МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Гончарука Івана Сергійовича
(прізвище, ім'я, по батькові здобувача)

на тему «Розробка програмного забезпечення пошуку та публікації новин»
(повна назва теми)

за матеріалами праць провідних спеціалістів з розробки ПЗ та проектування БД
(повна назва бази дослідження)

науковий керівник к.е.н., доцент _____ Баран С.В.
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 11.06.2025 р. № 12
Завідувач кафедри _____
(підпис)

д.т.н., професор Зеленський О.С.
Наук. ступень, вчене звання *Ініціали, прізвище*

Кривий Ріг – 2025

ЗГОДА здобувача вищої освіти

Державного університету економіки і технологій про перевірку кваліфікаційної роботи на прояви академічного плагіату та розміщення в Репозитарії Університету

Я, ___Гончарук Іван Сергійович_____, підтримую політику Державного університету економіки і технологій з академічної доброчесності і відкритого доступу.

Засвідчую, що кваліфікаційна бакалаврська робота _____«Розробка програмного забезпечення пошуку та публікації новин»____ виконана самостійно та не містить академічного плагіату. Я не надавав і не одержував недозволену допомогу під час підготовки цієї роботи. Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Державного університету економіки і технологій ознайомлений. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення норм академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

Також я поінформований, що відповідно до «Положення про Репозитарій (електронну базу даних) Державного університету економіки і технологій» зазначена робота буде розміщена в Електронному архіві Університету (Репозитарії ДУЕТ). З умовами такого розміщення ознайомлений.

Дата підпис ініціали, прізвище

(власноруч)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____ Зеленський О.С.
(підпис) (Прізвище, ініціали)
« 11 » червня 2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ

1. Тема роботи «Розробка програмного забезпечення пошуку та публікації новин»

Керівник роботи к.е.н., доц. Баран С.В.
затверджені наказом закладу вищої освіти від «04» квітня 2025 р. № 222-ст

2. Строк подання здобувачем роботи до «09» червня 2025 р.

3. Зміст кваліфікаційної роботи, об'єкт, предмет та мета дослідження:

Розділ 1. Постановка задачі

Розділ 2. Проектування системи

Розділ 3. Проектування бази даних додатку

Розділ 4. Розробка web-сайту

Об'єкт дослідження: сайт

Предмет дослідження: розробка сайту

Мета кваліфікаційної роботи: *розробити сайт пошуку та публікації новин*

5. Дата видачі завдання «04» квітня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів МДР	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	04.04.2025-13.04.2025	
2	Підготовка розділу 2	14.04.2025-26.04.2025	
3.	Підготовка розділу 3	27.04.2025-15.05.2025	
4	Підготовка розділу 4	16.05.2025-08.06.2025	
5	Реєстрація завершеної кваліфікаційної роботи	09.06.2025	Реєстраційний №___ «09»червня 2025 р.
6	Отримання відгуку від наукового керівника	10.06.2025	
7	Подання кваліфікаційної роботи на перегляд завідувачу кафедри	11.06.2025	
8	Отримання зовнішньої рецензії	12.06.2025	
9	Попередній захист кваліфікаційної роботи на кафедрі	13.06.2025	
10	Підготовка до захисту в ЕК	16.06.2025-21.06.2025	

Завдання підготував науковий керівник _____ __Баран С.В___
(підпис) (прізвище та ініціали)

Завдання одержав _____ ___Гончарук І.С.
_
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

на кваліфікаційну бакалаврську роботу

«Розробка програмного забезпечення пошуку та публікації новин»

Гончарука Івана Сергійовича

Кваліфікаційна бакалаврська робота на здобуття освітньо-кваліфікаційного рівня бакалавра зі спеціальності 121 «Інженерія програмного забезпечення» – Державний університет економіки і технологій – Кривий Ріг, 2025.

У бакалаврській дипломній роботі розроблене програмне забезпечення перегляду новин на основі алгоритмів веб розробки. Програмний додаток розроблено на мові Java Script з використанням бібліотеки Node.js, з використанням технологій passport, express тощо.

Ключові слова: НОВИНИ, JAVASCRIPT, САЙТ, ВЕБ-РЕСУРС.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД(база даних)	Впорядкований набір логічно взаємопов’язаних даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів.
СУБД	Система управління базами даних.
ПЗ	Програмне забезпечення.
ADO .NET	ActiveX Data Object для .NET – технологія доступу і управління базами даних для платформи .NET

ЗМІСТ

ВСТУП

Розвиток цифрових технологій сприяв появі нових способів подання та споживання інформації. Зростання попиту на онлайн-ресурси для ознайомлення з подіями, оглядами та статтями стимулює створення вебпорталів, орієнтованих на широке коло користувачів. У цьому контексті дедалі частіше застосовуються методи створення інтерактивних вебсторінок, які дозволяють реагувати на дії користувача, автоматично оновлювати вміст і забезпечувати гнучкий механізм керування матеріалами.

JavaScript є основним інструментом для розробки інтерфейсної частини подібних систем. Ця мова застосовується в більшості сучасних браузерів і підтримує створення компонентів, що відповідають за взаємодію між користувачем та даними. Її поєднання з HTML, CSS та засобами обробки запитів дає змогу формувати системи, які не потребують постійного перезавантаження сторінки для оновлення даних.

Мета дипломного проекту — створити вебпортал новин, у якому реалізовано базовий функціонал публікації, редагування та пошуку матеріалів. Передбачено також розмежування доступу для різних типів користувачів, перегляд новин за категоріями та інтеграцію з базою даних.

Крім вебпорталу, у межах проекту реалізовано окрему настільну програму, призначену для роботи зі схемами розміщення товарів на складі. Цей компонент дає змогу створювати умовну модель приміщення та розміщувати в ньому об'єкти з урахуванням заданих параметрів. У межах однієї роботи продемонстровано застосування двох підходів до побудови прикладного програмного забезпечення: веборієнтованого та десктопного.

Запропонована система охоплює різні сфери застосування програмування і демонструє можливість реалізації функціоналу для обробки інформації, візуального представлення даних та організації взаємодії з користувачем.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ

1.1. Характеристика задачі

Поширення електронних медіа спричинило суттєві зміни у способах отримання інформації. Онлайн-новини поступово витісняють друковані джерела, а зростання мобільного доступу до мережі формує нові вимоги до організації вмісту. У відповідь на це виникає потреба у створенні вебпорталів, здатних оперативно надавати текстові та мультимедійні матеріали з можливістю інтерактивної взаємодії з користувачем.

У межах дипломного проекту передбачено створення вебпорталу, орієнтованого на публікацію новин. Це електронний ресурс, який надає користувачам можливість переглядати нові матеріали, здійснювати пошук за змістом, переходити між тематичними розділами та ознайомлюватися з архівом попередніх публікацій. Окрім цього, адміністратор сайту має отримати засоби для створення, редагування, категоризації та видалення вмісту.

Основна задача полягає у побудові системи, здатної обробляти великий обсяг новинних матеріалів, організовувати їх за різними критеріями та забезпечувати просту навігацію. У реалізації беруть участь три ключові складові: клієнтська частина, серверна логіка та база даних. Клієнтська частина створюється з використанням HTML, CSS та JavaScript. Серверна логіка може реалізовуватися з використанням будь-якого середовища, що підтримує обробку HTTP-запитів і взаємодію з базами даних. База даних містить записи про новини, категорії, користувачів, коментарі, позначки часу та допоміжну інформацію.

Користувачі вебпорталу поділяються на дві групи:

– відвідувачі, які споживають інформацію;

– адміністратори, які її додають і редагують.

Відвідувачам доступні такі функції:

– перегляд списку новин;

– ознайомлення з повним текстом кожної новини;

– навігація за тематиками через категорії;

– пошук за ключовими словами.

Адміністраторам, які проходять авторизацію через спеціальну форму, надається можливість:

– створювати нові публікації з додаванням заголовка, основного тексту, зображень, тегів;

– редагувати вже існуючі записи;

– змінювати категорії та призначати нові;

– вилучати матеріали, які втратили актуальність або були додані помилково.

З технічного погляду, вебпортал має підтримувати обробку асинхронних запитів — тобто обмін даними між сторінкою користувача та сервером без повного перезавантаження. Це дозволяє динамічно оновлювати новини, коментарі, результати пошуку тощо. JavaScript виконує ключову роль у цій частині: за його допомогою обробляються події на сторінці, формуються запити до сервера, виводиться оновлений контент.

Для представлення новин на головній сторінці використовуються короткі анонси. Кожен анонс містить заголовок, дату публікації та скорочений фрагмент тексту. При натисканні на заголовок користувач переходить на сторінку повної новини, де подано повний текст, галерею зображень, інформацію про автора, а також розділ з коментарями, якщо він реалізований.

Система категорій дозволяє структурувати новини за тематиками. Кожна новина пов’язується з однією або кількома категоріями. Натискаючи на назву категорії, користувач потрапляє на сторінку, де зібрано всі новини з відповідної теми. Це значно спрощує навігацію, особливо для користувачів, які цікавляться вузьким колом тем.

Механізм пошуку працює за принципом повнотекстового аналізу полів бази даних. Користувач вводить слово або фразу, а система повертає перелік

відповідних новин. Для оптимізації запитів передбачено попередню обробку введеного тексту, видалення зайвих символів, перетворення на нижній регістр та розбиття на ключові слова.

Інтерфейс створюється з урахуванням можливості використання на пристроях з різними розмірами екранів. Для цього застосовується адаптивна верстка: CSS-медіазапити дозволяють змінювати розташування елементів, шрифти, відступи залежно від ширини вікна браузера. Головне меню розміщується у верхній частині сторінки, містить посилання на категорії, кнопку входу до панелі керування та поле для пошуку.

Адміністративна панель — окрема частина системи, доступна лише після введення імені адміністратора та пароля. У ній реалізовано інтерфейс для створення нової новини, з можливістю попереднього перегляду та збереження чернетки. Окремо передбачено таблицю, в якій виводяться всі новини, що вже збережені у базі. Для кожного запису доступні кнопки «редагувати», «видалити» та «переглянути».

Технічна реалізація передбачає перевірку введених даних перед відправкою на сервер. Наприклад, поле заголовка не може бути порожнім, а розмір зображення не повинен перевищувати задану межу. Подібні перевірки дублюються і на серверній стороні.

Крім вебпорталу, у межах дипломного проекту створено настільну програму для моделювання планування складу. Цей додаток дозволяє розміщувати товарні об’єкти на умовному плані приміщення й змінювати їх положення з урахуванням розмірів та форми. Оскільки ця частина має допоміжний характер, вона описується стисло в окремому розділі.

1.2. Огляд існуючих web-сайтів

Один із найвідоміших міжнародних новинних порталів — BBC News (https://www.bbc.com/news). Його головна сторінка (Рис. 1.1.) складається з великого візуального блоку головної новини, далі йде стрічка з короткими

анонсами та підбірками за темами. При прокрутці поступово відкривається більше блоків — регіональні новини, аналітика, матеріали редакції. Всі вони подані через великі прев'ю з фото, лаконічними заголовками та позначкою часу.

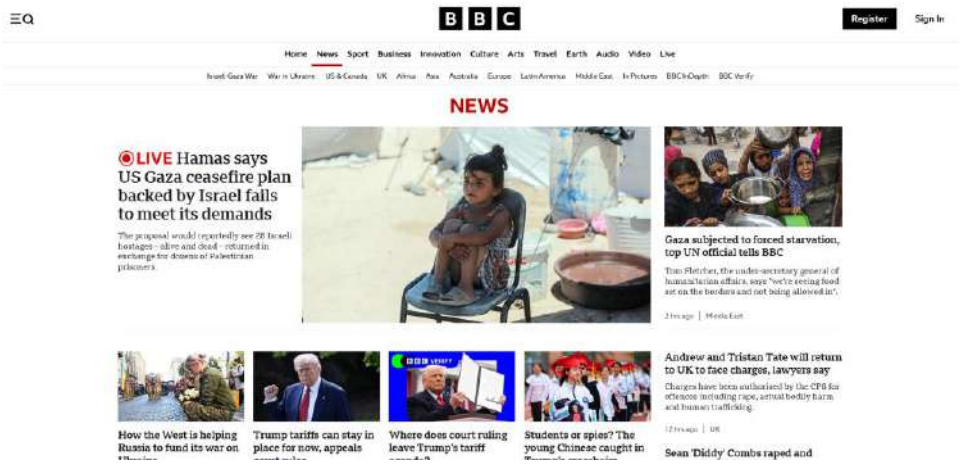

Рис. 1.1. Головна сторінка порталу

Навігація між розділами реалізована не через спадні меню, як часто буває, а через фіксовану верхню панель. Розділи не дублюються в підменю — кожен відкривається як окрема тематична сторінка з власною сіткою новин (Рис. 1.2.). При цьому URL-структура дуже чиста, що важливо для пошукової оптимізації.

Рис. 1.2. Сторінка культурних новин

Функціонально сайт дозволяє переходити між текстом і мультимедіа без перезавантаження сторінки. Наприклад, новини часто мають прикріплене відео або аудіо, яке автоматично підтягується на сторінці. Також реалізовано блок "живого оновлення" для динамічних подій, де кожне нове повідомлення додається поверх старих із зазначенням часу.

Офіційне державне агентство новин України — Укрінформ (https://www.ukrinform.ua). Головна сторінка (Рис. 1.3.) не перевантажена графікою: тут переважає текстовий контент, поділений на тематичні блоки — новини, аналітика, інтерв'ю. Візуальний акцент зроблено не на головній новині, а на стрічці новин — вона є фактично першим елементом, який бачить користувач.

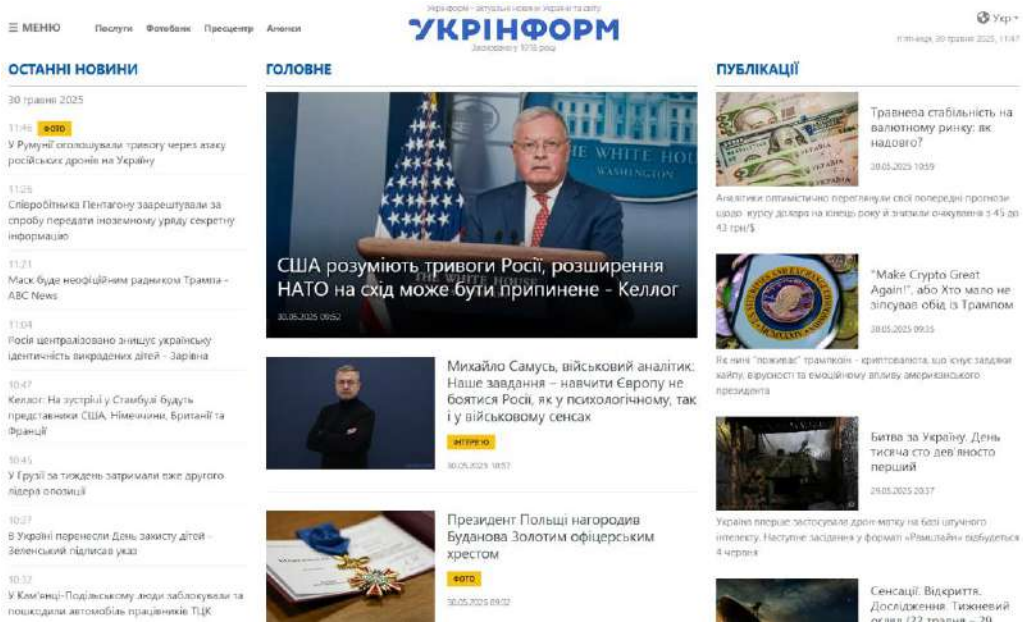

Рис. 1.3. Головна сторінка порталу

Меню (Рис. 1.4.) містить як основні розділи (політика, економіка, культура), так і менш очевидні, наприклад "Фото дня", "Історичний календар", "Погляди". При відкритті новини (Рис. 1.5.) користувач потрапляє на окрему сторінку з дуже мінімалістичним оформленням: текст, фото, джерело, дата — жодних зайвих блоків.

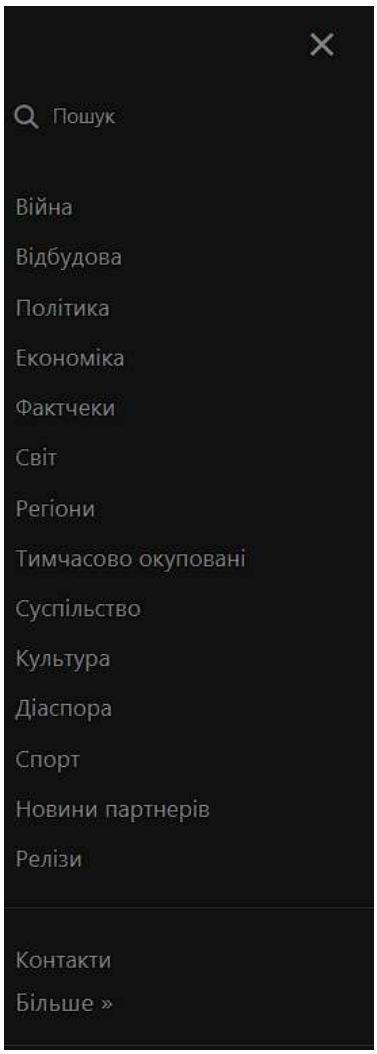

Рис. 1.4. Меню порталу

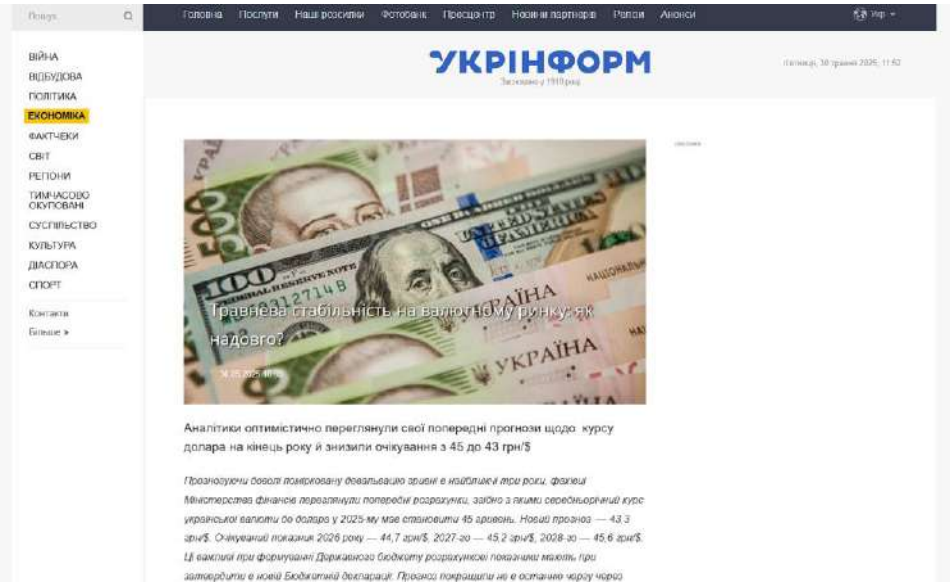

Рис. 1.5. Сторінка новини

Цікавою особливістю є модуль "Цитата дня", який генерується вручну редакторами та змінюється щодня. Також на сайті є інтеграція з Telegram, що дозволяє слідкувати за новинами без відкриття браузера (Рис. 1.6.).

Рис. 1.6. Футер сайту

Сайт добре працює в умовах обмеженого інтернету — мінімум важких елементів, усе оптимізовано для швидкого завантаження.

BBC News демонструє підхід, орієнтований на масову аудиторію — яскрава подача, візуальне навантаження, мультимедіа. Укрінформ більше нагадує "робочий інструмент" — тільки подача новин, акцент на джерелах, швидкий доступ до фактів. Обидва приклади показують різні стратегії організації контенту, що варто враховувати при проєктуванні власного сайту: все залежить від цілей — інформувати чи залучати.

1.3. Аналіз вимог до сайту

Сфера онлайн-інформування охоплює широкий спектр технічних і функціональних аспектів. Новинні вебпортали мають справу з великою кількістю контенту, який постійно оновлюється, групується за темами, подається в різних форматах. Для побудови ефективного інструменту комунікації між редакційним відділом та відвідувачами ресурсу необхідно врахувати як загальні особливості медійного контенту, так і специфіку його цифрової репрезентації.

Основу новинного порталу становить розподіл інформації за напрямами: політика, культура, спорт, економіка, міжнародні події, наука, технології,

місцеві новини. Кожна тематика об’єднує матеріали в логічно пов’язані підгрупи, що полегшує навігацію та дозволяє користувачам швидше орієнтуватися у вмісті.

Для ефективного представлення інформації портал реалізує поділ новин на дві основні форми подачі: короткі анонси (прев’ю) та повні тексти. Прев’ю включає заголовок, мініатюру зображення, дату публікації й вступне речення або фрагмент першого абзацу. Це дає змогу швидко ознайомитися з основними подіями. Натискання на заголовок веде до розгорнутої версії новини.

Кожна новина супроводжується метаданими: ім’ям автора, датою створення, списком тегів, категорією, списком пов’язаних публікацій. Часто також додаються галереї, відео або документи у вкладеннях.

Редактори та журналісти мають мати змогу працювати з панеллю керування, що дозволяє додавати нові записи, редагувати вже наявні, видаляти матеріали та змінювати їхню категоризацію. Така система керування контентом (CMS) може бути побудована без застосування сторонніх рішень, використовуючи тільки необхідні модулі, адаптовані до конкретних вимог.

З боку відвідувача важливо забезпечити просту й інтуїтивну навігацію, зокрема через головне меню, фільтрацію за датою, ключовими словами або категоріями. Для розуміння предметної області потрібно окремо розглянути технологічні компоненти, на яких будується будь-який вебпортал: інтерфейс користувача, програмна логіка, база даних.

Інтерфейс є першою точкою контакту між користувачем і контентом. У сучасних порталах використовується адаптивна верстка: сайт повинен коректно відображатися як на настільних комп’ютерах, так і на планшетах чи смартфонах. Компоненти інтерфейсу зазвичай включають:

1) панель навігації (категорії, пошук, перемикач теми, мова);
2) блок з анонсами новин;
3) підкатегорії або мітки;
4) сторінку повної новини;
5) окрему сторінку профілю користувача або редактора;

6) вікна авторизації та реєстрації.

Реалізація таких компонентів можлива з використанням HTML5, CSS3, JavaScript. Задля покращення взаємодії — асинхронні запити (AJAX, Fetch API), які дають змогу оновлювати вміст без перезавантаження сторінки [1].

Програмна логіка забезпечує функціональність обробки даних: надсилання форм, отримання новин із бази, сортування, фільтрація, пошук, автентифікація. Як середовище для розробки серверної частини буде використано Node.js.

Крім роботи з HTTP-запитами, важливим є керування доступом: відвідувачі бачать лише відкриті матеріали, а редактори мають змогу змінювати дані через захищену адміністративну панель.

Існуючі платформи публікації новин часто мають громіздкі CMS, у яких реалізовано надлишкову функціональність. Це ускладнює процес створення матеріалу, а іноді і знижує продуктивність. Вбудовані редактори не завжди підтримують потрібний рівень керування зображеннями, відео або вставкою структурованого HTML.

Ще однією особливістю є недоступність коду у SaaS-системах. При використанні готових хмарних рішень (наприклад, Wix, Tilda, WordPress.com) змінити архітектуру даних або додати специфічну поведінку стає неможливо без переходу на іншу платформу.

Рекламні блоки, банери, віджети соціальних мереж часто перевантажують сторінку та ускладнюють доступ до основного вмісту. Крім того, у багатьох безкоштовних системах не підтримується повноцінне керування правами доступу — важлива функція для багатокористувацьких ресурсів.

Ці фактори роблять створення власного вебпорталу оптимальним варіантом, якщо метою є реалізація ресурсу з чітко визначеними функціями, зрозумілою структурою та можливістю розширення.

Для правильної побудови інтерфейсу та логіки взаємодії необхідно розуміти, хто є потенційними користувачами. У випадку новинного сайту основними групами є:

відвідувачі — переглядають новини, шукають за ключовими словами, діляться посиланнями;

редактори — публікують новини, додають зображення, обирають категорії;

адміністратори — керують користувачами, змінюють категорії, слідкують за наповненням сайту.

Кожна з цих груп має свій набір доступних дій. Наприклад, редактор не може видалити чужу новину, якщо не має відповідного дозволу. У той же час адміністратор має змогу блокувати користувачів, переглядати статистику, змінювати структуру категорій.

Для реалізації таких сценаріїв використовуються механізми сесій, токенів доступу або інші засоби розмежування прав.

Новинний вебпортал може працювати у вигляді односторінкового застосунку (SPA) або багатосторінкової класичної структури. У першому випадку більшість логіки переміщується на сторону клієнта. У другому — кожен запит до сторінки обробляється сервером і віддає окремий HTML.

Для цього потрібно вибрати відповідний хостинг, який підтримує потрібну мову програмування, роботу з базою даних і обробку HTTP-запитів. Додатково необхідно враховувати:

1) зберігання зображень (локально або через зовнішні сервіси);
2) безпечне з’єднання (HTTPS, сертифікати);
3) захист від небажаних дій (наприклад, обмеження частоти запитів);
4) резервне копіювання даних.

Інфраструктура проекту залежить від обраної моделі розробки. Якщо портал реалізується як клієнт-серверна система, частину ресурсів розміщено на фронтенді, частину — на бекенді. Для цього можуть використовуватись окремі домени або піддомени.

Аналіз кількох популярних інформаційних ресурсів дозволяє виокремити загальні риси, які використовуються в сучасних новинних порталах:

головна сторінка відображає найбільш актуальні або рекомендовані матеріали;

1) категорії розміщені у вигляді горизонтального меню;
2) блок «останні новини» автоматично оновлюється;
3) підтримується багатомовність (для широкої аудиторії);
4) адаптивність під мобільні пристрої;
5) наявність фільтрації або хештегів.

У деяких порталах також реалізовано підписку на розсилку, систему коментарів, інструменти для редагування профілю користувача. Такі функції розширюють можливості, проте не є обов’язковими для базової реалізації.

Висновки до розділу 1

У ході розгляду основних аспектів, що передують безпосередній розробці системи, було сформовано чітке уявлення про характер завдань, які ставляться перед обома складовими проєкту — вебпорталом новинного спрямування та допоміжним програмним забезпеченням для складу. Основну увагу зосереджено на вебкомпоненті, що передбачає створення інформаційного ресурсу з гнучкою структурою, підтримкою інтерактивної взаємодії з користувачем.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Архітектура новинного вебпорталу

Побудова архітектури новинного вебпорталу здійснювалася з урахуванням простоти функціоналу, чіткої ролі користувача та базової логіки доступу до контенту. Загальна схема проекту передбачає взаємодію між клієнтською частиною, серверною логікою та базою даних. Уся система розділена на три основні рівні: інтерфейс користувача (frontend), серверна логіка (backend) та шар зберігання інформації (база даних). Такий підхід дозволяє розмежувати відповідальність між частинами та забезпечити зрозумілу структуру коду.

Цей компонент відповідає за відображення даних та взаємодію з користувачем. На головній сторінці сайту розміщується блок із трьома найновішими публікаціями. Вони завантажуються автоматично після відкриття сторінки. Якщо користувач ще не авторизований, доступ обмежується лише цими матеріалами.

Інтерфейс передбачає наявність форм для входу та реєстрації. У разі успішного входу користувачеві відкривається повна стрічка новин. Також стає доступною можливість редагування власного профілю. Кожна дія, що вимагає змін даних (наприклад, оновлення профілю), надсилає запит до серверної частини.

Верстка реалізована з використанням HTML та CSS. Інтерактивність забезпечується мовою JavaScript, яка обробляє події, формує запити до сервера, динамічно змінює вміст сторінки без повного перезавантаження.

Сервер виконує роль посередника між клієнтом і базою даних. Він обробляє всі запити, перевіряє автентичність користувача, надсилає відповіді з відповідними даними або повідомленнями про помилки.

Уся серверна логіка реалізована за допомогою Node.js. Для кожного запиту визначено маршрут (endpoint), який відповідає за певну дію: отримання

останніх новин, реєстрація нового користувача, авторизація, отримання повного списку новин, збереження змін у профілі.

Перед тим як надати доступ до повного контенту або дозволити редагування, сервер перевіряє, чи є користувач авторизованим. Для цього застосовується система токенів. Дані перевірки передаються в кожному запиті, що дозволяє контролювати права доступу. У структурі серверної частини реалізовані базові механізми захисту — хешування паролів, валідація введених даних, обробка некоректних запитів.

Для зберігання новинного контенту, облікових записів користувачів та налаштувань профілю використовується документно-орієнтована база даних MongoDB. Такий підхід дозволяє зберігати інформацію у форматі JSON-подібних об’єктів, що зручно для подальшої обробки на стороні JavaScript [8].

Кожна новина зберігається як окремий документ і містить такі поля, як заголовок, короткий опис, дата публікації, повний текст, автор. Запити на отримання новин виконуються з використанням сортування за датою, що дозволяє швидко відібрати останні публікації.

Облікові записи містять інформацію про логін, електронну пошту, хеш пароля та додаткові параметри профілю, які можуть бути змінені користувачем.

У процесі завантаження сторінки браузер надсилає запит до сервера для отримання останніх новин. Якщо користувач не авторизований, йому повертаються лише три записи. У разі входу — доступ розширюється, і сервер надає повний перелік. Профільна інформація зберігається окремо і передається тільки після авторизації.

Усі запити формуються на стороні клієнта JavaScript-кодом, а серверна частина Node.js відповідає за обробку, перевірку та повернення результатів. MongoDB виконує роль централізованого сховища, яке відповідає на запити через драйвер, підключений до Node.js.

Архітектура залишає простір для можливого розширення: наприклад, можна додати панель адміністратора, функцію коментування або

рейтингування новин без необхідності кардинально змінювати основу системи.

Блок-схема, яка створена на основі цієї структури (Рис. 2.1.), відображає основний потік роботи користувача і взаємодію між компонентами системи. Нижче наведено текстовий опис основних етапів:

1) Користувач відкриває сайт
2) Автоматичний запит до сервера (GET /news/short)
3) Відповідь: три останні новини
4) Користувач вирішує увійти або зареєструватися
5) Заповнення форми
6) Відправка запиту на сервер (POST /auth/login або POST /auth/register)
7) Сервер перевіряє або створює обліковий запис
8) Якщо успішно — створюється токен / сесія
9) Після входу
10) Клієнт надсилає запит (GET /news/all)
11) Сервер перевіряє автентифікацію
12) Відповідь: усі новини
13) Редагування профілю
14) Користувач відкриває форму редагування
15) Вносить зміни, відправляє запит (PUT /user/profile)
16) Сервер зберігає зміни в базі
17) Користувач переглядає новини
18) Клієнт відображає список

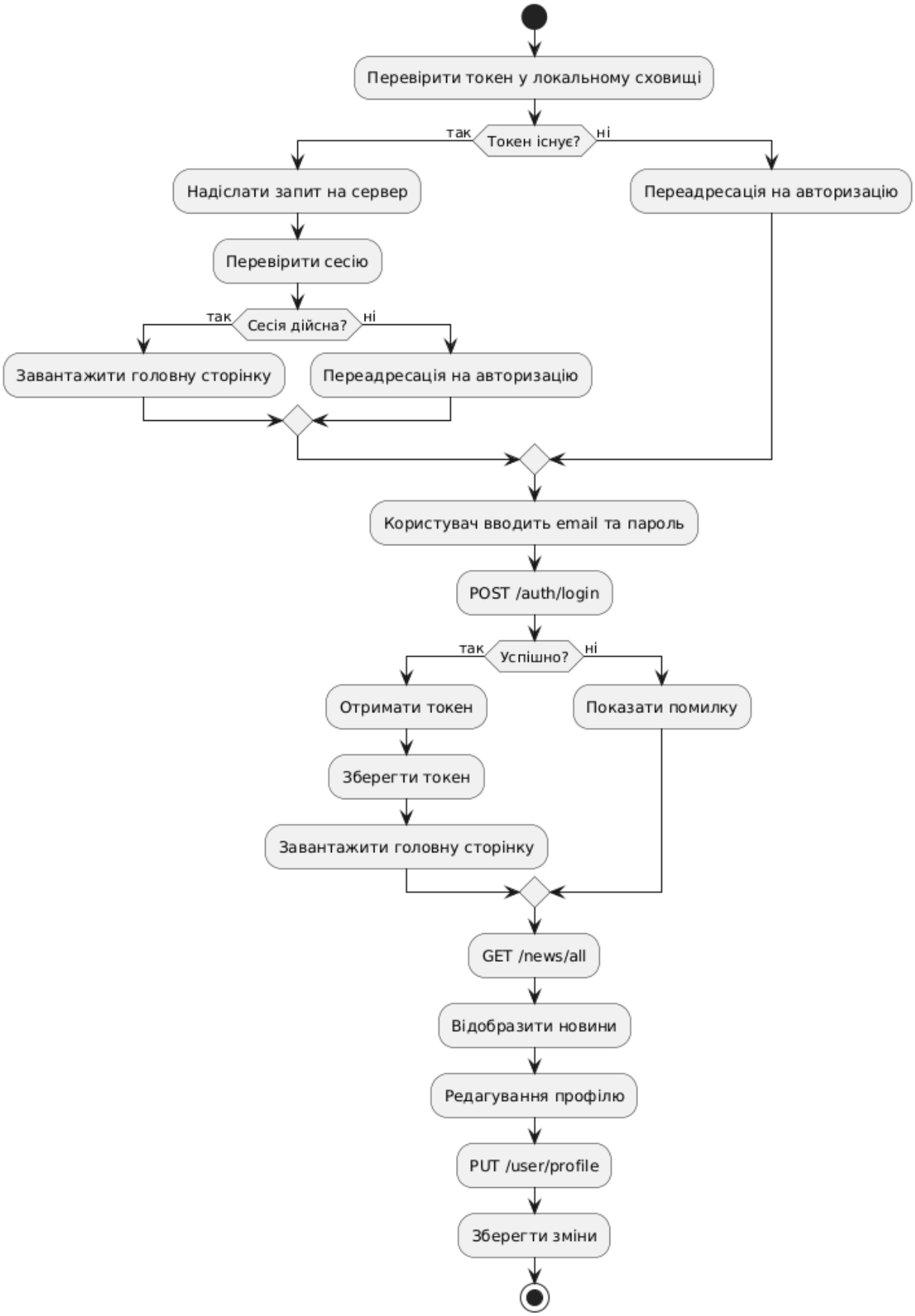

Рис. 2.1. Блок-схема роботи новинного порталу

2.2. Проектування інтерфейсу користувача

Інтерфейс користувача вебпорталу побудований відповідно до принципів простоти, зручності та послідовності. Враховуючи обмежену кількість функцій, основна увага приділена зрозумілості розташування елементів, логіці переходів між сторінками, а також збереженню єдиного стилю для всіх екранів.

Кожна сторінка має фіксовану структуру з чітко відокремленими зонами: хедер, контентна частина, футер. Таке компонування повторюється на всіх сторінках і підтримується для збереження візуальної цілісності та передбачуваності навігації.

Хедер присутній на всіх сторінках. Складається з:

1) Назви або логотипу сайту — розташований ліворуч, є посиланням на головну сторінку;
2) Меню навігації — розташоване праворуч і по центру.
3) Меню включає:
4) Головна — веде на стартову сторінку з новинами;
5) Увійти або Зареєструватися (якщо користувач неавторизований);
6) Мій профіль або Вийти (якщо користувач увійшов у систему).

Меню динамічно змінюється залежно від стану сесії користувача. Увійшовши, він бачить додаткові опції, зокрема доступ до сторінки редагування профілю.

Футер однаковий на всіх сторінках і містить лише текстовий блок. У ньому може бути зазначена інформація на зразок:

1) назва сайту;
2) короткий опис проєкту;
3) рік створення;
4) вказівка на навчальний або демонстраційний характер ресурсу.

Футер не містить функціональних елементів або навігації.

Головна сторінка, яку бачить будь-який відвідувач включає три найсвіжіші новини.

Кожна новина представлена у вигляді блоку з:

1) заголовком;
2) коротким описом або першим абзацом;
3) датою публікації;
4) кнопкою для перегляду повного тексту.

Коли користувач неавторизований, йому доступні лише ці три записи. Якщо користувач входить у систему, цей самий шаблон розширюється — новини завантажуються повністю, і їх може бути значно більше (при цьому структура відображення зберігається).

Сторінка входу містить форму з двома основними полями:

1) Електронна пошта або логін
2) Пароль

Під формою — кнопка підтвердження. Також є посилання на сторінку реєстрації. Якщо введені дані некоректні, відображається відповідне повідомлення про помилку. Сторінка не має зайвих елементів, крім хедера та футера.

Сторінка реєстрації візуально та структурно схожа на сторінку входу, але має додаткові поля:

1) Ім’я користувача
2) Електронна пошта
3) Пароль
4) Підтвердження пароля

Форма перевіряє правильність введення (наприклад, збіг пароля та підтвердження). У разі успіху користувач автоматично переходить до авторизованого режиму.

Розділ редагування профілю доступний тільки після входу. Структура включає:

1) текстову інформацію про поточні дані профілю;

2) форму з можливістю змінити:
3) ім’я користувача;
4) електронну пошту;
5) пароль;
6) кнопку для збереження змін;
7) повідомлення про успішне оновлення або помилки якщо введено некоректні дані.

Сторінка має мінімалістичний вигляд. Вона орієнтована на практичну функціональність і не містить великої кількості візуальних елементів.

Перехід між сторінками здійснюється через меню хедера або після виконання певних дій (наприклад, після входу користувач автоматично переходить на головну). Усі форми працюють асинхронно: введення даних не призводить до повного перезавантаження сторінки — взаємодія побудована на AJAX та fetch-запитах.

Інтерфейс реалізовано в світлій кольоровій гамі, з акцентами на заголовках та кнопках. Всі елементи мають чітку сітку розташування — новини розміщені у вертикальному списку, форми центровані, а елементи управління рознесені на зони. Весь контент адаптується до ширини екрана (ресурс сумісний із різними розширеннями браузера).

Висновки до розділу 2

У цьому розділі було сформовано загальне уявлення про внутрішню будову та зовнішній вигляд новинного вебпорталу. Описано логіку взаємодії між основними компонентами системи — клієнтом, сервером і базою даних, що разом утворюють просту, але цілеспрямовану архітектурну модель. Детально розглянуто структуру інтерфейсу користувача, включно з розміткою типових сторінок: головної, авторизації, реєстрації та редагування профілю.

РОЗДІЛ 3
ПРОЕКТУВАННЯ БАЗИ ДАНИХ ДОДАТКУ

3.1. Побудова та опис інфологічної моделі

Інфологічна модель бази даних відображає логічну структуру даних, необхідних для організації обліку товарів на складі, контролю їх надходження, зберігання та руху. Модель включає ключові сутності предметної області, їх атрибути та взаємозв'язки між ними.

Основними об'єктами бази даних складського обліку є

Виробнича програма — сутність, яка відображає дані про виробничу програму.

Виріб — сутність, яка описує виріб.

Матеріал — сутність, що відображає матеріал.

Витрати матеріалу — сутність, яка містить інформацію про те, який матеріал і в якій кількості витрачається на виготовлення певного виробу.

Заробітна плата – сутність, яка відображає заробітну плату робітників, які виробляють продукцію.

Складові елементи сутностей інфологічної моделі:

Виріб

1) Виріб_код – унікальний ідентифікатор виробу.
    a) Формат: 9(6)
    b) Відсоток наявності: 100%
    c) Обмеження: відсутні
    d) Частота використання: висока (щодня)
    e) Область допустимих значень: діапазон чисел від 1 до 999999
    f) Виводимість: ні
    g) Дублювання: ні
2) Назва_виробу – текстова назва виробу.
    a) Формат: X(100)
    b) Відсоток наявності: 100%

c) Обмеження: відсутні
d) Частота використання: висока (щодня)
e) Область допустимих значень: текст із буквами, цифрами та символами
f) Виводимість: ні
g) Дублювання: ні

Матеріал

1) Матеріал_код – унікальний ідентифікатор матеріалу.
    a) Формат: 9(6)
    b) Відсоток наявності: 100%
    c) Обмеження: відсутні
    d) Частота використання: висока (щодня)
    e) Область допустимих значень: діапазон чисел від 1 до 999999
    f) Виводимість: ні
    g) Дублювання: ні
2) Назва_матеріалу – текстова назва матеріалу.
    a) Формат: X(100)
    b) Відсоток наявності: 100%
    c) Обмеження: відсутні
    d) Частота використання: висока (щодня)
    e) Область допустимих значень: текст із буквами, цифрами та символами
    f) Виводимість: ні
    g) Дублювання: ні
3) Одиниця_виміру – одиниця вимірювання матеріалу.
    a) Формат: X(50)
    b) Відсоток наявності: 100%
    c) Обмеження: відсутні
    d) Частота використання: середня (щотижня)
    e) Область допустимих значень: кг, л, м, шт.

f) Виводимість: ні

g) Дублювання: так

4) Ціна – вартість одиниці матеріалу.

a) Формат: S9(8).99

b) Відсоток наявності: 100%

c) Обмеження: доступ обмежено для бухгалтерії

d) Частота використання: висока (щодня)

e) Область допустимих значень: діапазон від 0.01 до 999999.99

f) Виводимість: ні

g) Дублювання: так

Виробнича програма

1) Програма_код – унікальний ідентифікатор програми.

a) Формат: 9(6)

b) Відсоток наявності: 100%

c) Обмеження: відсутні

d) Частота використання: висока (щодня)

e) Область допустимих значень: діапазон чисел від 1 до 999999

f) Виводимість: ні

g) Дублювання: ні

2) Рік_програми – рік виконання програми.

a) Формат: 9(4)

b) Відсоток наявності: 100%

c) Обмеження: відсутні

d) Частота використання: низька (раз на рік)

e) Область допустимих значень: від 2000 до 2100

f) Виводимість: ні

g) Дублювання: так

3) Програма_випуску – кількість виробів, що випускаються за програмою.

a) Формат: 9(8)

b) Відсоток наявності: 100%

c) Обмеження: відсутні
d) Частота використання: середня (щомісяця)
e) Область допустимих значень: від 1 до 10000000
f) Виводимість: ні
g) Дублювання: так

4) Виріб_код – посилання на виріб.
a) Формат: 9(6)
b) Відсоток наявності: 100%
c) Обмеження: відсутні
d) Частота використання: висока (щодня)
e) Область допустимих значень: ідентифікатор існуючого виробу
f) Виводимість: ні
g) Дублювання: так

Витрати матеріалу

1) Витрати_код – унікальний ідентифікатор витрат.
a) Формат: 9(6)
b) Відсоток наявності: 100%
c) Обмеження: відсутні
d) Частота використання: висока (щодня)
e) Область допустимих значень: діапазон чисел від 1 до 999999
f) Виводимість: ні
g) Дублювання: ні

2) Норма_витрат – кількість матеріалів, необхідних для виготовлення одного виробу.
a) Формат: 9(6)
b) Відсоток наявності: 100%
c) Обмеження: відсутні
d) Частота використання: середня (щомісяця)
e) Область допустимих значень: від 1 до 999999
f) Виводимість: ні

g) Дублювання: так

3) Виріб_код – посилання на виріб.
   a) Формат: 9(6)
   b) Відсоток наявності: 100%
   c) Обмеження: відсутні
   d) Частота використання: висока (щодня)
   e) Область допустимих значень: ідентифікатор існуючого виробу
   f) Виводимість: ні
   g) Дублювання: ні
4) Матеріал_код – посилання на матеріал.
   a) Формат: 9(6)
   b) Відсоток наявності: 100%
   c) Обмеження: відсутні
   d) Частота використання: висока (щодня)
   e) Область допустимих значень: ідентифікатор існуючого матеріалу
   f) Виводимість: ні
   g) Дублювання: так

Заробітна плата

1) Плата_код – унікальний ідентифікатор запису про заробітну плату.
   a) Формат: 9(6)
   b) Відсоток наявності: 100%
   c) Обмеження: відсутні
   d) Частота використання: висока (щодня)
   e) Область допустимих значень: від 1 до 999999
   f) Виводимість: ні
   g) Дублювання: ні
2) Виріб_код – посилання на виріб, для якого нараховується заробітна плата.
   a) Формат: 9(6)
   b) Відсоток наявності: 100%

c) Обмеження: повинен посилатися на існуючий запис у таблиці "Виріб"

d) Частота використання: висока (щодня)

e) Область допустимих значень: ідентифікатор існуючого виробу

f) Виводимість: ні

g) Дублювання: так

3) Ставка_за_одиницю – ставка заробітної плати за виготовлення одиниці продукції.

a) Формат: S9(8).99

b) Відсоток наявності: 100%

c) Обмеження: відсутні

d) Частота використання: висока (щодня)

e) Область допустимих значень: від 0.01 до максимальної допустимої величини для типу MONEY

f) Виводимість: ні

g) Дублювання: так

4) Норма_праці – кількість трудових одиниць, необхідних для виготовлення одного виробу.

a) Формат: 9(6)

b) Відсоток наявності: 100%

c) Обмеження: відсутні

d) Частота використання: середня (щомісяця)

e) Область допустимих значень: від 1 до 999999

f) Виводимість: ні

g) Дублювання: так

Опис характеристик об’єктів:

Виріб

1) Спосіб звертання до екземплярів об'єкта – К (Виріб_код)
2) Структурна активність об'єкта – так

3) Обмеження на право звертання до екземплярів об'єкта – доступ дозволено адміністраторам і менеджерам виробництва
4) Кількість екземплярів об'єкта – 10000
5) Мінливість складу екземплярів – 5% на місяць

Матеріал

1) Спосіб звертання до екземплярів об'єкта – К (Матеріал_код)
2) Структурна активність об'єкта – так
3) Обмеження на право звертання до екземплярів об'єкта – доступ дозволено адміністраторам і бухгалтерам
4) Кількість екземплярів об'єкта – 5 000
5) Мінливість складу екземплярів – 10% на місяць

Виробнича програма

1) Спосіб звертання до екземплярів об'єкта – К (Програма_код)
2) Структурна активність об'єкта – так
3) Обмеження на право звертання до екземплярів об'єкта – доступ дозволено адміністраторам і планувальникам
4) Кількість екземплярів об'єкта – 10000
5) Мінливість складу екземплярів – 15% на рік

Витрати матеріалу

1) Спосіб звертання до екземплярів об'єкта – М (Виріб_код, Матеріал_код)
2) Структурна активність об'єкта – ні
3) Обмеження на право звертання до екземплярів об'єкта – доступ дозволено адміністраторам і технологам
4) Кількість екземплярів об'єкта – 50000
5) Мінливість складу екземплярів – 20% на місяць

Заробітна плата

1) Спосіб звертання до екземплярів об'єкта – К (Плата_код)
2) Структурна активність об'єкта – так
3) Обмеження на право звертання до екземплярів об'єкта – доступ дозволено адміністраторам і кадровим спеціалістам

4) Кількість екземплярів об'єкта – 10 000

5) Мінливість складу екземплярів – 15% на місяць

Опис відносин між об’єктами:

Виріб-Витрати матеріалу

1) Напрямок руху по структурному зв’язку – ВП

2) Кратність зв’язку – 1:М (один виріб може мати багато витрат матеріалів)

3) Спосіб упорядкування екземплярів підпорядкованого об'єкта – ДАТА Д (упорядкування за датою витрат у порядку зменшення, можливе дублювання)

4) Клас членства підпорядкованого об’єкта – обов’язковий (кожна витрата має бути пов’язана з певним виробом)

Матеріал-Витрати матеріалу

1) Напрямок руху по структурному зв’язку – ВП

2) Кратність зв’язку – 1:М (один матеріал може використовуватись у багатьох витратах)

3) Спосіб упорядкування екземплярів підпорядкованого об'єкта – КОД_МАТЕРІАЛУ (упорядкування за кодом матеріалу у порядку зростання)

4) Клас членства підпорядкованого об’єкта – обов’язковий (кожна витрата має містити конкретний матеріал)

Виробнича програма-Виріб

1) Напрямок руху по структурному зв’язку – ВПВ

2) Кратність зв’язку – 1:М (одна програма може включати декілька виробів, але кожен виріб належить тільки до однієї програми)

3) Спосіб упорядкування екземплярів підпорядкованого об'єкта – КОД_ВИРОБУ (упорядкування за кодом виробу у порядку зростання)

4) Клас членства підпорядкованого об’єкта – обов’язковий (кожен виріб обов’язково входить у певну виробничу програму)

Заробітна плата-Виріб

1) Напрямок руху по структурному зв’язку – ЗПВ

2) Кратність зв’язку – 1:M (один виріб може бути пов'язаний із декількома записами заробітної плати, але кожен запис заробітної плати відноситься лише до одного виробу)
3) Спосіб упорядкування екземплярів підпорядкованого об'єкта – ВИРІБ_КОД (упорядкування за кодом виробу у порядку зростання)
4) Клас членства підпорядкованого об’єкта – обов’язковий (для кожного запису заробітної плати обов’язково має бути визначений відповідний виріб)

3.2. Побудова та опис моделі сутність-зв’язок

Логічна модель описує структуру бази даних з урахуванням реалізації в конкретній СУБД. Вона містить детальний опис таблиць, атрибутів, ключів, зв’язків і обмежень, які забезпечують цілісність даних. Логічна модель побудована на основі інфологічної моделі та деталізує структуру з урахуванням вимог до нормалізації даних. Абстрактна структура інфологічної моделі трансформується у більш формалізовану систему, яка враховує конкретні вимоги до організації даних та забезпечує їх ефективне зберігання і обробку.

Логічна модель бази даних складу розроблена для управління інформацією про вироби, матеріали та виробничі програми. Вона включає таблиці, які відображають основні сутності системи управління виробництвом, та зв’язки між ними (Рис. 3.1.).

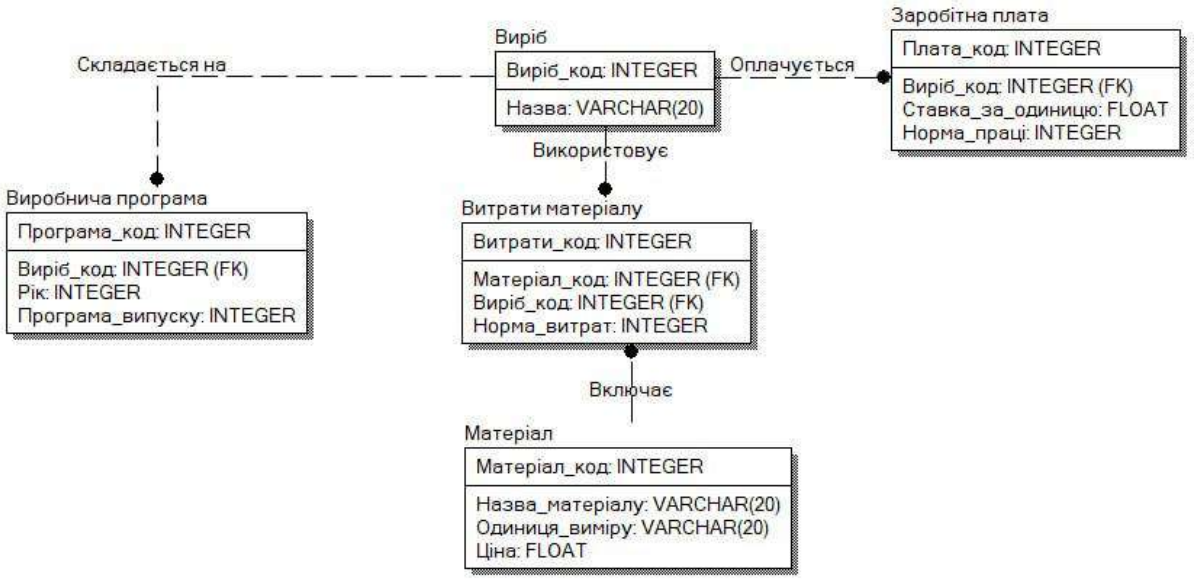

Рис. 3.1. Логічна модель бази даних обліку продукції

Логічна модель містить наступні сутності:

Виробнича програма — сутність, яка відображає дані про виробничу програму.

Виріб — сутність, яка описує виріб.

Матеріал — сутність, що відображає матеріал.

Витрати матеріалу — сутність, яка містить інформацію про те, який матеріал і в якій кількості витрачається на виготовлення певного виробу.

Заробітна плата – сутність, яка відображає заробітну плату робітників, які виробляють продукцію.

Атрибути всіх сутностей в ER-діаграмі наступні:

Виробнича програма:

1) Програма_код - номер програми (int, primary key)
2) Рік - рік програми (int)
3) Програма_випуску - програма випуску (int)
4) Виріб_код - посилання на сутність виробів (int, foreign key)

Матеріал:

1) Матеріал_код - номер матеріалу (int, primary key)
2) Назва - назва матеріалу (varchar)

3) Одиниця вимірювання (varchar)
4) Ціна (float)

Виріб:

1) Виріб_код - номер виробу (int, primary key)
2) Назва - назва виробу (varchar)

Витрати матеріалу:

1) Витрати_код - номер витрат (int, primary key)
2) Норма_витрат - норма витрат (int)
3) Виріб_код - посилання на сутність виробів (int, foreign key)
4) Матеріал_код - посилання на сутність матеріалів (int, foreign key)

Заробітна плата:

1) Плата_код – унікальний ідентифікатор запису заробітної плати (int, primary key).
2) Виріб_код – посилання на виріб, для якого визначено заробітну плату (int, foreign key).
3) Ставка_за_одиницю – ставка заробітної плати за одиницю виробу (float).
4) Норма_праці – кількість робочих годин, необхідних для виготовлення одиниці виробу (int).

Виробнича програма-Виріб (Складається на) має тип зв’язку 1:Б (один до багатьох), оскільки багато виробничих програм можуть містити один виріб. Потужність: мінімум — 0, максимум — не обмежено (виробнича програма може включати будь-яку кількість виробів). Обов’язковість: для виробничої програми — обов’язковий зв’язок (неможливо створити виробничу програму без виробів), для виробу — необов’язковий зв’язок (виріб може існувати без прив’язки до виробничої програми).

Виріб-Витрати матеріалу (Використовує) має тип зв’язку 1:Б (один до багатьох), оскільки один виріб може використовувати кілька різних матеріалів. Потужність: мінімум — 1 (кожен виріб обов’язково використовує хоча б один матеріал), максимум — не обмежено (виріб може використовувати необмежену кількість матеріалів). Обов’язковість: для виробу — обов’язковий

зв’язок (неможливо створити виріб без визначення витрат матеріалів), для витрат матеріалу — обов’язковий зв’язок (кожен запис про витрати матеріалу має бути прив’язаний до якогось виробу).

Матеріал-Витрати матеріалу (Включає) має тип зв’язку 1:Б (один до багатьох), оскільки один матеріал може використовуватися для створення багатьох виробів. Потужність: мінімум — 0 (матеріал може існувати в базі даних, навіть якщо він не використовується у виробництві), максимум — не обмежено (матеріал може бути використаний у будь-якій кількості виробів). Обов’язковість: для матеріалу — необов’язковий зв’язок (матеріал може існувати самостійно, без прив’язки до витрат матеріалів), для витрат матеріалу — обов’язковий зв’язок (неможливо зафіксувати витрати матеріалів без зазначення конкретного матеріалу).

Заробітна плата-Виріб (Оплачується) має тип зв’язку 1:Б (один до багатьох), оскільки багато записів про заробітну плату можуть бути прив’язані до одного виробу. Потужність: мінімум — 0, максимум — не обмежено (виріб може мати будь-яку кількість записів про заробітну плату). Обов’язковість: для заробітної плати — обов’язковий зв’язок (кожен запис про заробітну плату повинен бути прив’язаний до конкретного виробу), для виробу — необов’язковий зв’язок (виріб може існувати без прив’язки до заробітної плати).

Процес створення фізичної моделі бази даних починається після завершення проектування логічної моделі. Фізична модель відображає логічну структуру в конкретну реалізацію, яка враховує особливості обраної системи управління базами даних (СУБД) та технічні вимоги до роботи системи.

Фізична модель бази даних була створена з метою оптимізації зберігання та обробки інформації про компоненти складу (Рис. 3.2.).

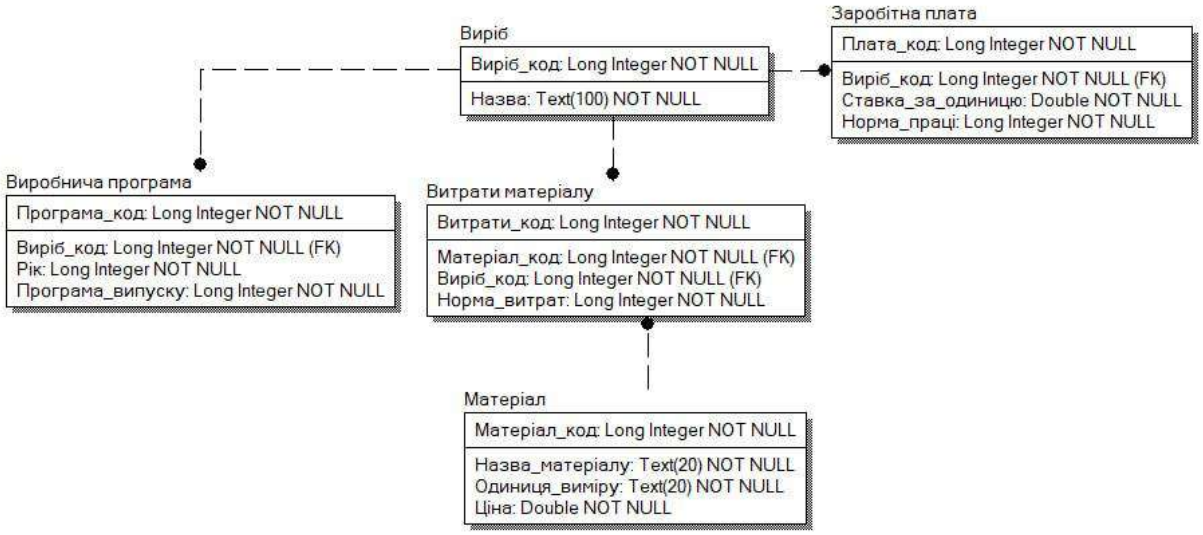

Рис. 3.2. Фізична модель бази даних обліку продукції

Атрибути перетворилися на стовпці, сутності перетворилися на таблиці та зазнали певних змін.

Атрибути всіх таблиць в ER-діаграмі наступні:

Виробнича програма:

1) Програма_код - номер програми (long int, primary key)
2) Рік - рік програми (long int)
3) Програма_випуску - програма випуску (long int)
4) Виріб_код - посилання на сутність виробів (long int, foreign key)

Матеріал:

1) Матеріал_код - номер матеріалу (long int, primary key)
2) Назва - назва матеріалу (text 20)
3) Одиниця вимірювання (text 20)
4) Ціна (double)

Виріб:

1) Виріб_код - номер виробу (long int, primary key)
2) Назва - назва виробу (text 20)

Витрати матеріалу:

1) Витрати_код - номер витрат (long int, primary key)
2) Норма_витрат - норма витрат (long int)

3) Виріб_код - посилання на сутність виробів (long int, foreign key)
4) Матеріал_код - посилання на сутність матеріалів (long int, foreign key)

Заробітна плата:

1) Плата_код – унікальний ідентифікатор запису заробітної плати (long int, primary key).
2) Виріб_код – посилання на виріб, для якого визначено заробітну плату (long int, foreign key).
3) Ставка_за_одиницю – ставка заробітної плати за одиницю виробу (double).
4) Норма_праці – кількість робочих годин, необхідних для виготовлення одиниці виробу (long int).

Відносини між таблицями теж зазнали певних перетворень.

Виробнича програма-Виріб — тип зв'язку 1:Б (один до багатьох), оскільки одна виробнича програма може містити багато виробів. Потужність: мінімум — 0, максимум — не обмежено (виробнича програма може включати будь-яку кількість виробів). Обов'язковість: для виробничої програми — обов'язковий зв'язок (неможливо створити виробничу програму без виробів), для виробу — необов'язковий зв'язок (виріб може існувати без прив'язки до виробничої програми).

1) Child Delete: NONE
2) Parent Delete: RESTRICT
3) Child Insert: RESTRICT
4) Parent Insert: NONE
5) Child Update: RESTRICT
6) Parent Update: RESTRICT

Виріб-Витрати матеріалу — тип зв'язку 1:Б (один до багатьох), оскільки один виріб може використовувати кілька різних матеріалів. Потужність: мінімум — 1 (кожен виріб обов'язково використовує хоча б один матеріал), максимум — не обмежено (виріб може використовувати необмежену кількість матеріалів). Обов'язковість: для виробу — обов'язковий

зв’язок (неможливо створити виріб без визначення витрат матеріалів), для витрат матеріалу — обов’язковий зв’язок (кожен запис про витрати матеріалу має бути прив’язаний до якогось виробу).

1) Child Delete: NONE
2) Parent Delete: RESTRICT
3) Child Insert: RESTRICT
4) Parent Insert: NONE
5) Child Update: RESTRICT
6) Parent Update: RESTRICT

Матеріал-Витрати матеріалу — тип зв’язку 1:Б (один до багатьох), оскільки один матеріал може використовуватися для створення багатьох виробів. Потужність: мінімум — 0 (матеріал може існувати в базі даних, навіть якщо він не використовується у виробництві), максимум — не обмежено (матеріал може бути використаний у будь-якій кількості виробів). Обов’язковість: для матеріалу — необов’язковий зв’язок (матеріал може існувати самостійно, без прив’язки до витрат матеріалів), для витрат матеріалу — обов’язковий зв’язок (неможливо зафіксувати витрати матеріалів без зазначення конкретного матеріалу).

1) Child Delete: NONE
2) Parent Delete: RESTRICT
3) Child Insert: RESTRICT
4) Parent Insert: NONE
5) Child Update: RESTRICT
6) Parent Update: RESTRICT

Заробітна плата-Виріб (Оплачується) має тип зв’язку 1:Б (один до багатьох), оскільки багато записів про заробітну плату можуть бути прив’язані до одного виробу. Потужність: мінімум — 0, максимум — не обмежено (виріб може мати будь-яку кількість записів про заробітну плату). Обов’язковість: для заробітної плати — обов’язковий зв’язок (кожен запис про заробітну плату повинен бути прив’язаний до конкретного виробу), для виробу —

необов’язковий зв’язок (виріб може існувати без прив’язки до заробітної плати).

1) Child Delete: NONE
2) Parent Delete: RESTRICT
3) Child Insert: RESTRICT
4) Parent Insert: NONE
5) Child Update: RESTRICT
6) Parent Update: RESTRICT

3.3. Реалізація бази даних в СУБД

В якості СУБД розробки бази даних було обрано SQLite. Такий вибір зумовлений актуальністю СУБД в сучасній практиці управління базами даних, адаптивністю до різних потреб і здатністю задовольняти вимоги автоматизації облікових процесів і калькуляції собівартості продукції.

SQLite було включено до списку завдяки відкритому коду, що знижує витрати на впровадження та підтримку. Його кросплатформеність, легкість адміністрування та висока швидкість виконання запитів роблять цю СУБД оптимальною для малого та середнього бізнесу. SQLite забезпечує баланс між доступністю та продуктивністю, відповідаючи вимогам до сучасних інформаційних систем.

У СУБД SQLite була створена база даних для обліку витрат на виробництво продукції та калькуляції собівартості, яка реалізована за допомогою інструментів для управління реляційними структурами. Вона включає таблиці для збереження інформації про вироби, матеріали, виробничі програми та витрати матеріалів. Реалізовані зв’язки між таблицями забезпечують логічну відповідність між записами, підтримують цілісність даних та гарантують їх узгодженість під час оновлення або внесення змін.

Структура бази даних передбачає збереження відомостей про вироби, які є об’єктами виробничих процесів, із зазначенням їхніх найменувань. Зв’язки

між таблицями реалізовані за допомогою зовнішніх ключів, що дозволяє забезпечити взаємозв’язок даних на різних рівнях та виключити дублювання інформації. Крім основних таблиць, у базі даних створено представлення, які автоматизують процеси розрахунку собівартості продукції.

У наступній табл. 3.1 наведено опис властивостей стовпців таблиць, що використовуються для обліку витрат на виробництво продукції та калькуляції собівартості обраній раніше СУБД.

Таблиця 3.1

Опис властивостей стовпців таблиць

назва таблиці	ім'я таблиці	тип даних	властивості
виріб	виріб_код	int	not null, primary key
	назва_виробу	varchar(100)	not null
виробнича_програма	програма_код	int	not null, primary key
	рік_програми	year	not null
	програма_випуску	int	not null
	виріб_код	int	not null, foreign key
матеріал	матеріал_код	int	not null, primary key
	назва_матеріалу	varchar(100)	not null
	одиниця_виміру	varchar(50)	not null
	ціна	double(10, 2)	not null
витрати_матеріалу	витрати_код	int	not null, primary key
	норма_витрат	int	not null
	виріб_код	int	not null, foreign key
	матеріал_код	int	not null, foreign key

Продовження табл. 3.1

заробітна_плата	Плата_код	int	not null, primary key
	Виріб_код	int	not null, foreign key
	Ставка_за_одиницю	double(10, 2)	not null
	Норма_праці	int	not null

Для кожної таблиці бази даних визначені властивості стовпців, оскільки вони визначають структуру даних і забезпечують правильне зберігання та обробку інформації. Кожен стовпець має визначений тип даних і відповідні обмеження, що забезпечують цілісність та коректність введених значень.

Висновки до розділу 3

Процес розробки бази даних для автоматизованого обліку витрат на виробництво продукції та калькуляції собівартості включав проектування структури бази даних та розробку бази даних.

РОЗДІЛ 4
РОЗРОБКА WEB-САЙТУ

4.1. Обґрунтування вибору засобів розробки

Під час реалізації дипломного проекту використовувались різні інструменти та середовища програмування, кожне з яких виконувало конкретну функцію у відповідній частині системи. Враховуючи наявність двох незалежних компонентів — вебпорталу для новин та настільної програми для планування товарів на складі — було доцільно застосовувати різні мови програмування, бібліотеки та бази даних, оптимально пристосовані до особливостей кожного з об'єктів розробки.

Для створення клієнтської частини порталу використовувались HTML (розмітка), CSS (оформлення) та JavaScript (взаємодія з елементами сторінки). Цей набір інструментів дає змогу сформувати гнучкий та інтерактивний інтерфейс без використання сторонніх візуальних конструкторів.

HTML забезпечує структуровану організацію вмісту: заголовки, абзаци, блоки новин, списки, панель навігації, форми пошуку. CSS дозволяє адаптувати вигляд сайту до різних пристроїв, підтримуючи медіазапити та стилізацію згідно з темами або кольоровими схемами. JavaScript забезпечує реалізацію таких сценаріїв, як динамічне завантаження новин, пошук без перезавантаження сторінки, робота з формами входу, фільтрація та взаємодія з сервером через API-запити.

У ролі середовища для виконання серверної логіки обрано Node.js. Його основною перевагою є те, що він використовує одну й ту ж мову — JavaScript — на клієнтському та серверному боці. Це спрощує комунікацію між частинами системи та дозволяє зменшити кількість контекстних перемикань між мовами.

Node.js працює так, що не зупиняє виконання програми під час очікування відповіді від бази даних чи зовнішнього запиту. Це особливо зручно, коли багато користувачів одночасно читають новини, шукають

інформацію або надсилають запити. Крім того, для Node.js доступна велика кількість готових модулів через менеджер пакетів npm. Це дозволяє швидко додавати функції на зразок обробки запитів, логування, авторизації чи роботи з сесіями, не пишучи все з нуля.

MongoDB обрано як основну базу даних для вебпорталу. Вона зберігає інформацію у вигляді документів (формат JSON), що добре підходить до динамічного характеру новинного контенту. У MongoDB не потрібно задавати жорстку схему таблиць, тому структура документа може легко змінюватися у процесі розробки без втрати даних.

Кожна новина, користувач або категорія зберігається як окремий документ, що дозволяє ефективно здійснювати пошук за ключовими словами, дати, автором або тегами. MongoDB підтримує агрегування, що дає змогу формувати вибірки з урахуванням складних умов (наприклад, отримати найпопулярніші новини за останні три дні в категорії «технології»).

Десктопний додаток для роботи зі складом використовує реляційні структури для збереження локальних даних, як-от товар на складі, відвантаження або тимчасові таблиці для розрахунків. У таких випадках застосовується SQLite — компактна, вбудована база даних, яка не потребує окремого сервера [2]. Вона забезпечує надійне зберігання, контроль цілісності даних та підтримує запити SQL будь-якої складності. Взаємодія із SQLite у програмі реалізується через відповідні Qt-модулі.

Для створення десктопного застосунку було використано мову програмування C++ разом із фреймворком Qt. Qt дозволяє створювати зручний графічний інтерфейс із сучасними компонентами: таблицями, вкладками, формами введення, а також підтримує побудову кастомних елементів керування. Інтеграція з SQLite, обробка подій, робота з файлами та відображення вмісту складу — усе це реалізовано у межах єдиної системи.

Застосування C++ у поєднанні з Qt дає змогу ефективно керувати ресурсами, працювати з великою кількістю об’єктів — товарів, комірок, зон зберігання — та точно візуалізувати їх розміщення на складі. Додаток реалізує

просту візуалізацію схеми складу у дво- або тривимірному вигляді, що полегшує контроль і планування простору.

Графічна частина реалізується з використанням OpenGL — інтерфейсу для побудови 2D- або 3D-зображень. У застосунку OpenGL використовується для виводу плану складу, де кожен елемент — це окрема структура: коробка, зона, транспортна лінія.

Завдяки OpenGL можливе масштабування, обертання сцени, інтерактивна зміна розташування елементів, підсвітка вибраних об'єктів. Користувач може в реальному часі переміщати віртуальні товари, аналізувати просторові відстані або перевіряти доступність зон зберігання.

4.2. Розробка web-сайту

Інтерфейс користувача сайту реалізований за допомогою HTML, CSS та JavaScript. Після завантаження головної сторінки (index.html) користувач бачить заголовок сайту, короткий опис і список останніх новин. Якщо користувач неавторизований, відображаються три найновіші записи, а також кнопки "Вхід" і "Реєстрація", розміщені у верхньому правому куті (Рис. 4.1.).

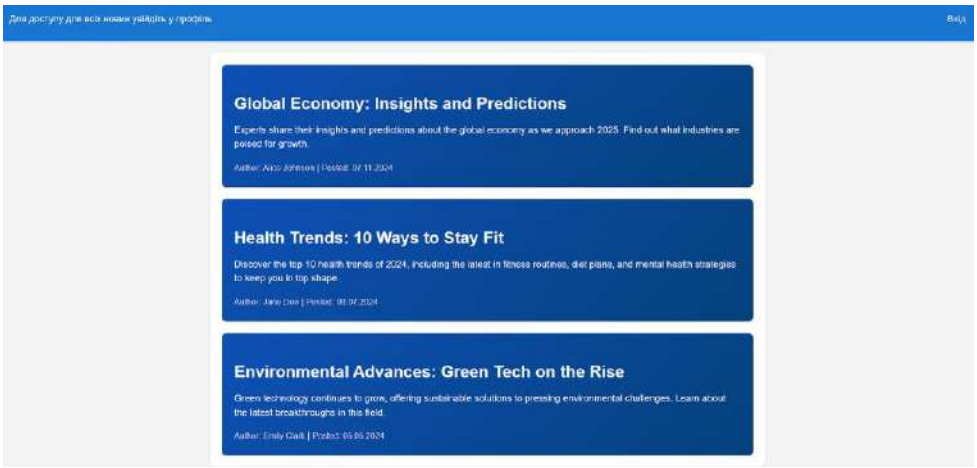

Рис. 4.1. Головна сторінка для неавторизованого користувача

Після натискання на кнопку "Вхід", відкривається форма авторизації. Вона має два поля для введення логіна і пароля та кнопку "Увійти" (Рис. 4.2.).

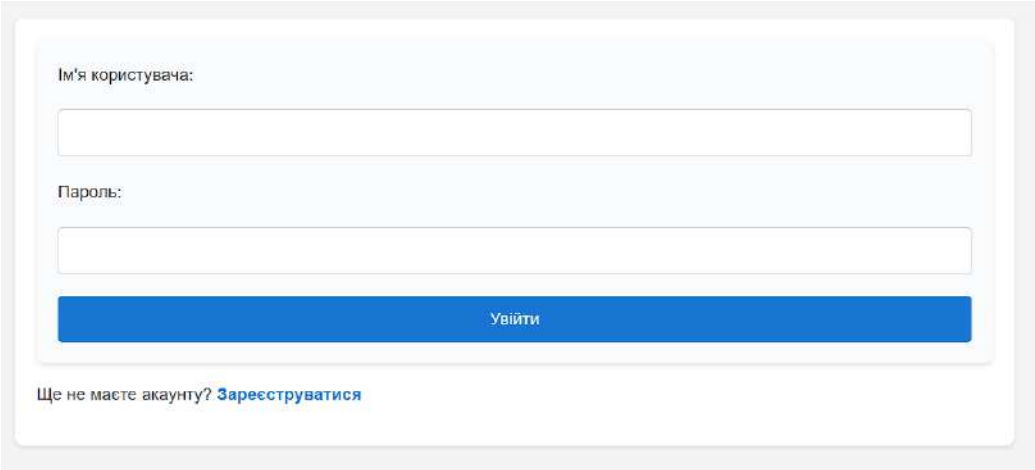

Рис. 4.2. Форма входу

Якщо натиснути кнопку "Реєстрація", відкривається сторінка з формою для створення облікового запису (Рис. 4.3.).

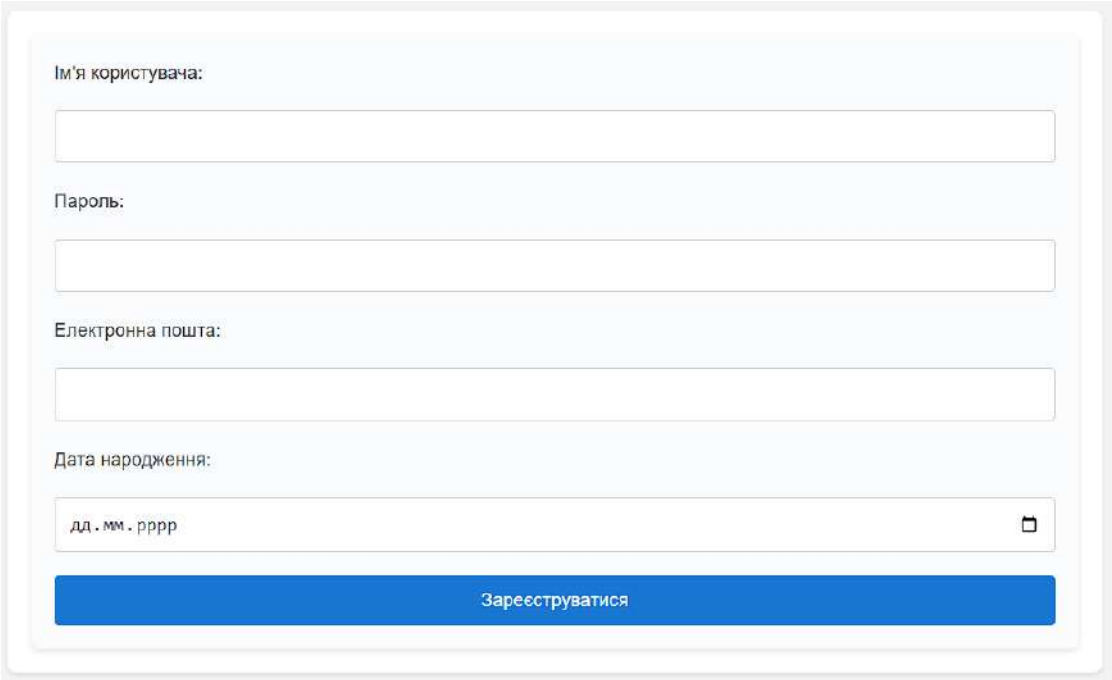

Рис. 4.3. Форма реєстрації

Користувач повинен ввести ім’я, пароль, електронну пошту, дату народження і натиснути "Зареєструватися". Якщо всі поля заповнені коректно, користувача перенаправляє назад на головну сторінку, де він вже перебуває в системі. Помилки, наприклад, при невідповідності паролів, спеціальним повідомленням.

У разі успішної реєстрації, користувач переадресовується на сторінку входу. У разі успішного входу користувача автоматично перенаправляє на

головну сторінку, яка тепер містить повний список новин і кнопку "Вийти", "Налаштування". Після входу користувач стають доступні до перегляду усі новини порталу, а також привітальне повідомлення з ніком користувача (Рис. 4.4.).

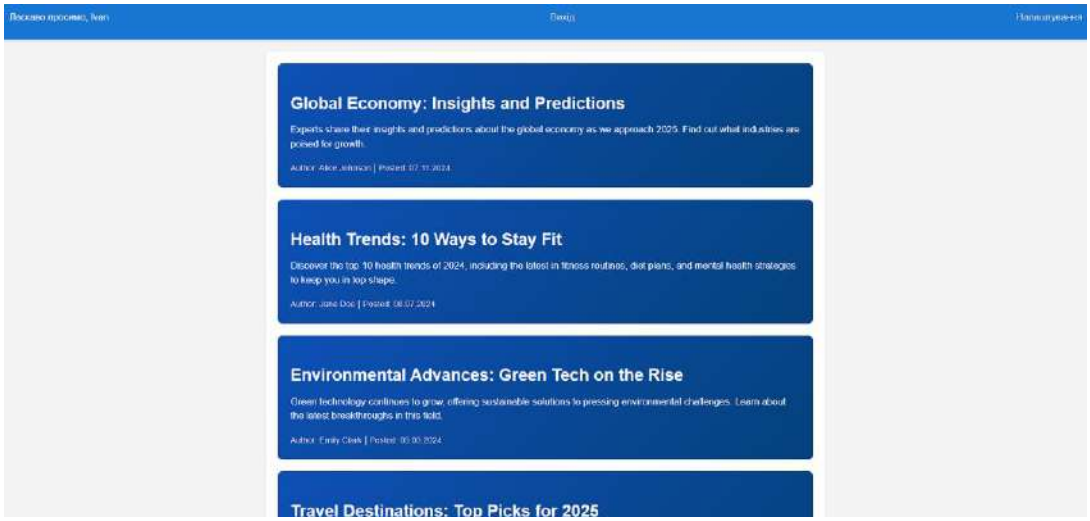

Рис. 4.4. Головна сторінка після авторизації

На сторінці налаштувань користувач може змінити основні дані свого профілю (Рис. 4.5.).

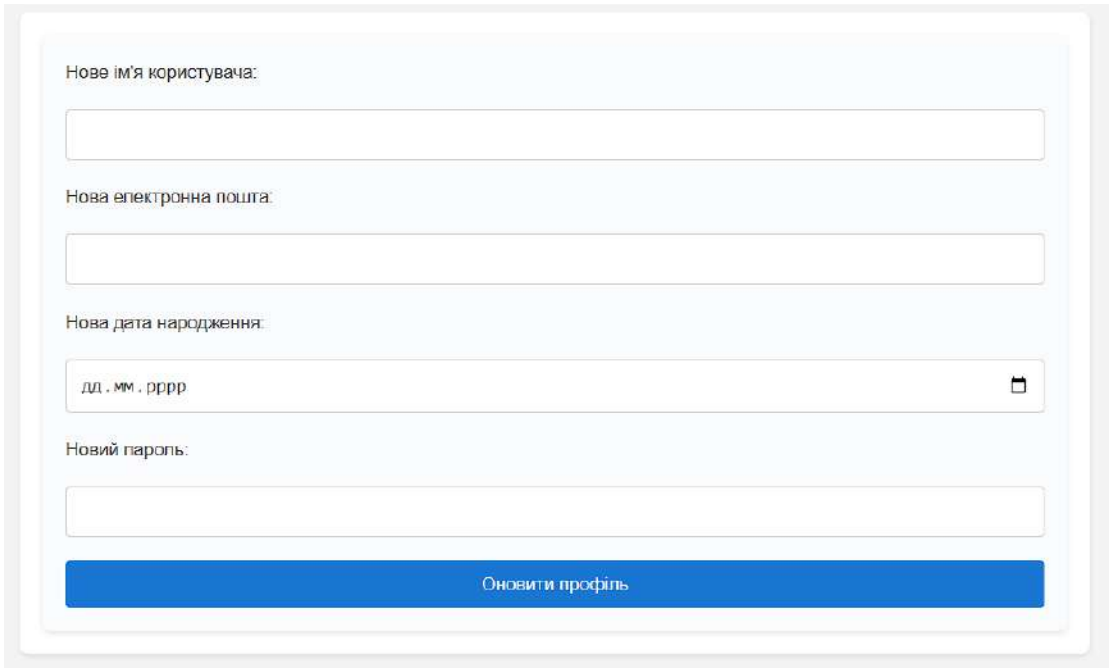

Рис. 4.5. Форма оновлення профілю

Він може ввести нове ім’я користувача, електронну пошту, дату народження та пароль у відповідні поля, після чого натиснути кнопку “Оновити профіль”. Усі введені дані передаються на сервер через захищений

запит. Якщо введена інформація відповідає вимогам, користувач бачить повідомлення про успішне оновлення. Усі поля очищаються автоматично, а дані профілю оновлюються на сервері. У випадку некоректного введення адреса електронної пошти або незаповнені поля користувачу виводиться відповідне повідомлення з поясненням причини.

Серверна частина веб-додатку реалізована на платформі Node.js із використанням фреймворку Express [Додаток В]. У самому кореневому файлі сервера створюється додаток, який обробляє HTTP-запити, налаштовує сесійну підтримку через express-session, а також підключає middleware для обробки тіла запитів: body-parser та вбудований express.json().

Після запуску сервер підключається до бази даних MongoDB за допомогою бібліотеки Mongoose. З’єднання виконується з базою news, яка зберігає користувачів та новини. Для обробки запитів на аутентифікацію використовується модуль Passport, зокрема локальна стратегія (passport-local). Паролі шифруються перед збереженням до бази даних за допомогою bcrypt. Авторизація користувачів у сесії виконується через Passport, який ініціалізується в середовищі Express.

Коли клієнт надсилає POST-запит на /api/auth/register, контролер перевіряє дані реєстрації: мінімальну довжину імені користувача, коректність email, валідність дати народження і достатній вік, а також унікальність логіна й пошти. У разі успіху, пароль хешується, і дані нового користувача зберігаються у колекції users.

При вході через POST-запит на /api/auth/login, дані перевіряються через Passport. Якщо облікові дані вірні, відбувається створення сесії, і користувач отримує доступ до захищених маршрутів. Доступ до таких маршрутів, як наприклад /api/auth/settings, контролюється функцією ensureAuthenticated, яка перевіряє, чи користувач автентифікований.

Маршрут PUT /api/auth/settings дозволяє користувачу змінити свої персональні дані. Сервер перевіряє, які поля були змінені, і для кожного

виконує валідацію. Якщо змінюється пароль, він заново хешується. Успішне оновлення даних завершується повідомленням, яке повертається у відповідь.

Для отримання короткої інформації про поточного користувача є маршрут GET /api/auth/me, який повертає лише ім’я користувача. Для завершення сесії користувач може скористатись маршрутом /api/auth/logout, який завершує його сесію через метод req.logout() і перенаправляє на головну сторінку.

Щодо новин — при запиті на маршрут /api/news, сервер запитує всі записи з колекції новин. Якщо користувач не автентифікований, йому повертаються лише перші три новини. Якщо ж автентифікація пройдена, то у відповідь надсилається повний список усіх новин, відсортованих за датою створення (від новіших до старіших).

У структурі моделей бази даних присутні дві основні схеми — для користувачів (UserSchema) та новин (NewsSchema). У схемі користувача зберігаються ім’я, пароль, email та дата народження, усі поля є обов’язковими. У моделі новин вказані заголовок, зміст, автор, а також автоматично додаються поля створення й оновлення запису завдяки опції timestamps.

У цьому проекті Passport.js використовується як основна система аутентифікації користувачів. Він інтегрований у серверний код через модуль passport і реалізує локальну стратегію аутентифікації — тобто перевірку користувача за ім’ям (логіном) та паролем.

У конфігураційному файлі passport.js створюється та налаштовується локальна стратегія. Вона використовує метод passport-local, у якому вказано, як саме слід перевіряти облікові дані. На кожну спробу входу користувача, функція verifyUser виконує пошук у базі даних MongoDB за іменем користувача. Якщо користувача знайдено, перевіряється відповідність хешованого пароля в базі зі введеним паролем (за допомогою bcrypt.compare). Якщо пароль збігається, аутентифікація вважається успішною і викликається done(null, user).

Після успішного входу Passport виконує серіалізацію користувача — тобто зберігає його унікальний ідентифікатор у сесію за допомогою passport.serializeUser. Коли надходить наступний запит від цього ж користувача, виконується десеріалізація (passport.deserializeUser) — тобто за збереженим у сесії ID витягується повна інформація про користувача з бази.

Passport працює разом з express-session, що дозволяє зберігати стан авторизації користувача між HTTP-запитами. Після входу користувач може переходити між сторінками, залишаючись автентифікованим. Якщо користувач робить запит до захищеного ресурсу, система перевіряє через middleware ensureAuthenticated, чи збережена в сесії авторизація.

4.3. Десктопний додаток на QT

У межах реалізації десктопного клієнта було створено повнофункціональне вікно для роботи з новинами, реалізоване з використанням бібліотеки Qt [Додаток Г]. Це вікно надає інструменти для перегляду, фільтрації, читання, додавання та видалення новин (Рис. 4.6.).

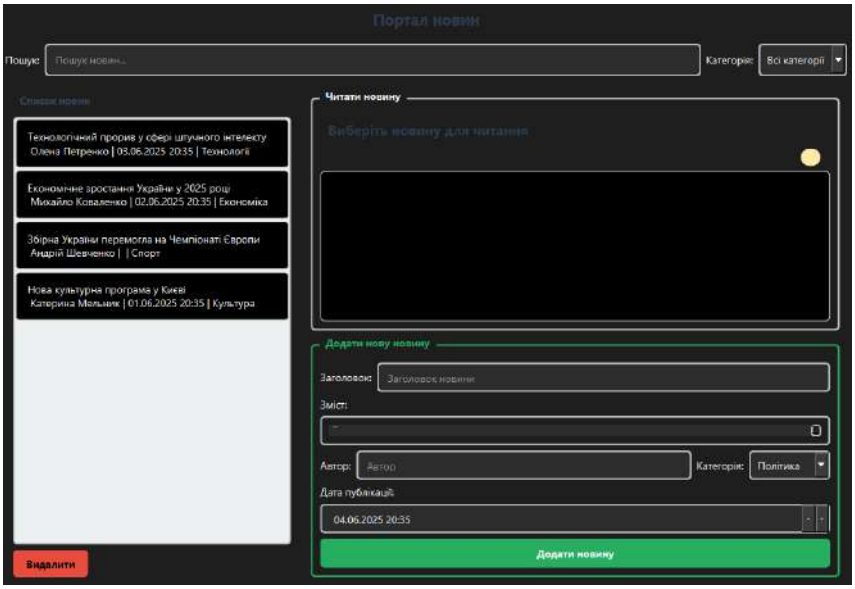

Рис. 4.6. Вікно новин

Верхня частина містить заголовок із назвою додатку, який чітко окреслює його призначення. Під заголовком розміщується панель пошуку та

фільтрації, що дає змогу користувачеві швидко знаходити новини за ключовими словами або обирати їх за тематичною категорією.

Основний простір вікна поділений на дві частини за допомогою горизонтального роздільника (сплітера). Ліва панель містить список новин, які відповідають критеріям пошуку або фільтру. Кожен елемент списку відображає основну інформацію про новину: назву, автора, дату публікації та категорію (Рис. 4.7.).

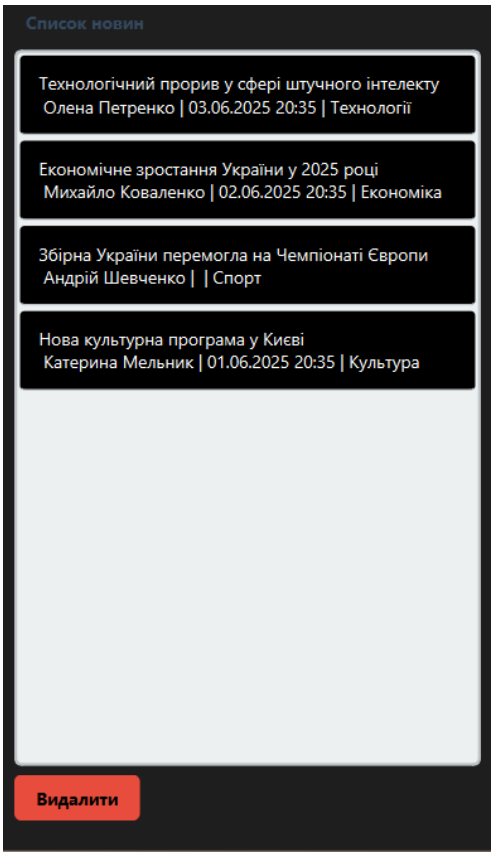

Рис. 4.7. Список новин

Права панель служить для перегляду повної інформації про вибрану новину (Рис. 4.8.). Вона включає заголовок, ім’я автора, дату та категорію, а також повний текст новини. Таким чином, користувач отримує змогу ознайомитись зі змістом, не залишаючи меж одного вікна.

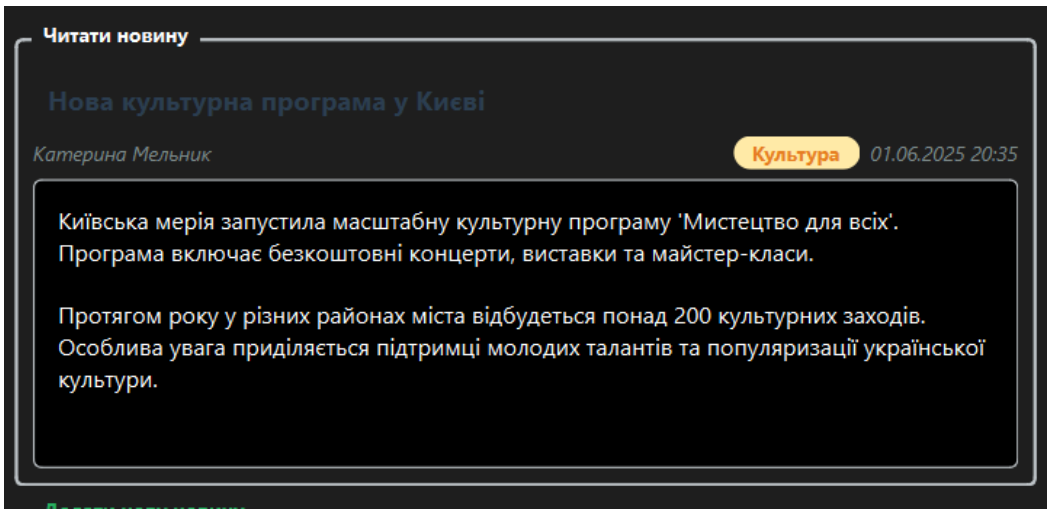

Рис. 4.8. Новина

Додатково, нижня частина правої панелі містить окремий блок для створення нової новини (Рис. 4.9). Цей блок оформлений у вигляді групи елементів введення, що дозволяють ввести заголовок, текст, автора, вибрати категорію та задати дату публікації. Новину можна швидко додати за допомогою відповідної кнопки, що розміщена в кінці форми.

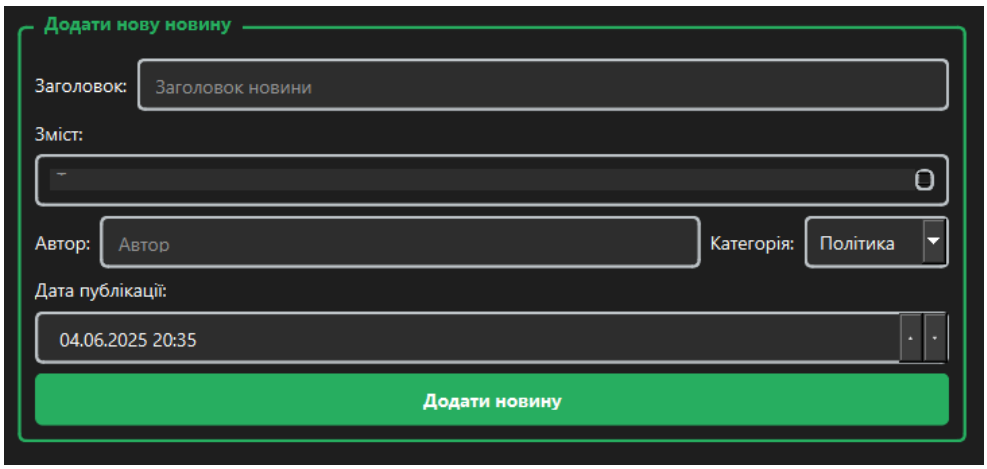

Рис. 4.9. Додати новину

Вся візуальна частина оформлена з використанням стилів CSS-подібного синтаксису, що забезпечує вигляд вікна: застосовано кольорові акценти, плавні переходи кольорів, закруглені кути, контрастність для елементів управління, а також інтуїтивно зрозумілу логіку розміщення елементів.

Десктопний додаток для моделювання складських приміщень — це інтерактивна система, яка дозволяє користувачам створювати, редагувати та

візуалізувати структуру складу з високою точністю. Основою програми є набір об'єктів, таких як стелажі, проходи та товари. Стелажі мають вигляд прямокутників з внутрішньою сегментацією, що імітує полиці (Рис. 4.10.).

Рис. 4.10. Стелаж

Проходи візуалізуються у вигляді відкритих зон зі стрілками напрямку, які вказують, куди може рухатися персонал (Рис. 4.11.).

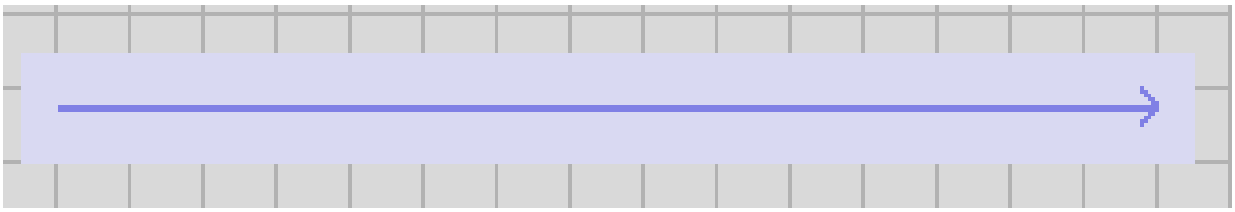

Рис. 4.11. Прохід

Товари позначені хрестоподібними мітками, що дозволяє швидко розпізнавати їх на схемі (Рис. 4.12.).

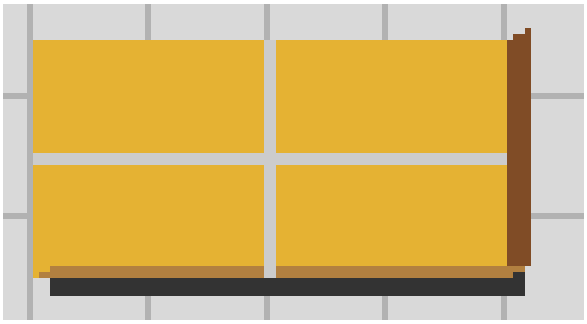

Рис. 4.12. Товар

Режими роботи програми охоплюють додавання об'єктів, вибір, переміщення та зміну розміру. У режимі додавання користувач клацає на потрібне місце на плані, і з'являється прозорий об'єкт — попередній перегляд із кольоровою індикацією, в якому зелений – допустиме розміщення (Рис. 4.13.), червоний – колізія (Рис. 4.14.).

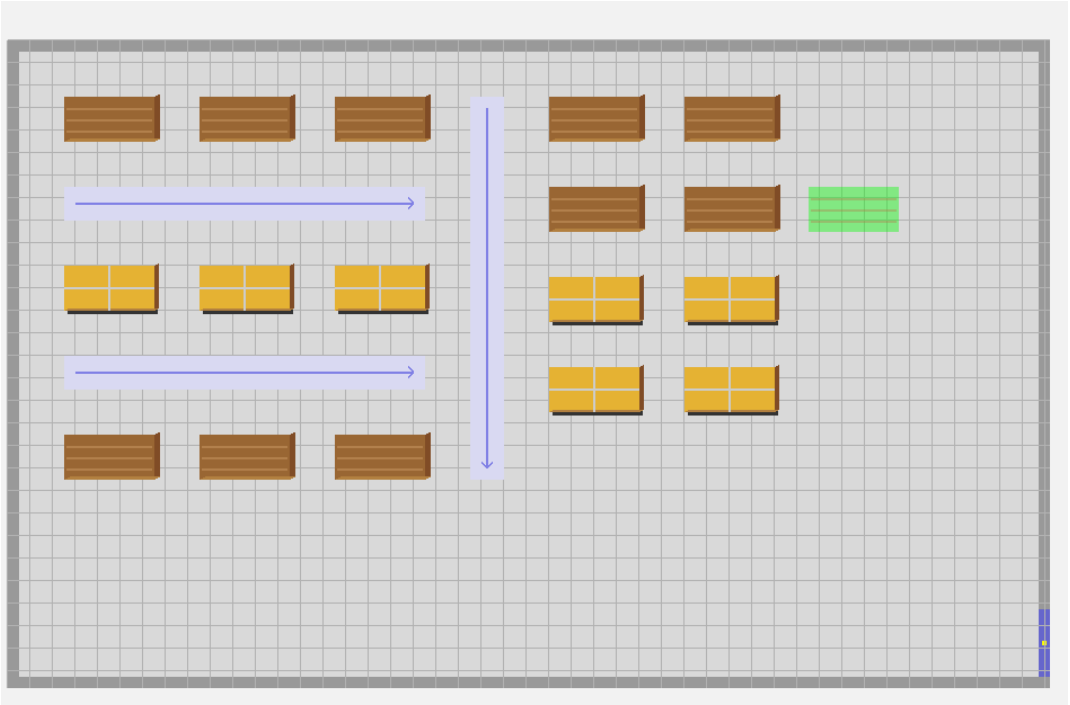

Рис. 4.13. Допустиме розміщення

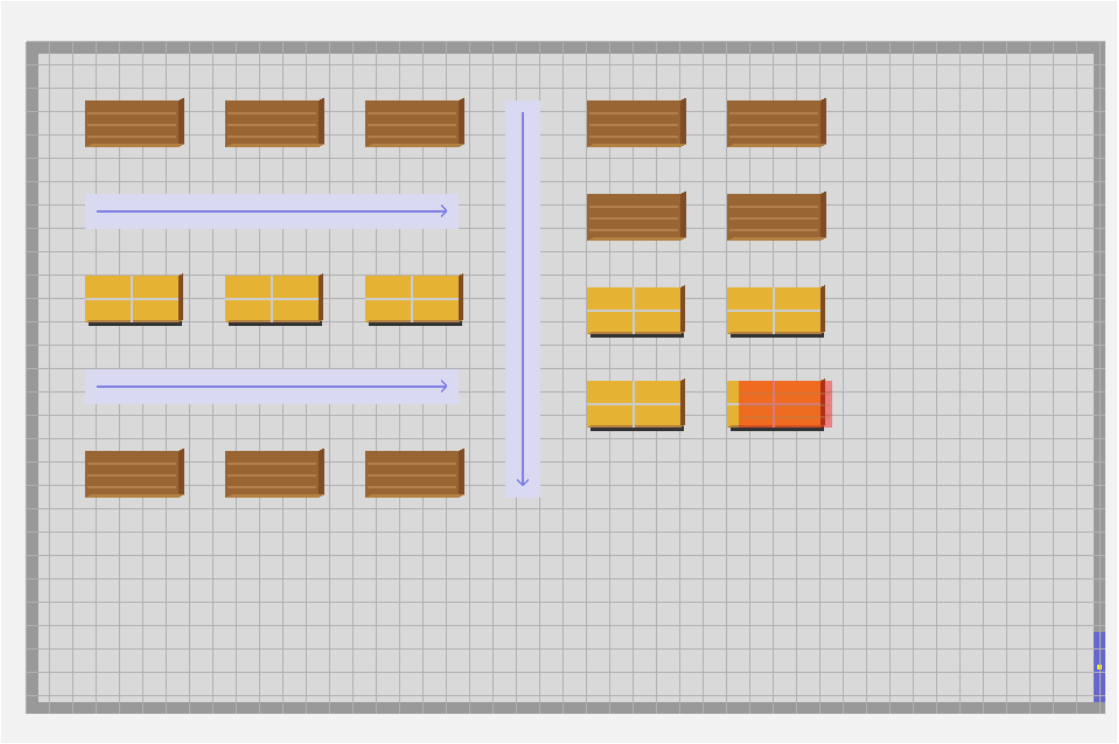

Рис. 4.14. Колізія

Виділений об'єкт обводиться помаранчевою рамкою з кутовими маркерами, які слугують для візуального контролю (Рис. 4.15.).

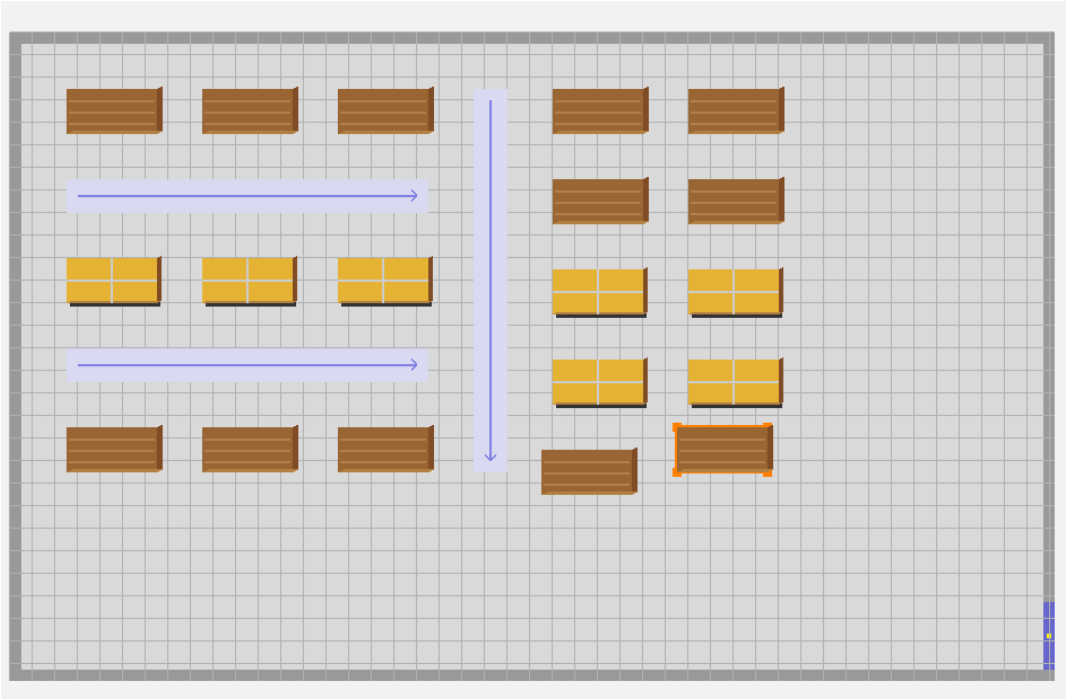

Рис. 4.15. Виділений об’єкт

Управління мишею та панель інструментів забезпечують максимальну зручність. Всі кнопки режимів роботи та типів об’єктів для додавання згруповані у верхній частині вікна. Активна кнопка підсвічується зеленим кольором, що полегшує орієнтацію. Лівий клік виконує додавання або виділення, а перетягування — переміщення чи зміну розміру, залежно від активного режиму.

Програма підтримує клавіатурні скорочення, які значно пришвидшують роботу: клавіші 1, 2, 3 активують додавання відповідно стелажів, проходів і товарів. Клавіші S, M, R переключають між режимами вибору, переміщення та масштабування. Окрім того, Delete видаляє обраний об'єкт, а Space дозволяє циклічно змінювати тип об'єкта. Для точного позиціонування передбачено стрілки на клавіатурі, які зміщують елемент на 20 пікселів по сітці.

Серед візуальних можливостей варто відзначити координатну сітку, до якої всі об'єкти автоматично прив'язуються, забезпечуючи рівномірність та логіку розміщення. Під час переміщення або зміни розміру з’являються направляючі та пунктирні лінії для кращої орієнтації.

Архітектура складського простору включає стіни, вхідні двері, загальну підлогу та робочу зону, в межах якої дозволено розміщення. Стартовий макет

автоматично генерується при запуску програми: формуються ряди стелажів, проходи між ними та початкові товари, що дозволяє одразу працювати з готовим середовищем (Рис. 4.16.).

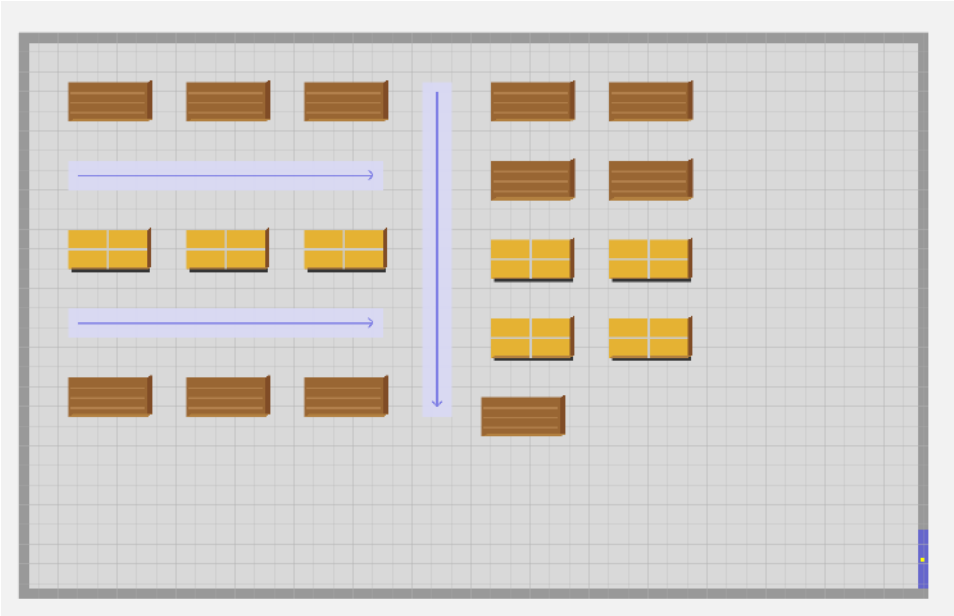

Рис. 4.16. Стартовий вигляд складу

У технічному плані додаток реалізує патерн Command, який дозволяє зберігати повну історію дій користувача (Рис. 4.17.).

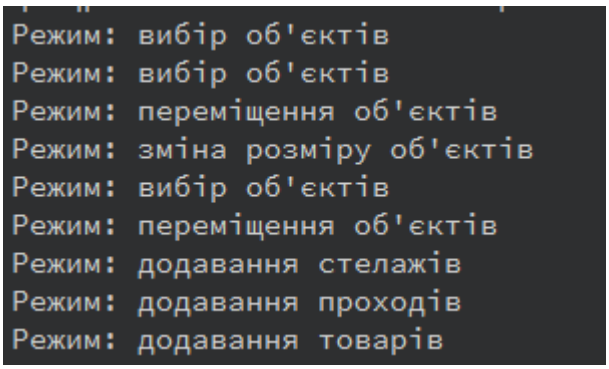

Рис. 4.17. Історія дій

Це дає змогу використовувати Ctrl+Z та Ctrl+Y для скасування та повторення дій. Крім того, реалізована перевірка колізій, яка не дозволяє об'єктам накладатися або виходити за межі робочої зони. Графічна частина працює на базі OpenGL, забезпечуючи плавну анімацію, ефективну візуалізацію та стабільну частоту оновлення до 60 кадрів на секунду.

У нижній частині вікна розташована інформаційна панель стану (Рис. 4.18.), яка в режимі реального часу відображає поточний режим роботи

(додавання, вибір, переміщення, зміна розміру) та тип об’єкта, що буде розміщено на схемі (стелаж, прохід або товар).

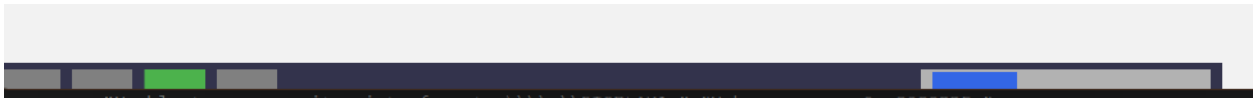

Рис. 4.18. Панель стану

Окрім візуального редактора, програма має спеціальне вікно для роботи з базою даних виробів, яке викликається через меню. Це діалогове вікно надає зручний інтерфейс для перегляду, додавання, редагування та видалення записів у таблиці виробів на підприємстві, що містить код та назву кожного виробу (Рис. 4.19.).

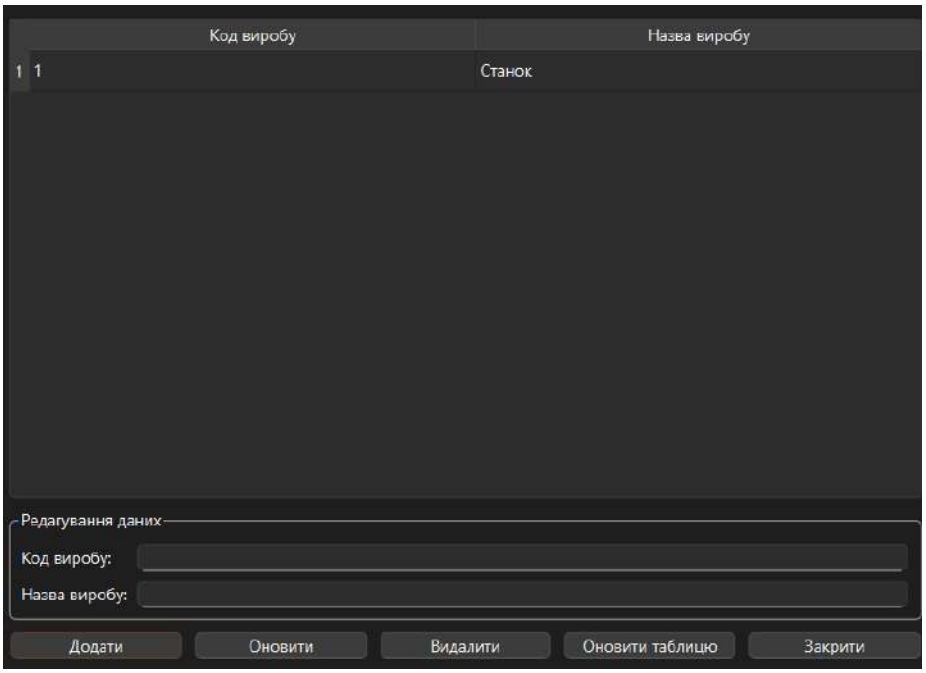

Рис. 4.19. Робота з виробами пыдприэмства

У верхній частині вікна розміщено таблицю записів, яка автоматично підтягує дані з бази. Рядки можна виділяти для редагування або видалення. Нижче знаходиться форма редагування, де користувач вводить або змінює код і назву виробу. Всі дії супроводжуються повідомленнями про успіх або помилки, наприклад — спроба додати дубльований код або залишити порожнє поле.

1) Під формою розміщена панель кнопок, яка включає:
2) Додати — для створення нового запису,
3) Оновити — для редагування вибраного запису,

4) Видалити — для видалення вибраного запису,
5) Оновити таблицю — щоб повторно завантажити дані,
6) Закрити — для виходу з діалогового вікна.

Висновки до розділу 4

У цьому розділі було реалізовано два ключові компоненти інформаційної системи: вебсайт для онлайн-доступу користувачів та десктопний застосунок для роботи зі складською картою й базою даних.

ВИСНОВКИ

У даній дипломній роботі було виконано повний цикл розробки інформаційної системи, що поєднує функціонал вебзастосунку та десктопного програмного забезпечення для обліку та візуального управління елементами складської інфраструктури підприємства. Проєкт охоплює всі етапи від постановки задачі й аналізу предметної області до безпосередньої реалізації та впровадження двох основних компонентів системи — вебпорталу та десктопного додатку.

У вступі було сформульовано мету та завдання дослідження — створення інтегрованої системи управління об'єктами складу з функціями візуального розміщення, редагування даних та підтримки користувацької взаємодії.

У розділі 1 розглянуто суть задачі, описано ключові процеси, які система повинна підтримувати: облік виробів, їх графічне розташування, взаємодія з базою даних. Проведено аналіз предметної області, що дозволило виділити основні об'єкти моделювання та виявити вимоги до інтерфейсу, структури даних і функціоналу. Особливу увагу приділено обґрунтуванню вибору інструментів: для десктопної частини було обрано фреймворк Qt завдяки його підтримці графіки (OpenGL), а для вебчастини — стек PHP + MySQL для реалізації інтерфейсу доступу до бази даних через браузер.

У розділі 2 представлено архітектуру системи, зокрема структуру взаємодії клієнтської частини з серверною та з базою даних. Було розроблено макети та описано інтерфейс користувача: у вебверсії зроблено акцент на зручності перегляду таблиць та довідкової інформації, у десктопній — на гнучкому управлінні об'єктами на плані складу.

Розділ 3 присвячено розробці бази даних, що лежить в основі всієї системи. Побудовано інфологічну модель, на основі якої сформовано модель "сутність–зв’язок", з подальшою реалізацією структури таблиць у СУБД SQLite. Базу даних оптимізовано для спільного використання обома

частинами проєкту. Враховано зв’язки між таблицями, обмеження цілісності та можливість масштабування.

У розділі 4 безпосередньо реалізовано вебзастосунок і десктопний додаток. У вебінтерфейсі реалізовано перегляд та редагування основних довідників: «Виріб», «Матеріали», «Виробнича програма». Окрему увагу приділено реалізації серверної логіки — обробці запитів, захисту даних та коректному оновленню таблиць. У десктопній версії створено функціонал візуального редагування плану складу: користувач може додавати, переміщувати або видаляти об’єкти, змінювати тип об'єкта, перемикатися між режимами, що значно спрощує управління реальним простором складу. Окремо реалізовано нижню інформаційну панель, яка відображає поточний режим роботи та тип об’єкта для розміщення.

У результаті виконання роботи було створено двокомпонентну програмну систему, що дозволяє:

1) ефективно керувати інформацією про складські об'єкти та вироби;
2) візуально моделювати розташування елементів складу;
3) швидко взаємодіяти з базою даних через як графічний, так і вебінтерфейс;
4) підтримувати введення, редагування та видалення записів з урахуванням зв’язків та обмежень цілісності.

Розроблена система відповідає вимогам предметної області, має зручний інтерфейс та може бути впроваджена на реальному підприємстві з мінімальними витратами на адаптацію. Крім того, вона створює основу для майбутнього розвитку, зокрема інтеграції зі складськими системами реального часу, автоматизованими роботами чи мобільними застосунками.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гук К.Р. *Проектування та розробка баз даних: навчальний посібник*. – Київ: Каравела, 2018. – 248 с.
2. Bashynskyi A., Fesenko G. *Modern Web Development with React and Node.js*. – К.: Наука, 2021. – 310 с.
3. Литвин В.О. *Основи створення баз даних*. – Львів: Видавництво ЛНУ, 2020. – 192 с.
4. Бондаренко І.І. *Системи управління базами даних: навч. посібник*. – К.: КНЕУ, 2017. – 246 с.
5. Sommerville I. *Software Engineering*. – 10th ed. – Pearson Education, 2016. – 832 p.
6. Freeman E., Robson E. *Head First Design Patterns*. – O’Reilly Media, 2021. – 694 p.
7. Grinberg M. *Flask Web Development: Developing Web Applications with Python*. – O’Reilly Media, 2018. – 258 p.
8. Welling L., Thomson L. *PHP and MySQL Web Development*. – 5th ed. – Addison-Wesley, 2016. – 672 p.
9. Reas C., Fry B. *Processing: A Programming Handbook for Visual Designers and Artists*. – MIT Press, 2014. – 672 p.
10. Eckel B. *Thinking in C++. Vol. 1: Introduction to Standard C++*. – 2nd ed. – Prentice Hall, 2000. – 800 p.
11. Stroustrup B. *The C++ Programming Language*. – 4th ed. – Addison-Wesley, 2013. – 1376 p.
12. Бабаєв Д.С. *Основи розробки клієнт-серверних додатків*. – Дніпро: Університет «Дніпровська політехніка», 2022. – 210 с.
13. Fain Y. *Java Programming for Beginners*. – Independently Published, 2020. – 420 p.
14. Zakas N.C. *Understanding ECMAScript 6: The Definitive Guide for JavaScript Developers*. – No Starch Press, 2016. – 352 p.

15. Souders S. *High Performance Web Sites: Essential Knowledge for Front-End Engineers*. – O'Reilly Media, 2007. – 168 p.
16. Williams N. *Pro SQLite*. – Apress, 2005. – 408 p.
17. Blanchette J., Summerfield M. *C++ GUI Programming with Qt 4*. – 2nd ed. – Prentice Hall, 2008. – 688 p.
18. McLaughlin B.D. *Head First Java*. – 3rd ed. – O'Reilly Media, 2022. – 720 p.
19. McFarland D. *CSS: The Missing Manual*. – 4th ed. – O'Reilly Media, 2020. – 630 p.
20. Duckett J. *HTML and CSS: Design and Build Websites*. – Wiley, 2011. – 490 p.
21. Meier R. *Professional Android*. – 4th ed. – Wrox, 2018. – 816 p.
22. Marcotte E. *Responsive Web Design*. – A Book Apart, 2011. – 150 p.
23. Černý M. *Developing Modern Desktop Applications with Qt*. – Springer, 2023. – 290 p.
24. Leach K. *Modern CMake for C++*. – Leanpub, 2021. – 220 p.
25. Sidorenko A. *Practical Database Design with SQLite and Qt*. – Independently published, 2020. – 268 p.

ДОДАТКИ

Додаток А

Html-файли сайту

Файл індексу

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta    name="viewport"    content="width=device-width,
initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Новини</title>
</head>
<body>
    <header>
        <p id="welcome-message">Для доступу для всіх новин
увійдіть у профіль</p>
        <a href="/login.html" id="auth-link">Вхід</a>
        <a      id="settings-link"      href="settings.html"
style="display: none;">Налаштування</a>
    </header>
    <main>
        <div id="news-container"></div>

    <script>
        async function fetchNews() {
            const response = await fetch('/api/news');
            if (!response.ok) {
                throw  new  Error('Не  вдалося  завантажити
новини');
            }

            const news = await response.json();
            const              newsContainer              =
document.getElementById('news-container');
            newsContainer.innerHTML = '';

            news.forEach(article => {
                const             articleDiv             =
document.createElement('div');
                articleDiv.className = 'news-article';
                articleDiv.innerHTML = `
                    <h2>${article.title}</h2>
                    <p>${article.content}</p>
                    <small>Author:    ${article.author    ||
'Невідомо'}</small> |
                    <small>Posted:                        ${new
Date(article.createdAt).toLocaleDateString()                ||
'Невідомо'}</small>
```

```
                    `;
                    newsContainer.appendChild(articleDiv);
                });
            }
            fetchNews();

            async function checkAuth() {
                const response = await fetch('/api/auth/me');
                if (!response.ok) {
                    throw new Error('Користувач не
авторизований');
                }
                const data = await response.json();
                const authLink = document.getElementById('auth-
link');
                const settingsLink =
document.getElementById('settings-link');
                const welcomeMessage =
document.getElementById('welcome-message');

                authLink.textContent = 'Вихід';
                authLink.href = '/api/auth/logout';
                settingsLink.style.display = 'inline-block';
                welcomeMessage.textContent = `Ласкаво просимо,
${data.username}`;
            }
            checkAuth();
        </script>
        </main>
    </body>
    </html>
```

Файл логіну

```
    <!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <link rel="stylesheet" href="styles.css">
        <title>Вхід</title>
    </head>
    <body>
        <header>
            <h1>Вхід на сайт</h1>
            <a href="/index.html">Головна</a>
        </header>
```

Продовження Додатку А

```
        <main>
            <form id="login-form">
                <label for="username">Ім'я користувача:</label>
                <input type="text" id="username" name="username"
required>
                <label for="password">Пароль:</label>
                <input       type="password"       id="password"
name="password" required>
                <button type="submit">Увійти</button>
            </form>
            <p>Ще       не       маєте       акаунту?       <a
href="/register.html">Зареєструватися</a></p>
        </main>

        <script>
            document.getElementById('login-
form').addEventListener('submit', async (e) => {
                e.preventDefault();
                const                username                =
document.getElementById('username').value;
                const                password                =
document.getElementById('password').value;

                const response = await fetch('/api/auth/login',
{
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify({ username, password })
                });

                const data = await response.json();

                if (response.ok) {
                    alert('Вхід успішний');
                    window.location.href = '/';
                } else {
                    alert(data.message || 'Невірний логін або
пароль');
                }
            });
        </script>
    </body>
    </html>
```

Продовження Додатку А

Файл реєстрації

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta    name="viewport"    content="width=device-width,
initial-scale=1.0">
    <title>Реєстрація</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Реєстрація</h1>
        <a href="/index.html">Головна</a>
    </header>

    <main>
        <form id="register-form">
            <label for="username">Ім'я користувача:</label>
            <input type="text" id="username" name="username"
required>

            <label for="password">Пароль:</label>
            <input       type="password"       id="password"
name="password" required>

            <label for="email">Електронна пошта:</label>
            <input   type="email"   id="email"   name="email"
required>

            <label                    for="dateOfBirth">Дата
народження:</label>
            <input       type="date"       id="dateOfBirth"
name="dateOfBirth" required>

            <button type="submit">Зареєструватися</button>
        </form>
    </main>

    <script>
        document.getElementById('register-
form').addEventListener('submit', async (e) => {
            e.preventDefault();
            const                username                =
document.getElementById('username').value;
            const                password                =
document.getElementById('password').value;
```

```
                const email = document.getElementById('email').value;
                const dateOfBirth = document.getElementById('dateOfBirth').value;

                const response = await fetch('/api/auth/register', {
                    method: 'POST',
                    headers: { 'Content-Type': 'application/json' },
                    body: JSON.stringify({ username, password, email, dateOfBirth })
                });

                const data = await response.json();
                alert(data.message);
                if (response.ok) {
                    window.location.href = '/login.html';
                }
            });
        </script>
    </body>
    </html>
```

Файл редагування профілю

```
    <!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Редагування профілю</title>
        <link rel="stylesheet" href="styles.css">
    </head>
    <body>
        <header>
            <h1>Налаштування</h1>
            <a href="/index.html">Головна</a>
        </header>

        <main>
            <form id="settings-form">
                <label for="username">Нове ім'я користувача:</label>
                <input type="text" id="username" name="username">

                <label for="email">Нова електронна пошта:</label>
```

Продовження Додатку А

```
                <input type="email" id="email" name="email">

                <label for="dateOfBirth">Нова дата
народження:</label>
                <input type="date" id="dateOfBirth"
name="dateOfBirth">

                <label for="password">Новий пароль:</label>
                <input type="password" id="password"
name="password">

                <button type="submit">Оновити профіль</button>
            </form>
        </main>

        <script>
            document.getElementById('settings-
form').addEventListener('submit', async (e) => {
                e.preventDefault();

                const                 username                =
document.getElementById('username').value;
                const                  email                  =
document.getElementById('email').value;
                const               dateOfBirth               =
document.getElementById('dateOfBirth').value;
                const                 password                =
document.getElementById('password').value;
                const token = localStorage.getItem('token');

                const         response         =         await
fetch('/api/auth/settings', {
                    method: 'PUT',
                    headers: {
                        'Content-Type': 'application/json',
                        'Authorization': 'Bearer ' + token
                    },
                    body:   JSON.stringify({   username,   email,
dateOfBirth, password })
                });

                if (!response.ok) {
                    const error = await response.json();
                    alert(error.message  ||  'Помилка  оновлення
профілю');
                    return;
                }

                const data = await response.json();
                alert(data.message);
```

```
                document.getElementById('username').value = '';
                document.getElementById('email').value = '';
                document.getElementById('dateOfBirth').value   =
'';
                document.getElementById('password').value = '';
            });
        </script>
    </body>
    </html>
```

Додаток Б

CSS-файл сайту

```
body {
    margin: 0;
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    color: #444;
    line-height: 1.6;
}

header {
    background-color: #1976d2;
    color: #fff;
    padding: 10px 20px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

header a {
    color: #ddd;
    text-decoration: none;
    margin-left: 15px;
    font-size: 16px;
}

header a:hover {
    color: #fff;
}

h1 {
    color: #1976d2;
    margin-bottom: 20px;
}

main {
    max-width: 900px;
    margin: 20px auto;
    padding: 20px;
    background-color: #fff;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

form {
    display: flex;
    flex-direction: column;
    gap: 20px;
    padding: 20px;
    background-color: #fafbfc;
    border-radius: 8px;
```

Продовження Додатку Б

```
        box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    }

    form input {
        padding: 12px;
        font-size: 16px;
        border: 1px solid #ccc;
        border-radius: 4px;
        box-sizing: border-box;
        background-color: #fff;
        color: #333;
        transition: all 0.3s ease-in-out;
    }

    form input:focus {
        border-color: #1976d2;
        outline: none;
        box-shadow: 0 0 5px rgba(25, 118, 210, 0.5);
    }

    form button {
        padding: 12px;
        font-size: 16px;
        background-color: #1976d2;
        color: white;
        border: none;
        border-radius: 4px;
        cursor: pointer;
        transition: background-color 0.3s ease, transform 0.3s
ease;
    }

    form button:hover {
        background-color: #1565c0;
        transform: translateY(-3px);
    }

    form button:active {
        transform: translateY(1px);
    }

    main a {
        color: #1976d2;
        text-decoration: none;
        font-weight: bold;
    }

    main a:hover {
        color: #1565c0;
    }
```

Продовження Додатку Б

```
#news-container {
    display: flex;
    flex-direction: column;
    gap: 20px;
}

.news-article {
    padding: 20px;
    background: linear-gradient(135deg, #0e51b6, #05417c);
    border: 1px solid #0c3c72;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.05);
    color: #fff;
}

.news-article h2 {
    font-size: 28px;
    margin-bottom: 10px;
    color: #fff;
}

.news-article p {
    margin: 0 0 15px;
    font-size: 16px;
    line-height: 1.5;
    color: #ffffff;
}

.news-article small {
    font-size: 14px;
    color: #e0e0e0;
}

.news-article .read-more {
    color: #e1f5fe;
    text-decoration: none;
    font-weight: bold;
    margin-top: 10px;
}

.news-article .read-more:hover {
    color: #b3e5fc;
}
```

Додаток В

Роутери сайту

Роутер аутентифікації

```
const User = require('../models/User');
const bcrypt = require('bcrypt');
const hashPassword = require('../utils/hashPassword');
const passport = require('passport');
const validator = require('validator');

module.exports = {
    register: async (req, res) => {
        const { username, password, email, fullName, dateOfBirth } = req.body;

        if (!username || username.length < 3) {
            return res.status(400).json({ message: 'Ім\'я користувача повинно бути не менше 3 символів' });
        }

        if (!email || !validator.isEmail(email)) {
            return res.status(400).json({ message: 'Невірний формат електронної пошти' });
        }

        if (!password || password.length < 6) {
            return res.status(400).json({ message: 'Пароль повинен бути не менше 6 символів' });
        }

        if (!dateOfBirth || !Date.parse(dateOfBirth)) {
            return res.status(400).json({ message: 'Невірний формат дати народження' });
        }

        const birthDate = new Date(dateOfBirth);
        const age = new Date().getFullYear() - birthDate.getFullYear();
        if (age < 16) {
            return res.status(400).json({ message: 'Для реєстрації потрібно бути старше 16 років' });
        }

        const existingUser = await User.findOne({ username });
        if (existingUser) {
            return res.status(400).json({ message: 'Користувач із таким ім\'ям вже існує' });
        }

        const existingEmail = await User.findOne({ email });
```

```
            if (existingEmail) {
                return      res.status(400).json({      message:
'Користувач із таким email вже існує' });
            }

            const hashedPassword = await hashPassword(password);

            const newUser = new User({
                username,
                password: hashedPassword,
                email,
                dateOfBirth
            });
            await newUser.save();

            res.status(201).json({ message: 'Користувач успішно
зареєстрований!' });
        },

        login: (req, res, next) => {
            passport.authenticate('local', (err, user, info) =>
{
                if (err) {
                    console.error('Помилка     аутентифікації:',
err);
                    return    res.status(500).json({    message:
'Внутрішня помилка сервера' });
                }
                if (!user) {
                    return    res.status(400).json({    message:
info.message || 'Невірний логін або пароль' });
                }

                req.logIn(user, (err) => {
                    if (err) {
                        console.error('Помилка входу:', err);
                        return  res.status(500).json({  message:
'Не вдалося увійти в систему' });
                    }
                    return res.json({ message: 'Успішний вхід',
user: { id: user._id, username: user.username } });
                });
            })(req, res, next);
        },

        updateProfile: async (req, res) => {
            const { username, email, dateOfBirth, password } =
req.body;

            const user = await User.findById(req.user._id);
            if (!user) {
```

```
                return      res.status(404).json({      message:
'Користувача не знайдено' });
            }

            if (username && username !== user.username) {
                if (username.length < 3) {
                    return    res.status(400).json({    message:
'Ім\'я користувача повинно бути не менше 3 символів' });
                }
                const   existingUser   =   await   User.findOne({
username });
                if (existingUser) {
                    return res.status(400).json({ message: 'Цей
нікнейм вже використовується' });
                }
                user.username = username;
            }

            if (email && email !== user.email) {
                if (!validator.isEmail(email)) {
                    return    res.status(400).json({    message:
'Невірний формат електронної пошти' });
                }
                const existingEmail = await User.findOne({ email
});
                if (existingEmail) {
                    return res.status(400).json({ message: 'Цей
email вже використовується' });
                }
                user.email = email;
            }

            if (dateOfBirth) {
                const birthDate = new Date(dateOfBirth);
                const   age   =   new   Date().getFullYear()   -
birthDate.getFullYear();
                if (age < 16) {
                    return res.status(400).json({ message: 'Вік
повинен бути не менше 16 років' });
                }
                user.dateOfBirth = dateOfBirth;
            }

            if (password && password.length < 6) {
                return  res.status(400).json({  message:  'Пароль
повинен бути не менше 6 символів' });
            }

            if (password) {
                user.password = await hashPassword(password);
            }
```

```
            await user.save();

            res.json({ message: 'Профіль оновлено успішно' });
        },

        getUserInfo: (req, res) => {
            res.json({ username: req.user.username });
        },

        logout: (req, res) => {
            req.logout(err => {
                if (err) {
                    console.error('Помилка виходу:', err);
                    return  res.status(500).json({  message:  'Не
вдалося вийти' });
                }
                res.redirect('/');
            });
        }
    };
```

Роутер новин

```
    const express = require('express');
    const router = express.Router();
    const News = require('../models/News');

    router.get('/', async (req, res) => {
        const news = await News.find().sort({ createdAt: -1 });
        const  limitedNews  =  req.isAuthenticated()  ?  news  :
news.slice(0, 3);
        res.json(limitedNews);
    });

    module.exports = router;
```

Додаток Г

Десктопний додаток на QT

```
#include "newsdialog.h"
#include <QMessageBox>
#include <QDateTime>
#include <QHeaderView>

NewsDialog::NewsDialog(QWidget *parent)
    : QDialog(parent)
{
    setWindowTitle("Портал новин");
    setMinimumSize(900, 600);
    resize(1000, 700);

    setupUI();
    populateNewsList();
}

void NewsDialog::showDialog(QWidget *parent) {
    NewsDialog *dialog = new NewsDialog(parent);
    dialog->setAttribute(Qt::WA_DeleteOnClose);
    dialog->show();
}

void NewsDialog::setupUI() {
    QVBoxLayout *mainLayout = new QVBoxLayout(this);

    QLabel *headerLabel = new QLabel("Портал новин");
    headerLabel->setStyleSheet("font-size: 18px; font-
weight: bold; color: #2c3e50; padding: 10px;");
    headerLabel->setAlignment(Qt::AlignCenter);
    mainLayout->addWidget(headerLabel);

    QHBoxLayout *searchLayout = new QHBoxLayout();
    searchEdit = new QLineEdit();
    searchEdit->setPlaceholderText("Пошук новин...");
    searchEdit->setStyleSheet("padding: 8px; border: 2px
solid #bdc3c7; border-radius: 5px;");
    connect(searchEdit, &QLineEdit::textChanged, this,
&NewsDialog::onSearchChanged);

    filterCombo = new QComboBox();
    filterCombo->addItems({"Всі категорії", "Політика",
"Спорт", "Технології", "Економіка", "Культура"});
    filterCombo->setStyleSheet("padding: 8px; border: 2px
solid #bdc3c7; border-radius: 5px;");
    connect(filterCombo,
QOverload<int>::of(&QComboBox::currentIndexChanged), this,
&NewsDialog::onFilterChanged);

    searchLayout->addWidget(new QLabel("Пошук:"));
```

```
        searchLayout->addWidget(searchEdit);
        searchLayout->addWidget(new QLabel("Категорія:"));
        searchLayout->addWidget(filterCombo);
        mainLayout->addLayout(searchLayout);

        mainSplitter = new QSplitter(Qt::Horizontal);

        QWidget *leftPanel = new QWidget();
        QVBoxLayout *leftLayout = new QVBoxLayout(leftPanel);

        QLabel *newsListLabel = new QLabel("Список новин");
        newsListLabel->setStyleSheet("font-weight: bold; color:
#34495e; padding: 5px;");
        leftLayout->addWidget(newsListLabel);

        newsList = new QListWidget();
        newsList->setStyleSheet(
            "QListWidget {"
            "    border: 2px solid #bdc3c7;"
            "    border-radius: 5px;"
            "    background-color: #ecf0f1;"
            "}"
            "QListWidget::item {"
            "    padding: 10px;"
            "    border-bottom: 1px solid #bdc3c7;"
            "    background-color: black;"
            "    margin: 2px;"
            "    border-radius: 3px;"
            "}"
            "QListWidget::item:selected {"
            "    background-color: #3498db;"
            "    color: black;"
            "}"
            "QListWidget::item:hover {"
            "    background-color: #e8f4f8;"
            "}"
            );
        connect(newsList, &QListWidget::currentRowChanged, this,
&NewsDialog::onNewsSelected);
        leftLayout->addWidget(newsList);

        QHBoxLayout *buttonLayout = new QHBoxLayout();
        deleteButton = new QPushButton("Видалити");
        deleteButton->setStyleSheet(
            "QPushButton {"
            "    background-color: #e74c3c;"
            "    color: black;"
            "    border: none;"
            "    padding: 8px 15px;"
            "    border-radius: 5px;"
            "    font-weight: bold;"
```

```
            "}"
            "QPushButton:hover {"
            "    background-color: #c0392b;"
            "}"
            );
        connect(deleteButton,    &QPushButton::clicked,    this,
&NewsDialog::onDeleteNews);
        buttonLayout->addWidget(deleteButton);
        buttonLayout->addStretch();
        leftLayout->addLayout(buttonLayout);

        mainSplitter->addWidget(leftPanel);

        QWidget *rightPanel = new QWidget();
        QVBoxLayout *rightLayout = new QVBoxLayout(rightPanel);

        QGroupBox *newsGroup = new QGroupBox("Читати новину");
        newsGroup->setStyleSheet(
            "QGroupBox {"
            "    font-weight: bold;"
            "    border: 2px solid #bdc3c7;"
            "    border-radius: 5px;"
            "    margin-top: 10px;"
            "    padding-top: 10px;"
            "}"
            "QGroupBox::title {"
            "    subcontrol-origin: margin;"
            "    left: 10px;"
            "    padding: 0 5px 0 5px;"
            "}"
            );
        QVBoxLayout *newsLayout = new QVBoxLayout(newsGroup);

        newsTitle = new QLabel("Виберіть новину для читання");
        newsTitle->setStyleSheet("font-size:  16px;  font-weight:
bold; color: #2c3e50; padding: 5px;");
        newsTitle->setWordWrap(true);

        QHBoxLayout *metaLayout = new QHBoxLayout();
        newsAuthor = new QLabel();
        newsAuthor->setStyleSheet("color:  #7f8c8d;  font-style:
italic;");
        newsDate = new QLabel();
        newsDate->setStyleSheet("color:   #7f8c8d;   font-style:
italic;");
        newsCategory = new QLabel();
        newsCategory->setStyleSheet("color:    #e67e22;    font-
weight: bold; background-color: #ffeaa7; padding: 2px 8px; border-
radius: 10px;");

        metaLayout->addWidget(newsAuthor);
```

```
        metaLayout->addStretch();
        metaLayout->addWidget(newsCategory);
        metaLayout->addWidget(newsDate);

        newsContent = new QTextEdit();
        newsContent->setReadOnly(true);
        newsContent->setStyleSheet(
            "QTextEdit {"
            "    border: 1px solid #bdc3c7;"
            "    border-radius: 5px;"
            "    background-color: black;"
            "    padding: 10px;"
            "    font-size: 14px;"
            "    line-height: 1.5;"
            "}"
            );

        newsLayout->addWidget(newsTitle);
        newsLayout->addLayout(metaLayout);
        newsLayout->addWidget(newsContent);
        rightLayout->addWidget(newsGroup);

        addNewsGroup = new QGroupBox("Додати нову новину");
        addNewsGroup->setStyleSheet(
            "QGroupBox {"
            "    font-weight: bold;"
            "    border: 2px solid #27ae60;"
            "    border-radius: 5px;"
            "    margin-top: 10px;"
            "    padding-top: 10px;"
            "}"
            "QGroupBox::title {"
            "    subcontrol-origin: margin;"
            "    left: 10px;"
            "    padding: 0 5px 0 5px;"
            "    color: #27ae60;"
            "}"
            );
        QVBoxLayout *addLayout = new QVBoxLayout(addNewsGroup);

        titleEdit = new QLineEdit();
        titleEdit->setPlaceholderText("Заголовок новини");
        titleEdit->setStyleSheet("padding: 8px; border: 2px solid
#bdc3c7; border-radius: 5px;");

        contentEdit = new QTextEdit();
        contentEdit->setPlaceholderText("Текст новини...");
        contentEdit->setMaximumHeight(100);
        contentEdit->setStyleSheet("padding:  8px;  border:  2px
solid #bdc3c7; border-radius: 5px;");
```

```
        authorEdit = new QLineEdit();
        authorEdit->setPlaceholderText("Автор");
        authorEdit->setStyleSheet("padding:  8px;  border:  2px
solid #bdc3c7; border-radius: 5px;");

        categoryCombo = new QComboBox();
        categoryCombo->addItems({"Політика",           "Спорт",
"Технології", "Економіка", "Культура"});
        categoryCombo->setStyleSheet("padding: 8px; border: 2px
solid #bdc3c7; border-radius: 5px;");

        dateEdit                      =                     new
QDateTimeEdit(QDateTime::currentDateTime());
        dateEdit->setStyleSheet("padding: 8px; border: 2px solid
#bdc3c7; border-radius: 5px;");

        addButton = new QPushButton("Додати новину");
        addButton->setStyleSheet(
            "QPushButton {"
            "    background-color: #27ae60;"
            "    color: white;"
            "    border: none;"
            "    padding: 10px 20px;"
            "    border-radius: 5px;"
            "    font-weight: bold;"
            "}"
            "QPushButton:hover {"
            "    background-color: #229954;"
            "}"
            );
        connect(addButton,      &QPushButton::clicked,      this,
&NewsDialog::onAddNews);

        QHBoxLayout *formRow1 = new QHBoxLayout();
        formRow1->addWidget(new QLabel("Заголовок:"));
        formRow1->addWidget(titleEdit);

        QHBoxLayout *formRow2 = new QHBoxLayout();
        formRow2->addWidget(new QLabel("Автор:"));
        formRow2->addWidget(authorEdit);
        formRow2->addWidget(new QLabel("Категорія:"));
        formRow2->addWidget(categoryCombo);

        addLayout->addLayout(formRow1);
        addLayout->addWidget(new QLabel("Зміст:"));
        addLayout->addWidget(contentEdit);
        addLayout->addLayout(formRow2);
        addLayout->addWidget(new QLabel("Дата публікації:"));
        addLayout->addWidget(dateEdit);
        addLayout->addWidget(addButton);
```

```
        rightLayout->addWidget(addNewsGroup);

        mainSplitter->addWidget(rightPanel);
        mainSplitter->setSizes({350, 650});

        mainLayout->addWidget(mainSplitter);
    }

    void NewsDialog::populateNewsList() {
        newsData.clear();

        NewsItem news1;
        news1.title = "Технологічний прорив у сфері штучного інтелекту";
        news1.content = "Сьогодні компанія TechCorp оголосила про революційний прорив у галузі штучного інтелекту. Нова технологія дозволяє машинам навчатися в 10 разів швидше, ніж раніше. Це відкриває нові можливості для медицини, освіти та транспорту.\n\nІнженери компанії працювали над цим проектом протягом трьох років. Результат перевершив усі очікування. Технологія вже проходить тестування в провідних університетах світу.";
        news1.author = "Олена Петренко";
        news1.category = "Технології";
        news1.date = QDateTime::currentDateTime().addDays(-1);
        newsData.append(news1);

        NewsItem news2;
        news2.title = "Економічне зростання України у 2025 році";
        news2.content = "За підсумками першого кварталу 2025 року, економіка України показала стабільне зростання на 3.5%. Це найкращий показник за останні п'ять років.\n\nОсновними факторами зростання стали розвиток IT-сектору, збільшення експорту сільськогосподарської продукції та приплив іноземних інвестицій. Експерти прогнозують збереження позитивної динаміки протягом всього року.";
        news2.author = "Михайло Коваленко";
        news2.category = "Економіка";
        news2.date = QDateTime::currentDateTime().addDays(-2);
        newsData.append(news2);

        NewsItem news3;
        news3.title = "Збірна України перемогла на Чемпіонаті Європи";
        news3.content = "Українська збірна з футболу здобула вражаючу перемогу на Чемпіонаті Європи, обігравши суперників з рахунком 3:1. Це історична перемога для українського спорту.\n\nГра проходила на стадіоні у Києві перед 70,000
```

```
    глядачів. Атмосфера була неймовірною. Гравці продемонстрували
високий рівень підготовки та командний дух.";
        news3.author = "Андрій Шевченко";
        news3.category = "Спорт";
        newsData.append(news3);

        NewsItem news4;
        news4.title = "Нова культурна програма у Києві";
        news4.content = "Київська мерія запустила масштабну
культурну програму 'Мистецтво для всіх'. Програма включає
безкоштовні концерти, виставки та майстер-класи.\n\nПротягом року
у різних районах міста відбудеться понад 200 культурних заходів.
Особлива увага приділяється підтримці молодих талантів та
популяризації української культури.";
        news4.author = "Катерина Мельник";
        news4.category = "Культура";
        news4.date = QDateTime::currentDateTime().addDays(-3);
        newsData.append(news4);

        filteredNews = newsData;
        filterNews();
    }

    void NewsDialog::filterNews() {
        newsList->clear();

        for (const auto &news : filteredNews) {
            QString itemText = QString("%1\n %2 | %3 | %4")
                                   .arg(news.title)
                                   .arg(news.author)

.arg(news.date.toString("dd.MM.yyyy hh:mm"))
                                   .arg(news.category);
            newsList->addItem(itemText);
        }
    }

    void NewsDialog::onNewsSelected() {
        int currentRow = newsList->currentRow();
        if (currentRow >= 0 && currentRow < filteredNews.size())
{
            const NewsItem &news = filteredNews[currentRow];

            newsTitle->setText(news.title);
            newsAuthor->setText(news.author);
            newsDate->setText(news.date.toString("dd.MM.yyyy
hh:mm"));
            newsCategory->setText(news.category);
            newsContent->setPlainText(news.content);
        }
    }
```

```
    void NewsDialog::onAddNews() {
        if    (titleEdit->text().isEmpty()    ||    contentEdit-
>toPlainText().isEmpty() || authorEdit->text().isEmpty()) {
            QMessageBox::warning(this, "Помилка", "Будь ласка,
заповніть всі поля!");
            return;
        }

        NewsItem newNews;
        newNews.title = titleEdit->text();
        newNews.content = contentEdit->toPlainText();
        newNews.author = authorEdit->text();
        newNews.category = categoryCombo->currentText();
        newNews.date = dateEdit->dateTime();

        newsData.prepend(newNews);

        clearForm();
        onFilterChanged();

        QMessageBox::information(this, "Успіх", "Новину успішно
додано!");
    }

    void NewsDialog::onDeleteNews() {
        int currentRow = newsList->currentRow();
        if (currentRow >= 0 && currentRow < filteredNews.size())
{
            QMessageBox::StandardButton         reply         =
QMessageBox::question(
                this, "Підтвердження",
                "Ви впевнені, що хочете видалити цю новину?",
                QMessageBox::Yes | QMessageBox::No
                );

            if (reply == QMessageBox::Yes) {
                const       NewsItem       &itemToDelete       =
filteredNews[currentRow];
                for (int i = 0; i < newsData.size(); ++i) {
                    if (newsData[i].title == itemToDelete.title
&&
                        newsData[i].date == itemToDelete.date) {
                        newsData.removeAt(i);
                        break;
                    }
                }

                onFilterChanged();
```

```
                newsTitle->setText("Виберіть новину для
читання");
                newsAuthor->clear();
                newsDate->clear();
                newsCategory->clear();
                newsContent->clear();
            }
        }
    }

    void NewsDialog::onFilterChanged() {
        QString filterText = filterCombo->currentText();
        QString searchText = searchEdit->text().toLower();

        filteredNews.clear();

        for (const auto &news : newsData) {
            bool matchesFilter = (filterText == "Всі категорії"
|| news.category == filterText);
            bool matchesSearch = (searchText.isEmpty() ||

news.title.toLower().contains(searchText) ||

news.content.toLower().contains(searchText) ||

news.author.toLower().contains(searchText));

            if (matchesFilter && matchesSearch) {
                filteredNews.append(news);
            }
        }

        std::sort(filteredNews.begin(), filteredNews.end(),
                  [](const NewsItem &a, const NewsItem &b) {
                      return a.date > b.date;
                  });

        filterNews();
    }

    void NewsDialog::onSearchChanged() {
        onFilterChanged();
    }

    void NewsDialog::clearForm() {
        titleEdit->clear();
        contentEdit->clear();
        authorEdit->clear();
        categoryCombo->setCurrentIndex(0);
        dateEdit->setDateTime(QDateTime::currentDateTime());
    }
```