МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Кафедра математичної інформатики

МАГІСТЕРСЬКА РОБОТА

на тему:

Розробка веб-платформи для комунікації та обміну знаннями в спільноті бджолярів

зі спеціальності 122 ”Комп’ютерні науки”

Виконав:
Студент 2 курсу
Групи БІ-2
Андрощук Едуард Олександрович

_____ (підпис)

Науковий керівник:
доктор наук, професор
ВОЛОДИМИР Заславський

_____ (підпис)

Київ — 2025

Анотація

Магістерська робота присвячена розробці повностекового веб-застосунку *Beekeepers Community Platform*, призначеного для об’єднання бджолярів, обміну знаннями та досвідом.

У роботі проведено аналіз предметної області, розглянуто існуючі рішення для спільнот та обґрунтовано вибір технологічного стеку, що включає React для фронтенду, NestJS для бекенду та MongoDB як базу даних.

Спроектовано архітектуру системи, визначено функціональні та нефункціональні вимоги. Розроблено ключові модулі, такі як форум для обговорень, база знань, інтерактивна карта для управління пасіками та полями, а також система автентифікації користувачів з верифікацією електронної пошти та можливістю входу через Google.

Описано процес розробки, структуру коду, принципи проектування API та основні компоненти системи. Розглянуто питання безпеки та реалізовано відповідні механізми.

Проведено тестування основних функціональних можливостей застосунку. За результатами роботи сформульовано висновки та окреслено напрямки для подальшого розвитку платформи.

Ключові слова: веб-застосунок, спільнота бджолярів, React, NestJS, MongoDB, форум, база знань, інтерактивна карта, автентифікація, OAuth.

Abstract

The Master's thesis is devoted to the development of the full-stack web application *Beekeepers Community Platform*, designed to unite beekeepers for knowledge and experience exchange.

The work includes an analysis of the subject area, a review of existing solutions for communities, and a justification for the chosen technology stack, which includes React for the frontend, NestJS for the backend, and MongoDB as the database.

The system architecture was designed, and functional and non-functional requirements were defined. Key modules have been developed, such as a forum for discussions, a knowledge base, an interactive map for managing apiaries and fields, and a user authentication system with email verification and Google Sign-In capability.

The development process, code structure, API design principles, and main system components are described. Security considerations were addressed, and appropriate mechanisms were implemented.

Testing of the main functional capabilities of the application was conducted. Based on the results, conclusions were formulated, and directions for further platform development were outlined.

Keywords: web application, beekeepers community, React, NestJS, MongoDB, forum, knowledge base, interactive map, authentication, OAuth.

Зміст

Перелік рисунків

Перелік таблиць

Вступ

Актуальність теми магістерської роботи зумовлена зростаючою потребою у спеціалізованих онлайн-платформах для нішевих спільнот, зокрема для бджолярів. Бджільництво є важливою галуззю сільського господарства та екології, і ефективний обмін знаннями, досвідом та оперативною інформацією між пасічниками може суттєво сприяти його розвитку. Існуючі загальні соціальні мережі та форуми не завжди враховують специфічні потреби бджолярської спільноти, такі як обговорення хвороб бджіл, методів догляду, медоносних рослин, а також координація дій щодо обробки полів та розташування пасік.

Метою даної магістерської роботи є розробка фулстек веб-застосунку *Beekeepers Community Platform*, що надасть бджолярам зручні інструменти для комунікації, обміну інформацією, доступу до бази знань та управління даними про власні пасіки та сільськогосподарські угіддя.

Для досягнення поставленої мети було визначено наступні завдання:

- Провести аналіз предметної області та існуючих рішень.
- Обґрунтувати вибір технологічного стеку для розробки.
- Спроектувати архітектуру клієнтської та серверної частин застосунку, а також схему бази даних.
- Реалізувати основний функціонал платформи, включаючи систему реєстрації та автентифікації, форум, базу знань та інтерактивну карту.
- Забезпечити базові механізми безпеки та валідації даних.
- Розробити інтуїтивно зрозумілий та адаптивний користувацький інтерфейс.

Об’єктом дослідження є процес проектування та розробки повностекового веб-застосунку для нішевої спільноти.

Предметом дослідження є архітектурні рішення, технології та інструменти для створення інтерактивної та функціональної платформи для бджолярів, що включає засоби комунікації, обміну знаннями та геоінформаційні функції.

Методи дослідження, що використовувалися в роботі, включають:

- Аналіз науково-технічної літератури та існуючих аналогів.
- Системний аналіз та проектування програмного забезпечення.
- Об’єктно-орієнтоване програмування.

- Використання сучасних веб-технологій та фреймворків: React для розробки клієнтської частини, NestJS (Node.js) для серверної частини, MongoDB як система управління базами даних.
- Застосування бібліотек Material-UI для користувацького інтерфейсу та Leaflet для реалізації картографічного функціоналу.

Наукова новизна (якщо є, сформулювати). Практичне значення отриманих результатів полягає у створенні готового до використання прототипу веб-платформи, що може бути впроваджена для підтримки спільноти бджолярів, покращення їх взаємодії та доступу до актуальної інформації.

Апробація результатів роботи (якщо є, вказати конференції, публікації).

Структура роботи: Магістерська робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1

Аналіз предметної області

1.1 Огляд існуючих рішень

Даний розділ присвячено аналізу існуючих веб-платформ та мобільних застосунків, що можуть використовуватися спільнотами для обміну інформацією, а також спеціалізованих рішень для бджолярів. Розглядаються популярні форумні системи, соціальні мережі для груп за інтересами, а також застосунки, що пропонують інструменти для ведення пасіки чи моніторингу умов медозбору. Аналіз включає порівняння їх функціональних можливостей, переваг та недоліків у контексті потреб цільової аудиторії проекту — української спільноти бджолярів.

1.2 Порівняння технологій для веб-розробки

Вибір технологічного стеку є ключовим етапом розробки будь-якого програмного продукту. Для проекту *Beekeepers Community Platform* розглядалися наступні категорії технологій:

1.2.1 Технології фронтенду

Для розробки клієнтської частини веб-застосунку основними кандидатами були такі JavaScript-бібліотеки та фреймворки як React, Angular та Vue.js.

1.2.1.1 React

React є популярною JavaScript-бібліотекою для створення користувацьких інтерфейсів, розробленою Facebook [2]. Її перевагами є компонентний підхід, велика екосистема, значна спільнота розробників, гнучкість та висока продуктивність завдяки використанню віртуального DOM. Для даного проекту React був обраний через його здатність ефективно будувати складні та інтерактивні інтерфейси, велику кількість готових компонентів (зокрема, інтеграція з Material-UI та картографічними бібліотеками як Leaflet), а також через наявний досвід розробника з цією технологією.

1.2.1.2 Angular

Angular — це комплексний фреймворк від Google, що надає повний набір інструментів для створення великих корпоративних застосунків. Хоча Angular є потужним рішенням,

для даного проекту його всеохоплююча структура та дещо вищий поріг входження були розцінені як надлишкові.

1.2.1.3 Vue.js

Vue.js — прогресивний JavaScript-фреймворк, відомий своєю простотою інтеграції та низьким порогом входження. Він є гарним вибором для багатьох проектів, однак екосистема React та доступність спеціалізованих бібліотек для React виявилися більш привабливими для специфічних завдань проекту (наприклад, використання RTK Query для управління станом API).

1.2.2 Технології бекенду

Для серверної частини розглядалися Node.js-фреймворки NestJS та Express.js, а також Django на Python.

1.2.2.1 NestJS (Node.js)

NestJS — це прогресивний фреймворк для створення ефективних, масштабованих серверних застосунків на Node.js [4]. Він побудований з використанням TypeScript та поєднує елементи об'єктно-орієнтованого програмування, функціонального програмування та функціонально-реактивного програмування. Архітектура NestJS, що базується на модулях, контролерах та сервісах, сприяє чіткій організації коду та його підтримці. Вбудована підтримка TypeScript, легка інтеграція з Passport.js для автентифікації, Swagger для документації API та TypeORM/Mongoose для роботи з базами даних роблять його чудовим вибором для розробки REST API. Обраний для проекту завдяки модульній структурі, підтримці TypeScript та хорошій інтеграції з інструментами, необхідними для проекту.

1.2.2.2 Express.js (Node.js)

Express.js є мінімалістичним та гнучким Node.js веб-фреймворком. Він надає базовий набір функцій для веб-застосунків та API, але вимагає більше ручного налаштування та структурування проекту порівняно з NestJS. Для проекту, що передбачає розширення функціоналу, більш структурований підхід NestJS був визнаний кращим.

1.2.2.3 Django (Python)

Django — високорівневий Python веб-фреймворк, що заохочує швидку розробку та чистий, прагматичний дизайн. Він включає багато готових компонентів, таких як ORM, адміністративна панель тощо. Однак, для даного проекту перевага була надана стеку на JavaScript/TypeScri для забезпечення однорідності технологій на фронтенді та бекенді.

1.2.3 Системи управління базами даних

Вибір СУБД ґрунтувався на потребах гнучкості схеми даних та роботи з геопросторовими даними.

1.2.3.1 MongoDB

MongoDB — це документо-орієнтована NoSQL база даних [3]. Її перевагами є гнучка схема, що дозволяє легко еволюціонувати структуру даних, горизонтальна масштабованість та вбудована підтримка геопросторових запитів, що є важливим для функціоналу карти вуликів та полів. Обрана для проекту завдяки гнучкості, підтримці GeoJSON та добрій інтеграції з Node.js через Mongoose.

1.2.3.2 PostgreSQL

PostgreSQL — потужна об'єктно-реляційна СУБД, відома своєю надійністю, відповідністю стандартам SQL та розширюваністю (наприклад, PostGIS для геоданих). Хоча PostgreSQL є чудовим вибором для багатьох застосунків, для даного проекту з потенційно змінною структурою даних спільноти та геопросторовими об'єктами, гнучкість MongoDB була визнана більш пріоритетною.

1.3 Обґрунтування вибору технологічного стеку

Вибір технологічного стеку для проекту *Beekeepers Community Platform* ґрунтувався на аналізі вимог, доступності інструментів, досвіді розробника та прагненні створити сучасний, масштабований та підтримуваний веб-застосунок. Було обрано наступний стек:

- **Клієнтська частина (Frontend):** React з TypeScript обрано через його популярність, велику екосистему, компонентний підхід та ефективність у створенні динамічних користувацьких інтерфейсів. Material-UI (MUI) використано як бібліотеку UI компонентів для швидкої розробки адаптивного та естетично привабливого дизайну, що відповідає сучасним веб-стандартам. React Router використано для навігації, а Redux Toolkit (зокрема RTK Query) — для управління станом та взаємодії з API.
- **Серверна частина (Backend):** NestJS (побудований на Node.js та Express/Fastify) з TypeScript обрано завдяки його модульній архітектурі, яка сприяє чіткій організації коду, вбудованій підтримці TypeScript, що підвищує надійність, та легкій інтеграції з іншими інструментами, такими як Passport.js для автентифікації та Swagger для автоматичної генерації документації API. Використання Fastify адаптера забезпечує високу продуктивність.
- **База даних:** MongoDB обрано як NoSQL документо-орієнтовану базу даних. Її гнучка схема даних є перевагою для проекту, де структура інформації може розвиватися. Вбудована підтримка GeoJSON та геопросторових індексів є критично важливою для реалізації функціоналу інтерактивної карти. Mongoose використано як ODM для взаємодії з MongoDB з боку NestJS.
- **Картографічний сервіс:** Leaflet разом з React-Leaflet обрано як легку та гнучку бібліотеку для відображення інтерактивних карт та маніпуляції геопросторовими даними (маркери, полігони) [1].

- **Інтернаціоналізація:** i18next з react-i18next використано для підтримки багатомовності інтерфейсу.
- **Контейнеризація:** Docker та Docker Compose використовуються для створення консистентного середовища розробки та спрощення розгортання застосунку.

Даний стек технологій дозволяє ефективно розробляти як клієнтську, так і серверну частини, забезпечуючи при цьому хорошу продуктивність, масштабованість та можливості для подальшого розвитку платформи.

РОЗДІЛ 2

Проектування системи

2.1 Функціональні та нефункціональні вимоги

Проектування веб-застосунку *Beekeepers Community Platform* базувалося на визначенні ключових функціональних та нефункціональних вимог, що забезпечують його корисність, надійність та зручність для користувачів.

2.1.1 Функціональні вимоги

- **Реєстрація та автентифікація користувачів:** Можливість створення облікового запису з використанням електронної пошти та паролю, верифікація email через надсилання підтверджувального листа, а також автентифікація за допомогою облікового запису Google (OAuth 2.0).
- **Управління профілем користувача:** Перегляд та редагування базової інформації профілю (наприклад, біографія, місцезнаходження, експертиза).
- **Форум для обговорень:** Створення нових тем для обговорення, публікація повідомлень у темах, можливість залишати коментарі до повідомлень, система вподобань (лайків) для постів.
- **База знань:** Доступ до каталогу статей та ресурсів з бджільництва, можливість пошуку та фільтрації матеріалів за категоріями (поточна реалізація з mock-даними).
- **Інтерактивна карта (Управління пасіками та полями):**
  - Відображення карти (Україна за замовчуванням).
  - Додавання точкових маркерів для вуликів із зазначенням назви та нотаток.
  - Додавання полігональних об’єктів для полів із зазначенням назви, типу культури, періоду цвітіння та запланованих дат обробки пестицидами/інсектицидами.
  - Перегляд, редагування (заплановано) та видалення (заплановано) маркерів та полігонів.
  - Відображення детальної інформації (метаданих) при виборі об’єкта на карті.
  - Фільтрація об’єктів на карті за різними критеріями (тип культури, період цвітіння для полів; тип вулика, стан для пасік).

  - Можливість отримання сповіщень про заплановані обробки полів поблизу пасік (майбутній функціонал).

- **Інтернаціоналізація:** Підтримка декількох мов інтерфейсу (українська, англійська).

2.1.2 Нефункціональні вимоги

- **Продуктивність:** Забезпечення прийнятного часу завантаження сторінок та швидкої відповіді сервера на запити користувачів.
- **Безпека:** Захист облікових записів користувачів (хешування паролів), валідація вхідних даних на клієнті та сервері, використання HTTPS у продакшн-середовищі, захист від поширених веб-вразливостей (наприклад, XSS через використання React, який екранує дані за замовчуванням).
- **Масштабованість:** Архітектура застосунку повинна дозволяти майбутнє масштабування для обслуговування зростаючої кількості користувачів та обсягів даних.
- **Надійність:** Система повинна бути доступною та стабільно працювати.
- **Зручність використання (Usability):** Інтерфейс має бути інтуїтивно зрозумілим, легким у навігації та адаптивним для різних розмірів екранів (десктоп, мобільні пристрої).
- **Підтримуваність коду:** Кодова база повинна бути добре структурованою, документованою (де необхідно) та легкою для модифікації та розширення.

2.2 Діаграми варіантів використання (Use Case Diagrams)

Тут мають бути представлені діаграми варіантів використання, що ілюструють взаємодію користувачів (акторів) із системою. Наприклад, для незареєстрованого користувача (перегляд публічного контенту, реєстрація), зареєстрованого користувача (вхід, участь у форумі, робота з картою, перегляд бази знань, управління профілем) та адміністратора (якщо передбачено).

2.3 Архітектура системи

Розроблений веб-застосунок має класичну клієнт-серверну архітектуру.

2.3.1 Загальна архітектура

Система складається з трьох основних компонентів:

- Клієнтська частина (Frontend): односторінковий застосунок (SPA), розроблений на React, відповідає за користувацький інтерфейс та взаємодію з користувачем.
- Серверна частина (Backend): REST API, розроблене на NestJS, відповідає за бізнес-логіку, обробку запитів, взаємодію з базою даних та автентифікацію.

- База даних: MongoDB, документо-орієнтована NoSQL база даних, використовується для зберігання всієї інформації застосунку, включаючи дані користувачів, пости форуму, статті бази знань, а також геопросторові дані для карти.

Взаємодія між клієнтом та сервером відбувається за протоколом HTTP(S) через RESTful API. Для розгортання використовується Docker, що забезпечує ізоляцію та портативність середовища.

2.3.2 Архітектура фронтенду

Клієнтська частина розроблена з використанням бібліотеки React та TypeScript. Управління станом реалізовано за допомогою Redux Toolkit, зокрема RTK Query для взаємодії з API та кешування даних. Навігація між сторінками забезпечується React Router. Для побудови користувацького інтерфейсу використано бібліотеку компонентів Material-UI (MUI), що дозволяє створювати адаптивні та візуально привабливі інтерфейси. Інтернаціоналізація реалізована за допомогою i18next та react-i18next. Картографічний функціонал побудований на базі Leaflet та React-Leaflet. Структура проекту включає розділення на компоненти (загального призначення та специфічні для функціоналу), сторінки, сервіси (API-зрізи RTK Query), хуки, контексти та утиліти.

2.3.3 Архітектура бекенду

Серверна частина розроблена на платформі Node.js з використанням фреймворку NestJS та TypeScript. Архітектура є модульною, де кожен функціональний блок (наприклад, автентифікація, користувачі, форум, карта) виділений в окремий модуль. Кожен модуль містить контролери (обробка HTTP-запитів), сервіси (бізнес-логіка) та DTO (Data Transfer Objects) для валідації вхідних даних. Для взаємодії з базою даних MongoDB використовується ODM Mongoose. Автентифікація реалізована за допомогою Passport.js (JWT, локальна стратегія, Google OAuth). Для автоматичної генерації документації API використовується Swagger (OpenAPI). Застосунок працює на базі веб-сервера Fastify для підвищення продуктивності.

2.3.4 Схема бази даних

База даних MongoDB зберігає дані у вигляді колекцій документів. Основні колекції включають:

- `users`: інформація про користувачів (email, хеш паролю, ім'я користувача, профільні дані, статус верифікації, токени, геодані тощо).
- `forumposts`: пости на форумі (заголовок, зміст, автор, коментарі, лайки).
- `hives`: дані про вулики (назва, нотатки, геокоординати GeoJSON Point, власник).
- `fields`: дані про поля (назва, тип культури, періоди цвітіння та обробки, геометрія GeoJSON Polygon, власник).

- (Інші колекції, якщо є, наприклад, для бази знань, подій).

Для геопросторових даних (координати вуликів та геометрія полів) використовуються відповідні індекси `2dsphere` для ефективних геозапитів.

2.4 Проектування UI/UX

Проектування користувацького інтерфейсу (UI) та досвіду взаємодії (UX) було спрямоване на створення інтуїтивно зрозумілої, зручної та візуально привабливої платформи. Основним інструментом для реалізації UI стала бібліотека компонентів Material-UI, яка надає широкий набір готових елементів дизайну, що відповідають сучасним стандартам Material Design.

Ключові рішення:

- **Адаптивний дизайн:** Забезпечення коректного відображення та функціонування на різних пристроях (десктопи, планшети, мобільні телефони).
- **Інтуїтивна навігація:** Використання бічної панелі навігації для доступу до основних розділів сайту та чіткої ієрархії сторінок.
- **Консистентність інтерфейсу:** Дотримання єдиного стилю оформлення елементів на всіх сторінках застосунку.
- **Інтерактивність:** Надання користувачам можливості легко взаємодіяти з елементами, такими як карта, форми, кнопки.
- **Зворотний зв'язок:** Інформування користувача про результати його дій (успіх, помилка, процес завантаження) за допомогою повідомлень та індикаторів.

2.4.0.1 Вимоги до форуму

- Реєстрація користувачів з верифікацією електронної пошти.
- Автентифікація (логін/пароль, можливість входу через Google OAuth).
- Зберігання даних користувачів (email, хеш паролю, ім'я користувача, роль, інформація про пасіку за бажанням).
- Управління профілем користувача.
- Відновлення паролю.

2.4.0.2 Вимоги до бази знань

- Створення, редагування та видалення статей/ресурсів адміністраторами.
- Категоризація та тегування матеріалів.

- Пошук по базі знань.
- Система коментарів до статей (за бажанням).
- Рейтинг статей (за бажанням).

2.4.0.3 Загальні нефункціональні вимоги

- **Безпека:** Захист від основних веб-вразливостей (XSS, CSRF, SQL/NoSQL ін’єкції), безпечне зберігання паролів, використання HTTPS.
- **Продуктивність:** Швидке завантаження сторінок та відгук інтерфейсу.
- **Масштабованість:** Архітектура повинна дозволяти додавання нового функціоналу та витримувати зростання кількості користувачів.
- **Надійність:** Система повинна бути доступною та стабільно працювати.
- **Зручність використання (Usability):** Інтуїтивно зрозумілий та легкий у використанні інтерфейс.
- **Адаптивність (Responsiveness):** Коректне відображення на різних пристроях (десктопи, планшети, мобільні телефони).
- **Інтернаціоналізація (i18n):** Підтримка української та англійської мов інтерфейсу.
- **Зворотний зв’язок:** Інформування користувача про результати його дій, помилки.

РОЗДІЛ 3

Реалізація системи

3.1 Огляд процесу розробки

Розробка веб-застосунку *Beekeepers Community Platform* велася з використанням гнучких підходів, що дозволяли ітеративно додавати функціонал та вносити корективи. Основними інструментами розробки були Visual Studio Code як інтегроване середовище розробки, Git для системи контролю версій (з використанням GitHub для хостингу репозиторію), Docker та Docker Compose для контейнеризації та управління середовищем розробки та розгортання. Робочий процес включав регулярні коміти, розробку у функціональних гілках (за потреби) та тестування ключових функцій на локальному середовищі перед розгортанням.

3.2 Організація коду

Структура коду проекту організована для забезпечення модульності, легкості підтримки та масштабування.

3.2.1 Структура проекту фронтенду

Клієнтська частина (директорія `client/`) розроблена на React з використанням Vite як інструменту для збірки. Основні директорії в `src/`:

- `components/`: Містить UI компоненти, розділені за функціональними ознаками (наприклад, `forum/`, `map/`, загальні компоненти).
- `pages/`: Компоненти, що відповідають за окремі сторінки застосунку (наприклад, `Home.tsx`, `Forums.tsx`, `MapPage.tsx`).
- `services/` (або `store/api/`): API-зрізи (slices) для RTK Query, що описують взаємодію з бекенд API (наприклад, `authApi.ts`, `forumApi.ts`, `mapApi.ts`).
- `store/`: Конфігурація Redux store (`store.ts`).
- `context/`: React Context API для управління глобальним станом, таким як автентифікація (`AuthContext.tsx`).
- `hooks/`: Спеціалізовані React хуки.
- `types/`: Загальні TypeScript типи та інтерфейси.

- `locales/`: Файли перекладів для інтернаціоналізації (`en/translation.json`, `uk/transl`
- `utils/`: Допоміжні функції та утиліти.
- `App.tsx`: Головний компонент застосунку, що налаштовує маршрутизацію.
- `index.tsx` (або `main.tsx`): Вхідна точка застосунку.

3.2.2 Структура проекту бекенду

Серверна частина (директорія `server/`) розроблена на NestJS. Проект структурований за модульним принципом:

- `src/`: Містить основний код застосунку.
- `src/MODULE_NAME/`: Кожен функціональний блок (наприклад, `auth/`, `users/`, `forum/`, `hives/`, `fields/`, `health/`, `email/`) виділений в окремий модуль NestJS.
- Кожен модуль зазвичай містить:
  - `*.module.ts`: Файл визначення модуля.
  - `*.controller.ts`: Контролер, що обробляє HTTP-запити.
  - `*.service.ts`: Сервіс, що інкапсулює бізнес-логіку.
  - `schemas/`: Mongoose схеми для визначення структури даних MongoDB.
  - `dto/`: Data Transfer Objects для валідації вхідних даних.
  - `guards/`, `strategies/`, `decorators/`, `types/` (для специфічних типів модуля, наприклад, в `auth/`).
- `src/main.ts`: Вхідна точка серверного застосунку, ініціалізація NestJS та Fastify.
- `src/app.module.ts`: Кореневий модуль застосунку.

3.2.3 Проектування API

API спроектовано за принципами RESTful. Всі ендпоінти мають префікс `/api/v1/`, що забезпечує версіонування. Для валідації даних, що надходять від клієнта, використовуються Data Transfer Objects (DTO) з декораторами `class-validator`. Документація API автоматично генерується за допомогою Swagger (OpenAPI) і доступна на ендпоінті `/docs` (у середовищі розробки), що спрощує тестування та інтеграцію.

3.3 Опис ключових компонентів та модулів

3.3.1 Автентифікація та авторизація

Система автентифікації реалізована з використанням Passport.js. Підтримуються:

- **Локальна стратегія:** Автентифікація за логіном (email) та паролем. Паролі зберігаються у хешованому вигляді (з використанням `crypto.pbkdf2Sync`).
- **JWT (JSON Web Tokens):** Використовуються для авторизації користувачів після успішного входу. Генеруються access та refresh токени.
- **Верифікація Email:** При реєстрації користувачу надсилається лист з унікальним токеном для підтвердження електронної пошти. Доступ до функціоналу обмежений до верифікації.
- **Google OAuth 2.0:** Реалізована можливість входу та реєстрації через обліковий запис Google.
- **Захист маршрутів (Guards):** `JwtAuthGuard` використовується для захисту ендпоінтів, що вимагають автентифікації. `LocalAuthGuard` використовується для обробки локального входу.

3.3.2 Форум

Модуль форуму дозволяє користувачам створювати теми для обговорення, залишати повідомлення та коментарі, а також висловлювати свою думку за допомогою лайків. Дані зберігаються в колекції `forumposts`. Фронтенд реалізований з використанням компонентів для списку постів, окремого поста та форми створення нового поста. Взаємодія з бекендом відбувається через відповідні API ендпоінти, реалізовані в `ForumController` та `ForumService`.

3.3.3 База знань

Модуль бази знань (на даний момент з демонстраційними даними на клієнті) передбачає зберігання та відображення статей, посібників та інших корисних матеріалів для бджолярів. Планується розширення функціоналу для управління контентом через адміністративну панель та реалізація повноцінного API для роботи з ресурсами бази знань.

3.3.4 Інтеграція інтерактивної карти

Одним з ключових функціональних блоків є інтерактивна карта, реалізована за допомогою Leaflet та React-Leaflet на фронтенді та GeoJSON з індексами `2dsphere` на бекенді (MongoDB). Користувачі можуть:

- Додавати на карту точкові маркери для позначення вуликів, вказуючи назву та нотатки.
- Малювати полігони для позначення полів, вказуючи назву, тип культури, період цвітіння та заплановані дати обробки.
- Переглядати розміщені об'єкти та їх метадані у спливаючих вікнах (Popups).

Бекенд надає API для збереження, отримання, оновлення та видалення цих геопросторових об'єктів та їх метаданих (колекції `hives` та `fields`).

3.3.5 Налаштування доставки транзакційних електронних листів

Для забезпечення надійної та безпечної доставки транзакційних електронних листів (наприклад, для верифікації email), платформа *Beekeepers Community Platform* інтегрується з сервісом Mailgun. Правильне налаштування DNS є критично важливим для доставки та відповідності сучасним галузевим стандартам, особливо тим, що висуваються великими провайдерами, такими як Google та Yahoo.

3.3.5.1 Автентифікація домену

Для автентифікації вихідних електронних листів до DNS-конфігурації домену проекту були додані наступні записи:

- **SPF (Sender Policy Framework):** TXT-запис, що вказує, які поштові сервери мають право надсилати електронні листи від імені домену. Це допомагає запобігти несанкціонованим відправникам (спуфінгу).
- **DKIM (DomainKeys Identified Mail):** TXT-запис, що містить публічний криптографічний ключ. Вихідні листи підписуються приватним ключем Mailgun, а одержувачі можуть перевірити підпис за допомогою публічного ключа в DNS, забезпечуючи цілісність та автентичність повідомлення.
- **DMARC (Domain-based Message Authentication, Reporting, and Conformance):** TXT-запис, який інструктує приймаючі поштові сервери, як обробляти листи, що не пройшли перевірки SPF або DKIM.
  - **Опції політики:**
    - `p=none`: Тільки моніторинг, без примусових дій.
    - `p=quarantine`: Відправляти листи, що не пройшли перевірку, до папки ”Спам”.
    - `p=reject`: Повністю блокувати листи, що не пройшли перевірку.
  - **Сучасна практика:** Через нові вимоги Google та Yahoo (з 2024 року), політика DMARC повинна бути встановлена щонайменше на `quarantine` або `reject` для використання у виробничому середовищі. Це захищає користувачів від фішингу та покращує доставку.

3.3.5.2 Поширення та верифікація DNS

Після додавання необхідних записів, поширення змін у DNS може зайняти до 48 годин. Верифікація виконується за допомогою панелі керування Mailgun та сторонніх інструментів (наприклад, MXToolbox), щоб переконатися, що всі записи коректно опубліковані та доступні глобально.

3.3.5.3 Безпека та відповідність стандартам

- **Відсутність жорстко закодованих облікових даних:** Усі API-ключі Mailgun та конфіденційні налаштування зберігаються у змінних середовища, а не в коді.
- **Постійний моніторинг:** Звіти DMARC надсилаються на визначену електронну адресу для моніторингу проблем автентифікації та потенційного зловживання.
- **Ескалація політики:** Проект спочатку використовує `p=none` для DMARC для моніторингу легітимного трафіку, а потім переходить до `quarantine` або `reject` для повної відповідності стандартам та захисту.

3.3.5.4 Підсумкова таблиця DNS-записів

Запис	Призначення	Приклад значення / Політика
SPF	Автентифікація відправника	`v=spf1 include:mailgun.org ~all`
DKIM	Підпис повідомлення	(Публічний ключ, наданий Mailgun)
DMARC	Політика та звіти	`v=DMARC1; p=quarantine; rua=mailto:admin@`

Таблиця 3.1. Приклади DNS-записів для автентифікації email.

Така конфігурація гарантує, що всі вихідні електронні листи автентифіковані, знижуючи ризик спаму та фішингу та відповідаючи останнім вимогам основних поштових провайдерів. Вона також підтримує моніторинг та постійне вдосконалення безпеки електронної пошти.

3.4 Безпека системи

При розробці увага приділялася аспектам безпеки:

- **Автентифікація та авторизація:** Використання JWT, захист маршрутів, OAuth 2.0.
- **Зберігання паролів:** Паролі користувачів хешуються на стороні сервера перед збереженням у базу даних (використано модуль `crypto` Node.js, зокрема `pbkdf2Sync`).
- **Валідація вхідних даних:** Усі дані, що надходять від клієнта на бекенд, валідуються за допомогою DTO та `class-validator`, що запобігає некоректним даним та потенційним атакам (наприклад, NoSQL ін'єкції на рівні структури даних).
- **Захист від XSS:** Використання React на фронтенді за замовчуванням екранує дані, що вставляються в DOM, що знижує ризик XSS-атак.
- **CORS:** Налаштовано політику Cross-Origin Resource Sharing для контролю доступу до API з боку фронтенду.
- **Використання HTTPS:** У продакшн-середовищі необхідно використовувати HTTPS для шифрування трафіку.

- **Управління секретами:** Чутливі дані (секрети JWT, ключі API для зовнішніх сервісів, рядок підключення до БД) зберігаються у змінних середовища та не включаються до системи контролю версій.

РОЗДІЛ 4

Тестування та розгортання

4.1 Тестування системи

Тестування є невід'ємною частиною процесу розробки програмного забезпечення, спрямованою на виявлення помилок та забезпечення відповідності функціональним та нефункціональним вимогам. Для веб-застосунку *Beekeepers Community Platform* було проведено кілька видів тестування.

4.1.1 Модульне тестування (Unit Testing)

Модульне тестування було зосереджено на перевірці окремих компонентів та функцій серверної частини (NestJS). Використовувався вбудований в NestJS тестовий фреймворк, що базується на Jest. Тестувалися сервіси, контролери (частково) та допоміжні утиліти. Основна увага приділялася тестуванню бізнес-логіки в сервісах, валідації даних та коректності відповідей API.

4.1.2 Інтеграційне тестування

Інтеграційне тестування передбачало перевірку взаємодії між різними модулями системи, зокрема між фронтендом та бекендом (API ендпоінти), а також взаємодію бекенду з базою даних MongoDB. На цьому етапі перевірялася коректність обробки запитів, передачі даних та їх збереження/отримання.

4.1.3 Тестування користувацького інтерфейсу (UI Testing)

На фронтенді проводилося ручне тестування користувацького інтерфейсу на різних пристроях та в різних браузерах (Chrome, Firefox, Safari) для забезпечення адаптивності та коректного відображення. Перевірялася робота інтерактивних елементів, форм, навігації та картографічного функціоналу.

4.1.4 Тестування безпеки

Здійснювалася базова перевірка на поширені веб-вразливості, такі як XSS (здебільшого покривається React), валідація вхідних даних для запобігання ін'єкціям на рівні API. Також перевірялася робота системи автентифікації та авторизації, зокрема захист маршрутів та валідність JWT токенів.

4.2 Розгортання застосунку

Для розгортання веб-застосунку *Beekeepers Community Platform* використовується контейнеризація за допомогою Docker та Docker Compose. Це дозволяє створити ізольоване та відтворюване середовище для роботи застосунку, що спрощує процес розгортання та управління залежностями.

Створено `Dockerfile` для клієнтської частини (React/Vite) та для серверної частини (NestJS). Файл `docker-compose.yml` описує сервіси для фронтенду, бекенду та бази даних MongoDB, а також налаштовує їх взаємодію та мережеві налаштування.

Для продакшн-розгортання рекомендується використання оберненого проксі-сервера (наприклад, Nginx) для обслуговування статичних файлів фронтенду, кешування, балансування навантаження (за потреби) та налаштування HTTPS.

4.3 Майбутні напрямки розвитку

Платформа *Beekeepers Community Platform* має потенціал для подальшого розвитку та розширення функціоналу. Можливі напрямки включають:

- Розширення функціоналу бази знань: додавання можливості користувачам пропонувати статті, система рецензування, коментарі.
- Розвиток картографічного сервісу: фільтри за типами культур, періодами цвітіння, сповіщення про обробку полів, інтеграція з погодними даними.
- Система приватних повідомлень між користувачами.
- Календар подій для бджолярів (виставки, ярмарки, семінари).
- Мобільний застосунок (React Native або нативні технології).
- Розширена аналітика та статистика для користувачів (наприклад, продуктивність пасік).
- Інтеграція з іншими сервісами (наприклад, маркетплейси для продукції бджільництва).

Висновки

У даній магістерській роботі було розроблено повностековий веб-додаток *Beekeepers Community Platform*. Метою роботи було створення платформи для спілкування, обміну досвідом та знаннями серед бджолярів, а також надання інструментів для управління пасіками та полями.

Основні досягнуті результати:

- Проведено аналіз предметної області та обґрунтовано вибір сучасного стеку технологій (React, NestJS, MongoDB).
- Спроектовано та реалізовано ключові функціональні модулі: система автентифікації (включаючи email верифікацію та Google OAuth), форум, база знань, інтерактивна карта.
- Забезпечено базові механізми безпеки та валідації даних.
- Створено адаптивний користувацький інтерфейс з використанням Material-UI.

Розроблений додаток успішно вирішує поставлені завдання, надаючи зручну та функціональну платформу для спільноти бджолярів.

Напрямки для подальшого розвитку включають розширення функціоналу карти (фільтрація, аналітика), впровадження системи сповіщень, розробку мобільного додатку та інтеграцію з іншими сервісами для бджолярів.

Список використаних джерел

[1] Volodymyr Agafonkin and contributors. Leaflet - an open-source javascript library for mobile-friendly interactive maps. https://leafletjs.com/, 2024. Accessed: YYYY-MM-DD;

[2] Meta Platforms, Inc. React - a javascript library for building user interfaces. https://react.dev/, 2024. Accessed: YYYY-MM-DD;

[3] MongoDB, Inc. Mongodb atlas - the multi-cloud developer data platform. https://www.mongodb.com/, 2024. Accessed: YYYY-MM-DD;

[4] NestJS. Nestjs - a progressive node.js framework. https://nestjs.com/, 2024. Accessed: YYYY-MM-DD;

ДОДАТКИ А

Приклад коду API

```
// Example: Backend - HivesController - create method
@Post()
@Version('1')
@ApiOperation({ summary: 'Create a new hive' })
@ApiResponse({ status: 201, description: 'Hive created successfully.' }
@ApiResponse({ status: 400, description: 'Invalid input.' })
create(@Body() createHiveDto: CreateHiveDto, @GetUser() user: JwtUserPa
  return this.hivesService.create(createHiveDto, user.userId);
}
```

ДОДАТКИ Б

Документація API (Swagger)

Документація API була автоматично згенерована за допомогою Swagger (OpenAPI) і доступна за ендпоінтом /docs на сервері розробки.

Placeholder for Swagger UI Screenshot

Рис. Б.1. Приклад інтерфейсу Swagger UI для API

ДОДАТКИ В

Інструкція користувача

Реєстрація та вхід

Опис процесу реєстрації та входу, включаючи верифікацію email та вхід через Google.

Використання форуму

Як створювати теми, писати повідомлення, коментувати.

Робота з картою

Як додавати та переглядати вулики та поля.