FishCast

Документація проєкту

Автор: Артем Політов

Рік: 2025

Місто: Дніпро

ЗМІСТ

1. АРХІТЕКТУРА ПРОЄКТУ 3
2. ФУНКЦІОНАЛ КОРИСТУВАЧА 4
2.1. Неавторизований користувач 4
2.2. Авторизований користувач 5
2.3. Сценарії використання 5
2.3.1. Перший запуск застосунку (неавторизований користувач) 5
2.3.2. Авторизація користувача 6
2.3.3. Робота з профілем користувача 6
2.3.4. Перегляд сторінки локації 7
2.3.5. Перемикання теми та мови 7
3. API ТА СТРУКТУРА БАЗИ ДАНИХ 7
3.1. Авторизация та користувачі 7
3.2. API для довідкових даних: локації, риби, регіони 8
3.3. Робота з MongoDB 9
3.4. Стек технологій бекенду 9
4. РЕАЛІЗАЦІЯ КЛІЄНТСЬКОЇ ЧАСТИНИ 9
4.1. Загальна архітектура клієнтської частини 9
4.2. Керування станом за допомогою Redux Toolkit та RTK Query 10
4.3. Обробка даних з OpenWeather API 11
4.4 Алгоритм розрахунку ймовірності кльову 12
4.5. Механізм авторизації на клієнті 12
4.6. Обрані локації. 13
4.7. Підтримка темної/світлої теми 14
4.8. Локалізація інтерфейсу та підтримка кількох мов 14
5. ВІЗУАЛІЗАЦІЯ ДАНИХ ТА АДАПТИВНІСТЬ ІНТЕРФЕЙСУ 14
6. ВИСНОВКИ ТА ПЛАНИ НА РОЗВИТОК 15

1. АРХІТЕКТУРА ПРОЄКТУ

Проєкт FishCast реалізовано як SPA (Single Page Application) із використанням фреймворку Next.js та мови TypeScript для забезпечення статичної типізації, підвищення надійності коду та зручності розробки. Уся клієнтська логіка виконується на стороні браузера — серверний рендеринг (SSR) та статична генерація сторінок (SSG) у проєкті не застосовуються.

Клієнтська частина працює як повноцінна односторінкова програма, яка динамічно підвантажує дані через RTK Query після первинного рендеру інтерфейсу. Це забезпечує швидку взаємодію користувача з додатком без повного перезавантаження сторінок.

Бекенд-частина розгорнута окремо та побудована на базі Express.js. Вона відповідає за обробку запитів, взаємодію з базою даних MongoDB через Mongoose, а також надає REST API для отримання й оновлення даних (локації, міста, області, види риб, користувачі тощо).

Комунікація між фронтендом і бекендом здійснюється по HTTP через REST API з використанням CORS. Авторизація реалізована через токени доступу (JWT) та cookie для збереження сесій.

Загальну архітектуру клієнтської та серверної частини наведено на діаграмі на рисунку 1.

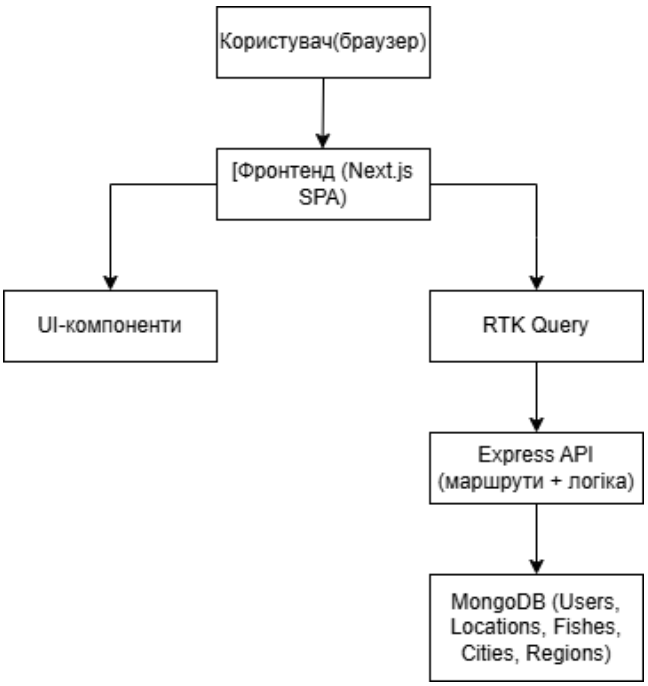

Рисунок 1 – Діаграма архітектури додатку

2. ФУНКЦІОНАЛ КОРИСТУВАЧА

2.1. Неавторизований користувач

Незареєстрований користувач має змогу взаємодіяти з основною частиною застосунку, отримуючи доступ до таких можливостей:

- Перегляд прогнозу погоди у вибраному населеному пункті.
- Прогноз кльову найпопулярніших видів риб у околицях населеного пункту або на конкретній риболовній локації, що міститься у базі даних.
- Відображення списку риб у яких найкращій кльов на поточний момент в околицях обраного населеного пункту.
- Відображення списку локацій поблизу обраного населеного пункту та можливість перейти на сторінку конкретної локації.
- Зміна теми (світла / темна) та мови інтерфейсу (українська / російська).

2.2. Авторизований користувач

Після авторизації користувач отримує розширений функціонал:

- Автоматичне встановлення населеного пункту, який користувач вказав при реєстрації.
- Можливість додавання локацій до вибраного для швидкого доступу.
- Доступ до модального вікна профілю з переглядом обраних локацій.

Приклад інтерфейсу модального вікна профілю наведено на рисунку 2.

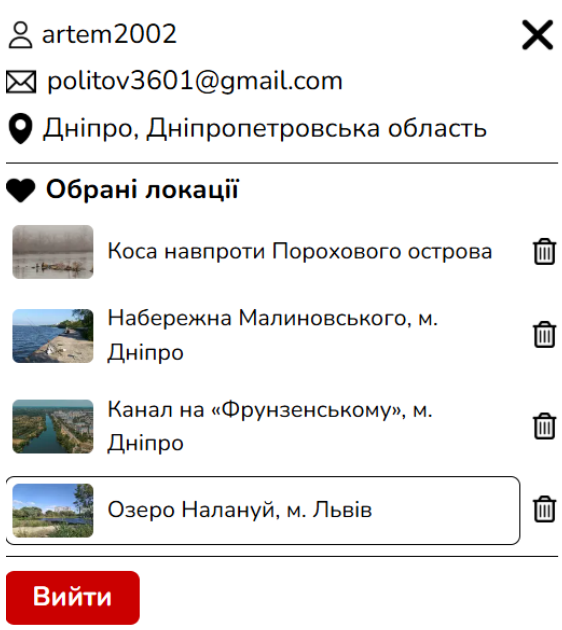

Рисунок 2 – Скріншот модального вікна профілю користувача

Усі функції працюють у реальному часі, завдяки динамічному обміну даними з API через RTK Query, без потреби у перезавантаженні сторінки.

2.3. Сценарії використання

2.3.1. Перший запуск застосунку (неавторизований користувач)

Користувач відкриває застосунок та потрапляє на головну сторінку. У верхній частині екрана він бачить хедер з логотипом зліва, а праворуч — перемикачі теми, мови та кнопку «Увійти».

На лівій боковій панелі поки що відображаються лише заголовки розділів «Кращий кльов поблизу» та «Локації поблизу», але дані в них ще не доступні.

У центральній частині розташовані два фрейми:

- «Погода» — користувач обирає область і населений пункт, після чого бачить поточні погодні умови та прогноз.
- «Кльов риби» — після вібору міста користувач може вибрати рибу, переглянути її опис, поточний рівень кльову та погодинний прогноз на кілька діб.

2.3.2. Авторизація користувача

Натиснувши кнопку «Увійти», користувач бачить модальне вікно з формою авторизації. Звідси можна перейти до реєстрації: заповнити ім’я, email, обрати область і населений пункт, задати пароль. Поля перевіряються на валідність.

Після успішної реєстрації відкривається вікно входу з підтвердженням. Після входу користувач автоматично повертається на головну сторінку, де:

- встановлюється його обраний населений пункт;
- заповнюються блоки «Кращий кльов» і «Локації поблизу»;
- кнопка «Увійти» замінюється на ім’я користувача.

2.3.3. Робота з профілем користувача

Клік по імені користувача відкриває модальне вікно профілю з правого боку. У ньому відображається особиста інформація (ім’я, email, область, населений пункт) та список обраних локацій.

Кожна локація у списку дозволяє:

- перейти на відповідну сторінку локації;
- видалити її з обраного.

2.3.4. Перегляд сторінки локації

Користувач відкриває сторінку однієї з риболовних локацій (через бокову панель або з профілю). Сторінка завантажується без перезавантаження застосунку.

На ній відображаються:

- основна інформація: зображення, опис, карта та кнопка повернення;
- прогноз кльову для риб у цій локації — всі доступні риби сортуються за поточним рівнем активності.

Якщо користувач авторизований, він бачить іконку серця, що дозволяє додати локацію до обраного.

2.3.5. Перемикання теми та мови

Незалежно від авторизації, користувач у будь-який момент може перемкнути тему (світла/темна) або мову (українська/російська). Зміни застосовуються миттєво, без перезавантаження сторінки.

3. API ТА СТРУКТУРА БАЗИ ДАНИХ

У проєкті FishCast реалізовано REST API за допомогою Express.js та бази даних MongoDB з використанням ODM-бібліотеки Mongoose. Дані з клієнтської частини надсилаються через HTTP-запити до відповідних маршрутів сервера, де обробляються контролерами, взаємодіють із моделями бази даних і повертають результат.

3.1. Авторизация та користувачі

Для аутентифікації використовується JWT (JSON Web Token). Після успішного входу користувач отримує токен, який зберігається на клієнті (у localStorage) і передається в заголовку Authorization для захищених запитів.

- POST /register. Створення нового користувача: пароль хешується через bcrypt, перевіряється унікальність email, зберігаються ім’я, email, регіон/місто та порожній список обраних локацій.
- POST /login. Авторизація: перевірка логіна та пароля, у разі успіху — видача JWT.
- GET /me. Повертає дані поточного користувача (за токеном). Здійснюється запит до MongoDB через User.findById(), пароль виключається, favoriteLocations підвантажуються через .populate().
- POST /favorite/:locationId. Додає локацію до обраного. Якщо локація ще не в масиві favoriteLocations користувача — додається.
- DELETE /favorite/:locationId. Видаляє локацію з обраного.

Всі захищені маршрути використовують authMiddleware, який дістає токен з заголовка, перевіряє його валідність (jwt.verify) та додає req.user з ID користувача до запиту.

3.2. API для довідкових даних: локації, риби, регіони

Ці ендпоінти забезпечують клієнт актуальними довідковими даними, необхідними для роботи застосунку:

- Локації (/api/locations) — містять назву водойми, координати, область, місто та популярні види риб у цій локації.
- Види риб (/api/fish) — включають назву риби, короткий опис, фото та погодні умови, за яких кльов найімовірніше.
- Регіони та міста (/api/regions, /api/cities) — дозволяють будувати ієрархію вибору геолокації вручну, якщо користувач не надає доступ до геопозиції.

Для обробки запитів використовуються стандартні методи find, findOne, findById з бібліотеки Mongoose. У разі потреби реалізовано фільтрацію за параметрами (наприклад, ?regionId= або ?cityId= для локацій).

3.3. Робота з MongoDB

Дані зберігаються в таких колекціях:

- users — користувачі з авторизацією та обраними локаціями.
- locations — риболовні місця з координатами.
- fish — види риб.
- regions / cities — довідники адміністративних одиниць.

Взаємозв’язки реалізовані через ID:

- location.regionId → regions
- location.cityId → cities
- user.favoriteLocations → locations

3.4. Стек технологій бекенду

- Express.js — серверна логіка, роутінг
- MongoDB + Mongoose — база даних і моделювання
- bcryptjs — хешування паролів
- jsonwebtoken — генерація та перевірка токенів
- dotenv — зберігання конфігурацій (наприклад, секретного ключа JWT)

4. РЕАЛІЗАЦІЯ КЛІЄНТСЬКОЇ ЧАСТИНИ

4.1. Загальна архітектура клієнтської частини

Клієнтська частина застосунку FishCast реалізована на базі Next.js 14, що поєднує серверний та клієнтський рендеринг для високої продуктивності та базової SEO-оптимізації.

Структура проєкту модульна: кожен компонент має власну папку зі стилями на SCSS, що забезпечує ізольованість і зручність підтримки. Навігація побудована на App Router, з розмежуванням публічних і захищених сторінок. Доступ до приватного контенту надається лише після авторизації.

Стан керується через Redux Toolkit, а запити до API — через RTK Query з кешуванням, повторним використанням запитів і контролем завантаження. Для локального стану застосовуються useState або useReducer.

Функціональність включає перегляд погоди й прогнозу кльову за вибраним містом, пошук риболовних локацій у списку або на інтерактивній мапі. Авторизовані користувачі можуть зберігати улюблені локації та бачити персоналізовану інформацію.

Застосунок підтримує мультимовність (українська та російська мови), реалізовану вручну без сторонніх бібліотек. Також доступні світла та темна теми, вибір яких зберігається у localStorage.

Інтерфейс повністю адаптивний, оптимізований для різних типів пристроїв. Захищені маршрути перевіряють наявність токена, який додається до HTTP-запитів автоматично.

Архітектура клієнта орієнтована на продуктивність, масштабованість і зручність використання.

4.2. Керування станом за допомогою Redux Toolkit та RTK Query

У проєкті використовується **Redux Toolkit** для централізованого керування станом застосунку та RTK Query — для ефективної роботи з API без потреби у ручному збереженні кешу чи статусів запитів.

Стан у Redux Toolkit розділений на логічні слайси:

- userSlice — зберігає дані авторизованого користувача та статус авторизації;
- citySelectionSlice, regionsDataSlice — містять дані обраної області та міста;
- fishDataSlice — дані про обрану рибу;
- themeSlice — керування темною/світлою темою;
- localizationSlice — керування мовою інтерфейсу (реалізовано без сторонніх бібліотек).

Слайси реалізовані через createSlice, а передача даних у стейт — через PayloadAction. Для глобального зберігання використано окремий файл store.ts, з підключенням middleware RTK Query.

У Redux зберігається переважно глобальний стан, який має значення для багатьох компонентів (наприклад, дані про користувача чи тему). Локальні стани (useState) використовуються для модалок, тимчасових інпутів тощо.

Для інтеграції з бекендом створено окремі сервіси API **RTK Query** (userApi, weatherApi, fishApi, locationsApi, regionsApi, cityApi), що дозволяють взаємодіяти з базою даних та відправляти запити на сервер.

Токен зберігається у localStorage, зчитується безпосередньо при виконанні запитів.

RTK Query автоматично кешує відповіді. Також у запиті getUserData використовується параметр refetch() для повторного завантаження даних після авторизації. Власне кешування, providesTags, invalidatesTags або polling у проєкті наразі не застосовуються.

4.3. Обробка даних з OpenWeather API

У проекті використовується OpenWeather API для отримання метеоданих. Реалізовано два основні запити: поточна погода (/weather) та погодинний прогноз на 4 дні (/forecast). Запити виконуються за координатами населеного пункту або риболовної локації, які зберігаються у базі даних.

Обробка запитів реалізована за допомогою RTK Query — з використанням createApi та автоматичної генерації хуків. Отримані дані кешуються та оновлюються згідно з політикою RTK Query. Інтерфейси відповіді строго типізовані, що спрощує обробку даних у компонентах.

Отримані метеопоказники (температура, вологість, тиск, швидкість і напрямок вітру, хмарність, опади тощо) використовуються для формування погодної інформації, відображення у UI та подальшого аналізу — зокрема для прогнозу кльову риби.

4.4 Алгоритм розрахунку ймовірності кльову

Алгоритм розрахунку ймовірності кльову базується на аналізі погодних параметрів, отриманих з API погоди. Основні вхідні дані включають температуру повітря, атмосферний тиск, вологість, швидкість та напрямок вітру, а також час доби а також вид риби.

На основі цих даних застосовується набір правил, що відображають вплив кожного параметру на активність риби. Наприклад, зміна тиску в певному діапазоні збільшує шанси кльову, оптимальна температура води підтримує активність, а сильний вітер навпаки знижує ймовірність.

В алгоритмі передбачена нормалізація вхідних значень, після чого обчислюється сумарний індекс активності що представляє собою число від 0 до 5. На основі цього числа генерується шкала кльову що представляє собою іконки з прозорими та непрозорими рибками.

Алгоритм є досить спрощеним і базується на традиційних емпіричних знаннях, проте дозволяє надати користувачу орієнтовний прогноз кльову для планування риболовлі.

4.5. Механізм авторизації на клієнті

Після успішної авторизації клієнт отримує accessToken, який зберігається в localStorage. Дані користувача підтягуються через запит getUserData (endpoint /users/me) з передачею токена в заголовку Authorization. Отримані дані зберігаються у Redux-стейті через setAuthorizedUserData, статус авторизації фіксується прапором isUserAuthorized.

При ініціалізації застосунку виконується перевірка токена через запит до /api/check-auth. Якщо токен дійсний — користувач вважається авторизованим, дані записуються в Redux store, і місто користувача одразу використовується як активне місто в інтерфейсі. У разі недійсного токена (401) — токен видаляється, викликається userLogout, і користувач вважається неавторизованим.

Перед відправкою форм реєстрації та авторизації виконується базова валідація на клієнті: перевірка на заповненість, формат email, довжину пароля, наявність вибраного населеного пункту (для реєстрації).

Реєстрація (/users/register) вимагає обов’язкового вибору населеного пункту (місто/область). Перед відправкою форм реєстрації та авторизації виконується базова валідація: перевірка на заповненість, формат email, довжину пароля, наявність обраного міста.

4.6. Обрані локації.

Зареєстровані користувачі мають можливість додавати риболовні локації до списку обраного. Цей функціонал реалізований за допомогою двох ендпоінтів у userApi:

- addLocationToFavorites — додає локацію до обраного (POST-запит на favorite/:locationId з токеном у заголовку),
- removeLocationFromFavorites — видаляє локацію з обраного (DELETE-запит на favorite/:locationId з токеном).

У компонентах використовується логіка, що перевіряє, чи поточна локація вже є в списку обраного (userData.favoriteLocations). При натисканні на кнопку додавання або видалення з обраного виконується відповідний запит, а після нього — оновлення даних користувача через refetchUserData(). Це забезпечує миттєве оновлення інтерфейсу без перезавантаження сторінки.

Для відображення обраних локацій (наприклад, у модальному вікні профілю) дані з userData.favoriteLocations зіставляються з усіма наявними локаціями, що дозволяє показати користувачу тільки ті, які він додав до обраного.

4.7. Підтримка темної/світлої теми

Застосунок підтримує перемикання між темною та світлою темами. Поточна тема зберігається в Redux-стейті (themeSlice.currentTheme). Для перемикання використовується action setTheme.

У компонентах стан теми читається через useSelector, а стилі елементів змінюються динамічно за допомогою умовних класів.

Зміна теми зберігається в localStorage і ініціалізується при старті додатку, завдяки чому налаштування зберігаються між сесіями користувача.

4.8. Локалізація інтерфейсу та підтримка кількох мов

Інтерфейс застосунку підтримує дві мови: українську та російську. Поточна мова зберігається в Redux store, а також дублюється у localStorage, що дозволяє зберігати мовні налаштування користувача між сесіями. Зміна мови здійснюється через відповідний action у Redux, після чого інтерфейс динамічно оновлюється.

Текстові елементи застосунку локалізовані вручну на основі значення поточної мови. Для динамічного контенту, такого як описи риб чи назви водоемів, у базі даних передбачено поля для кожної з підтримуваних мов.

5. ВІЗУАЛІЗАЦІЯ ДАНИХ ТА АДАПТИВНІСТЬ ІНТЕРФЕЙСУ

У додатку реалізована візуалізація погодних умов та прогнозу кльову риби на основі даних, отриманих із зовнішніх API. Відображаються поточні погодні параметри, включаючи температуру повітря, опис стану погоди, іконку, напрямок та швидкість вітру, вологість, хмарність та тиск. Також є погодинний прогноз, який дозволяє користувачеві простежити зміни погодних умов протягом доби та кількох наступних днів.

Прогноз кльову розраховується за власною формулою, в якій враховуються метеоумови, що впливають на активність риби: температура

повітря, атмосферний тиск, хмарність, швидкість, напрям вітру та ін. Результат подається у вигляді шкали, що відображає рівень кльову від 0 до 5 у вигляді іконок. Це дозволяє оцінити ймовірність успішної риболовлі в конкретний час та місці.

На сторінці рибальської локації додатково вбудована інтерактивна карта, що відображає координати вибраної водойми. Карта підвантажується через iframe з використанням OpenStreetMap та автоматично масштабується за координатами. Маркер на карті вказує точне розташування водойми, що допомагає користувачу зорієнтуватися.

Інтерфейс програми адаптований під різні розміри екранів за допомогою медіазапитів. Всі елементи коректно відображаються як на десктопних, так і мобільних пристроях, забезпечуючи зручний користувацький досвід.

6. ВИСНОВКИ ТА ПЛАНИ НА РОЗВИТОК

Робота над цим проєктом стала для мене важливим етапом у закріпленні й поглибленні знань з фронтенд-розробки. Я вже мав деякий досвід роботи з React, Redux і базовий рівень володіння TypeScript, тому проєкт дозволив мені краще структурувати знання, навчитися застосовувати їх у комплексному додатку та впевненіше почуватися у роботі з типізацією. Також я вперше почав використовувати RTK Query — інструмент, який значно спростив роботу з API-запитами та зменшив обсяг шаблонного коду. Окремо варто відзначити знайомство з Next.js, який я раніше не використовував у реальних проєктах, та роботу з серверною частиною: налаштування Express-сервера, підключення MongoDB через mongoose, реалізацію авторизації та захисту маршрутів.

У планах на розвиток додатку — додати можливість створення нових риболовних локацій через графічний інтерфейс, реалізувати автоматичне визначення геолокації користувача для покращення персоналізації, а також удосконалити формулу прогнозу кльову, зробивши її більш адаптивною до реальних погодних умов і різних видів риб.