**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**імені ІГОРЯ СІКОРСЬКОГО»**

факультет інформатики та обчислювальної техніки

(повна назва інституту/факультету)

кафедра інформаційних систем та технологій

(повна назва кафедри)

|  |  |
| --- | --- |
|  |  |

**Курсова робота**

з дисципліни «Програмування. Частина 2. Структури даних та алгоритми»

на тему: Веб-сайт прогноз погоди

Виконав : студент \_\_1\_ курсу, групи \_\_\_\_ІС-21\_\_\_\_\_

(шифр групи)

Лях Артемій Володимирович \_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові) (підпис)

Науковий керівник \_\_\_а.к. ІСТ ФІОТ В.О. Міщенко\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

Лях А. В \_\_\_\_\_\_\_\_\_

(підпис)

Київ – 2023

[ВСТУП 3](#_Toc136201302)

[1 ВИМОГИ ДО СИСТЕМИ 4](#_Toc136201303)

[**1.1 Функціональні вимоги до системи** 4](#_Toc136201304)

[**1.2 Нефункціональні вимоги до системи** 4](#_Toc136201305)

[2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ 5](#_Toc136201306)

[**2.1 Діаграма прецедентів** 5](#_Toc136201307)

[**2.2 Опис сценаріїв використання системи** 6](#_Toc136201308)

[3 АРХІТЕКТУРА СИСТЕМИ 11](#_Toc136201309)

[4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ 17](#_Toc136201310)

[**4.1 Загальна структура проекту** 17](#_Toc136201311)

[**4.2 Компоненти рівня доступу до даних** 20](#_Toc136201312)

[**4.3 Компоненти рівня бізнес-логіки** 21](#_Toc136201313)

[**4.4 Компоненти рівня інтерфейсу користувача** 23](#_Toc136201314)

[ВИСНОВКИ 26](#_Toc136201315)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 27](#_Toc136201316)

# ВСТУП

Мета розробки веб-сайту прогнозу погоди полягає в наданні користувачам точної й актуальної інформації про погодні умови для конкретного регіону або місцезнаходження. Основною метою є забезпечення користувачів надійною погодною інформацією, яка допоможе їм приймати обґрунтовані рішення щодо планування активностей, вибору одягу, безпеки тощо.

Актуальність розробки прогнозу погоди пояснюється наступними факторами:

1. Зростаюча потреба в точній інформації про погоду: Погода має значний вплив на наше повсякденне життя і прийняття рішень. Люди шукають надійну погодну інформацію для планування подорожей, організації заходів, сільськогосподарської діяльності та багатьох інших сфер.
2. Технологічний розвиток: Завдяки зростанню технологій Інтернету та мобільних пристроїв, доступ до погодної інформації став більш доступним для користувачів. Розробка веб-сайту прогнозу погоди є актуальною, оскільки вона надає зручний та швидкий доступ до актуальних даних про погоду.
3. Точність прогнозування: Сучасні технології прогнозування погоди стають все точнішими і надійнішими. Проте, щоб ці прогнози були доступними для публіки, виникає потреба в створенні інтерфейсу, який зможе легко передати інформацію користувачам.

Отже, щоб забезпечити зручність у використанні точної й актуальної інформації про погодні умови, необхідно створити веб-сайт, в якому користувач має можливість переглядати прогнози, а адміністратор може створювати або видаляти публікації.

# 1 ВИМОГИ ДО СИСТЕМИ

## **1.1 Функціональні вимоги до системи**

1. Створення прогнозу погоди:
   1. Адміністратор створює новий прогноз погоди, вводячи всі необхідні дані: регіон, місто, температуру, стан погоди, вологість, опади, видимість, дату.
2. Видалення прогнозу погоди:
   1. Адміністратор має таблицю зі всіма даними й може видаляти будь-який рядок із прогнозом погоди.
   2. У користувача в пошуку автоматично видаляється прогноз погоди.
3. Пошук прогнозу погоди:
   1. Користувач вводить в полі пошуку регіон, місто, дату й отримує список прогнозів.
   2. Якщо прогнозів за вказаними даними не існує, повідомити про це користувача.

## **1.2 Нефункціональні вимоги до системи**

* Авторизація. Щоб передивлятися, або створювати прогнози погоди, користувач має пройти авторизацію.
* Веб-інтерфейс. Система має бути реалізована, як веб-сайт.
* Відкритий вихідний код. Щоб в подальшому розвивати проєкт й зробити його відкритим для користувачів й розробників, система має бути опублікована на GitHub, або інші веб-застосунки.
* Безпека. Система повинна забезпечити безпеку інформації користувача, а також повинна розділяти права користувачів на користувацькі й адміністраторські.
* Масштабованість. Архітектура системи повинна побудована так, щоб додавання, або оновлення сервісу в майбутньому, не вплинуло на виконання вже створеного функціоналу.

# 2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

## **2.1 Діаграма прецедентів**

Діаграма прецедентів системи представлена на рис. 2.1.

Акторами є користувачі системи: незареєстрований користувач (гість), зареєстрований користувач та адміністратор.

Незареєстрований користувач не має доступу до прогнозів погоди. Це зроблено для того, щоб доступний функціонал був розділений тільки між зареєстрованим користувачем й адміністратором.

Зареєстрований користувач має змогу переглядати прогнози погоди й виходити з облікового запису.

Адміністратор, як і зареєстрований користувач, має змогу переглядати прогнози погоди, виходити з облікового запису, а також може створювати й видаляти прогнози.

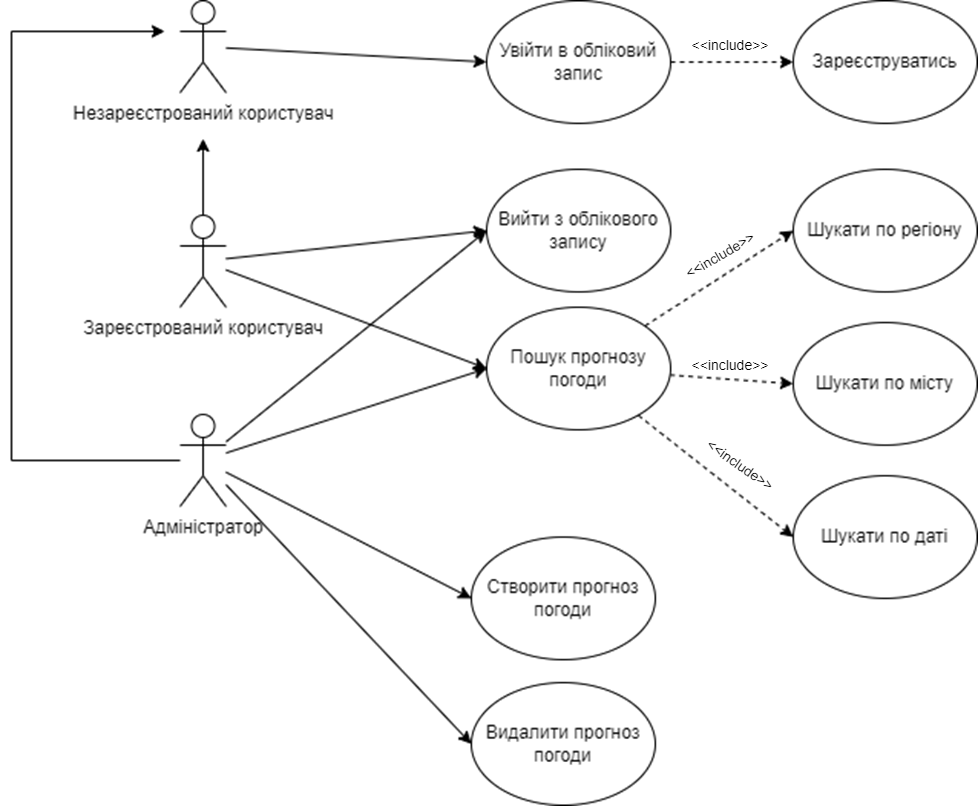


Рис. 2.1 –­­­­­­ Діаграма прецедентів

## **2.2 Опис сценаріїв використання системи**

Детальні описи сценаріїв використання наведено у таблицях 2.1 – 2.5.

Таблиця 2.1 – Сценарій використання «Реєстрація облікового запису»

|  |  |
| --- | --- |
| Назва | Реєстрація облікового запису |
| ID | 1 |
| Опис | Щоб передивлятися інформацію про прогноз, незареєстрований користувач повинен створити обліковий запис. |
| Актори | Гість |
| Вигоди компанії | Якщо гість не має бажання реєструватися то він не буде користуватись даним веб-сайтом. |
| Частота користування | Один раз |
| Тригери | Гість вводить електронну пошту й пароль. |
| Передумови | При переході на веб-сайт в перший раз, гість бачить головну сторінку й кнопку “Log in”. При переході на форму авторизації, гість бачить “Not registered”, натискає й переходить на форму реєстрації. |
| Постумови | Користувач потрапляє на головну сторінку, тепер на місці “Log in” він бачить статус “USER”, або “ADMIN”. |
| Основний розвиток | Гість створює обліковий запис й тепер має можливість переглядати прогнози погоди. |
| Альтернативні розвитки | — |
| Виняткові ситуації | — |

Таблиця 2.2 – Сценарій використання «Вхід в обліковий запис»

|  |  |
| --- | --- |
| Назва | Вхід в обліковий запис |
| ID | 2 |
| Опис | Щоб передивлятися інформацію про прогноз, або додавати/видаляти дані, користувач повинен ввійти в свій обліковий запис. |
| Актори | Користувач, адміністратор |
| Вигоди компанії | Вхід в обліковий запис для перегляду, або редагування прогнозів. |
| Частота користування | Рідко, оскільки сервіс запам’ятав користувача, й наступного разу автоматично зареєструє його. |
| Тригери | Гість вводить електронну пошту й пароль. |
| Передумови | Якщо користувач не входив в обліковий запис, або вийшов з нього, то щоб користуватися інформацію про погоду, він має в меню нажати на “Log in” й заповнити форму входу в обліковий запис. |
| Постумови | Користувач потрапляє на головну сторінку й тепер може передивлятись інформацію про прогноз погоди. |
| Основний розвиток | Користувач входить в обліковий запис й тепер має можливість переглядати прогнози погоди. |
| Альтернативні розвитки | Користувач намагається зайти на неіснуючий обліковий запис, вводячи невірний логін, або пароль. |
| Виняткові ситуації | Якщо користувач раніше входив в обліковий запис, то йому більше не потрібно авторизовуватись. |

Таблиця 2.3 – Сценарій використання «Пошук прогнозу погоди»

|  |  |
| --- | --- |
| Назва | Пошук прогнозу погоди |
| ID | 3 |
| Опис | Пошук прозгозу погоди за датою, регіоном, містом |
| Актори | Користувач, адміністратор |
| Вигоди компанії | Користувачі будуть постійно заходити на сторінку пошуку погоди. |
| Частота користування | Постійно |
| Тригери | Користувач перешодить на сторінку /forecast |
| Передумови | Після потрапляння на головну сторінку користувач в меню нажимає “Forecast”. |
| Постумови | Користувач потрапляє на сторінку пошуку погоди. |
| Основний розвиток | Користувач може отримати необхідну інформацію про погоду у вказаному місті, регіоні та визначеній даті. |
| Альтернативні розвитки | — |
| Виняткові ситуації | Якщо користувач не зареєстрований, то він не зможе перейти на сторінку прогнозів погоди. |

Таблиця 2.4 – Сценарій використання «Додавання прогнозу погоди»

|  |  |
| --- | --- |
| Назва | Додавання прогнозу погоди |
| ID | 4 |
| Опис | Додавання адміністратором прогнозу погоди |
| Актори | Адміністратор |
| Вигоди компанії | Користувачі завжди будуть отримувати актуальну інформацію про погоду. |
| Частота користування | Часто |
| Тригери | Адміністратор в меню переходить на вкладку “Admin” |
| Передумови | Адміністратор потрапляє на сторінку додавання/видалення прогнозів погоди. |
| Постумови | Новий прогноз погоди публікується й додається в таблицю. |
| Основний розвиток | Адміністратор вводить всі необхідні дані про прогноз погоди й публікує їх для користувачів. |
| Альтернативні розвитки | Адміністратор заповнив не всі поля, тому він не зможе опублікувати новий прогноз. |
| Виняткові ситуації | — |

Таблиця 2.5 – Сценарій використання «Видалення прогнозу погоди»

|  |  |
| --- | --- |
| Назва | Видалення прогнозу погоди |
| ID | 5 |
| Опис | Видалення адміністратором прогнозу погоди |
| Актори | Адміністратор |
| Вигоди компанії | Якщо адміністратор ввів некоректно дані, або прогноз неактуальний, то його можна видалити. |
| Частота користування | Часто |
| Тригери | Адміністратор на сторінці /admin видаляє прогноз погоди з таблиці. |
| Передумови | Адміністратор портапляє на сторінку додавання/видалення прогнозів погоди. |
| Постумови | Прогноз погоди видаляється з таблиці. |
| Основний розвиток | Адміністратор видаляє необхідний прогноз з таблиці. |
| Альтернативні розвитки | — |
| Виняткові ситуації | — |

# 3 АРХІТЕКТУРА СИСТЕМИ

Загальна архітектура системи наведена на рис. 3.1

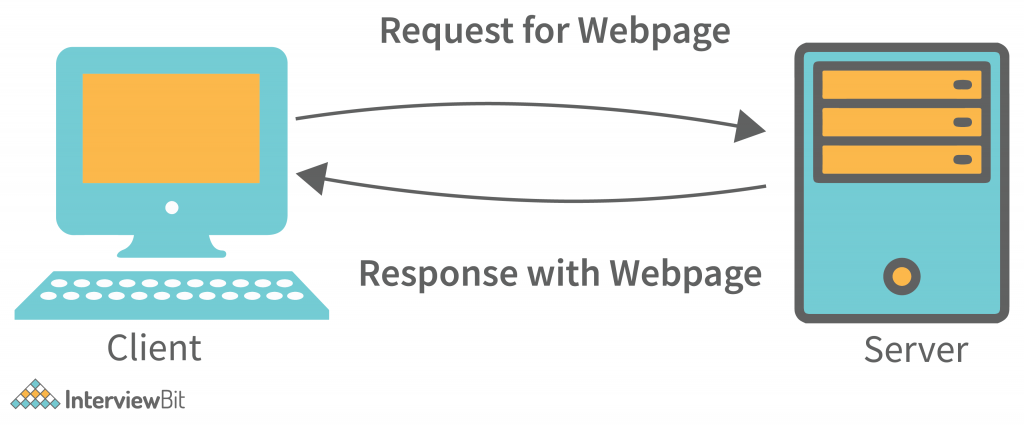


Рис. 3.1 — Загальна архітектура системи

Система розроблена на основі клієнт-серверної архітектури. На місті клієнта виступає фронтенд частина, яка взаємодіє з користувачем й відправляє запити на сервер. Сервер у свою чергу оброблює отримані дані, взаємодіє з базою даних й повертає відповідь на запит клієнта.

Фронтенд (Frontend) — інтерфейс, з яким буде взаємодіяти користувач. Він містить в собі список сторінок, форми запитів, візуальні елементи. Користувач, взаємодіючи з фронтендом, буде отримувати дані, які повертатиме бекенд. Фронтенд побудований на JavaScript з використанням фреймворку React.

Бекенд (Backend) — серверна частина проєкту, яка відповідає за обробку даних, бізнес-логіку та взаємодію з базою даних. Вона займається обробкою запитів, збереженням та отриманням інформації з бази даних, а також виконанням різних операцій, необхідних для функціонування веб-додатка. Коли користувач робить пошук, додає, або видаляє прогноз погоди, фронтенд робить запити на бекенд й відправляє необхідні дані, які бекенд буде оброблювати. Побудований бекенд на TypeScript з використанням фреймворку NestJS.

**Архітектура клієнтської частини**, розроблена на класичній реалізації, оскільки веб-сайт складається зі статичних сторінок й не має складних компонентів.

Схема класичної реалізації наведена на рис. 3.2

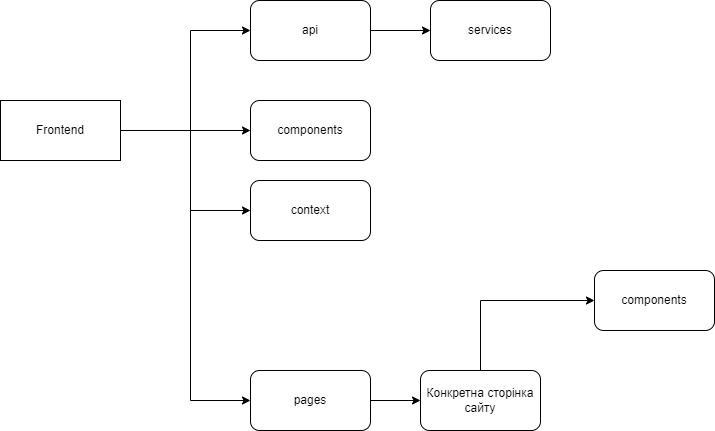


Рис 3.2 — Архітектура фронтенда

Клієнтська частина складається з:

* Pages (сторінки) ‒ кожна сторінка сайту створюється в окремій папці з однойменною назвою до URL, під яким буде ця сторінка знаходитись (наприклад, під посиланням на /forecast знаходиться сторінка, яка розташована в папці forecast).
* Components (компоненти) ‒ компоненти дозволяють організувати код й покращити його читабельність. Компоненти містять частини html коду, які можуть використовуватись в інших сторінках. (наприклад в проєкті реалізований компонент navbar (меню), який відображається на всіх сторінках сайту).
* Context (контекст) ‒ контексти використовуються для створення глобальних змінних в коді й полегшують взаємодію між різними сторінками.
* API ‒ ця частина архітектури відповідає за взаємодію клієнтської частини з серверною частиною. Для масштабованості створюються окремі сервіси, які будуть відправляти запити на о кремі частини сервера.

Під час розробки клієнтської частини були використані наступні технології:

1. React ‒ фреймворк, на якому побудована вся клієнтська частина проєкту. Ця технологія дозволяє легко створювати інтерактивні інтерфейси й ефективно й швидко оновлювати необхідні компоненти веб-сторінки.
2. React-bootstrap ‒ додатковий інструмент, який має готові елементи інтерфейсу і який взаємодіє з React. Ці елементи інтерфейсу базовані на веб-пакеті bootstrap, який містить колекцію компонентів веб-сторінки. Завдяки цьому інструменту можна швидко зробити візуальну частину сайта.
3. Axios ‒ інструмент, завдяки якому можна взаємодіяти з серверною частиною сайта. Він відправляє запити й необхідні дані на бекенд й отримує відповідь з серверу, яку можна в подальшому обробити й показати користувачу.

**Архітектура серверної частини** розроблена на основі DDD (Domain Driven Design) патерна.

DDD патерн допомагає організувати код й розбити шари серверу на незалежні компоненти. Його суть полягає в тому, щоб розділяти логіку, пов'язану з будь-яким модулем системи на шари. Модулі складаються з сервісів, контролерів й репозиторіїв. Ці компоненти не мають залежити одне від одного. Коли сервіс викликає репозиторій, репозиторій нічого не знає про сервіс. Така архітектура допомогає знизити складність впровадження нового функціоналу та модифікації в програмі. Основні шари серверної частини, які використовуються в проєкті, наведені на рис. 3.3



Рис. 3.3 ‒‒ Шари з яких складається проєкт

* Model (модель) ‒ сутність, завдяки якій будеється таблиця в базі даних й яка описує об’єкт, що повертатимтеться після запиту до цієї таблиці.
* Repository (репозиторій) ‒ компонент, що працює з базою даних, робить запити й який ніяк не взаємодіє з бізнес-логікою проєкту.
* Service (сервіс) ‒ містить в собі бізнес-логіку й функціонал, який оброблюватиме дані, що приходять з контролеру.
* Controller (контролер) ‒ відповідає за обробку вхідних запитів і повернення відповідей клієнту. Клієнт відправляє URL запит на сервер з даними, необхідними для контролера, й отримує дані оброблені сервісами.

Під час розробки клієнтської частини були використані наступні технології:

1. NestJS ‒ це платформа для створення ефективних, масштабованих програм Node.js на стороні сервера. Nest надає готову архітектуру, яка дозволяє розробникам створювати легко тестовані, масштабовані, слабозв'язані та прості в обслуговуванні додатки.
2. Mongoose ‒ це бібліотека моделювання об’єктних даних (ODM) для Node.js. Вона забезпечує простий спосіб взаємодії з базами даних MongoDB шляхом визначення схем і моделей, які представляють структуру даних.
3. Bcrypt ‒ бібліотека, що містить хеш функції. Її доцільно використовувати для хешування паролів й конфіденційних даних.
4. @nest/jwt ‒ доповнення до фреймворку Nest, яке дозволяє створювати jwt-токени для шифрування інформації про користувача. Цей токен можна використовувати для позначення звичайного користувача й адміністратора.

**Архітектура бази даних** складається з двох таблиць: User, Weather. Оскільки вони не мають реляційних зв’язків, в проєкті була використана нереляційна база даних MongoDB.

Таблиця User наведена на рис. 3.4

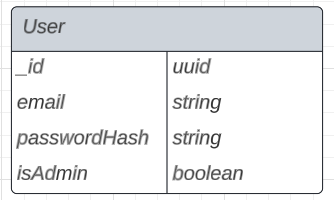


Рис 3.4 ‒‒ Таблиця User

* \_id ‒ ідентифікатор користувача
* email ‒ електронна пошта користувача.
* passwordHash ‒ захешований пароль користувача.
* isAdmin ‒ поле, яке визначає користувача, як звичайного користувача й адміна.

Таблиця Weather наведена на рис. 3.5

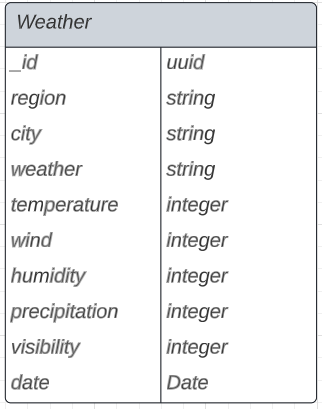


Рис 3.5 ‒‒ Таблиця Weather

* \_id ‒ ідентифікатор користувача
* region ‒ область
* city ‒ місто
* weather ‒ стан погоди (наприклад: сонячно, похмурно, дощ й т.п.)
* temperature ‒ температура
* wind ‒ швидкість вітру
* humidity ‒ вологість
* precipitation ‒ рівень опадів
* visibility ‒ видимість
* date ‒ дата, на яку робиться прогноз

# 4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

## **4.1 Загальна структура проекту**

Загальна структура проекту представлена на рис.4.1

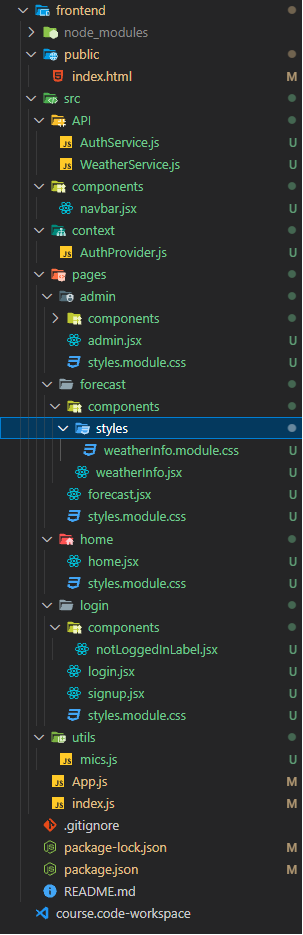
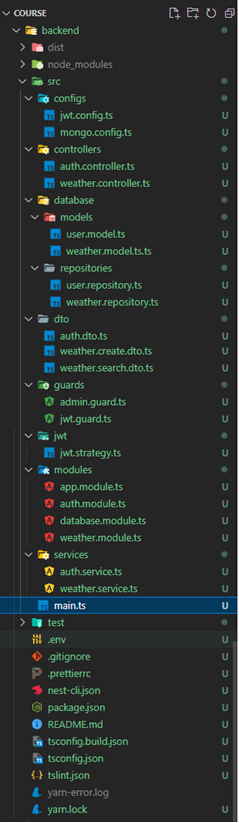


Рис. 4.1 ‒‒ Загальна структура проєкту

Проєкт складається з двох загальних папок: fronted та backend.

* Frontend ‒ головна директорія клієнтської частини.
  + public ‒ директорія, в якій містяться html сторінки, завдяки яким формуються сторінки сайту. Зазвичай в ній містяться небагато сторінок, оскільки вони доповнюються компонентами, що знаходяться в інших директоріях.
  + src ‒ директорія, в якій знаходяться всі компоненти, логіка й допоміжні елементи клієнтської частини.
    - api ‒ директорія, що містить сервіси, які взаємодіють з серверною частиною.
    - components ‒ директорія, що містить глобальні компоненти, які будуть використані в усіх сторінках.
    - context ‒ директорія, що містить контексти сторінок. Контексти допомагають сторінкам взаємодіяти між собою.
    - pages ‒ директорія в якій знаходяться компоненти, що формують веб-сторінку. Кожна сторінка має свої унікалькі елементи й стилі, отже найкращим рішенням буде розбити реалізацію кожних сторінок на окремі папки.
      * admin ‒ директорія, що містить компоненти й реалізацію сторінки адміністратора.
      * forecast ‒ містить компоненти й реалізацію сторінки пошуку прогнозів погоди.
      * home ‒ містить компоненти й реалізацію головної сторінки.
      * login ‒ містить компоненти й реалізацію сторінки реєстрацій й авторизації.
    - utils ‒ містить допоміжні функції.
  + app.js ‒ головний компонент клієнту, що формує сторінки веб-сайту.
  + index.js ‒ головний файл клієнту, в якій налаштовується й запускається клієнтська частина.
* Backend ‒ директорія, що містить в собі серверну частину.
  + src ‒ директорія, в якій знаходяться всі елменти серверної частини.
    - configs ‒ директорія, в якій містяться налаштування бібліотек й бази даних.
    - controllers ‒ папка з усіма контролерами серверної частини.
    - database ‒ директорія в якій містяться моделі й репозиторії сутностей, що створюються в таблицях.
      * models ‒ моделі бази даних.
      * repositories ‒ репозиторії кожної моделі.
    - dto ‒ директорія, в якій містяться вимоги до даних, що будуть приймати контролери від клієнтської частини.
    - guards ‒ директорія, що містить обмеження для клієнтської частини у використанні деяких контроллерів.
    - jwt ‒ директорія, що містить функціонал у використанні jwt-токена. Ці токени необхідні для позначення авторизованого користувача.
    - models ‒ містить моделі серверної частини. Моделі поєднують в собі сервіси, контролери, репозиторії й таким чином будують інфраструктуру сервера.
    - services ‒ містить в собі всю бізнес-логіку проєкту.
    - main.ts ‒ головний файл проєкту, завдяки якому запускається сервер.
  + test ‒ директорія для тестування серверного функціоналу.

## **4.2 Компоненти рівня доступу до даних**

База даних побудована з двох моделей. Щоб з ними взаємодіяти, були створені окремі репозиторії до кожної моделі. Також були створені допоміжні класи в директорії dto, які виступають у якості аргументів в деяких методах сервісів й контролерів.

Основні сутності та інтерфейси рівня доступу до даних наведені на рис. 4.2

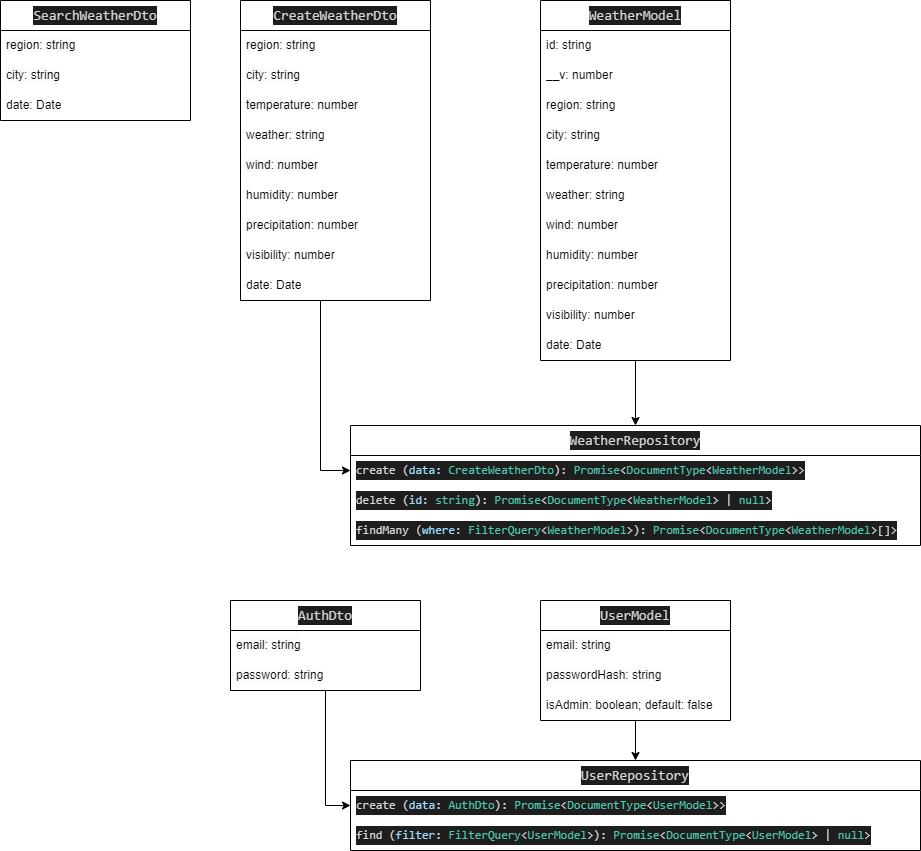


Рисунок 4.2 – Основні сутності та інтерфейси рівня доступу до даних

* UserModel ‒ використовується для опису даних користувача веб-сайту в таблиці user в базі даних.
* AuthDto ‒ описує дані, які необхідні для методів створення, регістрацї й авторизації користувача.
* UserRepositor ‒ використовується для взаємодії серверу з таблицею user в базі даних.
* WeatherModel ‒ використовується для опису елементів в таблиці weather в базі даних.
* CreateWeatherDto ‒ описує дані, які необхідні для створення прогнозу погоди.
* SearchWeatherDto ‒ описує дані, які необхідні для пошуку прогнозу погоди.
* WeatherRepository ‒ використовується для взаємодії серверу з таблицею weather в базі даних.

## **4.3 Компоненти рівня бізнес-логіки**

В бізнес-логіці реалізований весь функціонал, який необхідний для обробки даних клієнтської частини. Цей функціонал використовує методи в репозиторіях. Контролери в свою чергу використовують сервіси для передачі даних з клієнтської частини й повертають клієнту оброблені дані з сервісу.

Компоненти рівня бізнес-логіки наведені на рис. 4.3



Рис. 4.3 ‒‒ Компоненти рівня бізнес-логіки

* JwtStrategy ‒ описує налаштування jwt-токену й містить метод за який буде шифрувати дані.
* AdminGuard ‒ ставить “мітку” методам в контролерах, які може викликати тільки адміністратор.
* AuthService ‒ містить методи авторизації, валідації користувача, а також пошук й створення користувача в базі даних.
* AuthController ‒ створює посилання, за якии користувач має змогу авторизуватися, або зареєструватися.
* WeatherService ‒ містить методи пошуку, видалення, створення прогнозу погоди в базі даних.
* WeatherController ‒ створює посилання, за якии користувач, або адміністратор мають змогу створити, видалити, знайти прогнози прогоди.

Серверна частина надає список запитів, за якими взкаємодіє клієнтська частина. В таблиці 4.3.1 наведені усі можливі запити на сервер.

|  |  |  |
| --- | --- | --- |
| HTTP-метод | Запит | Опис |
| POST | /api/auth/register | Запит до реєстрації |
| POST | /api/auth/login | Запит до авторизації |
| POST | /api/weather | Запит на створення прогнозу погоди |
| GET | /api/weather | Запит на пошук прогнозу погоди |
| DELETE | /api/weather/:id | Запит на видалення прогнозу погоди. |

Таблиця 4.3.1 ‒ таблиця усіх можливих запитів на сервер

## **4.4 Компоненти рівня інтерфейсу користувача**

Наступні скріншоти описують взаємодію користувача з веб-інтерфейсом проєкту.

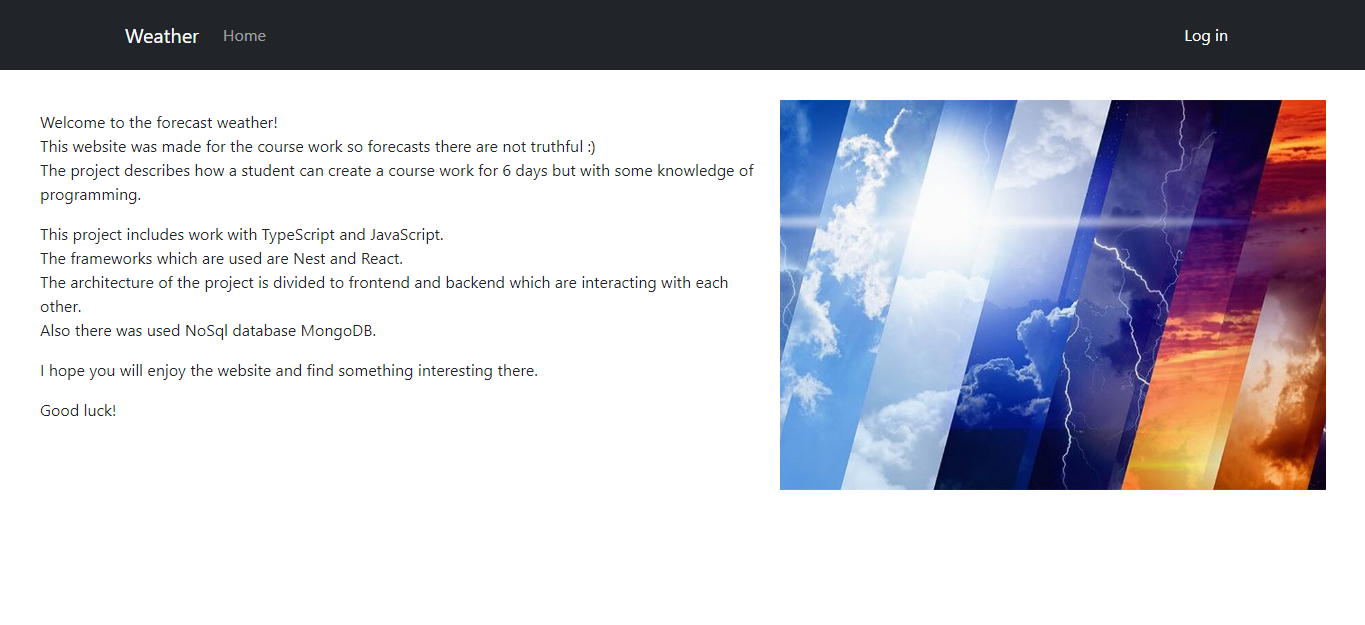


Рис. 4.4.1 ‒ головна сторінка веб-сайту незареєстрованого користувача

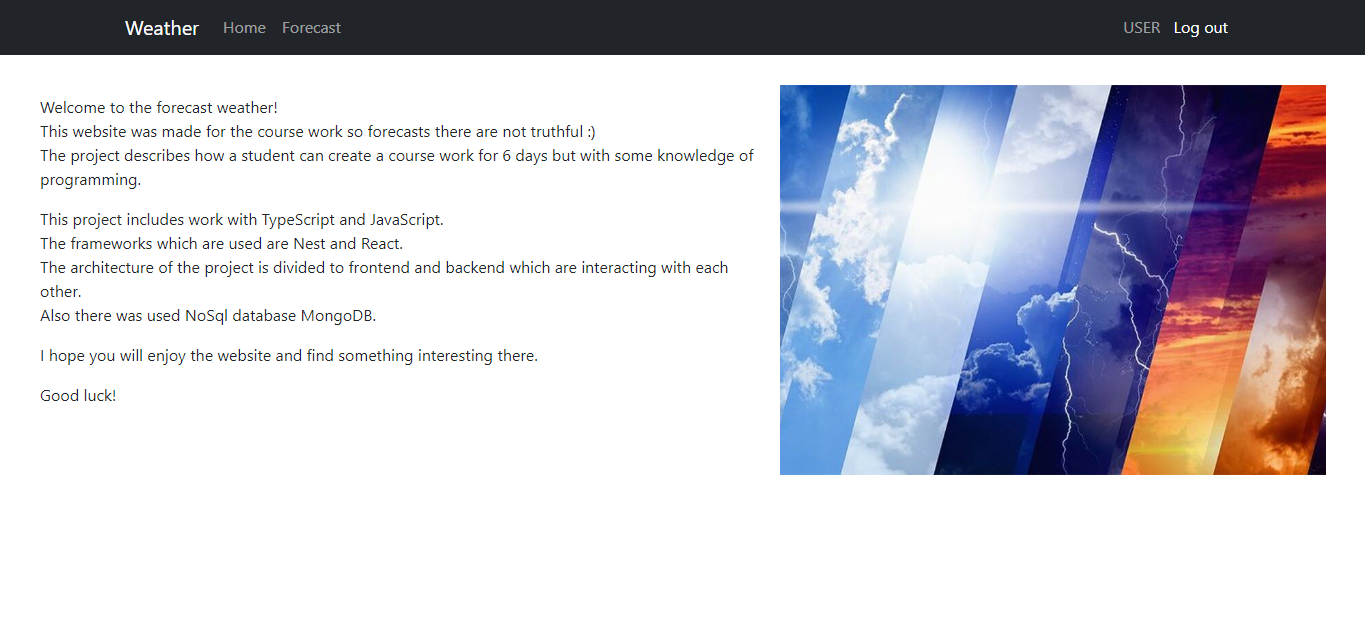


Рис. 4.4.2 ‒ головна сторінка зареєстрованого користувача

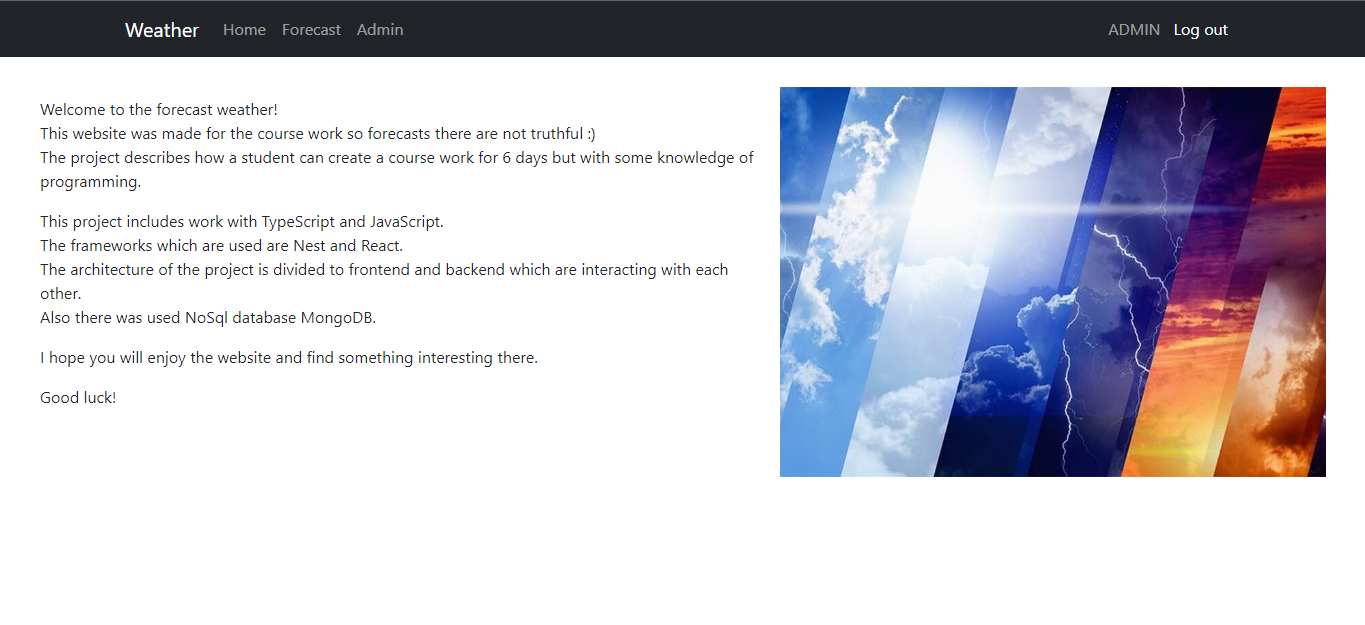


Рис. 4.4.3 ‒ головна сторінка адміністратора

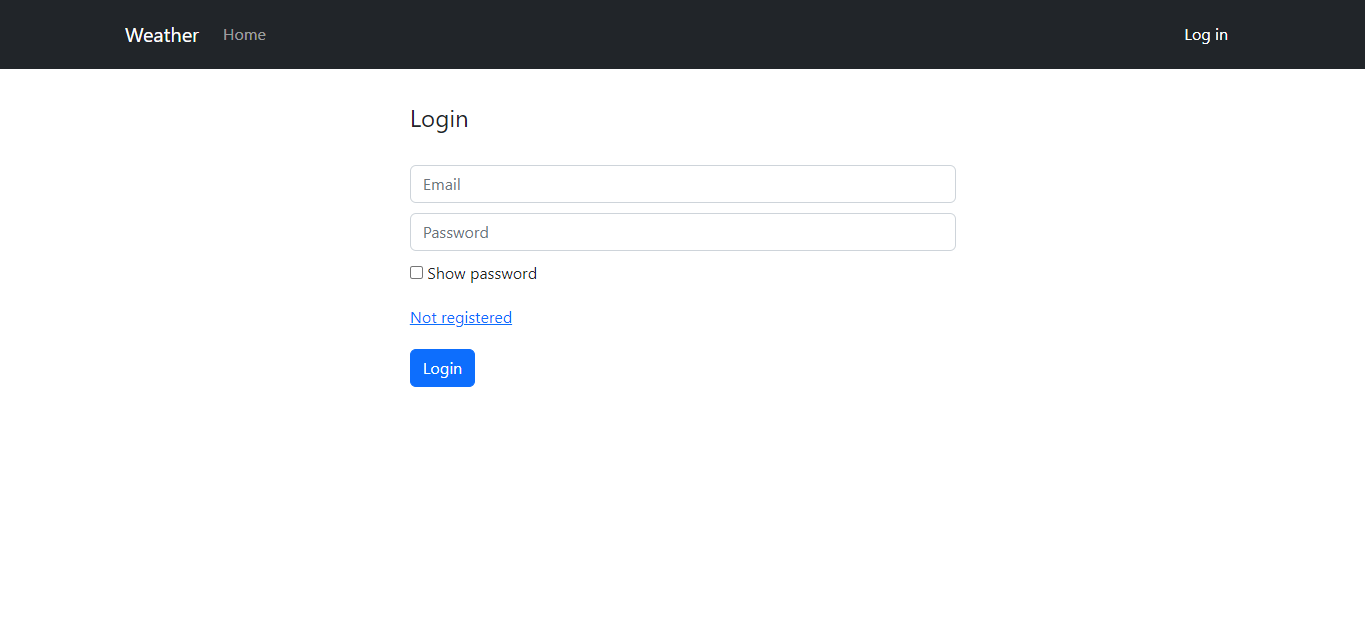


Рис. 4.4.4 ‒ форма авторизації

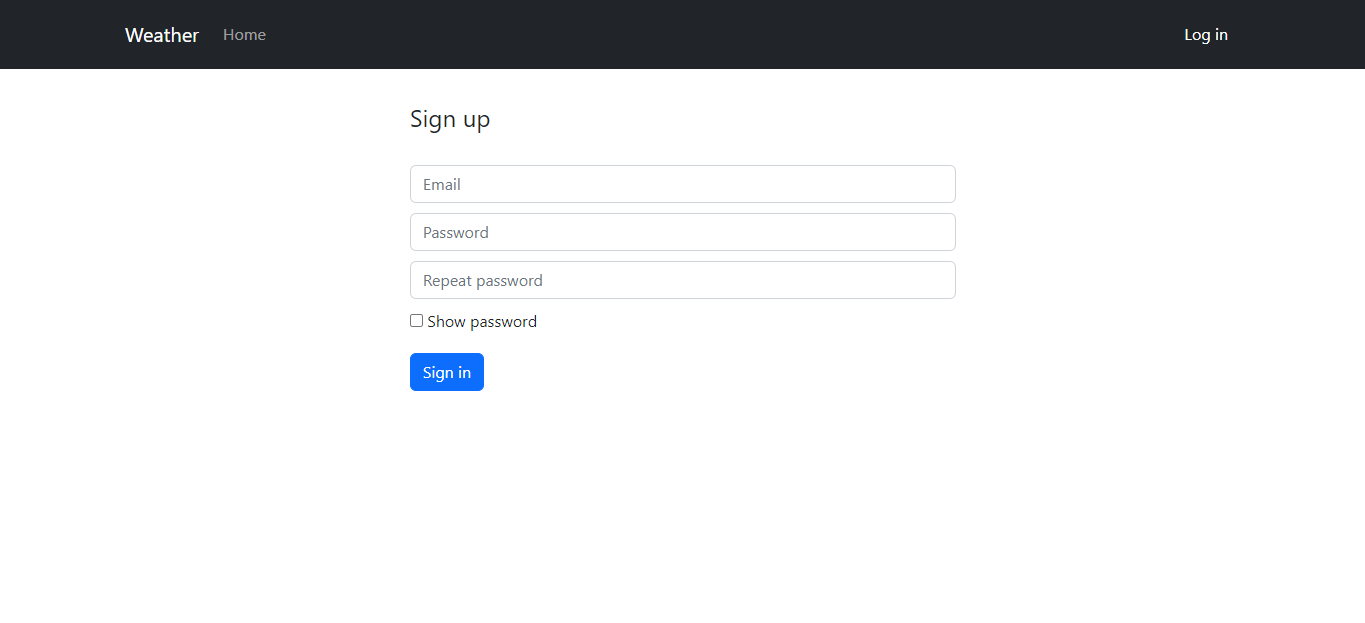


Рис. 4.4.5 ‒ форма реєстрації

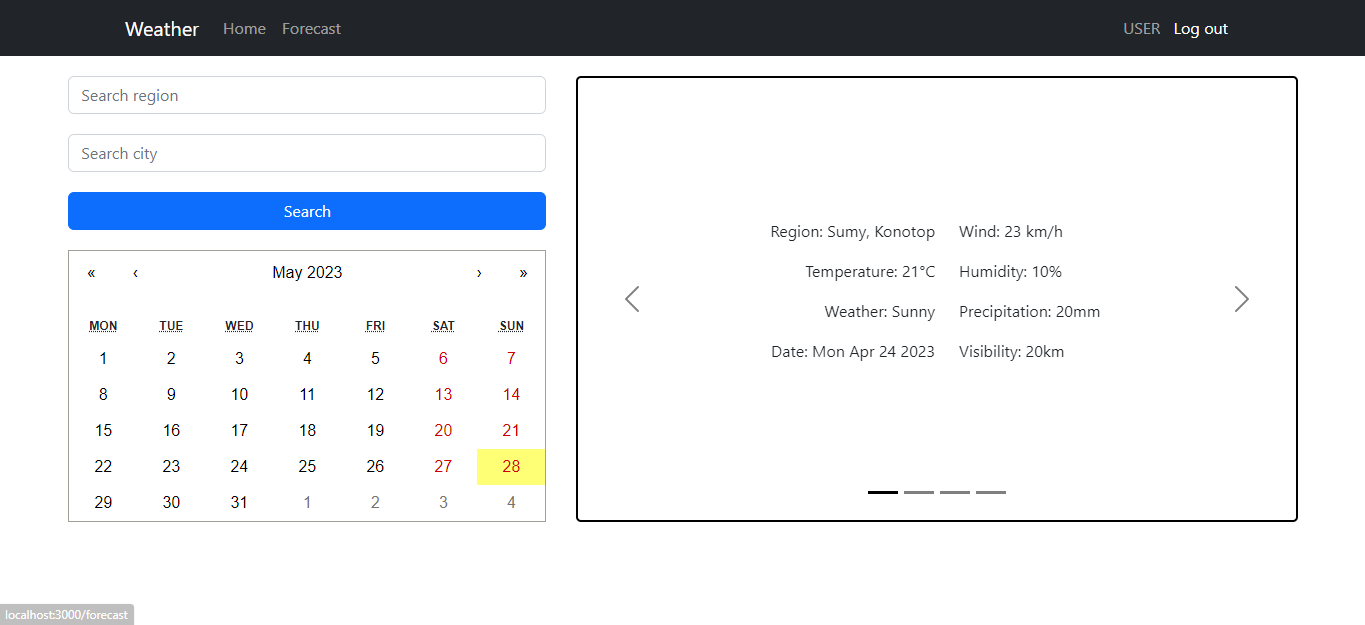


Рис. 4.4.5 ‒ сторінка пошуку прогнозу погоди

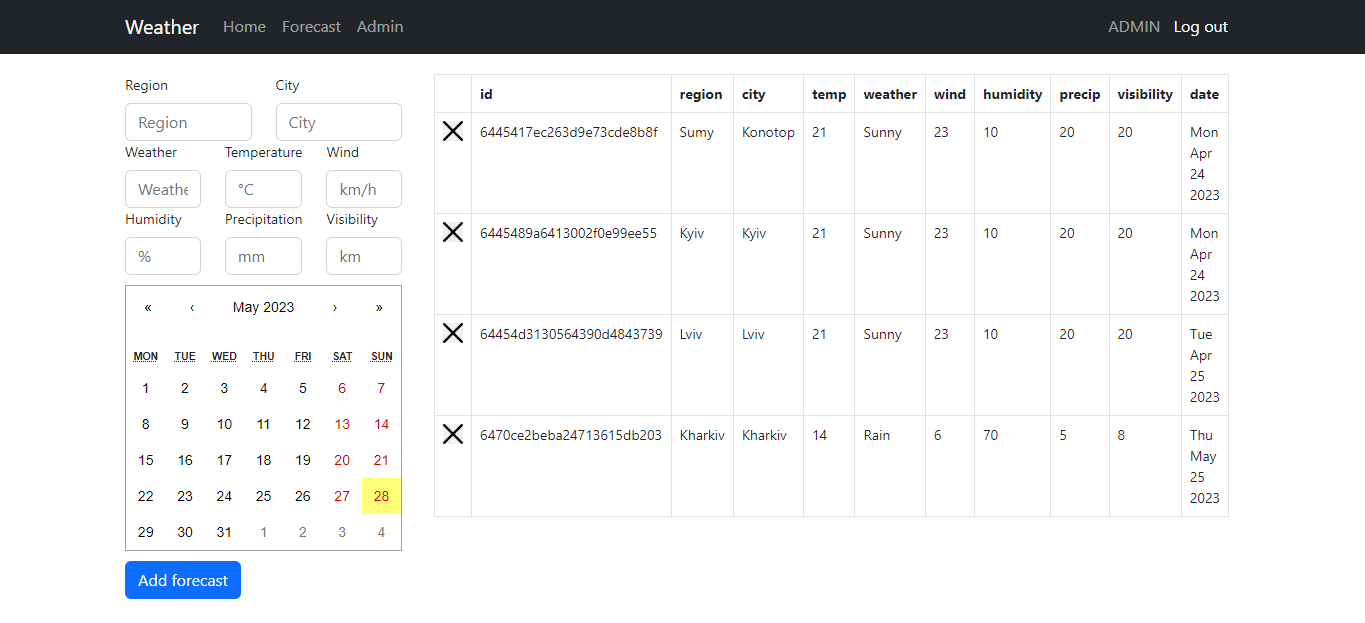


Рис. 4.4.6 ‒ сторінка адміністратора

# ВИСНОВКИ

В ході курсової роботи, було створено веб-сайт прогнозу погоди, який виконує всі необхідні функції для взаємодії з користувачем. Під час розробки проєкту, була використана клієнт-серверна архітектура з використанням сучасних фреймворків й мов програмування. Були виконані усі функціональні вимоги, а саме:

* Створення прогнозу погоди
* Видалення прогнозу погоди
* Пошук прогнозу погоди

Також були дотримані усі нефункціональні вимоги:

* Авторизація. Було створено безпечну систему авторизації, яка розділяє користувачів й надає їм права користуватися пошуком прогнозу погоди.
* Веб-інтерфейс. Система реалізована, як веб-сайт.
* Відкритий вихідний код. Весь код проєкту був опублікований на GitHub. Посилання на репозиторій знаходиться в додатковій інформації.
* Безпека. В проєкті відбувається хешування паролів, а також були використані jwt-токени для розділення користувачів на користувачів й адміністраторів.
* Масштабованість. Архітектура проєкту дозволяє створювати додатковий функціонал, який ніяк не вплине на роботу створеного.

На місті клієнтської частини виступають веб-сторінки. Було використано мову програмування JavaScript й фреймворк React, який може створювати динамічні односторінкові сайти. Цей фреймворк також допомагає полегшити написання коду й пришвидшує розробку сторінок. Також був використаний пакет веб-компонентів Bootstrap, який містить в собі готові частини сторінки з готовим дизайном. Додатково була використана бібліотека Axios, яка дає змогу клієнтської частини взаємодіяти з серверною.

На місті серверної частини було використано мову програмування TypeScript та фреймворк NestJS, який допомагає легко створити серверну частину. Сервер був побудований на патерні DDD, який розділяє окремі сутності й бізнес-логіку на модулі. Також була впроваджена база даних MongoDB в якій зберігаються дві таблиці user та weather.

Підсумовуючи, отриманий продукт є простим у використанні й має легкий функціонал. Він включає в собі всі функціональні й нефункціональні вимоги, а також він доступний для подальшого покращення й розвитку, завдяки відкритій архітектурі.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сайт прогнозу погоди [електронний ресурс] - режим доступу до ресурсу: <https://www.accuweather.com/>
2. Сучасний підручник з JavaScript [електронний ресурс] - режим доступу до ресурсу: <https://uk.javascript.info/>
3. TypeScript tutorial [електронний ресурс] - режим доступу до ресурсу: <https://youtube.com/playlist?list=PL4cUxeGkcC9gUgr39Q_yD6v-bSyMwKPUI>
4. Курс на Ua.Udemy Master NestJS 9 - Node.js Framework 2023 [електронний ресурс] - режим доступу до ресурсу: <https://ua.udemy.com/course/master-nestjs-the-javascript-nodejs-framework/>
5. Crash course з NestJS [електронний ресурс] - режим доступу до ресурсу: <https://www.youtube.com/watch?v=xzu3QXwo1BU&list=PL_cUvD4qzbkw-phjGK2qq0nQiG6gw1cKK>
6. NestJS офіційний сайт [електронний ресурс] - режим доступу до ресурсу: <https://nestjs.com/>
7. React JS фундаментальный курс от А до Я [електронний ресурс] - режим доступу до ресурсу: <https://youtu.be/GNrdg3PzpJQ>
8. React офіційний сайт [електронний ресурс] - режим доступу до ресурсу: <https://react.dev/>
9. MongoDB cloud [електронний ресурс] - режим доступу до ресурсу: <https://cloud.mongodb.com/>
10. Stackoverflow [електронний ресурс] - режим доступу до ресурсу: <https://stackoverflow.com/>
11. ChatGPT [електронний ресурс] - режим доступу до ресурсу: <https://chat.openai.com/>
12. Backend архітектура [електронний ресурс] - режим доступу до ресурсу: <https://dou.ua/forums/topic/33590/>
13. Frontend архітектура [електронний ресурс] - режим доступу до ресурсу: https://habr.com/ru/articles/667214/

ДОДАТКОВА ІНФОРМАЦІЯ

Посилання на репозиторій бекенду: <https://github.com/ArtemLyah/ForecastWeather-backend>

Посилання на репозиторій фронтенду: <https://github.com/ArtemLyah/ForecastWeather-frontend>