МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ
Навчально-науковий інститут автоматики, кібернетики та обчислювальної техніки

"До захисту допущена"
Зав. кафедри комп'ютерних наук та прикладної математики
_____
«___» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

Застосування технологій React та .NET для розробки порталу новин

Виконав: **Біра Роман Богданович** _____
(прізвище, ім'я, по батькові) (підпис)
група КН-41

Керівник: **к. т. н., доцент Жуковська Н. А.** _____
(науковий ступінь, вчене звання, посада, прізвище, ініціали) (підпис)

Рівне – 2023

Національний університет водного господарства та природокористування

Навчально наукоувий інститут автоматики, кібернетики та обчислювальної техніки

Кафедра **комп'ютерних наук та прикладної математики**

Освітньо-кваліфікаційний рівень **бакалавр**

Галузь знань 12 «Інформаційні технології»

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

“____” _____20___року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Бірі Роману Богдановичу

1. Тема роботи *Застосування технологій React та .net для розробки порталу новин*

керівник роботи к.т.н., доцент Жуковська Наталія Анатоліївна

затверджені наказом вищого навчального закладу від *19 квітня 2023 року С № 449.*

2. Термін подання роботи студентом *30 травня 2023 р.*

3. Вихідні дані до роботи інтернет форуму та засоби розробки веб ресурсів.

4. Зміст розрахунково-пояснювальної записки: *У першому розділі здійснено огляд технології React та застосування його для розробки інтерфейсів. Другий розділ присвячений вивченню застосування платформи .NET для розробки сайтів.. У третьому розділі розглядаються можливості SQLite при роботі з додатками, які містять бази даних. Розділ чотири містить опис структури та функціональних можливостей новинного порталу.*

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) *мультимедійна презентація.*

6.Консультанти розділів роботи(проекту)

Розділ	Прізвище, ініціали, посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Розділ 1	доцент Жуковська Н. А.		
Розділ 2	доцент Жуковська Н. А.		
Розділ 3	доцент Жуковська Н. А.		
Розділ 4	доцент Жуковська Н. А.		

7. Дата видачі завдання : 30.09.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Огляд літератури за обраною тематикою		
2	Проектування макету сайту		
3	Вибір технологій розробки		
4	Проектування структури бази даних		
5	Програмна реалізація фронтенду проекту		
6	Програмна реалізація бекенду проекту		
7	Підготовка виступу і презентації		

Студент _____ Біра Р.Б.

(підпис) (прізвище та ініціали)

Керівник роботи _____ Жуковська Н. А.

(підпис) (прізвище та ініціали)

ЗМІСТ

РЕФЕРАТ

Кваліфікаційна робота: 57 с., 12 малюнків, 15 джерел.

Метою кваліфікаційної роботи є розробка порталу новин, клієнтської та серверної частин веб-ресурсу.

Об’єкт дослідження – архітектура новинних сайтів та візуалізація контенту клієнтської частини додатку, структура серверної частини порталу новин.

Предмет дослідження – методи, засоби та сучасні технології розробки сайтів та веб-ресурсів.

Методи дослідження – технологія React, платформа .NET, SQLite, Entity -фреймворк.

При сучасному розвитку IT-технологій будь-хто може отримати «найсвіжішу» інформацію незалежно від свого місцеперебування. Портал новин – це мережевий ресурс, що складається з поєднаних за змістом та навігацією веб-сторінок, основаних на новинному контенті. Створення такого сайту передбачає проектування надійного і зручного додатку, здатного витримувати великий наплив відвідувачів. Найважливіше для інтернет-додатків такого типу є здатність працювати швидко і безвідмовно при високому трафіку. Такі ресурси дозволяють бути в курсі найважливіших подій у різних сферах: суспільство, економіка, культура, спорт та інших. За статистикою, більшість людей черпають інформацію саме з таких сайтів. Інформація завжди була головною цінністю та запорукою успіху у будь-який справі. Перевагами інтернет-ЗМІ є оперативність надання інформації, швидке оновлення, накопичення та зберігання інформації в одному сховищі, аналізування запитів користувачів.

Ключові слова: React, .NET, SQLite, дизайн системи, DOM.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

DOM	–	Document Object Model, специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами.
.NET	–	це платформа від Microsoft, яка дозволяє створювати програмні додатки.
React.js	–	front-end бібліотека JavaScript для побудови користувацьких інтерфейсів на основі UI-компонентів.
SQLite	–	це вбудована реляційна база даних, з відкритим вихідним кодом, втілена у вигляді бібліотеки.
Ant Design	–	Дизайн система, що використовується для розробки інтерфейсу..
NPM	–	пакетний менеджер який входить до складу Node.js мови програмування Javascript.
API	–	інтерфейс, який визначає взаємодію між кількома програмними додатками або змішаними апаратно-програмними посередниками
CLR		Common Language Runtime, загальномовне виконуюче середовище, компонент пакету Microsoft .NET Framework, віртуальна машина, на якій виконуються всі мови платформи .NET Framework
UWP		універсальна платформа Windows

ВСТУП

За останні роки інтернет-технології значно змінили наше сприйняття інформації. Новини стали доступні не тільки через телебачення, радіо та газети, але й через Інтернет. Створення новинного порталу - це важливий крок у розвитку медіа-сфери, який дозволяє забезпечити швидкий та простий доступ до новин для користувачів з усього світу. У роботі будуть розглянуті основні аспекти створення новинного порталу, включаючи вибір технологій, дизайн та контент.

Створення сайту новин задача складніша, ніж може здатись на перший погляд. Новинні сайти користуються популярністю та вирізняються значними навантаженнями та відвідуваністю. Навіть регіональні сайти новин (ЗМІ), при правильному підході до редакційної політики, можуть мати десятки, а то й сотні тисяч відвідувань щодня. Тому важливим є визначення цілей та мети порталу. Необхідно встановити, яку аудиторію ви хочете залучити, які теми новин ви будете публікувати, які функції порталу будуть доступні користувачам, джерела новин та ін.

Важливо зробити сайт новин популярним. Для цього потрібно забезпечити якісний та цікавий контент, висвітлювати актуальні теми та події, публікувати оригінальні матеріали та аналітику. Користувачу потрібен зрозумілий та адаптивний інтерфейс. Тому потрібно потурбуватись про зручну навігацію та швидку завантаження сторінок. Користувачі не будуть залишатися на сайті, якщо він завантажується довго або навігація є незручною. Для залучення нової аудиторії потрібно рекламувати сайт у соціальних мережах та інших платформах.

Розраховані на широку аудиторію новинні портали повинні подавати інформацію якісно, використовуючи унікальний дизайн та нестандартні рішення до відображення контенту. В цьому проекті ми розглянемо технології, що застосовуються для розробки інтернет ресурсів в контексті розробки новинного порталу.

РОЗДІЛ 1. ЗАСТОСУВАННЯ REACT ДЛЯ СТВОРЕННЯ ІНТЕРФЕЙСІВ КОРИСТУВАЧА

1.1. Огляд технології React

React - це бібліотека JavaScript для створення інтерфейсів користувача. Вона дозволяє створювати веб-додатки зі складними інтерактивними інтерфейсами, що змінюються динамічно, без перезавантаження сторінки. React дозволяє розробникам будувати складні інтерфейси з використанням декларативного підходу до програмування. Основною метою React є полегшення створення інтерактивних користувацьких інтерфейсів шляхом розбиття їх на компоненти. Ці компоненти можна перевикористовувати в різних частинах додатку, що зменшує кількість коду та полегшує його розуміння. Крім того, React забезпечує швидкий та ефективний рендеринг компонентів завдяки використанню віртуального DOM (Document Object Model), що дозволяє зменшити час «перемальовування» сторінки при зміні її стану.

React забезпечує набір базових компонентів, таких як кнопки, форми, таблиці тощо, які можна використовувати для створення складних інтерфейсів. Крім того, React надає можливість створювати власні компоненти, які можуть бути складнішими та спеціалізованими, підтримує створення компонентів за допомогою класів (class components) та за допомогою функціональних компонентів (functional components). React також надає можливість керувати станом компонентів використовуючи внутрішній state-об'єкт. Для кращого керування станом додатку React можна поєднувати з іншими бібліотеками та технологіями, такими як Redux, MobX, GraphQL, React Router та іншими. За допомогою React Native можна також розробляти мобільні додатки для IOS та Android.

На даний момент React є однією з найпопулярніших бібліотек для розробки веб-інтерфейсів та має велику спільноту розробників та користувачів.

Розвиток React продовжується, з’являються нові можливості та функції, що забезпечує його актуальність у сучасному веб-програмуванні.

Найбільш яскравими можливостями React є:

- спрощення створення інтерактивних інтерфейсів. Потрібно лише описати, як різні частини інтерфейсу виглядають у кожному стані додатку і React ефективно оновить та відрендерить лише потрібні компоненти, коли дані зміняться;
- декларативні інтерфейси роблять код програми більш передбачуваним і його набагато легше налагоджувати;
- можливість створення інкапсульованих компонентів, які керують власним станом, та з яких потім будуються складні інтерфейси;
- можливість розробляти нові функції в React, не переписуючи існуючий код;
- React також може рендеритись на сервері, використовуючи Node, і приводити в дію мобільні додатки, які використовують React Native.

Головне завдання React – забезпечення виведення на екран того, що можна побачити на веб-сторінках. Застосування цієї технології значно полегшує створення інтерфейсів завдяки розбиттю кожної сторінки на маленькі фрагменти, які називають компонентами.

Для того щоб створювати та підтримувати React-додатки, розробнику потрібно знати[1]:

- компоненти React;
- рендерінг ReactDOM;
- класи компонентів та функціональних компонентів;
- JSX;
- стани (state);
- обробку подій;
- асинхронний метод setState;
- властивості (props);

- посилання (refs).

Компонент React — це ділянка коду, яка є частиною веб-сторінки. Кожен компонент – це JavaScript-функція, яка повертає частину коду, який представляє фрагмент сторінки. Для формування сторінки потрібно викликати ці функції у певному порядку, зібрати разом результати викликів та відобразити їх для користувача.

Наприклад, якщо записати компонент в середині тега <script> файлу index.html з type, встановленим у "text/babel":

```
<script type="text/babel">
  function OurFirstComponent() {
    return (
      // Тут буде код, що демонструє фрагмент користувацького інтерфейсу
    );
  }
</script>
```

то при виклику функції OurFirstComponent(), у відповідь ми отримаємо фрагмент сторінки[1].

React використовує мову програмування яка називається JSX. Вона схожа на HTML, але працює всередині JavaScript, що відрізняє її від HTML. Можна додати сюди звичайний HTML для того, щоб він потрапив до інтерфейсу користувача, наприклад [1]:

```
<script type="text/babel">
  function OurFirstComponent() {
    return (
      <h1>Hello, I am a React Component!</h1>
    );
  }
</script>.
```

Коли ми викликаємо OurFirstComponent(), вона повертає фрагмент JSX-коду. Розробник може використовувати так званий ReactDOM для виведення

того, що представляє цей код, на сторінку, а можна написати власний компонент на JSX. Наприклад:

ReactDOM.render(<OurFirstComponent />, placeWeWantToPutComponent);

Стандартний підхід в роботі з React полягає в тому що, розробник може викликати компоненти так, ніби він працюєте з HTML.

Компоненти React можна розміщувати в інших компонентах[1]:

```
<script type="text/babel">
  function OurFirstComponent() {
    return (
      <h1>I am the child!</h1>
    );
  }
  function Container() {
    return (
      <div>
        <h1>I am the parent!</h1>
        <OurFirstComponent />
      </div>
    );
  }
  const placeWeWantToPutComponent = document.getElementById('hook');
  ReactDOM.render(<Container />, placeWeWantToPutComponent);
</script>
```

Ось що виведе вищенаведений код:

Рис.1.1. Приклад формування сторінки з використанням React [1]

Саме так відбувається збирання сторінки із фрагментів, написаних на React – вкладенням компонентів один в один.

Компоненти можна писати як класів JavaScript. Їх називають класами компонентів. Класи компонентів повинні містити функцію, яка називається render(). Ця функція повертає JSX-код компонента [1]:

```
class Container extends React.Component {
  render() {
    return (
      <div>
        <h1>I am the parent!</h1>
        <OurFirstComponent />
      </div>
    );
  }
}
const placeWeWantToPutComponent = document.getElementById('hook');
ReactDOM.render(<Container />, placeWeWantToPutComponent);
```

Їх можна використовувати так само, як функціональні компоненти, наприклад, звертаючись до них за допомогою конструкції:

```
<AClassComponent />;
```

Якщо вам потрібні компоненти без стану, перевагу слід віддати функціональним компонентам, їх легше читати.

У JSX-код можна розташовувати змінні JavaScript та викликати функції. Зокрема, після виконання цього фрагменту, сторінка буде виглядити як на рис.1. 2.

```
class Container extends React.Component {
 render() {
   const addNumbers = (num1, num2) => {
     return num1 + num2;
   };
   return (
     <div>
       <h1>The sum is: { addNumbers(1, 2) }</h1>
       <OurFirstComponent />
     </div>
   );
 }
}
```

Рис.1.2. Застосування змінних в якості функціональних компонентів [1]

Компоненти, що базуються на класах, можуть зберігати інформацію про поточну ситуацію. Ця інформація називається станом (state), вона зберігається у JS-об'єкті. Стан(state) – це інструмент, що дозволяє оновлювати інтерфейс користувача, ґрунтуючись на подіях. Коли стан компонента змінюється, він

знову викличе функцію render(). Ми можемо змінити стан за допомогою this.setState(), якщо передамо цій функції об'єкт, який представляє новий стан. Компонент на сторінці завжди представлятиме свій поточний стан. React самостійно забезпечуватиме таку поведінку компонентів. Компоненти можуть «спілкуватися» один з одним, використовуючи властивості (props). Властивості – це інформація, що колективно використовується батьківським компонентом і компонентами-нащадками. Властивостями не обов'язково мають бути якісь дані, це можуть бути і функції. React підтримує безліч конфігурацій збирання. За допомогою інструментів типу Create React App всю рутинну роботу з формування каркаса програми можна автоматизувати.

1.2. Особливості архітектури React

Архітектура React базується на компонентах, які представляють собою незалежні, повторно використовувані блоки веб-інтерфейсу. Компоненти можуть мати свій власний стан та властивості, які можуть бути передані від батьківського компонента до дочірнього.

Особливості архітектури React:

- Віртуальний DOM: React використовує віртуальний DOM, що забезпечує більш ефективне оновлення сторінки, оскільки він не перезавантажує всю сторінку з кожним оновленням, а тільки необхідні частини.
- Декларативний код: React дозволяє описувати структуру веб-інтерфейсу декларативно, без використання імперативних команд. Це полегшує розуміння та підтримку коду.
- Розділення на компоненти: Компоненти React дозволяють розділяти веб-інтерфейс на невеликі блоки, що полегшує їх перевикористання та розуміння.
- Підтримка JSX: JSX дозволяє описувати структуру веб-інтерфейсу в синтаксисі JavaScript, що полегшує написання коду та підтримку його.
- Єдиний напрямок потоку даних (Unidirectional Data Flow): React дозволяє передавати дані з батьківського компонента до дочірнього за

допомогою властивостей (props). Це забезпечує єдиний напрямок потоку даних, що полегшує розуміння коду та підтримку його.

- Підтримка стану (State): React дозволяє зберігати стан компонента, що забезпечує можливість робити інтерактивні веб-інтерфейси.
- Підтримка навігації: React Router дозволяє створювати навігацію на сторінках, що дозволяє створити односторінковий веб-додаток з навігацією, не оновлюючи сторінку, під час навігації користувача.

Одна з особливостей React – це те, як він змушує розробникаміркувати над додатком в процесі створення. Уявімо, що ми хочемо створити У цьому таблицю продуктів з пошуком за допомогою React. Опишемо хід такого процесу в контексті використання React. Припустимо, що у нас вже є JSON API і макет дизайну сайту (Рис.1.3.).

Рис.1.3. Макет дизайну сайту [2]

Першим корком розробника має бути визначення меж кожного компонента (і підкомпонента) в макеті, а також присвоєння їм імен. Для визначення, що має бути компонентом можна скористатись одним з важливих принципів об’єктно-орієнтованого програмування *принципом єдиної відповідальності*, з якого слідує, що кожний компонент в повинен виконувати

якесь одне завдання. Якщо функціонал компонента збільшується з плином часу, то його слід розбити на дрібніші підкомпоненти.

Як правило інтерфейс та модель даних мають подібну інформаційну архітектуру, тому розділити інтерфейс на частини не важко. Можна у інтерфейсі користувача виокремити компоненти, кожен з яких буде відображати частину моделі даних, як показано на рис.1.4.

Рис.1.4. Виокремлення компонентів макета [2]

Отже, було виокремлено п’ять компонентів додатку:

1. *FilterableProductTable* – весь блок;
2. *SearchBar* – блок для введення даних користувача;
3. *ProductTable* – відображення даних на основі запиту користувача;
4. *ProductCategoryRow* – блок заголовку кожної категорії;
5. *ProductRow* – виведення даних про окремий продукт.

Після визначення компонентів у макеті будують ієрархічне дерево впорядкування компонентів. Для нашого прикладу воно може бути таким:

- FilterableProductTable
  - SearchBar
  - ProductTable

- ProductCategoryRow
- ProductRow

Наступним кроком є реалізація спроектованого макету. Найлегше створити статичну версію, яка використовує модель даних і рендерить інтерфейс. Але такий підхід вимагає кодування у великій кількості, практично без творчого процесу. Створення інтерактивних компонентів в додатку – глибокий розумовий процес в якому мало рутинної роботи.

Компоненти, які створюються у статичній версії, використовують інші компоненти передаючи дані через *props*(властивості). Властивості це спосіб передачі даних від батьківських до дочірних елементів. Таке поняття як стан(state) у статичній версії не використовується. Стан передбачає собою дані, які змінюються з часом, а це інтерактивність.

Писати код можна як зверху до низу (з FilterableProductTable), так і навпаки (з ProductRow). Простіші додатки зручніше починати з компонентів, які знаходяться вище за ієрархією. У більш складних додатках зручніше в першу чергу створювати і тестувати підкомпоненти. В результаті розробник буде мати бібліотеку компонентів, які можна використовувати повторно. Оскільки ми розглядаємо статичну версію, то компоненти матимуть тільки методи render(). Компонент, який вищий за ієрархією (FilterableProductTable) буде передавати дані через props. Після внесення змін в базову модель даних і повторного виклику root.render(), будуть відображені зміни в інтерфейсі. Завдяки односторонньому потоку даних (або односторонній прив’язці) код працює швидко і залишається зрозумілим.

Для розробки інтерактивного інтерфейсу користувача застосовують state, якщо потрібно, щоб модель даних могла змінюватися з часом. Різниця між властивостями і станом полягає в наступному: властивості передаються в компонент подібно до параметрів функції, тоді як state знаходиться у компоненті, це дещо нагадує оголошення змінних усередині функції.

Перш за все потрібно подумати про мінімальний набір змінних станів, які будуть у додатку і дотримуватися принципу розробки DRY (Don’t Repeat Yourself – не повторюйся). Програмісту потрібно насамперед визначити мінімальну кількість необхідних станів для додатку, все інше обчислюється за необхідності.

Отже, список станів додатку може бути таким:

- початковий список товарів;
- пошуковий запит, введений користувачем;
- значення прапорця;
- відфільтрований список товарів.

Розглянемо кожний елемент вищенаведеного списку і визначимо, який з них є state. Для цього потрібно дати відповідь на таки питання:

1. Чи передається він від батьківського компонена через props?
2. Чи залишається він незмінним в часі?
3. Чи можна обчислити його на основі будь-якої іншої частини state або props у своєму компоненті?

Якщо відповідь на кожне з цих питань «так», то швидше за все це не state.

Початковий список товарів передається через props, отже це не state. Пошуковий запит і прапорець змінюються з часом, також їх не можна обчислити з інших даних, тому вони цілком можуть бути state. Відфільтрований список товарів не state, тому що його можна обчислити з вихідного списку, пошукового запиту та значення прапорця. У підсумку список state виглядає наступни чином:

- пошуковий запит;
- значення прапорця.

Наступне що потрібно з’ясувати, який з компонентів майбутнього додатку буде містити в собі state або буде змінювати його. Для цього потрібно застосувати таку стратегію: визначити компоненти, які рендерять щось на основі цього стану, знайти спільний батьківський компонент (компонент, розташований над іншими компонентами, яким потрібен цей стан) або спільний батьківський компонент чи будь-який компонент, що стоїть вище за ієрархією, повинен містити стан. Якщо не вдається знайти відповідний компонент, то його створюють виключно для стану та розміщають вище за ієрархією над загальним спільним батьківським компонентом. Розглнемо цю стратегію на нашому прикладі. Завдання ProductTable – відфільтрувати список товарів, базуючись на стані, а завдання SearchBar – відобразити стан для пошукового запиту та прапорця. Спільний батьківський компонент для обох – FilterableProductTable. Отже, є сенс помістити текст фільтра та значення прапорця в FilterableProductTable.

Ще одним завданням є забезпечити потік даних у зворотний бік. Наприклад, зробімо так, щоб компоненти форми у самому низу ієрархії могли оновлювати стан у FilterableProductTable. Оскільки потік даних у React односторонній, нам потрібно, щоб при змінах значень у пошуковій формі змінювався стан у FilterableProductTable. Тому що компоненти повинні оновлювати тільки той стан, що належить їм, FilterableProductTable передасть функцію зворотнього виклику у SearchBar. У свою чергу, SearchBar викликатиме цю функцію зворотнього виклику кожен раз, коли треба оновити стан. Щоб отримувати повідомлення про зміни елементів форми, ми можемо використовувати подію onChange. Функції зворотнього виклику, передані з FilterableProductTable, викличуть setState(), і додаток оновиться. Таким чином потік даних через додаток буде прямим і зрозумілим.

1.3. Використання дизайн-системи Ant Design при розробці програмних інтерфейсів

Складно уявити веб-додаток розроблений на React без бібліотеки компонентів. Розробник може поступово наповнювати таку бібліотеку в міру необхідності або використовувати колекцію з готових рішень, яка складається npm-модулів. Якщо розробник має достатньо часу на роботу над проєктом, бібліотека формується поступово ним обобисто. Як правило, перед початком проєкту роблять вибір на користь використання готових бібліотек компонентів, щоб мінімізувати витрати часу на написання власних «велосипедів». Враховуючи популярність React, його компонентний підхід та популяризацію дизайн-систем, може здатись, що розробник має широкий вибір таких систем. Насправді бібліотек компонентів, які відповідають потребам React, можна перерахувати на пальцях.

Розберемось, чому ж варто обрати саме на Ant Design. Проектні можливості Ant Design надають дизайнерам внутрішні стандарти для оцінки, дають змогу розробнику створювати дизайн та шаблони проектування та пропонують документацію та загальні рішення для конкретних проектних проблем. Чотири основні переваги Ant Design полягають в наступному:

- *природність*, тобто можливість створювати інтуїтивний користувацький інтерфейс, що базується на природних можливостях і здібностях людини сприймати інформацію через візуальні канали, а застосування сенсорних каналів, таких як слух і дотик, може створити багатший і природніший досвід взаємодії людини з комп’ютером;
- *надійність*, тобто дотримуючись правил проектування та модульного дизайну, можна зменшити командну ентропію та зробити процес проектування ефективнішим. Інкапсуляція складних або багаторазово використовуваних частин може забезпечити обмежені інтерфейси для взаємодії з іншими модулями, тим самим покращуючи надійність і зручність обслуговування;

- *можливість розширення функціоналу,* тобто при розширенні функціоналу продукту потрібно враховувати не лише покращення функціонування додатку але і зручність використання його клієнтами.

Отже, Ant Design – це повноцінна дизайн-система, візуальна мова, яка має власні принципи, стайлгайди та бібліотеку компонентів, яка розроблена і підтримується Alibaba Group. Разом з Ant Design Alibaba Group підтримують і два - фреймворк на основі популярного стека React – Redux, React-Router. Ant Design написаний на TypeScript, стилізований за допомогою Less та портований на Angular та Vue.

Зосередимось на тому, що може запропонувати Ant Design як бібліотека компонентів для React, яка є досить потужною і ознайомитись з нею можна на офіційному сайті [3]. Більшість компонентів можна використовувати окремо від Ant Design, використовуючи модулі react-component. Ant Design має дві особливості, які вигідно виділяють його серед аналогічних бібліотек: таблиці та форми.

При роботі з таблицями використовуються вбудована плагінація, але можна також написати свою. Доступна фільтрація за різними умовами відбору з опціями, але описувати функції сортування та фільтрації потрібно власноруч. За замовчуванням таблиці не «вміють» фільтрувати записи за введеним рядком, потрібно написати свій користувацький фільтр. Про це докладно описано в документації. Таблиці Ant Design надають досить гнучкий API для відображення відповіді на запит користувача. В наборі інструментів для роботи з таблицями є функції, що дозволяють приховати рядки або розгортати їх знову. Можливе об’єднання клітин та фіксація колонок і заголовків у таблицях. API для роботи з таблицями загалом досить гнучкий і дозволяє довільне оновлення комірок. Серед недоліків слід виділити те, що таблиці Ant Design погано підходять для відображення великої кількості рядків. Більше сотні рядків вже здатні істотно просадити продуктивність програми. Дефолтні стилі таблиць,можуть привести до того, що навіть приклади з офіційної документації без обмеження максимальної ширини таблиць можуть виглядати

не дуже гарно. Цей недолік усувається, але коли стикаєшся з цим вперше, то це є несподіванкою.

Робота з формами в Ant Design надає багато можливостей. У формах можна додавати правила валідації простими об'єктами, синхронізувати значення полів із Redux Store, зберігати значення полів, що використовуються за замовчування, окремо, щоб потім застосувати їх викликом одного методу. Новачки часто починають використовувати компоненти форм Ant Design окремо, тому перед тим, як починати використовувати форми на проекті зі встановленим Ant Design, важливо уважно ознайомитися з розділом документації по Form, щоб не виконувати зайвої роботи, а застосовувати вже розроблений інструментарій.

Ant Design надає багато корисних користувацьких компонентів для розробки форм. В основному це поля, що вже стали стандартом, перемикачі і селектори з деякими особливостями [3]:

- Input та InputNumber — це два різні компоненти;
- DatePicker вміє вибирати лише один день чи період;
- RangePicker часто не працює на мобільних пристроях, доводиться використовувати два DatePicker;
- TimePicker виконаний у вигляді трьох об'єднаних селекторів (годин, хвилин, секунд), таке рішення може здатися незвичним;
- API компонента Upload не надто гнучка.

Звичайно, бібліотека Ant Design не позбавлена недоліків. До них відноситься, наприклад, погана адаптація до мобільних пристроїв (є окремий Ant Design Mobile). І все ж таки ця технологія вигідно виділяється масштабом, складовими, великим набором готових рішень. Існує навіть офіційний бойлерплейт для адмін-панелей (Ant Design Pro). Ant Design рекомендується використовувати для швидкого початку не дуже вимогливих до веб-дизайну проектів, MVP-версій, проектів, які не передбачають широкого охоплення аудиторії.

РОЗДІЛ 2. ЗАСТОСУВАННЯ ПЛАТФОРМИ .NET ТА .NET 7 ДЛЯ РОЗРОБКИ САЙТІВ

2.1. Основні принципи та концепції, що лежать в основі платформи .NET

Платформа .NET Framework — це технологія, яка підтримує створення та використання вебдодатків та програм Windows. Основна мета, що враховувалися під час розробки платформи .NET Framework – забезпечення узгодженого об'єктно-орієнтованого середовища програмування для локального збереження та виконання об'єктного коду, розподіленого в Інтернеті або для віддаленого виконання.

Платформа .NET надає розробникові середовище виконання коду, в якому [4]:

- зведено до мінімуму ймовірність конфліктів у процесі розгортання програмного забезпечення та управління його версіями;
- гарантується безпечне виконання коду, включаючи код, створений невідомим або стороннім виробником;
- виключені проблеми з продуктивністю середовищ виконання скриптів або інтерпретованого коду;
- забезпечуються єдині принципи розробки для різних типів програм, таких як додатки Windows та вебдодатки;
- забезпечується взаємодія на основі промислових стандартів, що гарантує інтеграцію коду платформи .NET Framework з будь-яким іншим кодом.

2.2 Ключові складові платформи .NET

Ядром .NET Framework вважається CLR (Common Language Runtime) – «віртуальна машина .NET», команди якої визначаються об'єктно-орієнтованою мовою CIL (Common Intermediate Language, загальна проміжна мова). Другим

важливим компонентом платформи .NET Framework є Framework Class Library (FCL, єдина бібліотека класів для всіх мов платформи .NET).

CLR є основою платформи NET Framework, його можна вважати фактором, який керує кодом під час його виконання та надає основні сервіси, такі як управління пам'яттю, управління потоками та віддалену взаємодію. CLR накладає умови суворої типізації та інші види перевірки точності коду, що забезпечують безпеку та надійність. Фактично основним завданням CLR є керування кодом.

Бібліотека класів є комплексним об'єктно-орієнтованим набором типів, що можуть повторно використовуватись та застосовуються для розробки додатків різної складності: від додатків, що запускаються з командного рядка до додатків з графічним інтерфейсом (GUI) та таких, що використовують останні технологічні можливості ASP.NET (веб-форми та веб-служби XML).

Платформа .NET Framework не лише надає кілька базових середовищ виконання, але й підтримує їх розробку незалежними виробниками. Наприклад, ASP.NET розміщує середовище виконання та забезпечує масштабоване середовище для керованого коду на стороні сервера. ASP.NET працює безпосередньо з середовищем виконання, щоб забезпечити виконання додатків ASP.NET та веб-служб XML.

Середовище CLR керує пам'яттю, виконанням потоків, виконанням коду, перевіркою безпеки коду, компіляцією та іншими системними службами. Ці засоби є внутрішніми для керованого коду, який виконується у CLR. Також забезпечується надійність коду, через реалізацію інфраструктури суворої типізації та перевірки коду, яку називають системою загальних типів (CTS). CTS виконує самоопис всього керованого коду. Різні мовні компілятори корпорації Microsoft та інших незалежних виробників створюють керований код, що задовольняє системі загальних типів. Це означає, що керований код може приймати інші керовані типи та екземпляри, при цьому гарантує правильність типів та сувору типізацію.

Кероване середовище виконання дозволяє уникнути багатьох проблем, які часто виникають з програмним забезпеченням. Наприклад, автоматично керує розміщенням об'єктів та посиланнями на об'єкти, звільняючи їх, коли вони більше не використовуються. Забезпечує автоматичне керування пам'яттю, що дозволяє виключити дві найпоширеніші помилки додатків: втрат пам'яті та недійсних посилань на пам'ять. Середовище виконання також підвищує продуктивність розробників завдяки використання переваг таких як, бібліотеки класів і компонентів, написаних розробниками інших мов програмування. Мовні компілятори, призначені для платформи .NET Framework, роблять засоби .NET Framework доступними для існуючого коду, написаного на відповідних мовах, суттєво полегшуючи процес перенесення вже написаних програм на нову платформу. Хоча середовище виконання розроблялося для підтримки наступних версій програмного забезпечення, воно також підтримує програмне забезпечення попередніх версій. Взаємодія керованого та некерованого кодів дозволяє розробникам використовувати необхідні компоненти COM та бібліотеки DLL.

Хоча загальномовне середовище виконання надає багато стандартних служб часу виконання, керований код ніколи не інтерпретується. Засіб компіляції на вимогу дозволяє виконувати весь керований код машинною мовою комп'ютера, де він запускається. Нарешті, середовище виконання може розміщуватися у високопродуктивних серверних додатках, таких як Microsoft SQL Server та IIS (Internet Information Services). Така інфраструктура дозволяє використовувати керований код для написання своєї логіки програм, користуючись при цьому високою продуктивністю найкращих виробничих серверів.

Бібліотека класів платформи .NET Framework є об'єктно-орієнтованою і складається з набору типів, які тісно інтегруються з середовищем CLR. Вона надає типи, від яких керований код користувача може наслідувати функції. Це не тільки полегшує роботу з типами .NET Framework, але й скорочує час на вивчення нових засобів платформи .NET Framework. Крім того, компоненти

незалежних виробників можна легко поєднувати з класами платформи .NET Framework.

Наприклад, у класах колекцій .NET Framework реалізується набір інтерфейсів для розробки власних класів колекцій. Користувацькі класи колекцій легко поєднуються з класами .NET Framework. Типи .NET Framework дозволяють вирішувати типові завдання програмування, включаючи роботу з рядками, збирання даних, підключення до баз даних та доступ до файлів. На додачу до цих звичайних можливостей бібліотека класів містить типи, що підтримують багато спеціалізованих сценаріїв розробки.

.NET Framework можна використовувати для розробки наступних типів програм і служб:

- Консольні програми;
- Програми із графічним інтерфейсом Windows (Windows Forms);
- Програми Windows Presentation Foundation (WPF);
- Програми ASP.NET;
- служби Windows;
- Сервісно орієнтовані програми, які використовують Windows Communication Foundation (WCF);
- Програми, які підтримують бізнес-процеси Windows Workflow Foundation (WF).

Класи Windows Forms є повним набором типів, які істотно спрощують розробку графічних інтерфейсів користувача Windows. Під час написання веб-форм ASP.NET можна використовувати класи веб-форм.

2.3. Застосування платформи .NET та .NET 7

Технологія .NET, заснована корпорацією Microsoft, з’явилась в кінці 1990- х під назвою Next Generation Windows Services (NGWS). .NET розвивалась швидко. Першу бета-версію .NET 1.0 випустили наприкінці 2001 року. У 2002 році була опублікована першу версію .NET Framework. В

подальшому компанія представила ще дев'ять оновлень .NET Framework, сім з яких були впроваджені з оновленим форматом Visual Studio. Це платформа з відкритим кодом для розробки різних застосунків: вебдодатків, мобільних додатків, десктопних додатків, інтернет-ресурсів та ігор. .NET Framework сумісний з різними мовами програмування, бібліотеками, та редакторами кодів програм. Цією платформою користуються розробники програмного забезпечення багатьох успішних компаній.

Якщо на початку .NET застосунки могли запускатись тільки на Windows, то на даний час вебдодатки, побудовані з використанням .NET, також можуть бути розміщені на будь-якому хостингу на базі Linux. А .NET MAUI дозволяє додавати програми на основі .NET для macOS. Є можливість запускати програми на Android або iOS-пристрої Xamarin. Починаючи з версії .NET 6 додатки можуть запускатись на x86 і x64, Arm32 і Arm64. .NET чудово підходить для розробки корпоративних систем фінансових і банківських установ, страхових компаній тощо. Це надійний і безпечний Framework, коли мова йде про обробку великих обсягів даних і високих навантажень на платформу.

Впродовж 12 років .NET був платформою із закритим вихідним кодом. Проте, політика Microsoft змінилася, і код для серверної частини платформи став відкритим. З того часу кількість функцій фреймворків почала збільшуватися і, починаючи з 2022 року, .NET став повністю безкоштовним незалежно від мети, для якої ви його використовуєте.

.NET широко використовується в розробці ігор та є кросплатформним і складається з багатьох інструментів, бібліотек і мов програмування. Ідеально підходить як для 2D, так і для 3D-ігор, працює з ігровими двигунами, такими, як Unity, Monogame, Stride, NeoAxis та багатьма іншими.

Платформа .NET Framework продовжує активно розвиватись та доповнюватись новими можливостями. Нова версія платформи ".NET 7" включає середовище виконання з компілятором RyuJIT JIT, специфікації API, бібліотеки WPF та інші інструменти. Завдяки .NET 7 розробник може

створювати кросплатформні програми для веб-браузера, хмари, комп'ютера, пристроїв Інтернету речей і мобільних пристроїв. У цій новій версії .NET бібліотека базових класів (BCL, Base Class Library) була уніфікована для використання в різних типах програм: програми для настільних систем, веб-додатків, хмарних платформ, мобільних додатків, ігор, вбудованих програм і систем машинного навчання. Для розробки різних типів програм можна використовувати загальний SDK, середовище виконання та набір бібліотек. Є можливість прив'язувати програму до API, сумісного з версією .NET 7.

У версії .NET 7 покращена підтримка архітектури ARM64 і продовжено роботу над досягненням паритету продуктивності для програм .NET під час роботи на архітектурах x86 і ARM64, покращена ефективність кешу L3 під час виконання в системах ARM64. Команди LSE використовуються для обмеження доступу до пам'яті паралельного потоку, що призводить до зменшення затримки виконання коду на 45%.

До бібліотеки було додано драйвери, які використовують векторні типи Vector64, Vector128 і Vector256, а функції EncodeToUtf8 і DecodeFromUtf8 були переписані на основі команд векторної графіки, що підвищило їх продуктивність на 60%. Для окремих функцій (NarrowUtf16ToAscii, GetIndexOfFirstNonAsciiChar) приріст продуктивності досягає 35%. Загалом швидкість проходження тестів на платформі ARM64 зросла на 10-60%.

.NET Upgrade Assistant дозволяє полегшити перенесення старих програм на версії .NET 6 або .NET 7. У новій версії розширено підтримку портування додатків ASP.NET на ASP.NET Core, додано парсери коду та засоби перевірки для WinForms, WPF і бібліотек класів, реалізовано підтримку аналізу виконуваних файлів, додано підтримку UWP. А також запропоновано загальні інтерфейси для математичних функцій, передбачена можливість визначення статичних елементів у віртуальних інтерфейсах, що дозволяє застосовувати загальні методи програмування для виконання математичних операцій без точної інформації про тип значень.

Серед окремо визначених змін можна назвати наступні: додано підтримку для компіляції в самодостатні виконувані файли, де весь проект спочатку компілюється до рідного коду цільової платформи без використання проміжного коду та без використання JIT; .NET SDK реалізує можливість обмежити використання наданих шаблонів проектів (наприклад, можна визначити, для яких операційних систем дійсний шаблон); у NuGet додався режим централізованого керування пакетами, який дозволяє керувати взаємними зв’язками для кількох проектів одночасно.

РОЗДІЛ 3. ОСОБЛИВОСТІ SQLITE ТА ЇЇ ВИКОРИСТАННЯ

3.1. Загальна інформація про реляційні бази даних та їх застосування у додатках

Виникнення технологій баз даних припадає на початок 60-х років XX століття. Їх швидкому розвитку сприяли потреби в обробці інформації, досягнення в суміжних областях інформаційних технологій таких, як операційні системи, мови програмування, технічне забезпечення. Із самого початку було зрозуміло, що цей напрям має самостійне значення і буде відігравати одну з ключових ролей у побудові інформаційних систем різного призначення. Зароджувалися певні ідеї щодо управління ресурсами даних, формувалися основи методології побудови систем баз даних.

Під *інформацією* розуміють будь-які відомості про будь-яку подію, сутність, процес, які є об'єктом певних операцій: передачі, перетворення, зберігання або використання даних. *Дані* можна визначити, як інформацію зафіксовану у певній формі, яка придатна для подальшої обробки, зберігання і передачі. Функції управління інформацією в інформаційних системах виконують системи управління базами даних. Відсутність централізованих методів керування доступом до інформації була однією з причин появи СУБД. Наступною причиною стала необхідність забезпечення ефективної паралельної роботи багатьох користувачів з одним тим самим файлом даних.

Розглянемо основні поняття, якими оперують в теорії баз даних. *База даних(БД)* – це будь-яка пов’язана між собою за певними ознаками інформація, що зберігається і організовується певним чином, як правило у вигляді таблиць, це свого роду електронна картотека, сховище даних, що складається з одного або кількох файлів. *Система управляння базами даних (СУБД)* - сукупність мовних та програмних засобів, призначених для створення, ведення та сумісного використання БД багатьма користувачами. *Банк даних* - система спеціально організованих даних(баз даних, програмних, технічних, мовних, організаційно-методичних засобів), призначених для створення, ведення та сумісного використання БД багатьма користувачами. З поняттям база даних

тісно пов’язане поняття *предметної області*. Під предметною областю розуміють об’єкт для якого створюється база даних та реальні інформаційні процеси в системі: галузь виробництва, підприємство та ін.

До головних функцій СУБД належать такі:

- управління даними у зовнішній і оперативної пам'яті;
- управління транзакціми і паралельним доступом до даних;
- відновлення БД;
- підтримка мов БД;
- контроль доступу до даних;
- підтримка цілісності даних;
- підтримка незалежності даних;
- підтримка обміну даними.

Банк даних містить не тільки дані, що зберігаються в базі, але і опис БД. Опис БД належить до метаінформації, або метеданих, тобто інформації про інформацію. Опис БД часто називають *схемою даних*.

Централізоване сховище метаінформації називається *словником даних*. Це каталог даних, який використовується для централізованого накопичення і опису ресурсів даних. Словник даних відповідає за визначення всіх елементів даних та містить:

- імена, типи і розміри елементів даних;
- імена зв'язків;
- обмеження даних по підтримці цілісності;
- схеми БД (зовнішня, концептуальна і внутрішня), а також відображення між ними;
- імена користувачів і їх права доступу до даних;
- статистичну інформація.

Програмні засоби БД включають ядро СУБД, транслятори, утиліти, прикладні програми. *Мовні засоби* поділяються на мови опису даних (МОД) і мови маніпулювання даними (ММД). МОД призначені для опису схеми БД або

її частини, з їх допомогою виконується опис типів даних, їх структур і зв'язків між ними. Відповідно до отриманого опису СУБД знаходить в програмі необхідні дані, перетворює їх і передає в прикладну програму. ММД виконуть функції вибірки з БД даних за певними умовами, зміну даних, додавання даних, вилучення даних та ін.

ММД розподіляються на процедурні та непроцедурні (декларативні). При користуванні процедурними мовами треба вказати, які дії і над якими об'єктами необхідно виконати, щоб отримати результат. У непроцедурних мовах вказується, що треба отримати у відповідь, а не як цього досягти. Процедурні мови можуть розподілятись за основними інформаційними одиницями, якими вони маніпулюють. Це можуть бути мови, що орієнтовані на обробку даних по записах або мови, що орієнтовані на операції над множинами записів. Прикладами непроцедурних мов є мови, засновані на реляційному численні. До них належить мова структурована запитів SQL.

На кожному етапі з банком даних пов’язана своя категорія користувачів. Осіб, які використовують відомості, що зберігаються в базі даних, називають *кінцевим користувачами*. Це основна категорія користувачів, для яких створюється БД. Від них не вимагається якихось особливих знань в області обчислювальної техніки та мов програмування. Інша категорія користувачів баз даних – це розробники. До них відносяться *прикладні та системні програмісти*.

Особливе місце серед всіх категорій користувачів займає *адміністратор БД*(АБД). Це може бути один або кілька користувачів, які на стадії розробки відповідають за оптимальну організацію з точки зору одночасної роботи багатьох користувачів, на стадії експлуатації – за коректність роботи даного банка інформації в багатокористувацькому режимі. На стадії розвитку та реорганізації АБД відповідає за можливість коректної реорганізації банка без зміни чи припинення його експлуатації.

Адміністрування даними і базою диних передбачає управління інформаційними ресурсами, проектування БД, управління реалізацією

застосувань, підтримку цілісності даних, захист даних, спостереження за поточною продуктивністю системи, а також реорганізацію БД при необхідності. Тож для такої роботи призначають *адміністратора даних* та *адміністраторі бази даних*, функції яких дещо відрізняються.

Адміністратор даних – людина, яка відповідає за управління даними (планування БД, розробку і супроводження стандартів, прикладних алгоритмів і ділових процедур), а також за концептуальне і логічне проектування БД. *Адміністратор БД* – людина, яка відповідає за фізичну реалізацію БД (фізичне проектування і втілення проекту), за забезпечення безпеки і цілісності даних, за супроводження операційної системи, а також за забезпечення максимальної продуктивності застосувань і користувачів. Отже, адміністратор даних і адміністратор БД виконують функції управління структурою БД, управління паралельною обробкою, розподіл прав і обов'язків при обробці, забезпечення безпеки БД, відновлення БД, управління СУБД, підтримка репозиторію даних.

На початку 70-х років були сформовані теоретичні основи сучасних технологій БД і остаточно сформувався самостійний напрям інформаційних технологій – наука про бази даних. Головною подією цього періоду стало виникнення *реляційних баз даних*. В основі реляційної моделі лежить поняття відношення. Реляційна модель є простою і зрозумілою, а її фізичне представлення добре реалізується на комп'ютері. До недоліків реляційних моделей можна віднести складність реалізації ієрархічних і мережних зв'язків. У 70-х роках почали формуватися підходи в БД, які пов'язані з використанням апарата логіки в якості моделі даних. Ці роботи привели до створення дедуктивних БД. Розвиток цього напряму поклав початок створенню баз знань.

У 80-х роках XX століття реляційні БД набули домінуючого положення. Однак уже тоді існувало і постійно розширювалося коло застосувань, для яких ця технологія була неадекватною. Це стосується мультимедійних застосувань, систем, які оперують просторовими даними і т. ін. В середині 80-х років успіхів досягло об'єктно-орієнтоване програмування. Під його впливом і в зв'язку з необхідністю реалізації нетрадиційних застосувань, вимоги яких погано

узгоджуються з можливостями реляційних систем, почались роботи з практичної реалізації об'єктно-орієнтованих БД. Об'єктно-орієнтовані БД у порівнянні з реляційними БД мають можливість відображати інформацію про складні взаємозв'язки об'єктів, визначати функції обробки окремих записів. До недоліків об'єктно-орієнтованих моделей належить висока понятійна складність, низька швидкість виконання запитів, незручність обробки даних.

Бажання розробників реляційних БД зберегти перші позиції і підвищити ефективність реляційної моделі, привели до розробки об'єктно-реляційної моделі, на основі якої почали розроблятися об'єктно-реляційні БД. Ці БД дозволяють забезпечити високу наочність представлення інформації і підвищити ефективність її обробки. До недоліків об'єктно-реляційних моделей можна віднести складність забезпечення цілісності і несуперечливості даних.

На початку 90-х років почало збільшуватися значення інформаційного забезпечення систем підтримки прийняття рішень. Це привело до необхідності створення багатомірних СУБД і на їх основі розробки інформаційних сховищ. Ці системи використовують історичну інформацію, яка представляється в агрегованому вигляді. На основі цієї інформації виконується аналітична обробка, прогнозування даних, а також інтелектуальний аналіз даних. В цей час також велика увага приділялася розвитку розподілених систем. Успіхи в розробці комп'ютерних мереж стимулювали дослідження в технологіях розподілених БД, були розроблені архітектурні концепції клієнт – сервер.

Одним з найбільших досягнень 90-х років в області інформаційних технологій стало створення відкритої глобальної розподіленої неоднорідної гіпермедійної інформаційної системи, яка використовує комунікаційну мережу Internet – WWW (Web). З самого початку виконувались спроби інтегрувати системи БД у Web. Одним з напрямів роботи є інтеграція структурованих і слабо структурованих даних у Web, проводяться роботи зі створення БД на мові XML.

У 2000-ні рр. головним нововведенням є підтримка та застосування XML у БД. У даний час в багатьох комп'ютерних компаніях здійснюються роботи зі

створення цифрових бібліотек − інформаційних систем, які призначені для зберігання, пошуку, обробки, аналізу і розповсюдження інформаційних ресурсів різної природи – структурованих і слабо структурованих даних, мультимедійної інформації, повнотекстових документів. Для створення таких систем використовуються Web-технології, розробляються методи інтеграції неоднорідних ресурсів, розпізнавання і пошуку інформаційних ресурсів.

3.2. Порівняння SQLite з іншими системами управління базами даних

У сучасних інформаційних системах база даних є важливою складовою будь-якого веб-сайту чи проекту розробки. На ринку IT- технологій є різні бази даних на вибір: Oracle, MySQL, SQL Server, PostgreSQL, SQLite та ін. Кожна з них має свої переваги та недоліки. Усі бази даних використовуються для керування даними, їх підтримки та маніпулювання даними на найпростішому рівні. Існує два типи моделей даних: реляційні(RDBMS) та нереляційні(NoSQL). NoSQL — це неструктурована модель бази даних, яка все ще розвивається. Системи управління реляційними базами даних є більш структурованою і широко використовується в проєктуванні різних додатків.

Для роботи над некомерційними проектами краще підходять безкоштовні СУБД, які є у вільному доступі. СУБД Oracle підійде великим корпораціям, які працюють з чималими обсягами даних через високу ціну ліцензованого продукту та високе споживання системних ресурсів. SQL Server найкраще використовувати компаніям, які застосовують продукти Microsoft. У цієї СУБД високе споживання ресурсів та ціна на ліцензію. Якщо розглядати безкоштовні СУБД, то варто зупинити увагу на MySQL або PostgreSQL. Обидв платформи підходять для розробки вебдодатків, мають простий інтерфейс та економне споживання ресурсів.

Найбільш популярним серед розробників на даному етапі є MySQL, тому здійснимо порівняння саме цього продукту з SQLite. MySQL — це система керування реляційною базою даних, безкоштовне програмне забезпечення з

відкритим вихідним кодом, ліцензоване за Загальною публічною ліцензією GNU, яке підтримується корпорацією Oracle. дуже активно використовуєтьс при розробці додатків для доступу до записів бази даних і керування ними. SQLite також є системою керування реляційною базою даних для вбудованих систем. Це механізм транзакційної бази даних SQL, який є автономним, безсерверним і не потребує налаштування. У таблиці 1 наведено порівняння цих двох СУБД за основними параметрами.

Таблиця 1. Порівняльний аналіз MySQL та SQLite

MySQL	SQLite
Проєкт з відкритим вихідним кодом, яким володіяє Oracle	Проєкт з відкритим кодом, який є у вільному доступі
Для зв'язку через мережу знадобиться архітектура клієнт-сервер. Потребує використання сервера для функціонування.	Автономна база даних, яка не потребує сервера. Механізм бази даних інтегрований у програму
Підтримує багато типів даних, серед них: короткий текст (Tinytext), текст (Text), середній текст (Mediumtext), довгий текст(Longtext), чілочисельний (Integer), десятковий (Decimal)., подвійної точності (Double), з плаваючою крапкою (Float), дата (Date), час (Time), символьний (Varchar) та ін.	Підтримує Blob, Integer, Null, Text і Real.
Перед копіюванням або експортом інформацію необхідно стиснути в один файл. Це буде трудомістким завданням для великих баз даних. Розмір сервера MySQL приблизно 600 МБ	Розмір бібліотеки становить приблизно 250 КБ. Інформація зберігається в одному файлі, завдяки чому її легко тиражувати, без налаштувань, тому операцію можна виконати з мінімальною підтримкою.
Добре розроблена система керування користувачами, підтримує багатокористувацький режим роботи та різні рівні дозволів.	Не має можливості керування користувачами, непридатний для бегатокористувацьких баз даних
Добре розроблена система керування користувачами, підтримує багатокористувацький режим роботи та різні рівні дозволів.	Не має можливості керування користувачами, непридатний для бегатокористувацьких баз даних

Масштабований, може обробляти більшу базу даних з меншими зусиллями.	Є хорошим вибором для невеликих баз даних, необхідна пам’ять збільшується в міру розширення бази даних. Оптимізація продуктивності складніша.
Має кілька вбудованих механізмів безпеки: ім’я користувача, пароль та автентифікації SSH.	Немає вбудованого методу аутентифікації. Будь-хто має доступ до файлів бази даних.
Вимагає додаткових налаштувань	Простий у налаштуванні і не вимагає багато конфігурацій.

3.3. Як обрати систему управління базами даних для свого додатку

Основне завдання СУБД – це створення, керування та зберігання інформації у базі даних.

В залежності від моделі даних чи виду доступу, розглядають наступні СУБД:

- Мережеві;
- Реляційні;
- Об'єктно-орієнтовані;
- Клієнт-серверні;
- Файл-серверні;
- Вбудовані.

Основні функції СУБД такі:

- керування базою в оперативній пам'яті;
- управління базою у зовнішній пам'яті;
- підтримка мов, які має база даних;
- копіювання та відновлення при потребі.

Щоб визначитись, яка СУБД найкраще підходить для проекту, потрібно їх ретельно розглянути та проаналізувати. Кожна з них має свої переваги та недоліки, тому, щоб вибрати відповідну, слід зіставити усі «за» та «проти».

Розглянемо 5 найкращих СУБД за рейтингом 2022 року: MySQL, PostgreSQL, SQLite, Oracle та MongoDB.

MySQL – СУБД для баз даних реляційного типу. Серед плюсів варто відмітити подвійне ліцензування: поширюється за умовами ліцензії GPL або ліцензією від Oracle. MySQL легко інтегрується з великою кількістю платформ: Linux, macOS, багатьма версіями Windows, та ін. Підтримує широкий асортимент мов програмування: API, C, C++, Java, Python, PHP, Компонентний Паскаль, та інші. Недоліками є фрагментарне використання SQL (якщо раніше ви працювали зі стандартною мовою, при впровадженні СУБД можуть виникнути проблеми), діри в безпеці (при виконанні деяких операцій MySQL може спровокувати DDos-атаку на базу даних), платна техпідтримка (навіть для безкоштовних версій). MySQL підходить для розробки простих сайтів, часто нею користуються під час створення інтернет-магазинів.

PostgreSQL – це СУБД для баз даних об'єктно-реляційного типу. Плюсами цієї СУБД є індексування географічних та геометричних об'єктів, функція успадкування, викристання надійних механізмів транзакцій, додаткова система розширення мов програмування, вбудована підтримка слабо структурованих даних, високопродуктивність реплікацій, широкий набір типів даних. Але з системою PostgreSQL часто виникають труднощі при оновленні. PostgreSQL є найкращою для обробки даних усередині системи, в СУБД є явна перевага – повнотекстовий пошук, який використовується у проєктах, де потрібна ця функція.

SQLite – система керування базами даних компактного вбудованого типу, вихідний код бібліотеки якої зберігається у спільному доступі. Переваги СУБД: підтримка динамічного типування даних, у нових версіях системи введено обмеження, що перевіряються із загальним набором тестів. Основний недолік полягає у відсутності системи користувачів, через невеликі обсяги бази даних, що трохи ускладнює використання даної СУБД. SQLite користуються при створенні вебдодатків, де є необхідність використання бази даних.

Oracle – об'єктно-реляційна система управління базами даних, одна з провідних у світі. Плюсами Oracle є підтримка багатьох апаратно-програмних платформ від Linux до Windows один із найкращих способів зберігання даних, довіра до цієї системи досить висока, якісне програмування. Мінуси – високе споживання системних ресурсів, висока ціна ліцензованого продукту, скдадна у впровадженні та обслуговуванні.

MongoDB – СУБД документно-орієнтованого типу. Серед переваг відмінна масштабованість, підтримка json, можливість роботи з будь-якими видами даних. Проте, транзакції із системою MongoDB зазвичай складні, тому не всі можуть впоратися із таким завданнями. Для роботи з реляційними базами доведеться вручну переписати код. MongoDB ідеально підходить для проєктів, де наперед відома структура даних, а також там, де потрібно зберегти великий обсяг даних з урахуванням гнучкості структури.

Складно визначити, яка з вищезгаданих СУБД найкраща. Програмування процес творчій та залежить від запиту замовника та уподобань розробників, тому кожен розробник самостійно вирішує, яка система йому підходить найкраще. Розробка додатків різного типу вимагає застосування різних СУБД, комусь підійде проста у використанні, а комусь буде потрібна СУБД зі складними алгоритмами та схемами. Варто шукати найкращу СУБД для кожного завдання окремо, враховуючи особливості кожного проєкту. Багато залежить і від самого програміста, адже вміння працювати з різними СУБД показник професіоналізму.

РОЗДІЛ 4. ПРОГРАМНА РЕАЛІЗАЦІЯ ПОРТАЛУ НОВИН

4.1. Види новинних сайтів та вимоги до їх створення

Поняття «Інтернет-журналістика» з’явилося відносно недавно і представляє собою різновид журналістики. Близькими до цього терміну є також назви «мережева журналістика», «вебжурналістика», «онлайн-журналістика», «цифрова журналістика». Все вище назване передбачає поширення журналістських матеріалів через мережу Інтернет. Інтернет-журналістика поєднує собою всі різновиди медіа і передбачає збір, перевірку, підготовку, подачу інформації та організацію редакційної роботи. Важливим критерієм є подання нової інформації.

Новинний сайт являє собою інтернет-портал, де можна розмістити інформаційний контент, пов'язаний з різними резонансними фактами. Такий ресурс повинен бути надійним, зручним у використанні та здатним витримувати великий наплив відвідувачів.

Розрізняють наступні види новинних сайтів:

- регіональні засоби масової інформації;
- міжнародний новинний портал;
- агрегатор для засобів масової інформації;
- державний новинний портал;
- загально тематичний новинний сайт;
- тематичний новинний ресурс.

Регіональні засоби масової інформації – це, як правило, міські та обласні портали. Такі ресурси висвітлюють події з життя місцевих громад. Наприклад, новини місцевого бізнесу, довідкову інформацію, інформацію від комунальних служб, події культури і політики, тощо. Такі сайти важливо оптимізувати під локальні пошукові запити.

Міжнародні новинні портали публікують контент кількома мовами і, зазвичай, мають багатомовний інтерфейс. Такі сайти розраховані на великий трафік і високе навантаження. Щоб забезпечити стабільну роботу міжнародних порталів, застосовують такі архітектурні та технологічні рішення, як CDN і кешування.

Агрегатори збирають і публікують анонси або матеріали зі списку обраних медіа. Інформація для таких сайтів збирається автоматично. Розробка такого виду ресурсу передбачає проектування веб-сайту, який збирає різний контент (оголошення, тури, новини, інформацію про товари та послуги, відгуки та ін.) з різноманітних джерел. Інформація систематизується та розміщається в одному місці. Технічно агрегацію реалізують за допомогою API, RSS-стрічок або парсинга.

Завдання державних порталів - проінформувати громадян про нові укази, акти, внутрішні зміни в структурі управління тієї чи іншої установи. Свій портал повинен бути у інстанцій всіх рівнів - від міністерств до місцевих адміністрацій.

На *загально тематичному новинному сайті*, як правило, публікуються матеріали резонансних подій, які відбуваються в середині країни або на міжнародному рівні. Подібні ресурси розраховані на найбільше широку аудиторію та не мають визначеної географічної або іншої прив’язки.

Тематичний новинний ресурс містить контент, присвячений конкретній галузі, виду спорту, розділу науки або іншій сфері. Мета створення нішевого сайту – продемонструвати експертність у визначеній галузі та сформувати осередок аудиторії, зацікавленої в певній інформації, за рахунок глибини розкриття теми та якості публікацій.

Оскільки новинні портали націлені на широку аудиторію, то найважливішим для додатків такого типу є здатність працювати швидко і безвідмовно працювати при будь-яких навантаженнях. Щоб забезпечити стабільну роботу такого ресурсу використовують проектування масштабованої архітектури, оптимізовані функціональні модулі. балансування навантаження

на сервер, кешування графіки та інших статичних елементів. Важливі і популярні новини виділяють в окремі блок за допомогою грамотного UX-дизайну. Як правило, новинні сайти використовують рекламну модель монетизації. Для керування великою кількістю банерів, потрібен зручний інтерфейс. Розродники створюють для цього спеціальні модулі, які дозволяють налаштовувати формат, місце і час показу реклами, готувати рекламну, автоматизувати та налаштовувати інші рекламні процеси.

Як будь який інформаційний ресурс, новинні сайти мають певні вимоги до створення та використання. При створенні вебсайту для ЗМІ слід врахувати наступні фактори:

- інформацію потрібно подавати якісно, використовувати унікальний дизайн та нестандартні рішення до відображення контенту, не використовувати шаблони;
- готувати контент у форматах зручних для сприйняття та подальшого поширення, створювати різні формати новин (відеоновини, фотопублікації, статті, аналітичні дописи, прямі трансляції, збірні публікації, інтерв'ю, репортажі та ін.);
- сайт повинен витримувати значні навантаження, потрібно правильно розробити відповідну архітектуру, щоб він був розрахований на збільшення відвідуваності та бути швидким відповідно до вимог Google Page Speed;
- інформація на новинному ресурсі повинна відповідати поглядам редакції, сайт має бути захищеним від втручання сторонніх осіб у його роботу та в інформаційне наповнення, стійкий до різного типу зломів та маніпуляцій;
- адмін-панель сайту новин повинна бути мегазручною та швидкою, мати безліч корисних функцій, таких як перевірка контенту на унікальність, швидкий переклад, вставка та обробка статичних та анімованих зображень;
- редакція має доповнювати функціонал свого сайту, створювати нові функції та формати подачі інформації, мати можливість представляти контент та адаптовувати інтерфейси під різні потреби та задачі;

- новинний портал має бути мультимовним. Вебсайт новин в Україні має виходити обов’язково українською, але потрібно враховувати і англомовну аудиторію. Отже, структура сайту, готовність адмін-панелі, експортів новин, API, мобільних додатків це фактори, важливі фактори при виборі продукту.

4.2. Функціональні можливості веб-ресурсу

Нами розроблений новинний портал ТАНОМА NEWS. Усі сторінки нашого сайту містять хедер та футер. Хедер (з англ., header) – це шапка сайту, або ще іншими словами заголовок чи колонтитул. На хедері розміщено посилання для повернення на головну сторінку, для переходу на сторінку про сайт, та для переходу на сторінку контактів. Також у ньому розміщено логотип сайту та поточну дату.

Футер (з англ. footer) – блок, розташований в нижній частині сайту. Зазвичай це місце для додаткової інформації, куди можна вставити посилання для переходу по розділах і сторінках. Футер, також, містить логотип сайту посилання на ті ж сторінки що й хедер та контактні дані компанії, короткі відомості про неї. Ще на футері міститься посилання на сторінку введення логіна та пароля для входу авторів. У разі якщо автор аутентифікований тоді дане посилання зміниться на посилання для переходу на сторінку автора та кнопку виходу.

На рис 4.1. відображено головну сторінку сайту. У її центральній частині міститься меню з переліком категорій та авторів. Та безпосередньо перелік карток з новинами. При натисканні на категорію чи автора в меню відбувається фільтрування карток відповідно по категорії чи автору. Також дане меню має функцію згортання. Меню генеруються відповідно даних отриманих з сервера, які зберігаються у базі даних. Картка новини на головній сторінці складається з зображення новини, за умови його наявності (якщо воно відсутнє, то відображається стандартне зображення), назви новини, її опису та імені автора. При виборі читачем картки на головній сторінці, він перенаправляється на

сторінку з цією новиною. Дана сторінка містить заголовок новини, її короткий опис, зображення (за його наявності) текст новини та ім’я автора (рис.4.2).

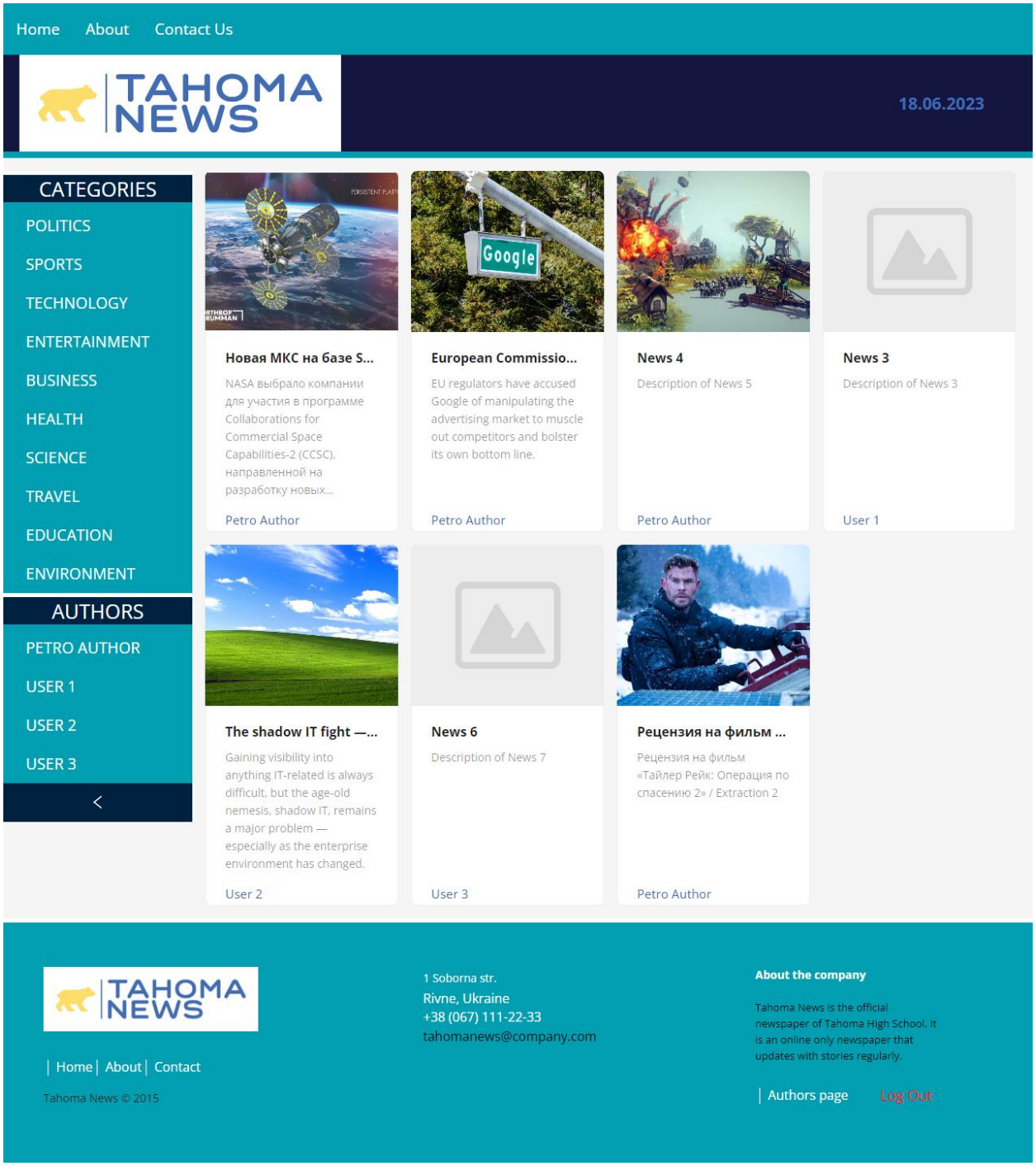

Рис.4.1. Головна сторінка сайту.

Рис. 4.2. Сторінка новин

Наступні дві сторінки (рис.4.3 та рис.4.4) міститься інформація про офіс та дані про компанію, контактні дані компанії.

Рис.4.3. Сторінка з інформацією про новинний портал.

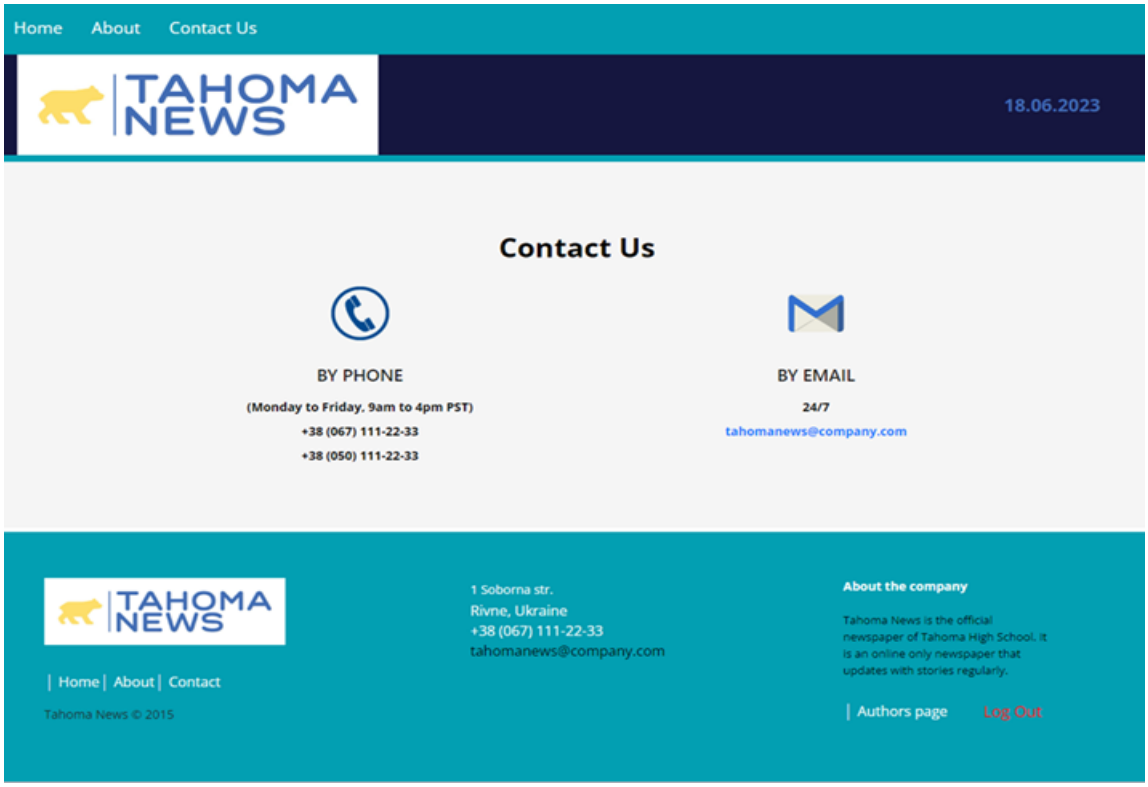

Рис.4.4. Контактна інформація

Для того, щоб мати можливість публікувати новини, потрібно авторизуватись. Це можливо здійснити, перейшовши на сторінку автора для введення логіну та паролю (рис.4.5.). У разі помилкового, або неповного введення користувач отримає відповідні повідомлення. Також у разі переходу неавторизованого користувача на сторінку автора він буде скерований дану сторінку . Після коректного вводу логіна і пароля у футері з’являться кнопка для переходу на сторінку автора та кнопка виходу. Сторінка автора містить таблицю з новинами автора та кнопку додавання новини. В даній таблиці автор бачить короткі дані по новині, може публікувати свою новину, оновити, чи видалити її. Видалення доступне лише для неопублікованих новин. Дана таблиця має можливість сортування по ознаці чи опублікована новина, а також фільтрації по категорії. Також за умови великої кількості новин у автора, вони будуть розділені на сторінки.

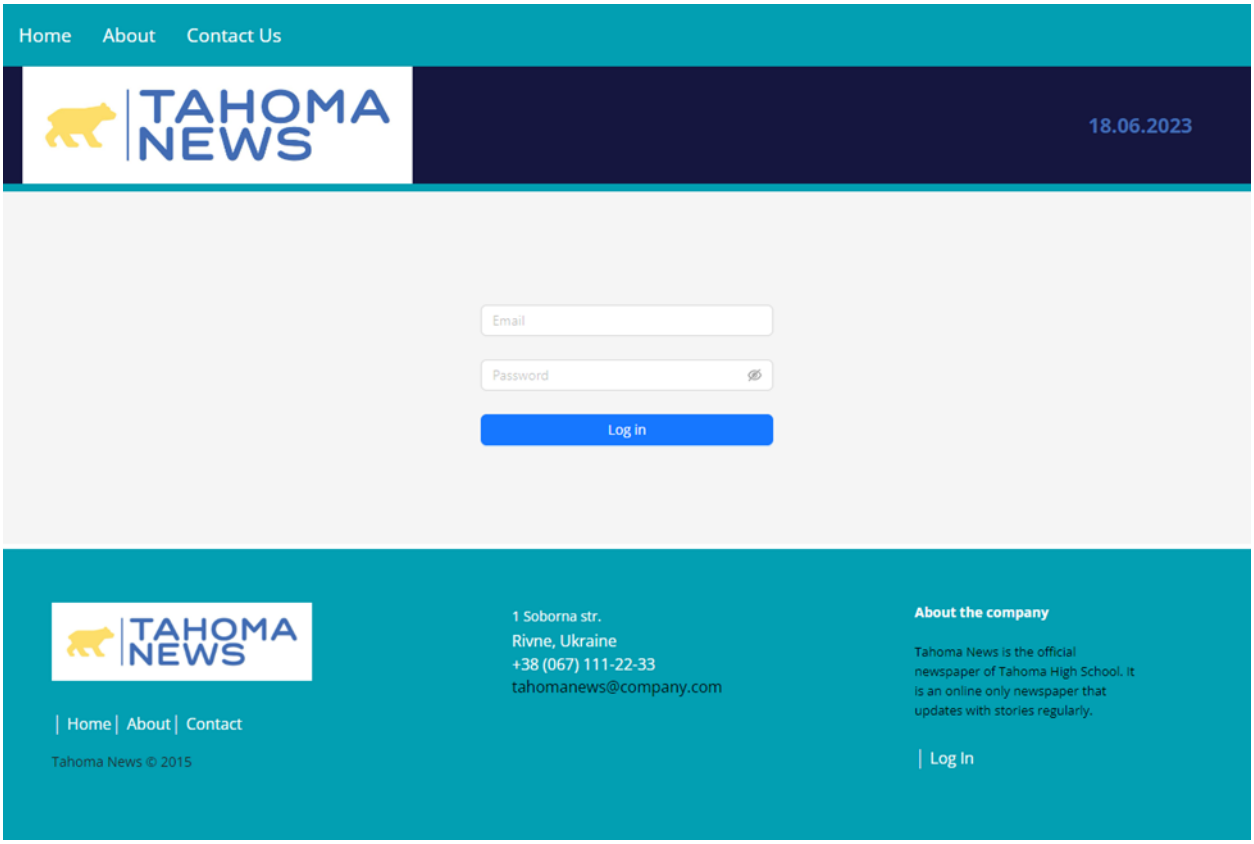

Рис.4.5. Сторінка авторизації

У відповідь на кожну вдалу чи невдалу дію автору показується модульне вікно як на наступному зображенні (рис.4.6.):

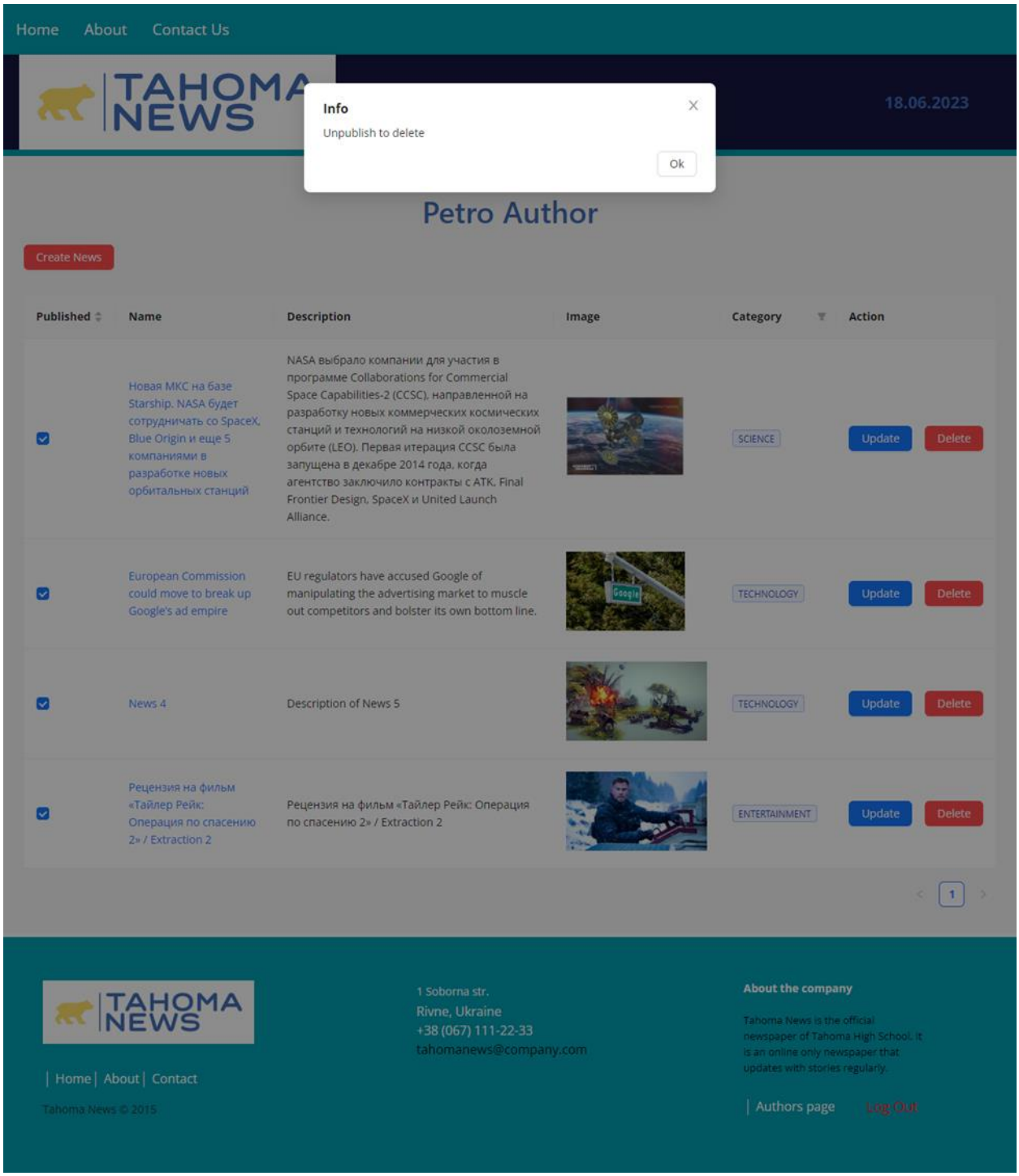

Рис. 4.6. Сторінка автора

При видаленні новини автор повинен спочатку відмінити її публікацію. І лише після цього йому стане доступна можливість видалення. Про результати видалення автор буде проінформований у відповідним повідомленням. При натисканні на кнопку «Створити новину» автор скеровується на наступну

сторінку (рис.4.7.). Тут він має можливість додати усі дані для створення новини.

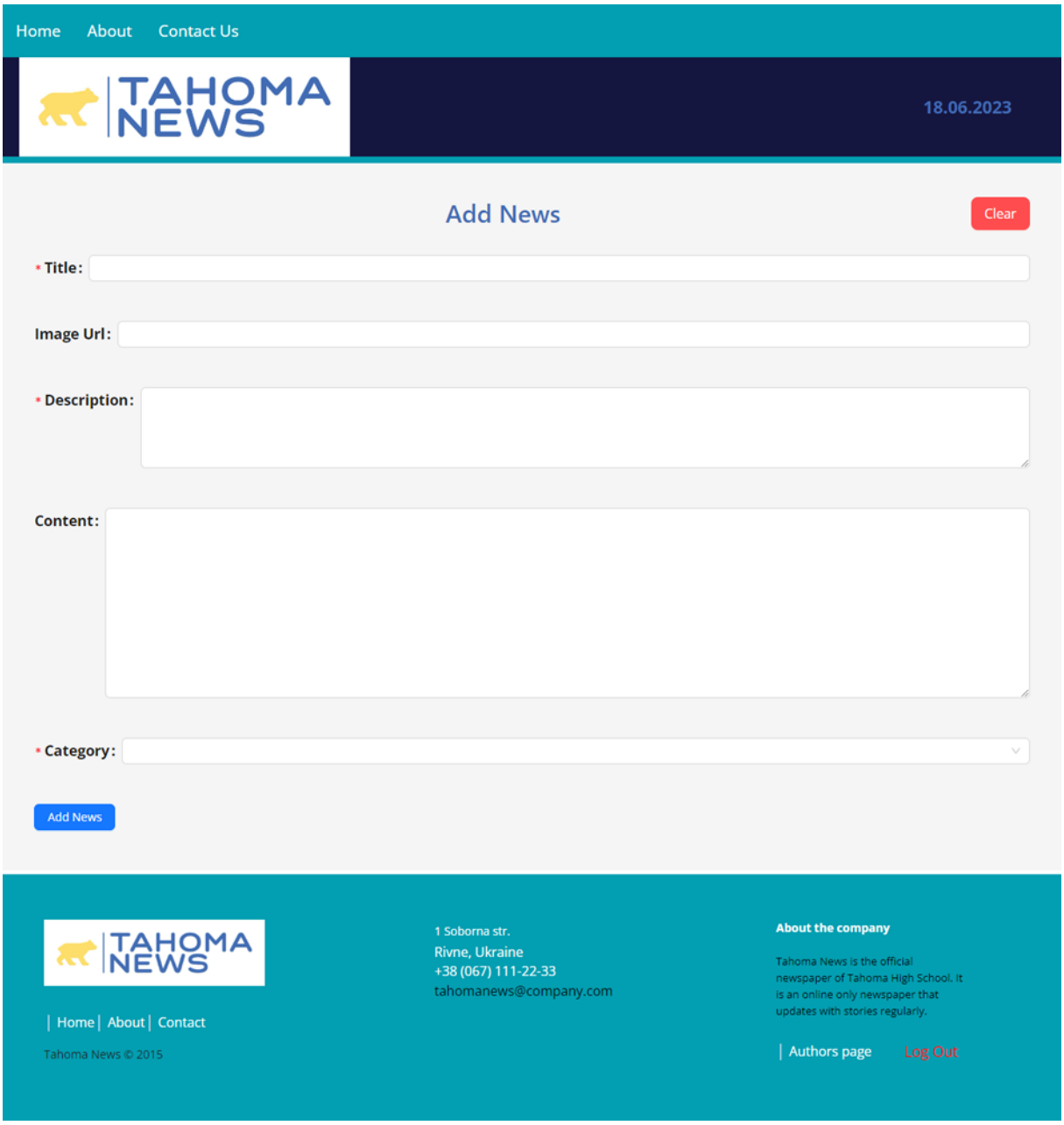

Рис.4.7. Сторінка додавання новини

Обов’язкові поля позначені зірочкою. Також є можливість очистити форму введення. Після натискання на кнопку Додати новину, відбувається валідація введених полів. Якщо автор, щось не заповнив з’явиться відповідний підпис під полем. Після обробки запиту сервером, автор отримає повідомлення на модульному вікні з трьома кнопками. Які дозволять йому перейти на сторінку автора, очистити форму, та просто закрити модульне вікно. При

натисканні кнопки «Update» на сторінці автора. Користувач перенаправляється на відповідну сторінку. Сторінка має вигляд, аналогічний до сторінки додавання новини. Тут автор може видаляти чи змінювати новини. Для збереження змін необхідно натиснути на кнопку «Save».

Розроблений нами новинний портал містить базу даних, в який міститься інформація про новини та авторів. Для проектування бази даних використовувалась СКБД SQLite. Спроектовано реляційну базу даних, яка складається з 5-ти взаємопов'язаних таблиць (Рис.4.8.). Все приведено до 3-ї нормальної форми згідно теорії нормалізації.

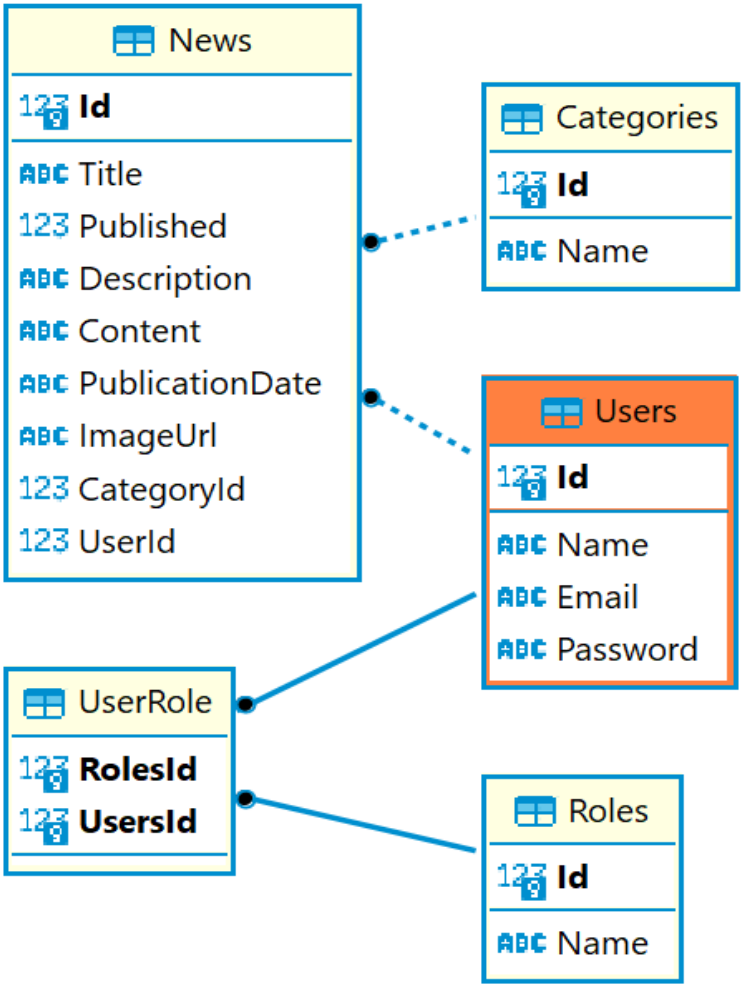

Рис.4.8. Структурна схема бази даних

Для розробки, фронтенду нами було застосовано технології React та Art.Disign. Проектування бекунду виконувалось із застосуванням технології

EntityFrameworkCore. Фрагменти програмного коду розробки бекенду та фронтенду подані в додатках.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи був розроблений інформаційний ресурс – новинний портал, призначення якого надати користувачу можливість публікувати пости та повідомлень на одній платформі. Для проектування порталу застосовувались сучасні технології, які надають можливість розробляти серверну та клієнтську частини сайтів та вебдодатків.

При підготовці до виконання роботи здійснений огляд основних технологій, що застосовуються для розробки подібних інтернет-ресурсів та їх архітектури. Для проектування фронтенду були обрані інструменти React та Art.Design, які на наш погляд найбільше підходять для даного проекту. Для роботи над бекендом ми обрали EntityFrameworkCore. Додаток містить базу даних, спроектовану засобами SQLite, де міститься інформація про дописувачів та їх публікації.

Основна мета роботи була досягнута: розроблено зручний, інтуїтивний користувацький інтерфейс, враховано основні вимоги до подібних ресурсів.

Розроблений новинний портал має наступні функціональні можливості:

- Меню з переліком категорій новин та авторів;
- Можливість створення та видалення публікацій;
- Додавання та оновлення постів;
- Сторінку авторизації;
- Фільтрування повідомлень відповідно до категорії чи автора;
- Редагування профілю користувача;
- Оновлення контенту автором публікації.

Основними перевагами додатку є гнучкість та універсальність візуалізації контенту, написання постів з використанням гіпертекстової розмітки, хороша швидкодія, простота використання. Розроблений новинний портал може використовувати будь-який користувач з вільним доступом до мережі Інтернет.

ДОДАТКИ

Додаток А

Фрагмент написання фронтенду

```
      import { useNavigate } from 'react-router-dom';
import {Layout, Menu} from "antd";
import {NewsLayout} from "../../components/NewsLayout";
import {MenuItemType} from "antd/lib/menu/hooks/useItems";
import {CategoryService} from "../../services/categoryService";
import {NewsService} from "../../services/newsService";
import {NewsDto} from "../../models/NewsDto";
import HomePageNewsCard from "../../components/HomePageNewsCard";
import {Content} from "antd/lib/layout/layout";
import {UserService} from "../../services/userService";
import "./HomePage.css";

type HomePagePropsType = {
    categoryService: CategoryService
    newsService: NewsService
    userService: UserService
}
const HomePage: React.FC<HomePagePropsType> = ({categoryService, newsService,
userService}) => {
    const navigate = useNavigate();
    const {Sider} = Layout;
    const [collapsed, setCollapsed] = useState(false);
    const [categories, setCategories] = useState<MenuItemType[]>([]);
    const [authors, setAuthors] = useState<MenuItemType[]>([]);
    const [selectedCategoryKey, setSelectedCategoryKey] = useState("");
    const [selectedAuthorKey, setSelectedAuthorKey] = useState("");
    const [showNews, setShowNews] = useState<NewsDto[]>([]);
    const [currentPage, setCurrentPage] = useState<number>(1);
    const [pages, setPages] = useState<number>(1);
    const onPage = 12;

    useEffect(() => {
        getLastNews();
        getCategories();
        getAuthors();
    }, []);

    useEffect(() => {
        if (selectedCategoryKey !== "") {
            getCategoryNews();
            return;
        }

        if (selectedAuthorKey !== "") {
            getAuthorNews();
            return;
        }
    }, [selectedAuthorKey, selectedCategoryKey, currentPage]);

    const getCategories = async () => {
        const cat = await categoryService.getCategories();
        if (cat && cat.length > 0) {
            const menu = cat.map<MenuItemType>((c, i) => ({
                key: c.id, label: c.name, onClick: (e) => {
                    setSelectedCategoryKey(e.key);
                    setSelectedAuthorKey("");
                }
```

```
            }));
            setCategories(menu);
        }
    };

    const getAuthors = async () => {
        const authors = await userService.getAuthors();
        if (authors && authors.length > 0) {
            const menu = authors.map<MenuItemType>((c, i) => ({
                key: c.id || 0, label: c.name, onClick: (e) => {
                    setSelectedAuthorKey(e.key);
                    setSelectedCategoryKey("");
                }
            }));
            setAuthors(menu);
        }
    };

    const getLastNews = async () => {
        const news = await newsService.getLastNews(currentPage, onPage);
        if (news) {
            setShowNews(news);
        }
    };

    const getCategoryNews = async () => {
        const categoryId = parseInt(selectedCategoryKey);
        const news = await newsService.getCategoryNews(categoryId, currentPage,
onPage);
        if (news) {
            setShowNews(news);
        }
    };

    const getAuthorNews = async () => {
        const authorId = parseInt(selectedAuthorKey);
        const news = await newsService.getAuthorNews(authorId, currentPage,
onPage);
        if (news) {
            setShowNews(news);
        }
    };

    const onCardClick = (newsId: number) => {
        navigate('/news',{state: newsId});
    }

    return (
        <NewsLayout>
            <Layout className="home-layout" hasSider>
                <Sider width={"235px"} className="home-categories" collapsible
collapsed={collapsed} onCollapse={(value) => setCollapsed(value)}>
                    <div className={"sider-title"}>CATEGORIES</div>
                    <Menu selectedKeys={[selectedCategoryKey]} mode="inline"
items={categories}/>
                    <div className={"sider-title"}>AUTHORS</div>
                    <Menu selectedKeys={[selectedAuthorKey]} mode="inline"
items={authors}/>
                </Sider>
                <Content key={"home-content"} className={"home-content"}>
                    {
                        showNews.map(n => <HomePageNewsCard
                            cardClick={onCardClick}
```

```
                            authorClick={(id) => {
                                setSelectedAuthorKey(id.toString());
                                setSelectedCategoryKey("");
                            }} newsDto={n}/>)
                    }
                </Content>
            </Layout>
        </NewsLayout>
    );
};

export default HomePage;
```

Додаток В

Фрагмент написання бекенду

```
      using NewsPortal.Entities;
using NewsPortal.Models;
using NewsPortal.Services.Contracts;

namespace NewsPortal.Controllers;

/// <summary>
/// AuthorController
/// </summary>
[ApiController]
[Route("api/author")]
//[Authorize(Roles = "author")]
public class AuthorController : ControllerBase
{
    private readonly IAuthorService _authorService;

    /// <summary>
    /// initializes new readers controller
    /// </summary>
    /// <param name="authorService">authorService</param>
    /// <param name="readerService">readerService</param>
    public AuthorController(IAuthorService authorService)
    {
        _authorService = authorService;
    }

    /// <summary>
    /// gets Author.
    /// </summary>
    /// <param name="id">id</param>
    /// <returns>author.</returns>
    [HttpGet("{id:int}")]
    public async Task<ActionResult<User?>> GetAuthorById(int id)
    {
        var user = await _authorService.GetAuthorById(id).ConfigureAwait(false);
        return Ok(user);
    }

    /// <summary>
    /// Retrieves news items by author.
    /// </summary>
    /// <param name="authorId">The ID of the author.</param>
    [HttpGet("news/{authorId:int}")]
    public async Task<ActionResult<IEnumerable<NewsDto>>> GetNewsByAuthor(int
authorId)
```

```
    {
        var news = await _authorService.GetNewsByAuthorAsync(authorId);
        return Ok(news);
    }

    /// <summary>
    /// Creates a new news item.
    /// </summary>
    /// <param name="newsDto">The news item information.</param>
    /// <returns>The created news item.</returns>
    [HttpPost("news")]
    public async Task<ActionResult<bool>> CreateNews(NewsDto newsDto)
    {
        try
        {
            var news = await _authorService.CreateNewsAsync(newsDto);
            return Ok(news);
        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }

    /// <summary>
    /// Updates an existing news item.
    /// </summary>
    /// <param name="newsDto">The updated news item information.</param>
    /// <returns>The updated news item.</returns>
    [HttpPut("news")]
    public async Task<ActionResult<NewsDto>> UpdateNews(NewsDto newsDto)
    {
        try
        {
            var news = await _authorService.UpdateNewsAsync(newsDto);
            return Ok(news);
        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }

    /// <summary>
    /// Deletes a news item by ID.
    /// </summary>
    /// <param name="userId">The user ID.</param>
    /// <param name="newsId">The news item ID.</param>
    /// <returns>True if the news item was deleted successfully, false
otherwise.</returns>
    [HttpDelete("news/{userId:int}/{newsId:int}")]
    public async Task<ActionResult<bool>> DeleteNews(int userId, int newsId)
    {
        try
        {
            var result = await _authorService.DeleteNewsAsync(userId, newsId);
            return Ok(result);
        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }
```

```
    /// <summary>
    /// Publishes a news item by ID.
    /// </summary>
    /// <param name="userId">The user ID.</param>
    /// <param name="newsId">The news item ID.</param>
    /// <param name="publish">publish trigger</param>
    /// <returns>True if the news item was published/unpublished successfully,
false otherwise.</returns>
    [HttpPost("publish/{userId:int}:{newsId:int}:{publish:bool}")]
    public async Task<ActionResult<bool>> PublishNews(int userId, int newsId,
bool publish)
    {
        try
        {
            var result = await _authorService.PublishNewsAsync(userId, newsId,
publish);
            return Ok(result);
        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }
}
```

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. RUVDS.com. URL: https://habr.com/ru/companies/ruvds/articles/343022/ (дата звернення 14.05.2023).
2. Філософія React. URL: https://uk.legacy.reactjs.org/docs/thinking-in-react.html (дата звернення 20.05.2023).
3. Ant Design of React. URL: https://ant.design/docs/react/introduce/(дата звернення 1.06.2023).
4. Документація .NET. URL: https://learn.microsoft.com/ru-ru/dotnet/core/releases-and-support (дата звернення 28.05.2023).
5. SQLite. URL: https://sqlite.org/index.html (дата звернення 24.05.2023).
6. React.js. URL: https://uk.reactjs.org (дата звернення 20.05.2023).
7. Frontend and Backend. URL: https://en.wikipedia.org/wiki/Front_end_and_back_end (дата звернення 25.05.2023).
8. Database. URL: https://en.wikipedia.org/wiki/Database (дата звернення 05.04.2022).
9. NPM. URL: https://www.npmjs.com (дата звернення 12.05.2023).
10. MongoDB. URL: https://www.mongodb.com (дата звернення 20.05.2023).
11. Інтернет-журналістика та блогінг. URL: http://kzgizh.knukim.edu.ua/entrant/specialty/internet-zhurnalistyka-ta-blohinh (дата звернення 25.05.2023).
12. Mongoose. URL: https://mongoosejs.com/ (дата звернення 16.05.2023)
13. NoSQL: URL: https://www.mongodb.com/nosql-explained (дата звернення 18.05.2023).
14. Passport.js. URL: https://www.passportjs.org (дата звернення 22.05.2023).
15. Redux. URL: https://redux.js.org (дата звернення 24.05.2023).