УДК 004.414;658.5.011;
UDC 004.414;658.5.011;
DOI: 10.33744/2308-6645-2025-1-60-012-019

# ПРОЕКТУВАННЯ ОПТИМІЗОВАНОГО БЛОГ-ДОДАТКУ

*Безверхий О.І.,* доктор фізико-математичних наук, Національний транспортний університет, Київ, Україна, o_bezver@ukr.net, orcid.org/0000-0002-0834-6335
*Дедул О.С.,* Національний транспортний університет, Київ, Україна, sasha15200201@gmail.com, orcid.org/0009-0009-1722-2869

# DESIGNING OPTIMIZED BLOG APPLICATION

*Bezverkhiy O.I.,* Doctor of physical and mathematical Science, National transport university, Kyiv, Ukraine, o_bezver@ukr.net, orcid.org/0000-0002-0834-6335
*Dedul O.S.,* National Transport University, Kyiv, Ukraine, sasha15200201@gmail.com, orcid.org/0009-0009-1722-2869

Блог-додаток є веб-платформою, яка надає користувачам можливість читати, створювати та публікувати статті на різноманітні теми. Додаток будем проектувати за принципом односторінкового додатка (SPA), що означає, що користувачам надається безперервний досвід взаємодії без перезавантаження сторінок. Це забезпечує високу швидкість роботи додатка та зменшує затримки під час переходу між сторінками.

Основна функціональність додатка включає публікацію статей, управління користувацьким контентом, а також соціальну взаємодію через коментарі та вподобання. Користувачі можуть реєструватися на платформі, створювати особисті профілі, писати статті, залишати коментарі та відповідати на них. Крім того, додаток надає можливість стежити за вподобаними авторами та отримувати сповіщення про нові публікації.

Інтерфейс додатка розроблений таким чином, щоб забезпечити інтуїтивно зрозумілу навігацію. У додатку є головна стрічка новин, де відображаються останні та найпопулярніші статті, а також категорії, за якими можна здійснювати фільтрацію контенту. Для зручності користувачів передбачено функцію пошуку статей за ключовими словами та автором. Для зменшення навантаження на сервер і покращення продуктивності додаток використовує технології кешування та оптимізації запитів, що дозволяє забезпечити плавний досвід роботи навіть при значній кількості контенту та великій кількості користувачів.

**Основні функціональні вимоги**

1. Управління статтями: можливість створення, редагування, видалення і перегляду статей. Користувачі повинні мати можливість легко створювати контент та керувати ним через зручний інтерфейс.
2. Коментарі: можливість користувачам залишати коментарі до статей, а також взаємодіяти з іншими користувачами через функціонал відповідей на коментарі.
3. Аутентифікація та авторизація: підтримка реєстрації нових користувачів, входу в систему, виходу, а також захист доступу до певних функцій додатка (наприклад, можливість створення або редагування статей тільки для авторизованих користувачів).
4. Пошук та фільтрація статей: можливість пошуку статей за ключовими словами, а також фільтрації статей за різними критеріями, такими як дата публікації, популярність або автор.
5. Підтримка багатомовності: додаток має підтримувати кілька мов (наприклад, українську та англійську), що дозволить користувачам вибирати мову інтерфейсу.
6. Оптимізація продуктивності: використання сучасних підходів до оптимізації SPA, таких як код-спліттинг, відкладене завантаження компонентів, оптимізація перерисовок та кешування даних.

**Вимоги до продуктивності та оптимізації**

1 Код-спліттинг: додаток має підтримувати розділення коду на окремі частини (чанки), які завантажуються лише тоді, коли вони дійсно забезпечити швидкий запуск додатка.

2 Відкладене завантаження (Lazy Loading): компоненти, які не є критично важливими для початкового завантаження сторінки, мають бути завантажені лише тоді, коли користувач взаємодіє з відповідною частиною додатка. Це дозволить зменшити навантаження на мережу і прискорити перше завантаження сторінки.

3 Оптимізація перерисовок: використання React.memo(), useMemo(), та useCallback() для уникнення зайвих перерисовок компонентів, що покращить загальну продуктивність додатка. Також необхідно розбивати стан на дрібніші частини для зменшення кількості перерисовок.

4 Віртуалізація списків: для рендерингу списків статей або коментарів має використовуватись віртуалізація з використанням таких бібліотек, як React Virtualized або React Window. Це дозволить зменшити навантаження на браузер при роботі з великими обсягами даних.

5 Кешування запитів: для зменшення навантаження на сервер і підвищення продуктивності має використовуватися RTK Query для кешування запитів до API. Це дозволить зберігати результати запитів і уникати повторного запиту до сервера для одних і тих самих даних, що знижує затримки і покращує користувацький досвід.

6 Service Workers: використання Service Workers для кешування статичних ресурсів додатка та забезпечення офлайн-режиму. Це дозволить користувачам мати доступ до статей навіть за відсутності інтернет-з'єднання, покращуючи загальну зручність використання додатка.

7 Управління станом: використання Redux Toolkit або Zustand для управління станом додатка. Це дозволить організувати структуру стану, розділити його на окремі частини, забезпечити асинхронні операції та зменшити кількість повторних рендерів.

**Нефункціональні вимоги**

1. Кросбраузерність: додаток має коректно працювати у всіх сучасних браузерах (Chrome, Firefox, Safari, Edge).

2. Реагування на зміну розміру екрану: підтримка адаптивного дизайну для коректної роботи як на настільних комп'ютерах, так і на мобільних пристроях.

3. Безпека: забезпечення безпеки даних користувачів, включаючи шифрування паролів та захист від атак типу CSRF та XSS.

Для розробки блог-додатка з урахуванням усіх вимог необхідно використовувати сучасні інструменти і підходи для оптимізації продуктивності. Код-спліттинг, відкладене завантаження, оптимізація перерисовок, віртуалізація списків і кешування запитів дозволять створити високопродуктивний, стабільний і зручний для користувачів додаток, який відповідає сучасним вимогам до веб-розробки. Цей підхід забезпечить якісну взаємодію з користувачами та високий рівень задоволеності від роботи з додатком.

**Опис архітектури додатку**

Feature Slice Design — це архітектурний підхід до розробки додатків, який забезпечує модульність, ізольованість та ефективність шляхом розподілу додатка на окремі функціональні слайси (feature slices). Кожен слайс представляє окрему бізнес-логічну частину додатка і включає в себе всі необхідні елементи для реалізації цієї частини, такі як компоненти інтерфейсу, логіка управління станом, стилі, запити до API та інші ресурси. Цей підхід дозволяє краще організувати проект, забезпечити незалежність модулів, що, в свою чергу, значно полегшує підтримку та масштабування додатка.

Feature Slice Design відрізняється від традиційної організації проектів, де код розподіляється за типами файлів (наприклад, окремі папки для компонентів, стилів, тестів і т.д.). Замість цього, у Feature Slice Design додаток розбивається на логічні функціональні блоки, де кожен слайс об'єднує всі типи файлів, необхідних для його роботи. Це дозволяє забезпечити високу когезію і низьке зв'язування між частинами додатка, що покращує модульність і знижує складність коду.

**Основні Принципи Feature Slice Design**

1. Модульність та ізольованість: Кожен слайс повністю ізольований від інших, що дозволяє уникати взаємозалежностей та робить додаток більш стабільним і передбачуваним.

2. Зручність підтримки: Зміни в одному слайсі не впливають на інші частини додатка, тому внесення нових функцій або виправлення помилок стає швидшим і простішим.

3. Командна робота: Завдяки ізоляції функціональних частин, розробники можуть одночасно працювати над різними слайсами без ризику виникнення конфліктів у коді.

У контексті блог-додатка Feature Slice Design є ідеальним рішенням для організації коду і забезпечення масштабованості. Блог-додаток має кілька ключових функціональних областей, таких як управління статтями, коментарями, користувачами та навігацією. Кожна з цих областей може бути реалізована як окремий слайс, що дозволяє зосередити всі пов'язані з конкретною функціональністю ресурси в одному місці.

**Оптимізація через Feature Slice Design**

Feature Slice Design не тільки сприяє покращенню структури додатка, але й значно спрощує процес оптимізації. Ось як цей підхід дозволяє підвищити продуктивність блог-додатка:

1 Код-спліттинг (Code Splitting): Кожен слайс може бути реалізований як окремий чанк, що дозволяє завантажувати тільки необхідні початковий обсяг коду, який завантажується, і покращує швидкість роботи додатка. Наприклад, компоненти, що відповідають за редагування статті, завантажуються тільки тоді, коли користувач переходить до редагування.

2 Відкладене завантаження (Lazy Loading): У кожному слайсі можна реалізувати відкладене завантаження компонентів і даних, які не є критичними для початкового відображення сторінки. Це дозволяє зменшити навантаження на браузер та покращити користувацький досвід. Наприклад, список коментарів можна завантажувати лише тоді, коли користувач прокручує сторінку до кінця статті.

3 Локалізоване управління станом: Кожен слайс має свої власні редюсери та дії, що забезпечує локалізацію стану і запобігає зайвим перерисовкам компонентів. Це особливо важливо для великих додатків, де зменшення кількості перерисовок значно впливає на загальну продуктивність. Використання таких бібліотек, як Redux Toolkit або Zustand, дозволяє створювати ізольовані сховища для кожного слайсу, забезпечуючи швидкий доступ до даних і легкість управління ними.

4 Віртуалізація списків (List Virtualization): Для компонентів, що відображають великі списки даних, таких як статті або коментарі, можна використовувати бібліотеки на кшталт React Virtualized або React Window. Віртуалізація допомагає уникнути зайвого рендерингу невидимих елементів і покращити продуктивність додатка.

5 Кешування запитів (Caching): Завдяки локалізованій структурі кожного слайсу, можна ефективно використовувати RTK Query для кешування запитів до API. Це дозволяє уникнути повторних запитів до серверу і використовувати закешовані дані, що покращує швидкість завантаження сторінок і знижує навантаження на сервер.

**Переваги Feature Slice Design для Блог-Додатка**

1. Масштабованість: Кожен функціональний модуль додатка розробляється незалежно, що дозволяє легко додавати нові можливості або змінювати існуючі без впливу на інші частини додатка.

2. Полегшена підтримка: Оскільки кожен слайс ізольований, підтримувати та оновлювати код стає набагато легше. Це також зменшує ймовірність виникнення помилок під час внесення змін.

3. Ефективність розробки: Команди можуть одночасно працювати над різними частинами додатка, не перетинаючись між собою і не створюючи конфліктів у коді. Це значно пришвидшує процес розробки і забезпечує високу продуктивність роботи.

**Опис структурних компонентів FSD**

Feature Slice Design передбачає розподіл функціональних частин додатка на окремі структурні елементи, такі як Widgets, Pages, Shared, Features, App, та Entities. Кожен із цих елементів відіграє свою роль у створенні модульного та масштабованого додатка, що дозволяє ефективно організувати код і забезпечити його підтримку. Розглянемо кожен із цих елементів детальніше.

Widgets — це невеликі, переважно універсальні компоненти, які використовуються в багатьох місцях додатка для повторюваних елементів інтерфейсу. Вони можуть включати(рис.1) кнопки, інпут-форми, невеликі картки або інформаційні блоки. Відмінність виджетів від інших компонентів полягає в тому, що вони не мають специфічної бізнес-логіки, а виконують лише одну універсальну функцію, наприклад, відображення інтерфейсного елементу.

Widgets використовуються в різних частинах додатка і можуть бути інтегровані у Features або Pages. Їхня модульність і багаторазове використання робить їх важливим елементом для створення уніфікованого дизайну та функціональності додатка.

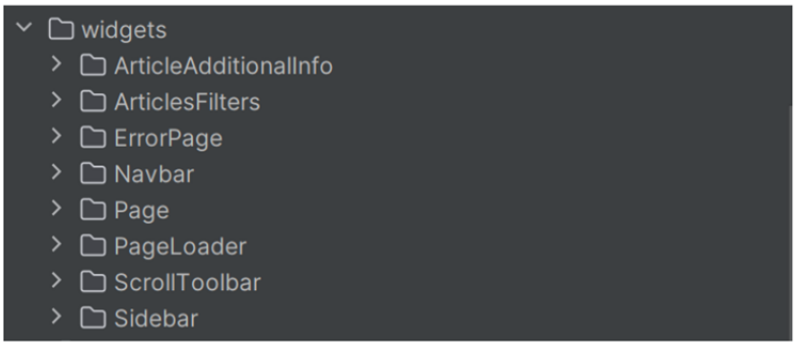


Рисунок 1 – Структура Widgets
Figure 1 – Structure of Widgets

Pages — це кінцеві компоненти, які зв'язують усі інші елементи додатка разом і забезпечують взаємодію користувача з різними функціями. Кожна сторінка реалізує певний сценарій використання, такий як перегляд статті, редагування профілю або створення нового допису. Вони часто об'єднують кілька різних слайсів або функціональних частин у рамках одного інтерфейсу. Наприклад(рис.2), сторінка статті може включати основний контент статті, список коментарів, віджети для навігації та додаткові елементи, такі як авторський блок.

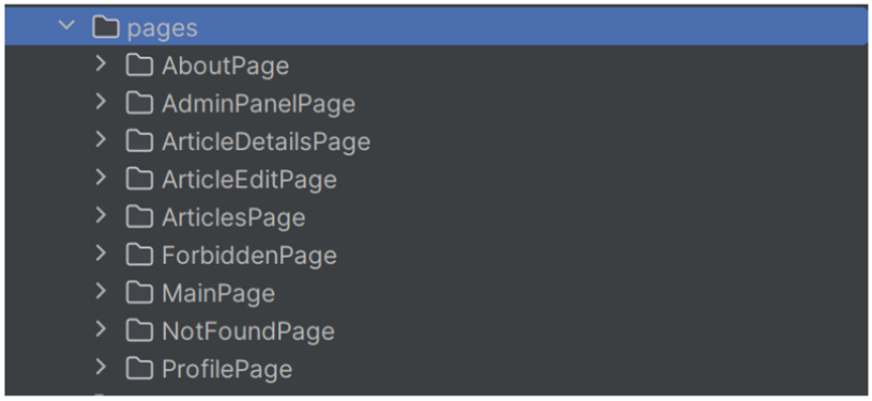


Рисунок 2 – Структура Pages
Figure 2 – Structure of Pages

Shared — це модуль, який містить загальні ресурси, що можуть використовуватися в різних частинах додатка. Це можуть бути(рис. 3) утиліти, загальні компоненти, константи, типи даних, стилі, а також API-запити, які можуть бути потрібні в багатьох місцях додатка. Модуль Shared дозволяє зменшити дублювання коду і забезпечити централізоване зберігання загальних ресурсів.
Наприклад, у Shared можуть знаходитися компоненти, такі як кнопки, модальні вікна або навіть інструменти для роботи з датами, які використовуються в декількох слайсах. Це значно полегшує підтримку і дозволяє швидко змінювати загальні елементи у всьому додатку.

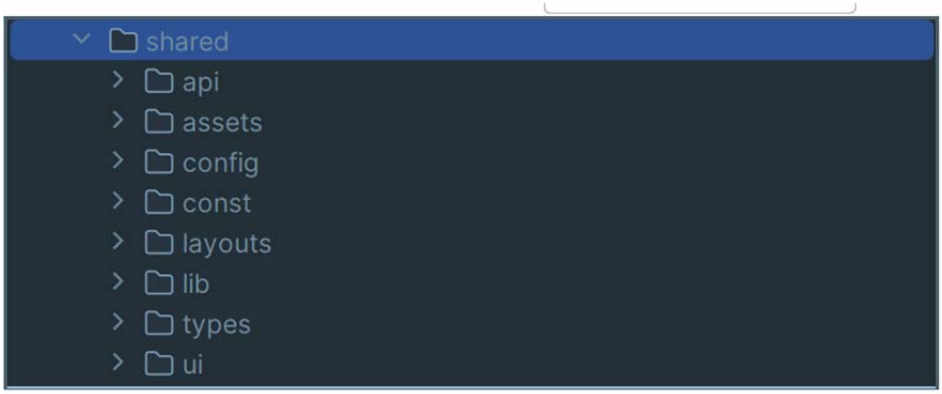


Рисунок 3 – Структура Shared
Figure 3 – Shared structure

Features — це ключові функціональні модулі додатка, що реалізують певні бізнес-логічні можливості. Кожна Feature включає(рис. 4) всі необхідні елементи для реалізації конкретної функціональності: компоненти, управління станом, стилі та API-запити. Наприклад, Feature для керування статтями (Articles Feature) може включати компоненти для створення, редагування, перегляду статей, а також логіку управління станом і запити до API для отримання або збереження даних. Features можуть бути інтегровані на різних сторінках, і їхня ізольованість забезпечує простоту додавання нової функціональності без впливу на інші частини додатка. Це робить додаток більш стабільним і передбачуваним у підтримці.

App — це модуль, який включає загальну конфігурацію та інфраструктуру всього додатка. Це місце, де знаходиться головна точка входу додатка, маршрутизація (наприклад, за допомогою React Router), а також глобальні налаштування, такі як тема, налаштування контексту або загальні перехоплювачі помилок.

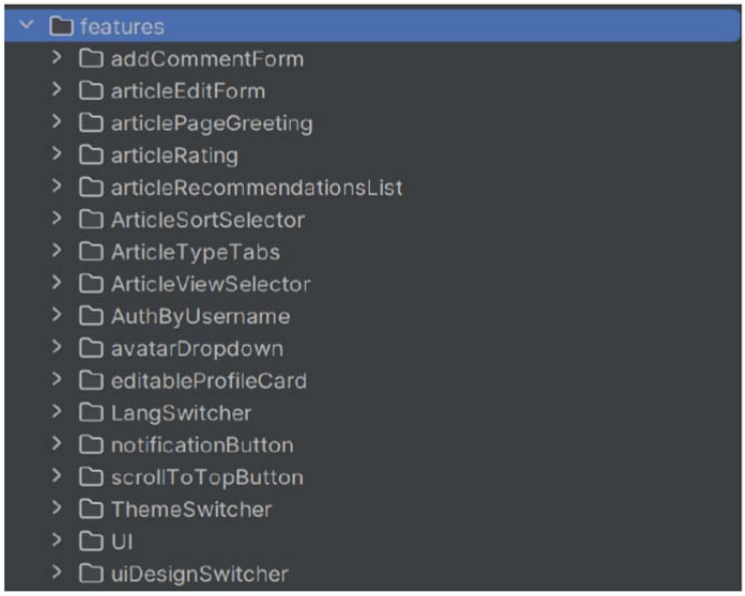


Рисунок 4 – Структура Features
Figure 4 – Structure of Features

App модуль також відповідає за ініціалізацію ключових частин додатка(рис. 5), включаючи інтеграцію з API, налаштування управління станом і глобальні події. Це основа, на якій будується весь додаток, і вона забезпечує зв'язок між різними компонентами та модулями.


Рисунок 5 – Структура App
Figure 5 – App structure

Entities — це модуль, який включає моделі даних та всю логіку, пов'язану зі зберіганням і обробкою даних, які представляють бізнес-сутності додатка. Наприклад (рис. 6), для блог-додатка це можуть бути сутності, такі як Article, User, Comment.

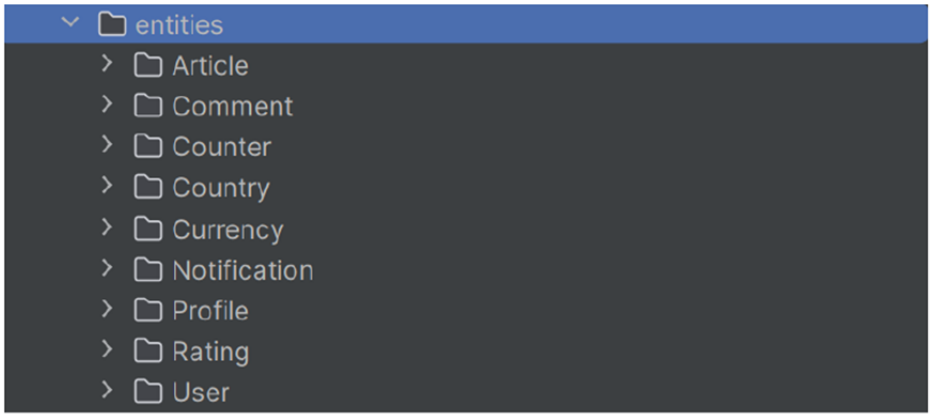


Рисунок 6 – Структура Entities
Figure 6 – Structure of Entities

Кожна з цих сутностей містить логіку, яка дозволяє працювати з даними, включаючи CRUD-операції (створення, читання, оновлення, видалення).

Entities забезпечують централізовану обробку даних, що дозволяє зберігати код, який пов'язаний зі взаємодією із сервером або маніпуляціями з даними, в одному місці. Це забезпечує кращу підтримку і дозволяє легко оновлювати моделі даних у разі зміни бізнес-вимог.

Feature Slice Design є потужним підходом до організації архітектури додатків, особливо для великих і складних проєктів, таких як блог-додаток. Цей підхід не тільки забезпечує модульність та ізольованість функціональних частин(рис. 7), але й дозволяє ефективно оптимізувати додаток, що забезпечує швидкий, стабільний і зручний користувацький досвід.

Для зберігання конфігурацій, даних користувачів та контенту блогу використовується json-server, який виступає як легка база даних у форматі JSON. Це рішення дозволяє швидко налаштувати API для тестування додатка та зберігати всі дані у простому форматі. База даних складається з наступних колекцій: notifications, articles, comments, users, profile, article-ratings, та profile-ratings. Кожна з цих колекцій представляє різні типи даних, які необхідні для роботи додатка.


Рисунок 7 – Глобальна структура папок
Figure 7 – Global folder structure

**Висновок.** Під час розробки інформаційної системи ключовим аспектом є чітке визначення задачі та вимог до системи. Ми розглянули основні вимоги до системи, зокрема її функціональні можливості, обмеження та цілі, що допомагають сформувати чітке уявлення про кінцевий продукт. Визначення вимог дозволяє забезпечити узгодженість між різними учасниками проекту та сприяє реалізації цілісного підходу до розробки. Опис архітектури додатку дав змогу чітко структурувати проект, що забезпечує масштабованість і зручність у підтримці. Опис архітектури дозволяє бачити, як взаємодіють різні компоненти системи, а також сприяє ефективній декомпозиції складних функціональностей. Архітектура визначає, як інтегруються модулі, як зберігаються та обробляються дані, а також як забезпечується ефективність і стабільність роботи додатка.

База даних є фундаментальним елементом для зберігання інформації та забезпечення її цілісності. Дизайн бази даних визначає, як дані структуровані та зберігаються, що дозволяє забезпечити швидкий доступ і легкість у їхній обробці. Використання JSON як бази даних у нашому проекті сприяло швидкій реалізації MVP та гнучкому управлінню даними, що дозволяє швидко змінювати структуру на етапі розробки.

Таким чином, кожен із розглянутих аспектів відіграє важливу роль у створенні масштабованої та якісної інформаційної системи. Від правильно поставленої задачі та побудованої архітектури до впровадження інструментів автоматизованої перевірки якості коду, тестування компонентів і використання бази даних — усе це забезпечує створення стабільного, функціонального та зручного продукту. Інтеграція сучасних технологій і підходів дозволяє зменшити витрати на підтримку, збільшити швидкість розробки та забезпечити відповідність додатка вимогам користувачів і бізнесу

## ПЕРЕЛІК ПОСИЛАНЬ

1. Дуглас Крокфорд. JavaScript: Сильні сторони. – Харків: Видавництво "Фактор", 2018. – 200 с.

2. Deshmukh S. Building a Single Page Application Web Front-end for E-Learning site [Електронний ресурс] / S. Deshmukh, A. Retawade, D. Mane – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/8819703/authors#authors .

3. React Documentation. [Онлайн ресурс] – Режим доступу: https://reactjs.org/docs/getting-started.html

4. Jest Documentation. [Онлайн ресурс] – Режим доступу: https://jestjs.io/docs/getting-started
5. Analyzing best practices on Web development frameworks: The lift approach [Електронний ресурс] / [M. PilarSalas-Zárate, G. Alor-Hernández, R. Valencia-García та ін.] – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S0167642314005735.
6. Babel Documentation. [Онлайн ресурс] – Режим доступу: https://babeljs.io/docs/en/

## REFERENCES

1. Douglas Crockford. JavaScript: Strengths. – Kharkiv: "Faktor" Publishing House, 2018. – 200 p.
2. Deshmukh S. Building a Single Page Application Web Front-end for E-Learning site [Electronic resource] / S. Deshmukh, A. Retawade, D. Mane – Resource access mode: https://ieeexplore.ieee.org /document/8819703/authors#authors .
3. React Documentation. [Online resource] – Access mode: https://reactjs.org/docs/getting-started.html
4. Jest Documentation. [Online resource] – Access mode: https://jestjs.io/docs/getting-started
5. Analyzing best practices on Web development frameworks: The lift approach [Electronic resource] / [M. PilarSalas-Zárate, G. Alor-Hernández, R. Valencia-García, et al.] – Resource access mode: https://www.sciencedirect.com/science/article/pii/S0167642314005735 .
6. Babel Documentation. [Online resource] – Access mode: https://babeljs.io/docs/en/

## РЕФЕРАТ

Безверхий О.І. Проектування оптимізованого блог-додатку / О.І. Безверхий, О.С. Дедул // Вісник Національного транспортного університету. Серія «Технічні науки». Науковий, науково-виробничий журнал. – К.: НТУ, 2025. – Вип. 1 (60).

У роботі проаналізовано сучасні технології оптимізації front-end розробки, включаючи налаштування інфраструктури проекту, модульну архітектуру та підходи до оптимізації продуктивності. Досліджено використання інструментів для конфігурації проекту, таких як Webpack, Vite, та Babel, з метою зменшення розміру бандла та підвищення швидкодії. Впроваджено оптимізацію за допомогою code splitting, lazy loading та кешування запитів.

Об’єкт дослідження: процес оптимізації front-end розробки у сучасних блог-додатках. Мета роботи: дослідити сучасні методи та технології оптимізації процесів front-end розробки, їх інтеграцію в реальний проект та оцінити ефективність впроваджених рішень. У роботі проаналізовано сучасні підходи до організації інфраструктури front-end проектів, включаючи використання модульних систем, а також гнучких підходів до конфігурації середовища розробки. Розроблено комплексну архітектуру для масштабованих веб-додатків, яка базується на принципах модульності, слабкої зв'язаності та повторного використання компонентів. Здійснено конфігурацію проекту з використанням таких інструментів, як ESLint для лінтингу коду, Jest для тестування, JSON Server для емуляції API, TypeScript для статичної типізації, а також React для створення користувацького інтерфейсу. Досліджено методи оптимізації бандла, включаючи розділення коду (code splitting), ліниве завантаження (lazy loading) та використання бандл-аналізаторів. Це дозволило визначити найбільш ефективні методи для створення оптимальної інфраструктури, яка підтримує гнучкість, масштабованість і зручність розробки.

КЛЮЧОВІ СЛОВА: FRONT-END, ОПТИМІЗАЦІЯ, WEBPACK, VITE, BABEL, TYPESCRIPT, UI-КОМПОНЕНТИ, ТЕМІЗАЦІЯ, I18N, LAZY LOADING, CODE SPLITTING, REDUX, ENTITYADAPTER, REACT, STORYBOOK, СКРИНШОТНІ ТЕСТИ, МОДУЛЬНА АРХІТЕКТУРА, АСИНХРОННІ РЕДЮСЕРИ.

## ABSTRACT

Bezverhiy O.I Dedul O.S.. Design of the application for management of software development processes. Visnyk National Transport University. Series «Technical sciences». Scientific, scientific and industrial journal. – K.: NTU, 2025. – Issue 1 (60).

The paper analyzes modern front-end development optimisation technologies, including project infrastructure configuration, modular architecture, and performance optimisation approaches. The use of

project configuration tools, such as Webpack, Vite, and Babel, to reduce bundle size and improve performance is investigated. Optimisation using code splitting, lazy loading, and request caching was implemented.

Object of study: the process of optimizing front-end development in modern blog applications. Purpose: to investigate modern methods and technologies for optimizing front-end development processes, their integration into a real project, and to evaluate the effectiveness of the implemented solutions. The paper examines contemporary approaches to organizing the infrastructure of front-end projects, including the utilization of modular systems and flexible configurations for the development environment. A comprehensive architecture for scalable web applications based on the principles of modularity, loose coupling, and component reuse is developed. The project was configured using tools such as ESLint for code linting, Jest for testing, JSON Server for API emulation, TypeScript for static typing, and React for creating a user interface. The methods of bundle optimization, including code splitting, lazy loading, and the use of bundle analyzers, were investigated. This allowed us to identify the most effective methods for creating an optimal infrastructure that supports flexibility, scalability, and ease of development.

KEYWORDS: FRONT-END, OPTIMISATION, WEBPACK, VITE, BABEL, TYPESCRIPT, UI COMPONENTS, THEMING, I18N, LAZY LOADING, CODE SPLITTING, REDUX, ENTITYADAPTER, REACT, STORYBOOK, SCREENSHOT TESTS, MODULAR ARCHITECTURE, ASYNCHRONOUS EDITORS.

**АВТОРИ:**

Безверхий Олександр Ігорович, д.ф.-м.н., професор, Національний транспортний університет професор кафедри інформаційних систем і технологій, Київ, Україна, e-mail: o_bezver@ukr.net, тел. +380666791995, Україна, 01010, м. Київ, вул. Омеляновича-Павленка, 1, к. 351, orcid.org/0000-0002-0834-6335.

Дедул Олександр Станіславович, Національний транспортний університет, студент групи КНм-2-1 факультету транспортних та інформаційних технологій, Київ, Україна, e-mail: sasha15200201@gmail.com, тел. +380680937561, Україна, 01010, м. Київ, вул. Омеляновича-Павленка, 1, orcid.org/0009-0009-1722-2869.

**AUTHORS:**

Bezverkhiy Oleksandr I., Doctor of Physical and Mathematical Sciences, Professor, National Transport University Professor of the Department Information Systems and Technologies, Kyiv, Ukraine, e-mail: o_bezver@ukr.net, tel. +380666791995, Ukraine, 01010, Kyiv, str. Omelyanovich-Pavlenka, 1, room 351, orcid.org/0000-0002-0834-6335.

Dedul Oleksandr S., National Transport University, student of the KNm-2-1 group of the Faculty of Transport and Information Technologies, sasha15200201@gmail.com, , tel. +380680937561, Ukraine, 01010, Kyiv, str. Omelyanovich-Pavlenka, 1, orcid.org/0009-0009-1722-2869.

**РЕЦЕНЗЕНТИ:**

Гавриленко Валерій Володимирович., д-ф.м-н., професор, Національний транспортний університет, Київ, Україна.

Івохін Євген Віктотрович.,д-ф.м-н., професор, Національний університет імені Тараса Шевченка, Київ, Україна.

**REVIEWERS:**

Gavrilenko Valeriy Volodymyrovych, Doctor of Physical and Mathematical sciences, professor, National Transport University, Kyiv, Ukraine.

Ivohin Evgen Viktorovych, Doctor of Physical and Mathematical sciences, professor, National university Taras Shevchenko, Kyiv, Ukraine.