УДК 04.092 DOI: 10.31673/2786-8362.2025.014921

Вишнівський В.В., д.т.н.; **Шикула О.М.,** д.ф.-м.н.; **Довженко Т.П.,** к.т.н.
Двуреченський Є.А., Сєрих С.О., к.т.н.

РОЗРОБКА ВЕБ-ДОДАТКУ «TASKMASTER» ЗА ДОПОМОГОЮ REACT, NODE.JS, MONGODB В ХМАРНОМУ СЕРЕДОВИЩІ

Vyshnivskyi V.V., Shykula O.M., Dovzhenko T.P., Dvurechenskyi Y.A., Sierykh S.O. Development of the web application "TaskMaster" using React, Node.js, MongoDB in a cloud environment. The article analyzed the market of web applications for task management, described existing analogues, their advantages and disadvantages. The client and server parts of the application were implemented. The client part includes components for entering, displaying and managing tasks, while the server part provides data storage and processing in the database.

Using modern technologies React, Node.js, MongoDB and other modern tools, a task management application was created, which ensures high reliability and efficiency of the application. The component approach makes it easy to add new functions and change existing ones without much effort. Thanks to the use of virtual DOM in React and optimized rendering, the application responds quickly to user actions. Using Redux allows you to centrally manage the state of the application, which ensures its stable operation even with a large number of tasks and simplifies its maintenance. Thanks to MongoDB Atlas, data is stored in the cloud, which ensures its availability and security.

Keywords: web application, task management, React, Node.js, MongoDB

Вишнівський В.В., Шикула О.М., Довженко Т.П., Двуреченський Є.А., Сєрих С.О. Розробка веб-додатку «TaskMaster» за допомогою React, Node.js, MongoDB в хмарному середовищі. В статті було проаналізовано ринок веб-додатків для управління завданнями, описані існуючі аналоги їх переваги і недоліки. Реалізовано клієнтську та серверну частини додатку. Клієнтська частина включає в себе компоненти для введення, відображення та управління завданнями, тоді як серверна частина забезпечує збереження та обробку даних у базі даних.

З використанням сучасних технологій React, Node.js, MongoDB та інших сучасних інструментів створено додаток для управління завданнями, що забезпечує високу надійність та ефективність додатку. Компонентний підхід дозволяє легко додавати нові функції та змінювати існуючі без значних зусиль. Завдяки використанню віртуального DOM у React та оптимізованого рендерингу, додаток швидко реагує на дії користувачів. Використання Redux дозволяє централізовано управляти станом додатку, що забезпечує його стабільну роботу навіть при великій кількості завдань та спрощує його підтримку. Завдяки MongoDB Atlas дані зберігаються в хмарі, що забезпечує їх доступність та безпеку.

Ключові слова: веб-додаток, управління завданнями, React, Node.js, MongoDB

Вступ

У сучасному світі ефективне управління завданнями є ключовим фактором успішної діяльності. Веб-додатки для управління завданнями стають все більш популярними завдяки їх здатності забезпечувати зручний і швидкий доступ до інформації. Все більше користувачів обирають веб-додатки для управління своїми завданнями та плануванням. Однак багато існуючих рішень мають свої недоліки, що обумовлює необхідність створення нових, більш функціональних та зручних додатків.

Особливо важливо створити інструмент, який задовольнятиме потреби різних користувачів: студентів, професіоналів, фрілансерів та домогосподарок. Цей додаток повинен забезпечити адаптацію до індивідуальних потреб кожного.

Тому створення функціонального та надійного веб-додатку для управління завданнями, який би відповідав сучасним вимогам користувачів та ринку, є важливим для планування свого часу і виконання поставлених завдань. Цей додаток має забезпечити користувачів зручним інтерфейсом для створення, редагування, видалення та перегляду завдань, а також ефективною системою збереження та обробки даних.

© Вишнівський В.В., Шикула О.М., Довженко Т.П., Двуреченський Є.А., Сєрих С.О. 2025

Постановка завдання. Основними задачами цього дослідження є:

- провести огляд та аналіз ринку існуючих рішень для управління завданнями, визначити їх переваги та недоліки;
- визначити проблеми, які має вирішити новий додаток, та обґрунтувати вибір технологій для його розробки;
- описати процес розробки додатку, включаючи створення та налаштування проекту, розробку клієнтської та серверної частин, інтеграцію всіх компонентів системи;
- розробити архітектуру додатку та описати основні функції та можливості “TaskMaster»;
- оцінити досягнуті результати, проаналізувати переваги розробленого додатку, а також визначити перспективи його подальшого розвитку.

Аналіз останніх досліджень. У сучасному світі ефективне управління завданнями є ключовим фактором успішної діяльності. Веб-додатки для управління завданнями, або так звані "to-do" додатки, стають все більш популярними завдяки їх здатності забезпечувати зручний і швидкий доступ до інформації та розумне управління своїми справами і як наслідок часом, тому "to-do" є невід’ємним інструментом для успішного тайм-менеджменту.

Розглянемо найбільш популярні рішення на ринку:

Todoist – це один з найпопулярніших "to-do" додатків, який надає користувачам можливість створювати та організовувати завдання. Основні функції Todoist включають:

- створення і редагування завдань;
- встановлення термінів виконання завдань;
- пріоритезація завдань;
- додавання описів та вкладень до завдань;
- сортування завдань за різними критеріями;
- підтримка колективної роботи.

Сильні сторони Todoist включають зручний інтерфейс, гнучкість в організації завдань та інтеграцію з багатьма іншими сервісами. Однак, деякі розширені функції доступні тільки у платній версії.

Microsoft To Do – це ще один популярний "to-do" додаток, який інтегрований з іншими продуктами Microsoft, такими як Outlook та Office 365. Основні можливості Microsoft To Do включають:

- створення і редагування завдань;
- встановлення термінів виконання та нагадувань;
- пріоритезація завдань;
- додавання нотаток до завдань;
- сортування завдань за різними критеріями;
- підтримка колективної роботи.

Сильні сторони Microsoft To Do полягають у його інтеграції з іншими продуктами Microsoft, що робить його зручним для користувачів Microsoft. Проте для користувачів, які не використовують інші продукти Microsoft, цей додаток може виявитися менш привабливим.

Google Keep – це простий і зручний "to-do" додаток від Google, який надає базові функції для управління завданнями. Основні функції Google Keep включають:

- створення і редагування завдань;
- встановлення нагадувань;
- додавання нотаток і списків;
- сортування завдань;
- інтеграція з іншими продуктами Google.

Сильні сторони Google Keep включають простоту використання та інтеграцію з іншими сервісами Google. Однак, цей додаток має обмежені можливості порівняно з більш потужними інструментами для управління завданнями.

Any.do – це ще один популярний "to-do" додаток, який пропонує комплексний набір функцій для управління завданнями. Основні функції Any.do включають:

- створення і редагування завдань;
- встановлення термінів виконання та нагадувань;
- пріоритезація завдань;
- додавання нотаток і вкладень до завдань;
- сортування завдань за різними критеріями;
- підтримка колективної роботи;
- інтеграція з календарями та іншими сервісами.

Сильні сторони Any.do включають багатофункціональність, зручний інтерфейс та можливість інтеграції з іншими сервісами, що робить його зручним для комплексного управління завданнями та часом.

Незважаючи на популярність і багатофункціональність таких рішень, існує низка проблем, які не повністю вирішуються цими додатками. Проаналізуємо проблемну область, щоб зрозуміти, які саме аспекти потребують покращення і як “TaskMaster» може вирішити ці проблеми.

Проблеми зручності використання: однією з основних проблем багатьох "to-do" додатків є складність інтерфейсу для нових користувачів. Багато додатків мають багатофункціональні інтерфейси, які можуть бути заплутаними і непростими для освоєння. Наприклад, користувачам може бути важко швидко знаходити необхідні функції або інструменти, що знижує ефективність роботи з додатком. Важливість інтуїтивно зрозумілого та простого інтерфейсу важко переоцінити, особливо для користувачів, які лише починають користуватися подібними інструментами.

Обмежені функції у безкоштовних версіях: багато популярних "to-do" додатків мають обмежені можливості у безкоштовних версіях, що може стати серйозною перепоною для користувачів, які не готові платити за розширений функціонал. Наприклад, у Todoist деякі розширені функції доступні тільки у платній версії, що обмежує можливості користувачів без підписки. Це створює бар'єр для доступу до повного функціоналу додатку і знижує його привабливість для широкої аудиторії.

Відсутність функцій сортування і пріоритезації: деякі "to-do" додатки мають обмежені можливості сортування та пріоритезації завдань. Наприклад, можливість сортування завдань за датою створення, пріоритетністю або алфавітним порядком може бути важливою для багатьох користувачів. Відсутність таких функцій знижує ефективність управління завданнями і може викликати незручності у роботі. Забезпечення можливості гнучкого сортування і пріоритезації завдань є важливим аспектом для будь-якого "to-do" додатку.

Аналіз проблемної області "to-do" додатків виявив кілька ключових проблем, які потребують вирішення. Веб-додаток “TaskMaster» спрямований на усунення цих проблем, пропонуючи простий і інтуїтивно зрозумілий інтерфейс, відсутність платної версії та гнучкі можливості сортування і пріоритезації завдань. Це дозволить “TaskMaster» у преспективі зайняти свою нішу на ринку "to-do" додатків і забезпечити користувачам ефективне управління завданнями та підвищення продуктивності.

Метою роботи є створення функціонального та надійного веб-додатку для управління завданнями, який би відповідав сучасним вимогам користувачів та ринку, забезпечував користувачів зручним інтерфейсом для створення, редагування, видалення та перегляду завдань, а також ефективною системою збереження та обробки даних.

Виклад основного матеріалу дослідження

Вибір технологій для розробки веб-додатку. Для розробки веб-додатку «TaskMaster» були обрані сучасні технології, які забезпечують високу продуктивність, надійність та масштабованість. Розглянемо основні технології, використані у проекті.

React [1] – одна з найпопулярніших бібліотек для створення користувацьких інтерфейсів. Вона дозволяє розробляти динамічні та інтерактивні інтерфейси веб-додатків за допомогою компонентного підходу. Завдяки віртуальному DOM і оптимізації оновлень React забезпечує високу продуктивність додатку навіть при великій кількості завдань і користувачів. Вибір React для розробки фронтенд-частини веб-додатку «TaskMaster» обґрунтований його високою продуктивністю, гнучкістю, зручністю використання, масштабованістю, широким вибором готових рішень та активною спільнотою. React легко інтегрується з іншими технологіями, що дозволяє використовувати його в поєднанні з іншими інструментами для створення повноцінних веб-додатків.

Node.js [2] – потужне середовище виконання JavaScript на сервері. Воно дозволяє створювати високопродуктивні серверні додатки з використанням одного і того ж мовного середовища як на клієнтській, так і на серверній стороні. Переваги: Висока швидкість обробки запитів, неблокуюча архітектура, активна підтримка спільноти та велика кількість доступних модулів. Вибір Node.js для розробки серверної частини веб-додатку «TaskMaster» обґрунтований його високою продуктивністю, здатністю обробляти велику кількість одночасних запитів, масштабованістю та єдиним стеком технологій з фронтенд-частиною. Ці переваги роблять Node.js ідеальним вибором для створення високонавантажених, масштабованих та продуктивних веб-додатків, що відповідають сучасним стандартам розробки.

MongoDB [3] – документно-орієнтована база даних NoSQL, яка забезпечує гнучкість та масштабованість. Вона ідеально підходить для зберігання структурованих та неструктурованих даних. MongoDB легко інтегрується з іншими інструментами та фреймворками, такими як Node.js, що дозволяє створювати повноцінні додатки, використовуючи єдиний стек технологій. Її переваги: простота у використанні, гнучка модель даних, висока продуктивність при великих навантаженнях, інтеграція з хмарними сервісами, такими як MongoDB Atlas. Вибір MongoDB для управління базою даних у веб-додатку «TaskMaster» обґрунтований її високою продуктивністю, гнучкістю, масштабованістю та легкістю інтеграції з іншими технологіями.

Окрім основних технологій, таких як React, Node.js та MongoDB, у розробці веб-додатку “TaskMaster» використовуються й інші важливі інструменти, що забезпечують високу продуктивність, зручність та гнучкість додатку. Серед них особливо виділяються TailwindCSS, React Router DOM, Express, Redux, Redux Toolkit та MongoDB Atlas.

TailwindCSS [4, 5] – це утилітарний CSS фреймворк, який надає велику кількість класів для створення стилів без написання власного CSS. TailwindCSS відрізняється від традиційних CSS фреймворків, таких як Bootstrap, своєю гнучкістю та можливістю налаштування під конкретні потреби проекту. Його переваги: швидкість розробки, гнучкість у налаштуванні, мінімізація CSS-коду.

React Router DOM [6] – бібліотека для маршрутизації в React. Її переваги: простота у налаштуванні маршрутів, підтримка динамічних маршрутів, управління навігацією.

Express [7] – мінімалістичний та гнучкий веб-фреймворк для Node.js, який спрощує розробку серверних додатків та API. Express забезпечує простий інтерфейс для створення маршрутів, обробки запитів та відповіді, а також підтримку проміжних обробників (middleware).

Redux та Redux Toolkit [8, 9] – популярна бібліотека та її офіційний пакет для управління станом додатків JavaScript, особливо корисна в поєднанні з React. Вона дозволяє централізовано зберігати стан додатка, роблячи його більш передбачуваним і полегшуючи відстеження змін.

MongoDB Atlas – хмарна служба баз даних, яка надає повністю керовану MongoDB інфраструктуру. Atlas забезпечує автоматичне масштабування, резервне копіювання, моніторинг та безпеку, що спрощує управління базами даних та забезпечує високу надійність і продуктивність.

Вибір TailwindCSS, React Router DOM, Express, MongoDB Atlas, Redux та Redux Toolkit для розробки веб-додатку «TaskMaster» обґрунтований їхньою зручністю, гнучкістю та здатністю покращувати продуктивність додатку. Ці технології забезпечують швидку розробку, зручну маршрутизацію, високу продуктивність та надійність, що робить їх ідеальним вибором для створення сучасних та ефективних веб-додатків. Використання Redux та Redux Toolkit дозволяє централізовано управляти станом додатку, зменшуючи складність коду та підвищуючи ефективність розробки. Ці інструменти забезпечують передбачуваність поведінки додатку та спрощують налагодження, що значно покращує загальну якість та зручність у підтримці коду.

Процес розробки веб-додатку. Після детального аналізу технологій та вибору інструментів для розробки веб-додатку, почалося створення самого додатку. Процес розробки включає декілька ключових етапів, кожен з яких є важливим для досягнення кінцевого результату. Основними етапами розробки це створення та налаштування проекту, розробка клієнтської частини та розробка серверної частини [9].

Процес створення та налаштування проекту розпочався з підготовки середовища розробки. Як основний інструмент для написання коду вибрано Visual Studio Code (VSCode) завдяки його зручності та підтримці великої кількості розширень, що підвищують продуктивність. Для створення проекту React використано команду npx create-react-app client, яка автоматично налаштовує середовище розробки з мінімальними налаштуваннями. Структуру проекту поділено на клієнтську і серверну частину. Структуру проекту представлена на рисунку 1.

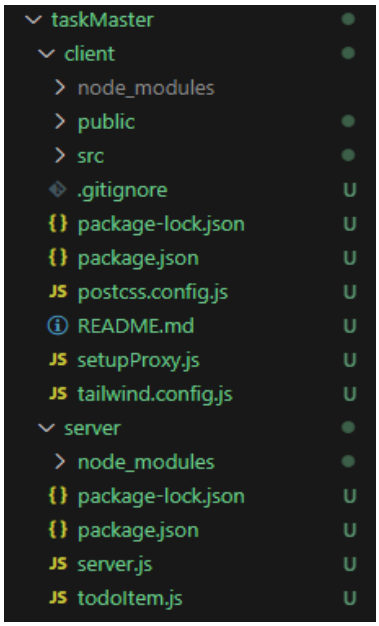

Рис. 1. Структура веб-додатку TaskMaster

Після створення основної структури проекту підключено всі необхідні технології за допомогою npm. Встановлено та налаштовано основні бібліотеки та фреймворки, такі як React, Node.js, MongoDB, TailwindCSS, Express, та Redux.

Наступним етапом стала розробка клієнтської частини додатку. Архітектура клієнтської частини базується на компонентному підході, що дозволяє розділити інтерфейс на незалежні, повторно використовувані компоненти. Це значно спрощує розробку, тестування та підтримку коду [10-12]. Основними компонентами клієнтської частини являються Header, TodoInput, TodoList, TodoPage. Розробка компонентів інтерфейсу та налаштування навігації між різними сторінками додатку здійснено за допомогою React Router DOM [13].

Далі було створено сервер. Головна задача сервера це приймати запити із клієнтської частини, обробляти їх і при необхідності брати та змінювати дані в базі даних [14]. Створення та налаштування серверу, обробки API запитів [15] здійснювалися за допомогою Node.js та

Express [16-18]. Для зберігання та обробки даних про завдання сервер підключено до бази даних MongoDB Atlas. Підключення здійснюється через URL-з'єднання, що містить інформацію про користувача, пароль та назву бази даних. Налаштовано механізми безпеки, такі як автентифікація користувачів та захист даних.

В результаті було створено веб-додаток «TaskMaster». Це високопродуктивний додаток завдяки використанню Node.js та MongoDB [19], що забезпечують його швидку та надійну роботу. Додаток має інтуїтивно зрозумілий та зручний інтерфейс із адаптивним дизайном для коректної роботи на комп'ютерах, планшетах і смартфонах. Він підтримує колективну роботу для командних проектів. Завдяки додатку користувачі можуть створювати, редагувати, видаляти та пріоритезувати завдання. Redux забезпечує централізоване управління станом додатку для передбачуваності та зручності. Централізоване управління станом додатку спрощує його налагодження та тестування. Для надійного зберігання та масштабованості даних використовується MongoDB Atlas. Хмарні сервіси забезпечують високий рівень безпеки даних.

Основні функції та можливості додатку. При вході в додаток, користувач опиняється на головній сторінці (Рисунок 1). Для використання додатку потрібно перейти у вікно TodoList, для цього потрібно нажати пкм по кнопці TodoList, після чого наш адрес із http://localhost:3001/ зміниться на http://localhost:3001/todolist. У вікні TodoList ми маємо всі функції, які нам потрібні для створення і перегляду всіх задач.

Рис. 2. Вигляд головної сторінки

Для *створення нової задачі* користувачу необхідно ввести наступне:

Заголовок: поле для вводу заголовку робить всі введені літери жирними і великими (Рисунок 3).

Завдання: поле для вводу завдання займає найбільшу кількість місця серед всіх елементів, тому у користувача є достатньо місця для вводу тексту (Рисунок 4).

Дата: дату можна ввести вручну або використати календар (Рисунок 5).

Час: час дату можна ввести вручну або використати меню для вибору часу (Рисунок 6).

Пріоритет: потрібно натиснути пкм і обрати пріоритет для задачі (Рисунок 7).

Після вводу всіх необхідних даних для задачі потрібно нажити кнопку Add task для внесення задачі у список справ.

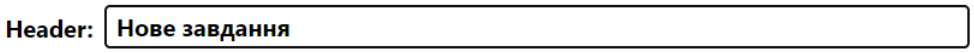

Рис. 3. Поле для вводу заголовку завдання

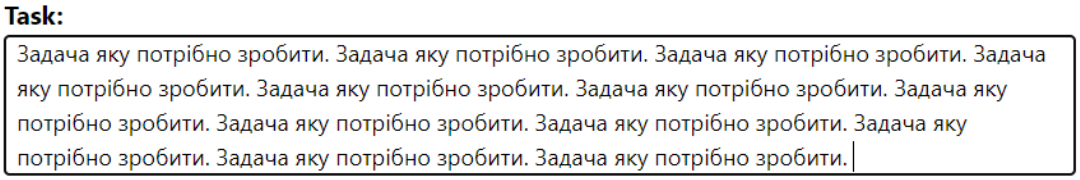

Рис. 4. Поле для вводу задачі

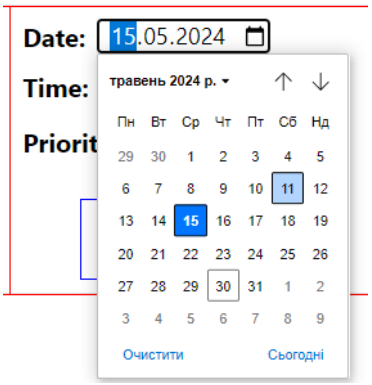

Рис. 5. Календар для обрання дати

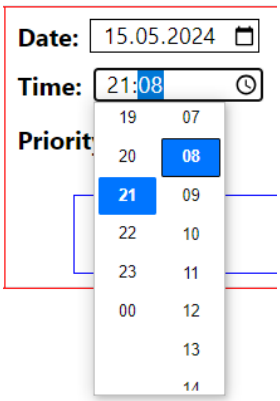

Рис. 6. Інтерфейс для вибору часу

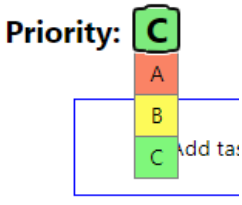

Рис. 7. Вигляд меню для виставлення пріоритету задачі

Після створення задачі, вона додається до списку TodoList (Рисунок 8). Основна функція списку задач це *перегляд всіх завдань* які додав користувач. Також список задач можна сортувати за датою створення, алфавітним порядком і пріоритетом (Рисунок 9). А самі задачі мають наступні функції (Рисунок 10):

Статус завдання: при виконанні завдання можна видалити його або встановити галочку, що завдання виконане.

Видалення: завдання можна видалити із списку задач натиснувши кнопку Delete.

Рис. 8. Вигляд списку задач

Редагування: якщо необхідно, задачу можна редагувати натиснувши кнопку Edit, при цьому завдання видаляється із списку задач, а всі данні які були введені у завданні переходять у місце для вводу нових задач. При даній функції користувач може без проблем виправити або додати нові дані при цьому не переписуючи все з нуля.

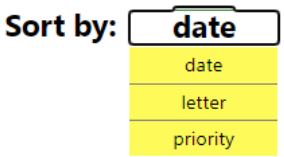

Рис. 9. Вибір методу сортування

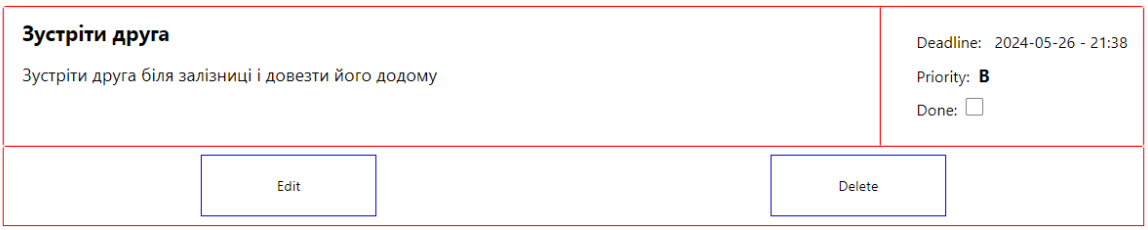

Рис. 10. Вигляд задачі

Висновки

В статті було проаналізовано ринок веб-додатків для управління завданнями, описані існуючі аналоги їх переваги і недоліки. Були розглянуті сучасні технології для створення веб додатку.

Налаштовано середовище розробки та створення базової структури проекту. Реалізовано клієнтську та серверну частини додатку. Клієнтська частина включає в себе компоненти для введення, відображення та управління завданнями, тоді як серверна частина забезпечує збереження та обробку даних у базі даних.

З використанням сучасних технологій React, Node.js, MongoDB та інших сучасних інструментів створено додаток для управління завданнями, що забезпечує високу надійність та ефективність додатку. Компонентний підхід дозволяє легко додавати нові функції та змінювати існуючі без значних зусиль. Завдяки використанню віртуального DOM у React та оптимізованого рендерингу, додаток швидко реагує на дії користувачів. Використання Redux дозволяє централізовано управляти станом додатку, що забезпечує його стабільну роботу навіть при великій кількості завдань та спрощує його підтримку. Завдяки MongoDB Atlas дані зберігаються в хмарі, що забезпечує їх доступність та безпеку.

Розробка додатку «TaskMaster» значно покращить розподіл власного часу і допоможе організувати задачі.

Список використаної літератури:

1. Schwarzmüller M. React Key Concepts: An in-depth guide to React's core features. Packt Publishing, 2025. 544 p.

2. Mammino L. Node.js Design Patterns. JS, 2020. 660 p.

3. Aleksendric M., Borucki A., Domingues L. Mastering MongoDB 7.0: Achieve data excellence by unlocking the full potential of MongoDB. Packt Publishing, 2024. 434 p.

4. Rappin N. Modern CSS with Tailwind. Flexible Styling Without the Fuss. Pragmatic Bookshelf, 2022. 104 p.

5. Bhat K. Ultimate Tailwind CSS Handbook: Build sleek and modern websites with immersive. Orange Education Pvt Ltd, 2023. 294 p.

6. React Router / React Router Documentation. URL: https://reactrouter.com/en/main.

7. Brown E. Web Development with Node and Express: Leveraging the JavaScript Stack. O'Reilly, 2019. 340 p.

8. Garreau M., Faurot W. Redux in Action. Manning, 2018. 312 p.

9. Redux Toolkit / Redux Toolkit Documentation. URL: https://redux-toolkit.js.org/.

10. Building Applications with React and Redux / Pluralsight. URL: https://www.pluralsight.com/courses/react-redux-react-router-es6.

11. Learning React: Functional Web Development with React and 18. Redux / O'Reilly Media. URL: https://www.oreilly.com/library/view/learning-react-2nd/9781492051718/.

12. Full-Stack Web Development with React / Coursera. URL: https://www.coursera.org/learn/full-stack-react.

13. React Hooks / React Documentation. URL: https://legacy.reactjs.org/docs/hooks-intro.html

14. HTTP Methods / Avior API Documentation. URL: https://www.contrive.mobi/aviorapi/HTTPMETHODS.html.

15 Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node. / Apress. URL: https://www.apress.com/gp/book/9781484243906.

16. Mastering Node.js / Packt Publishing. URL: https://www.packtpub.com/product/mastering-node-js/9781785888960.

17. Advanced Node.js Development / Udemy. URL: https://www.udemy.com/course/advanced-nodejs-development/.

18. Express in Action: Writing, Building, and Testing Node.js Applications / Manning Publications. URL: https://www.manning.com/books/express-in-action.

19. MongoDB: The Definitive Guide / O'Reilly Media. URL: https://www.oreilly.com/library/view/mongodb-the-definitive/9781491954454/.

Автори статті

Вишнівський Віктор – доктор технічних наук, професор, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

ORCID: 0000-0003-1923-4344

Шикула Олена – доктор фізико-математичних наук, професор, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

ORCID: 0000-0002-7385-2816

Довженко Тимур – кандидат технічних наук, доцент, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

ORCID: 0000-0002-0352-8391

Двуреченський Євгеній – студент, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

ORCID: 0009-0005-2588-4265

Серих Сергій – кандидат технічних наук, доцент, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

ORCID: 0009-0008-6033-3225

Authors of the article

Vyshnivskyi Viktor – Doctor of Science (technic), Professor, State University of Information and Communication Technologies, Kyiv, Ukraine.

Shykula Olena – Doctor of Sciences (physics and mathematics), Professor, State University of Information and Communication Technologies, Kyiv, Ukraine.

ORCID: 0000-0002-7385-2816

Dovzhenko Tymur – Candidate of Sciences (technical), Associate Professor, State University of Information and Communication Technologies, Kyiv, Ukraine.

ORCID: 0009-0008-6033-3225

Dvurechenskyi Yevhenii – student, State University of Information and Communication Technologies, Kyiv, Ukraine.

ORCID: 0009-0005-2588-4265

Sierykh Sergiy – Candidate of Sciences (technical), Associate Professor, State University of Information and Communication Technologies, Kyiv, Ukraine.

ORCID: 0009-0008-6033-3225