&TYPENAME=layer1&CRSNAME=EPSG:3
857&OUTPUTFORMAT=application/json',
```

```
            strategy: ol.loadingstrategy.bbox,
            crossOrigin: 'anonymous'
          });

          const pointSource = new ol.source.Vector({
            format: new ol.format.GeoJSON(),
            url:                              'http://localhost/cgi-
bin/mapserv.exe?map=/ms4w/apps/blacksea/blacksea.map&SERVICE=WFS
&VERSION=1.1.0&REQUEST=GetFeature&TYPENAME=point&CRSNAME=EPSG:38
57&OUTPUTFORMAT=application/json',
            strategy: ol.loadingstrategy.bbox,
            crossOrigin: 'anonymous'
          });

          // ================================
          // Стилі для різних шарів
          // ================================
          const seaLayer = new ol.layer.Vector({
            source: seaSource,
            style: new ol.style.Style({
              fill: new ol.style.Fill({
                color:    'rgba(173,    216,    230,    0.5)',    //
Полупрозорість для моря
              }),
              stroke: new ol.style.Stroke({
                color: '#000', // Чорний обвід для моря
                width: 1,
              }),
            }),
          });

          const landLayer = new ol.layer.Vector({
            source: landSource,
            style: new ol.style.Style({
              fill: new ol.style.Fill({
```

```
            color: 'rgba(34, 139, 34, 0.5)', // Полупрозорість
для землі
          }),
          stroke: new ol.style.Stroke({
            color: '#000', // Чорний обвід для землі
            width: 1,
          }),
        }),
      });

      const pointLayer = new ol.layer.Vector({
        source: pointSource,
        style: new ol.style.Style({
          image: new ol.style.Circle({
            radius: 7, // Розмір точок
            fill: new ol.style.Fill({
              color: 'rgba(255, 0, 0, 1)' // Червона заливка
для точок
            }),
            stroke: new ol.style.Stroke({
              color: '#000', // Чорний обвід точок
              width: 1
            })
          })
        })
      });

      // ================================
      // Функція додавання нового шару
      // ================================
      function addNewLayer() {
        const newLayer = new ol.layer.Vector({
          source: new ol.source.Vector({
            format: new ol.format.GeoJSON(),
            url: 'https://example.com/geojson',
```

```
            crossOrigin: 'anonymous'
          }),
          style: new ol.style.Style({
            fill: new ol.style.Fill({
              color: 'rgba(255, 165, 0, 0.5)', // Оранжева заливка для нового шару
            }),
            stroke: new ol.style.Stroke({
              color: '#000', // Чорний обвід
              width: 1,
            }),
          })
        });
        map.addLayer(newLayer); // Додаємо новий шар на карту
      }

      // ================================
      // Зміна стилю шару в реальному часі
      // ================================
      function changeLayerStyle(layer, newStyle) {
        layer.setStyle(newStyle); // Змінюємо стиль шару
      }

      const newSeaStyle = new ol.style.Style({
        fill: new ol.style.Fill({
          color: 'rgba(0, 191, 255, 0.5)', // Новий стиль заливки для моря
        }),
        stroke: new ol.style.Stroke({
          color: '#000',
          width: 2, // Товщий обвід
        }),
      });
      changeLayerStyle(seaLayer, newSeaStyle); // Змінюємо стиль шару моря
```

```
// ================================
// Карта з різними шарами
// ================================
const map = new ol.Map({
  target: 'map',
  layers: [
    new ol.layer.Tile({
      source: new ol.source.OSM() // Фоновий шар OSM
    }),
    seaLayer, // Шар моря
    landLayer, // Шар землі
    pointLayer // Шар точок
  ],
  view: new ol.View({
    center: ol.proj.fromLonLat([35, 45]), // Центр карти
    zoom: 5 // Масштаб
  })
});

// ================================
// Спливаюче вікно на карті
// ================================
const popup = new ol.Overlay({
  element: document.createElement('div'),
  positioning: 'bottom-center',
  stopEvent: true
});
popup.getElement().className = 'ol-popup'; // Стиль для
вспливаючого вікна
map.addOverlay(popup);

const closeButton = document.createElement('span');
closeButton.className = 'close-popup';
closeButton.innerHTML = '&times;';
```

```
popup.getElement().appendChild(closeButton);

closeButton.addEventListener('click', function() {
  popup.setPosition(undefined); // Сховати вікно
});

// Обробник для кліку на карті
map.on('singleclick', function (evt) {
  const features = map.getFeaturesAtPixel(evt.pixel); //
Отримуємо об'єкти на карті
  if (features.length > 0) {
    const properties = features[0].getProperties();
    const info = Object.entries(properties)
      .map(([key, value]) => `${key}: ${value}`)
      .join('<br>');
    popup.getElement().innerHTML = info;
    popup.getElement().appendChild(closeButton);
    popup.setPosition(evt.coordinate); // Позиція вікна
  } else {
    popup.setPosition(undefined);
  }
});

// ================================
// Оновлення легенди для шарів
// ================================
function updateLegendVisibility() {
  const seaVisible = seaLayer.getVisible();
  const landVisible = landLayer.getVisible();
  const pointVisible = pointLayer.getVisible();
  const legend = document.getElementById('legend');
  const legendItems = [];

  if (seaVisible) {
```

```
            legendItems.push('<div      class="legend-item"><div
class="color-rect"  style="background-color:  rgba(173,  216,  230,
0.5)"></div>Sea</div>');
          }
          if (landVisible) {
            legendItems.push('<div      class="legend-item"><div
class="color-rect"  style="background-color:  rgba(34,  139,  34,
0.5)"></div>Land</div>');
          }
          if (pointVisible) {
            legendItems.push('<div      class="legend-item"><div
class="color-circle"  style="background-color:  rgba(255,  0,  0,
1)"></div>Point</div>');
          }

          legend.innerHTML = legendItems.join('');

          if (!seaVisible && !landVisible && !pointVisible) {
            legend.style.display = 'none';
          } else {
            legend.style.display = 'flex';
          }
        }

        // Обробники для видимості шарів

document.getElementById('seaLayer').addEventListener('change',
function (e) {
          seaLayer.setVisible(e.target.checked);
          updateLegendVisibility();
        });


document.getElementById('landLayer').addEventListener('change',
function (e) {
```

```
            landLayer.setVisible(e.target.checked);
            updateLegendVisibility();
          });


document.getElementById('pointLayer').addEventListener('change',
function (e) {
            pointLayer.setVisible(e.target.checked);
            updateLegendVisibility();
          });

          // Ініціалізація легенди
          updateLegendVisibility();
        </script>
      </body>
      </html>
      /*
       Description:   Simple map to display on MS4W localhost
( http://127.0.0.1 )
                     Also configured for WMS, WFS, OGCAPI (Feature)
services, and
                        includes  GeoJSON,  Shapefile,  KMZ,  GML
output
       Data source:  NaturalEarth dataset, in SpatiaLite format.
       Other notes:  Open this mapfile in Notepad++, and use the
color syntax file
                     from
https:/C:/ms4w.com/trac/wiki/Notepad++MapServerStyle

      */

      MAP
        NAME "local"
        STATUS ON
        SIZE 600 400
```

```
  SYMBOLSET "../etc/symbols.txt"
  EXTENT -23422614 -11650014 23758273 18540865 #EPSG:3857
  #EXTENT -180 -90 180 90 #EPSG:4326
  UNITS meters #EPSG:3857
  #UNITS DD #EPSG:4326
  SHAPEPATH "./shp"
  IMAGECOLOR 255 255 255
  FONTSET "../etc/fonts.txt"
  IMAGETYPE "png"
  CONFIG "MS_ERRORFILE" "C:/ms4w/tmp/ms_error.log"
 CONFIG "ON_MISSING_DATA" "IGNORE"
 CONFIG "CPL_DEBUG" "ON"
 CONFIG "SHAPE_ENCODING" "UTF-8"

  #output projection
  PROJECTION
    "init=epsg:3857"        #Web        Mercator,        see
https://spatialreference.org/ref/epsg/3857/
  END

  WEB
    IMAGEPATH "C:/ms4w/tmp/ms_tmp/"
    IMAGEURL "/ms_tmp/"
    METADATA
       "wms_enable_request" "*"
      "ows_title"              "MS4W Demo WMS / WFS / OGCAPI
Server"
      "ows_abstract"              "This demonstration server
was setup by GatewayGeo (https://gatewaygeomatics.com/) and is
powered by MS4W (https:/C:/ms4w.com/)."
      "ows_keywordlist"         "MS4W, MapServer, OGCAPI"
      "ows_onlineresource"            "http://localhost/cgi-
bin/mapserv.exe?map=/ms4w/apps/blacksea/blacksea.map&"
      "oga_onlineresource"            "http://127.0.0.1/cgi-
bin/mapserv.exe/local-demo/ogcapi"
```

```
            "ows_service_onlineresource"
"https://gatewaygeomatics.com/"
            "ows_fees"                "none"
            "ows_accessconstraints"   "none"
            "ows_contactorganization" "GatewayGeo"
            "ows_contactposition"     "Director"
            "ows_role"                "Director"
            "ows_contactvoicetelephone" "none"
            "ows_contactfacsimiletelephone" "none"
            "ows_contactinstructions" "none"
            "ows_hoursofservice"      "none"
            "ows_addresstype"         "none"
            "ows_address"             "none"
            "ows_city"                "none"
            "ows_stateorprovince"     "none"
            "ows_postcode"            "none"
            "ows_country"             "none"
            "ows_srs"               "EPSG:3857 EPSG:4326 EPSG:4269"
            "ows_getfeatureinfo_formatlist"
"text/plain,text/html,application/json,application/vnd.ogc.gml,g
ml"
            "wfs_getfeature_formatlist"
"gml,application/json,shapezip,kmz,ogrgml,spatialite,ogrflatgeob
uf,csv" #can also be set at LAYER level, which will be used instead
            "ows_enable_request"      "*"
            "wms_allow_getmap_without_styles" "true"
            #"oga_html_template_directory"
"../../share/ogcapi/templates/html-plain/"
            "oga_html_template_directory"
"../../share/ogcapi/templates/html-bootstrap4/"
             "ows_encoding" "UTF-8"
            "ows_language" "ru-RU"
          END
        END
```

```
        /* Sample outputformats
             more            about            includes            at
https://mapserver.org/mapfile/include.html
        */
        INCLUDE "../includes/outputformat.include"

        /* Ocean */
        LAYER
          NAME "Sea"
          METADATA
             "wms_enable_request" "*"
            "ows_title" "Sea"
            "ows_abstract"     "World Oceans, NaturalEarth dataset,
2020"
            "ows_keywordlist"   "MS4W, MapServer, OGCAPI"
            "ows_include_items" "all"
            "gml_include_items" "all"
             "gml_featureid"     "ne_id"
            "gml_types"         "auto"
            "wfs_use_default_extent_for_getfeature" "false"
            "ows_extent" "-180 -90 180 90" #data's source extent,
this helps for performance
             "wfs_title" "Sea"
            "wfs_srs" "EPSG:4326 EPSG:3857"
            "gml_include_items" "all"
            "wfs_enable_request" "*"
            "wfs_getfeature_formatlist" "geojson"
             "wfs_getfeature_formatlist"
"geojson,application/json"
          END
          TYPE POLYGON
          STATUS ON


          DATA "AllSea"
```

```
        #the data's source projection
        PROJECTION
          "init=epsg:4326"              #latlong,              see
https://spatialreference.org/ref/epsg/4326/
        END
        CLASS
          NAME "AllSea"
          STYLE
            COLOR 134 204 249
          END
        END
        TEMPLATE "ttt.html"
      END # layer

      /* Countries */
      LAYER
        NAME "layer1"
        TYPE POLYGON
        STATUS ON
        DATA "O81STATE"
         PROJECTION
          "init=epsg:4326"
        END
        CLASS
          NAME "layer1"
          STYLE
            COLOR 255 255 255
            OUTLINECOLOR 0 0 0
            WIDTH 1
          END
        END
        METADATA
           "wms_enable_request" "*"
          "ows_title" "layer1"
```

```
          "ows_abstract"     "World Oceans, NaturalEarth dataset,
2020"
          "ows_keywordlist"   "MS4W, MapServer, OGCAPI"
          "ows_include_items" "all"
          "gml_include_items" "all"
           "gml_featureid"     "ne_id"
          "gml_types"         "auto"
          "wfs_use_default_extent_for_getfeature" "false"
          "ows_extent" "-180 -90 180 90" #data's source extent,
this helps for performance
           "wfs_title" "layer1"
          "wfs_srs" "EPSG:4326 EPSG:3857"
          "gml_include_items" "all"
          "wfs_enable_request" "*"
          "wfs_getfeature_formatlist" "geojson"
           "wfs_getfeature_formatlist"
"geojson,application/json"
        END
      END

      LAYER
        NAME "point"
        TYPE POINT
        STATUS ON
        DATA "123_dots"
         PROJECTION
          "init=epsg:4326"
        END
        CLASS
          NAME "point"
          STYLE
            COLOR 255 255 255
            OUTLINECOLOR 0 0 0
            WIDTH 6
          END
```

```
        END
        METADATA
           "wms_enable_request" "*"
          "ows_title" "point"
          "ows_abstract"     "World Oceans, NaturalEarth dataset,
2020"
          "ows_keywordlist"   "MS4W, MapServer, OGCAPI"
          "ows_include_items" "all"
          "gml_include_items" "all"
           "gml_featureid"     "ne_id"
          "gml_types"         "auto"
          "wfs_use_default_extent_for_getfeature" "false"
          "ows_extent" "-180 -90 180 90" #data's source extent,
this helps for performance
           "wfs_title" "point"
          "wfs_srs" "EPSG:4326 EPSG:3857"
          "gml_include_items" "all"
          "wfs_enable_request" "*"
          "wfs_getfeature_formatlist" "geojson"
           "wfs_getfeature_formatlist"
"geojson,application/json"
        END
      END



    END # Map File
```