МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

“До захисту допущений”
Зав. кафедри комп’ютерних наук та прикладної математики
д.т.н., професор Турбал Ю. В.
«___» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

«Розробка веб застосунку для керування бізнесом»

Виконав: Бортнік Юрій Андрійович
студент навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки
група ІПЗ-41і-2

_____
(підпис)

Керівник: Повшенюк Андрій Петрович

_____
(підпис)

Рівне – 2025

ЗМІСТ

РЕФЕРАТ

Кваліфікаційна робота: 43 сторінки, 3 рисунки, 12 наукових джерел.

Мета роботи: створення веб застосунку для автоматизованого управління бізнес-процесами малого та середнього підприємства з використанням сучасних вебтехнологій.

Інструменти реалізації: C#, ASP.NET Core, HTML5, CSS3, JavaScript, Bootstrap, SQL Server, Entity Framework Core, Chart.js, REST API.

Актуальність теми обумовлена зростаючою потребою у цифровій трансформації бізнесу, що вимагає ефективних інструментів для керування клієнтами, замовленнями, фінансами та аналітикою. Веб застосунки дозволяють забезпечити гнучкість, мобільність, надійність і швидке масштабування, що критично важливо в умовах сучасної конкуренції.

Ключові слова: автоматизація бізнесу, ASP.NET Core, веб застосунок, управління замовленнями, CRM, SQL Server, аналітика, REST API, малий бізнес.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД – система управління базами даних (у рамках проєкту використовувалась Microsoft SQL Server);

Frontend – клієнтська частина веб застосунку, реалізована з використанням HTML5, CSS3, JavaScript та Bootstrap;

Backend – серверна частина програмного забезпечення, розроблена на основі ASP.NET Core з використанням мови програмування C#;

ORM – Object-Relational Mapping, технологія, яка забезпечує взаємодію між об'єктами коду та реляційною базою даних (у проєкті використано Entity Framework Core);

API – інтерфейс прикладного програмування для обміну даними між клієнтською та серверною частинами системи;

JWT – JSON Web Token, стандарт для безпечної передачі інформації між сторонами через токени авторизації;

CRUD – Create, Read, Update, Delete – основні операції з даними, які реалізовано в системі для управління клієнтами, товарами, замовленнями тощо;

UI – User Interface, інтерфейс користувача, з яким взаємодіє кінцевий користувач;

UX – User Experience, досвід користувача при роботі з інтерфейсом системи;

REST API – архітектурний стиль побудови API, що використовує HTTP-запити для обміну даними між клієнтом і сервером;

Chart.js – JavaScript-бібліотека для побудови графіків та діаграм, що використовується для візуалізації бізнес-аналітики в системі.

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується стрімким зростанням ролі веб застосунків у процесах управління, планування та обліку бізнес-процесів. Усе більше компаній прагнуть автоматизувати свою діяльність, покращити взаємодію з клієнтами, оптимізувати облік фінансів, товарів, замовлень і співробітників. Особливо актуальним це є для малого та середнього бізнесу, який потребує простих, надійних та доступних інструментів для щоденного управління.

Одним із найбільш ефективних рішень у цій сфері є створення веб ресурсів – сайтів або веб платформ, що дозволяють керувати ключовими аспектами бізнесу через зручний інтерфейс. Такі системи забезпечують віддалений доступ до даних, багатокористувацький режим, інтеграцію з іншими сервісами (наприклад, платіжними системами, електронною поштою, CRM), а також гнучкість у налаштуванні відповідно до потреб конкретного підприємства.

Метою даної кваліфікаційної роботи є проєктування та розробка веб ресурсу для керування бізнесом, який дозволятиме власникам та адміністраторам малого бізнесу вести облік клієнтів, товарів, фінансових операцій, а також аналізувати основні показники діяльності за допомогою візуалізації даних.

Об'єктом дослідження є процес автоматизованого керування бізнес-процесами.

Предметом дослідження – інформаційна система керування, реалізована у вигляді вебресурсу.

У ході виконання кваліфікаційної роботи передбачається:

- провести аналіз існуючих систем та технологій керування бізнесом;
- визначити ключові функціональні та нефункціональні вимоги до системи;
- розробити архітектуру вебресурсу та структуру бази даних;
- реалізувати програмну частину вебресурсу з використанням ASP.NET Core для Backend та HTML/CSS/JS для Frontend;

провести тестування та аналіз результатів роботи вебресурсу.

Актуальність теми полягає у потребі малого бізнесу в доступних та адаптивних засобах автоматизації, які не потребують складного налаштування чи дорогого впровадження. Розроблений веб ресурс має слугувати прикладом такого рішення, що може бути використане в різних сферах підприємницької діяльності – від торгівлі до надання послуг.

Результати цієї роботи можуть бути використані як основа для створення повноцінних комерційних рішень, а також як навчальний приклад для студентів, які вивчають розробку інформаційних систем, веб технології та програмування.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ЗАСОБІВ РОЗРОБКИ

1.1. Особливості управління бізнесом у цифрову епоху

Управління бізнесом в умовах **цифрової трансформації** потребує швидкого реагування на зміни ринку, ефективного управління ресурсами та оперативного доступу до ключових показників діяльності. Сучасний бізнес, незалежно від його розміру, функціонує в умовах глобальної конкуренції та постійних інновацій, що вимагає гнучкості та адаптивності. **Малий і середній бізнес (МСБ)** часто стикається з особливими труднощами у веденні обліку клієнтів, товарів, фінансів та комунікацій, оскільки історично використовує застарілі або розрізнені інструменти (електронні таблиці, паперові документи, локальні програми). Ці методи не тільки уповільнюють процеси, але й збільшують ризик помилок, втрати даних та неефективного використання робочого часу, що безпосередньо впливає на прибутковість і конкурентоспроможність.

Зростаюча конкуренція змушує підприємців шукати способи оптимізації внутрішніх процесів, підвищення продуктивності праці та автоматизації рутинних операцій. Нездатність швидко адаптуватися до нових технологій та цифрових інструментів може призвести до значного відставання від конкурентів, втрати частки ринку та навіть до закриття бізнесу. В умовах, коли клієнти очікують швидкого та якісного обслуговування, а ринок диктує постійні зміни, ефективне управління стає критично важливим. У цьому контексті **створення власного вебресурсу**, який дозволяє керувати бізнесом через браузер із будь-якого пристрою, є не просто зручним, а й стратегічно важливим рішенням. Такий підхід забезпечує **мобільність**, **централізований доступ до даних** і можливість швидкого **масштабування функціоналу** відповідно до зростаючих потреб бізнесу. Це дозволяє підприємствам краще контролювати свої операції, аналізувати ефективність, виявляти вузькі місця та приймати обґрунтовані рішення на основі актуальних даних, що сприяє сталому розвитку.

1.2. Аналіз існуючих веб рішень для керування бізнесом

На ринку існує чимало систем, орієнтованих на управління підприємствами, які класифікуються за своїм основним призначенням: **CRM (Customer Relationship Management)** для управління взаємовідносинами з клієнтами, **ERP (Enterprise Resource Planning)** для комплексного планування ресурсів підприємства, **POS (Point of Sale)** для автоматизації торгових операцій тощо. До популярних рішень, що широко використовуються, можна віднести:

- **Bitrix24** – комплексна CRM-система, яка дозволяє керувати продажами, завданнями, проєктами та комунікацією з клієнтами, пропонуючи широкий спектр інструментів для спільної роботи. Вона має хмарну та коробкову версії, що дає певну гнучкість, але часто вимагає детального налаштування.
- **Zoho CRM** – гнучка хмарна платформа з широким спектром функцій для управління клієнтами, автоматизації маркетингу та продажу, що підходить для різних розмірів бізнесу. Відома своєю модульністю та інтеграцією з іншими продуктами Zoho Suite.
- **Odoo** – відкрита модульна ERP-система, яка включає облік, продажі, склад, фінанси, виробництво та інші бізнес-функції, дозволяючи компаніям інтегрувати всі свої процеси. Її перевага – відкритий код, але це також означає, що для повноцінного впровадження потрібні значні технічні знання.
- **МуСклад** – онлайн-сервіс, орієнтований на торгівлю, облік товарних залишків, управління закупівлями та фінансову звітність, популярний серед малого та середнього бізнесу в країнах СНД. Він простий у використанні для базових потреб, але його функціонал може бути обмеженим для складніших процесів.

Проте, незважаючи на їхню функціональність та популярність, ці рішення часто мають істотні **недоліки для малого бізнесу**:

- **Висока вартість ліцензій або підписок**: Більшість комерційних систем пропонують багаторівневі тарифи, які можуть бути непосильними для невеликих компаній, особливо з обмеженим бюджетом. Причому вартість зростає зі збільшенням кількості користувачів та необхідного функціоналу.

- **Надлишковий функціонал**: МСБ рідко використовує всі можливості великих ERP-систем. Наявність великої кількості невикористовуваних функцій може ускладнювати інтерфейс, збільшувати час на навчання персоналу та створювати відчуття перевантаженості системи.
- **Складність налаштування без технічних знань**: Впровадження та кастомізація складних систем часто вимагає залучення IT-фахівців або сторонніх консультантів, що є додатковими витратами та може затягнути процес запуску системи.
- **Залежність від зовнішнього провайдера (відсутність повного контролю над даними)**: Хмарні рішення, хоч і зручні, але залишають компанію залежною від постачальника послуг у питаннях безпеки даних, доступності, політики оновлень та конфіденційності. Це може бути критичним для бізнесу, що працює з конфіденційною інформацією або має суворі вимоги до безпеки. Крім того, припинення роботи провайдера може призвести до втрати доступу до даних.
- **Відсутність гнучкості та кастомізації**: Готові рішення рідко ідеально відповідають усім унікальним бізнес-процесам конкретної компанії. Можливості для глибокої кастомізації зазвичай обмежені або вимагають значних додаткових витрат.

Тому актуальним стає створення **власного вебресурсу**, який буде адаптовано під конкретні потреби компанії, з мінімально необхідним, але ефективним функціоналом та повним контролем над інфраструктурою та даними. Такий підхід дозволяє оптимізувати витрати на впровадження та підтримку, забезпечити максимальну гнучкість у розвитку та підвищити рівень безпеки інформації, оскільки всі дані знаходяться під прямим контролем власника системи.

1.3. Аналіз потреб користувачів і формування вимог до системи

Для успішної розробки системи вкрай важливо чітко визначити **потреби кінцевих користувачів** та бізнес-процеси, які вона повинна автоматизувати.

Детальний аналіз дозволяє уникнути розробки зайвого функціоналу та зосередитися на тому, що дійсно принесе користь. На основі аналізу типового малого підприємства, його щоденних операцій, больових точок та цілей, можна сформувати деталізований перелік основних **функціональних вимог** до системи:

- **Управління користувачами та ролями**:
  - **Реєстрація та авторизація користувачів**: Безпечний процес створення облікових записів та входу в систему.
  - **Розмежування доступу за ролями (адміністратор, менеджер)**: Адміністратор має повний доступ до всіх функцій та налаштувань, менеджер – обмежений доступ до даних та операцій, необхідних для його роботи (наприклад, управління клієнтами та замовленнями, але без доступу до системних налаштувань).
  - **Можливість редагування профілю користувача та зміни пароля**.
- **Управління клієнтською базою (CRM-функціонал)**:
  - **Ведення бази клієнтів**: Зберігання контактних даних (ПІБ, телефон, email, адреса), реквізитів компанії (для юридичних осіб).
  - **Історія взаємодії**: Запис усіх комунікацій (дзвінки, листи, зустрічі) з кожним клієнтом.
  - **Присвоєння статусів клієнтам**: Наприклад, "новий", "постійний", "VIP", "потенційний", "проблемний" для сегментації та пріоритезації роботи.
  - **Додавання приміток та файлів** до картки клієнта.
- **Управління товарами/послугами**:
  - **Облік товарів або послуг**: Каталог з назвою, описом, ціною, одиницею виміру.
  - **Управління залишками**: Відстеження кількості товарів на складі з можливістю автоматичного оновлення при продажу/закупівлі.
  - **Категоризація товарів**: Можливість групування товарів за категоріями для зручності пошуку та аналізу.
  - **Додавання зображень товарів**.

- **Управління замовленнями/угодами**:
  - **Створення та редагування замовлень**: Додавання товарів/послуг до замовлення, вказівка кількості, розрахунок суми.
  - **Присвоєння статусів замовленням**: Наприклад, "нове", "в обробці", "очікує оплати", "виконано", "скасовано", "доставлено".
  - **Відстеження історії змін замовлення**.
  - **Прив'язка замовлень до клієнтів та менеджерів**.
- **Фінансовий облік та звітність**:
  - **Облік доходів та витрат**: Можливість фіксації всіх фінансових операцій.
  - **Формування простих фінансових звітів**: Наприклад, звіти про прибутки/збитки за період, рух коштів.
  - **Візуалізація даних у вигляді графіків та діаграм**: Для наочного представлення динаміки продажів, доходів, витрат тощо.
- **Зручність використання та інтерактивність**:
  - **Можливість пошуку, фільтрації, сортування даних**: Швидкий доступ до необхідної інформації за різними критеріями (назвою, датою, статусом, клієнтом).
  - **Пагінація**: Для зручної роботи з великими обсягами даних.
  - **Експорт даних у формати Excel/CSV**: Для подальшого аналізу або інтеграції з іншими системами.
- **Адаптивний інтерфейс**:
  - **Робота з ПК та мобільних пристроїв**: Система повинна коректно відображатися та повноцінно функціонувати на екранах різного розміру (десктопи, планшети, смартфони) без втрати функціоналу.

Крім функціональних вимог, не менш важливими є **нефункціональні вимоги**, які визначають якість, надійність та зручність експлуатації системи:

- **Зручність інтерфейсу (User-Friendly Interface)**: Інтуїтивно зрозумілий дизайн, логічна навігація та мінімальна кількість кроків для

виконання типових операцій, що мінімізує час на навчання користувачів та підвищує продуктивність роботи.

- **Захищений доступ до даних**: Реалізація сучасних механізмів автентифікації (наприклад, двофакторна автентифікація), авторизації, шифрування даних (TLS/SSL для трафіку, шифрування чутливих даних у базі) для запобігання несанкціонованому доступу та захисту конфіденційної інформації. Регулярне резервне копіювання даних.
- **Швидкодія та продуктивність**: Оптимізація продуктивності системи для забезпечення швидкого завантаження сторінок (до 2-3 секунд), оперативного виконання запитів та обробки великих обсягів даних.
- **Масштабованість**: Архітектура системи повинна дозволяти легке розширення функціоналу (додавання нових модулів) та збільшення обсягів даних/кількості одночасних користувачів без значної перебудови або падіння продуктивності.
- **Надійність та відмовостійкість**: Система повинна бути стійкою до збоїв, забезпечуючи мінімальний час простою та відновлення після можливих несправностей.
- **Підтримка та супроводження**: Код повинен бути чистим, документованим та легким для розуміння, що дозволить легко здійснювати подальші оновлення та виправлення помилок.

Відповідно до цих вимог буде розроблено систему, яка є універсальною базою для управління підприємницькою діяльністю, забезпечуючи її ефективність, безпеку та адаптивність до мінливих умов ринку та зростаючих потреб бізнесу.

1.4. Обґрунтування вибору технологій для реалізації

Вибір технологічного стеку є ключовим етапом, що впливає на продуктивність, масштабованість, безпеку, вартість розробки та легкість подальшої підтримки майбутньої системи. З урахуванням сучасних вимог до безпеки, продуктивності, зручності розробки та підтримки, для реалізації

вебресурсу доцільно використати такі технології, які забезпечують оптимальний баланс цих факторів:

- **ASP.NET Core (Backend)**: Це кросплатформенна, високопродуктивна та відкрита фреймворк-технологія від Microsoft, що є оптимальним вибором для побудови **Backend-частини** (серверної логіки). Вона підтримує створення високоефективних REST API, має вбудовані, надійні механізми для реалізації авторизації та аутентифікації (наприклад, **ASP.NET Core Identity**), а також надає широкі можливості для доступу до баз даних через ORM. Її модульність, гнучкість та підтримка асинхронних операцій дозволяють створювати легкі, швидкі та масштабовані додатки, що здатні обробляти значні навантаження.
- **C# (Мова програмування Backend)**: Як основна мова програмування для серверної логіки, C# є сучасною, об'єктно-орієнтованою мовою з потужною екосистемою .NET. Вона забезпечує високу продуктивність, надійність, типову безпеку та зручність розробки завдяки сильній типізації, розширеним можливостям асинхронного програмування, збору сміття та великій кількості готових бібліотек та інструментів. Велика спільнота розробників та якісна документація також є перевагами.
- **SQL Server (Система управління базами даних)**: Це потужна, надійна та масштабована система управління реляційними базами даних (СУБД) від Microsoft. SQL Server прекрасно інтегрується з .NET-середовищем, що спрощує розробку та адміністрування. Він підтримує складні запити, транзакції, індексацію, реплікацію та розширені механізми безпеки, що забезпечує цілісність, доступність та конфіденційність даних. Альтернативою може бути PostgreSQL або MySQL для бюджетних рішень, але SQL Server є стандартним вибором у .NET-екосистемі.
- **HTML5 / CSS3 / JavaScript (Frontend)**: Це фундаментальний, стандартний і невід'ємний стек для створення будь-якого сучасного, інтерактивного та динамічного **вебінтерфейсу (Frontend)**. HTML5 забезпечує семантичну структуру вмісту, CSS3 відповідає за візуальне оформлення,

стилізацію та адаптивність (забезпечуючи гнучкий дизайн для різних пристроїв), а JavaScript додає інтерактивність, динамічний функціонал та можливість взаємодії з користувачем на стороні клієнта.

- **Bootstrap (Frontend Framework)**: Популярний, відкритий та гнучкий фреймворк для розробки адаптивного та мобільного інтерфейсу. Він значно прискорює процес верстки, надаючи велику бібліотеку готових, попередньо стилізованих компонентів (кнопки, форми, навігація, модальні вікна) та потужну сіткову систему. Це дозволяє створювати професійний вигляд з мінімальними зусиллями, забезпечує коректне відображення на всіх пристроях та сумісність з різними браузерами.
- **Chart.js або Plotly.js (Бібліотеки візуалізації даних)**: Для побудови інтерактивних та візуально привабливих графіків та діаграм, необхідних для фінансових звітів та аналізу ключових показників (KPI), будуть використані JavaScript-бібліотеки, такі як Chart.js або Plotly.js. Вони дозволяють наочно та зрозуміло представляти дані, роблячи їх більш доступними для аналізу та прийняття рішень.
- **Entity Framework Core (ORM)**: Це сучасний та потужний ORM (Object-Relational Mapper) для .NET, який значно спрощує взаємодію з базою даних. Він дозволяє розробникам працювати з об'єктами C# (моделями) замість прямого написання SQL-запитів, автоматично генеруючи їх. Це підвищує продуктивність розробки, зменшує кількість помилок, пов'язаних з SQL-ін'єкціями та синтаксисом, та покращує підтримку коду. EF Core підтримує міграції бази даних, що спрощує управління змінами у схемі.

Використання цих засобів дозволяє створити швидкий, зручний, безпечний та масштабований веб ресурс із повноцінною адміністративною панеллю. Цей технологічний стек є збалансованим вибором, що забезпечує гнучкість у розробці, високу продуктивність, легкість у подальшій підтримці та розвитку системи, а також доступ до великої спільноти розробників та ресурсів. Таке рішення буде економічно вигідним та ефективним для потреб малого та середнього бізнесу.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Архітектура вебресурсу

Проєктований веб ресурс для керування бізнесом реалізується на основі перевіреної та надійної **трирівневої архітектури (Three-Tier Architecture)**. Такий підхід передбачає логічне та фізичне розділення системи на три незалежні рівні, кожен з яких відповідає за певну групу функцій. Це забезпечує високу **модульність**, **гнучкість розробки**, **можливість масштабування** та легкість підтримки системи, а також сприяє **забезпеченню високої продуктивності** навіть при значній кількості одночасних користувачів.

1. **Презентаційний рівень (Frontend / User Interface Layer)**:

- **Призначення**: Цей рівень є безпосередньою точкою взаємодії користувача з системою. Він відповідає за відображення інтерфейсу користувача, збір вхідних даних, відображення результатів обробки та візуалізацію інформації.
- **Технології реалізації**: Використовуються стандартні веб-технології: **HTML5** для структури сторінок, **CSS3** для їхнього стилізованого вигляду та адаптивності, а також **JavaScript** для забезпечення інтерактивності та динамічного функціоналу. Для прискорення та стандартизації розробки інтерфейсу застосовується фреймворк **Bootstrap**, який забезпечує адаптивний дизайн та готові UI-компоненти. Візуалізація даних, таких як статистичні графіки та діаграми, реалізується за допомогою бібліотеки **Chart.js**. Додатково, для покращення користувацького досвіду, можуть бути використані **jQuery** для спрощення маніпуляцій елементами сторінки та **AJAX** для асинхронного обміну даними із сервером без перезавантаження сторінки, що робить інтерфейс більш чуйним.
- **Взаємодія**: Компоненти цього рівня відправляють запити на рівень бізнес-логіки та отримують від нього відповіді, представляючи їх у зручному для користувача вигляді.

2. **Рівень бізнес-логіки (Backend / Application Logic Layer)**:

- **Призначення**: Цей "мозок" системи відповідає за обробку всіх запитів користувача, реалізацію основних бізнес-правил та логіки (наприклад, перевірка даних, обробка замовлень, розрахунок цін), управління сесіями користувачів, автентифікацію та авторизацію, а також координацію взаємодії з рівнем доступу до даних.
- **Технології реалізації**: Розробка цього рівня здійснюється на платформі **ASP.NET Core** із застосуванням мови програмування **C#**. ASP.NET Core надає надійний і високопродуктивний фреймворк для побудови веб-додатків та API. Взаємодія між клієнтською частиною і сервером відбувається через **REST API**, що забезпечує гнучкість та дозволяє клієнту та серверу бути незалежними одне від одного. Система автентифікації та авторизації буде реалізована за допомогою **ASP.NET Core Identity** або на основі JWT-токенів для безпечного управління доступом.

3. **Рівень доступу до даних (Data Layer / Data Access Layer)**:

- **Призначення**: Цей рівень відповідає виключно за операції з базою даних: збереження нової інформації, оновлення існуючих записів, ефективний пошук та видалення даних. Він є своєрідним "мостом" між рівнем бізнес-логіки та фізичним сховищем даних, абстрагуючи бізнес-логіку від деталей реалізації бази даних.
- **Технології реалізації**: Для зберігання даних використовується **Microsoft SQL Server** – надійна та масштабована реляційна СУБД. Взаємодія з базою даних здійснюється через ORM-бібліотеку **Entity Framework Core**. Використання ORM дозволяє розробникам працювати з даними як з об'єктами C#, що значно спрощує розробку, зменшує кількість помилок та підвищує читабельність коду. Entity Framework Core також підтримує підхід **Code First**, що дозволяє генерувати схему бази даних на основі моделей C#, спрощуючи управління міграціями та змінами в структурі даних.

Переваги обраної архітектури:

- **Модульність та незалежність**: Кожен рівень може бути розроблений та масштабований незалежно, що спрощує командну роботу та прискорює розробку.
- **Гнучкість та масштабованість**: При зростанні навантаження можна масштабувати окремі рівні (наприклад, додати більше серверів для Frontend або Backend) без впливу на інші.
- **Безпека**: Розділення рівнів дозволяє реалізувати надійні механізми безпеки на кожному етапі, обмежуючи прямий доступ до бази даних.
- **Легкість підтримки та оновлення**: Зміни в одному рівні менше впливають на інші, що спрощує внесення правок та розширення функціоналу.

2.2. Проєктування бази даних

База даних є критично важливим компонентом системи, що забезпечує надійне та ефективне зберігання всієї інформації, необхідної для функціонування бізнесу. Для побудови оптимальної та ефективної структури БД була розроблена **логічна модель**, що включає такі основні таблиці, які покривають ключові аспекти бізнес-процесів малого підприємства:

Таблиця	**Призначення**	**Поля (приклад)**
Users	Зберігає інформацію про користувачів системи, які мають доступ до функціоналу (адміністратори, менеджери).	Id, Login, PasswordHash, Salt, Email, RoleId, FullName, IsActive
Roles	Довідник ролей користувачів для розмежування доступу.	Id, Name (напр., "Administrator", "Manager")
Clients	Детальні дані про клієнтів (фізичних або юридичних осіб), з якими працює бізнес.	Id, FirstName, LastName, CompanyName, Phone, Email, Address, Notes, CreatedAt
Categories	Довідник для класифікації товарів або послуг.	Id, Name, Description

Products	Каталог товарів або послуг, що пропонуються компанією.	Id, Name, Description, Price, Unit (од. виміру), CategoryId, StockQuantity (залишок)
Orders	Основна таблиця для обліку замовлень або угод, що фіксує загальні дані про транзакцію.	Id, ClientId, UserId (відповідальний менеджер), OrderDate, StatusId, TotalAmount
OrderStatuses	Довідник можливих статусів замовлення (напр., "Нове", "В обробці", "Виконано", "Скасовано").	Id, Name
OrderItems	Деталізація кожного замовлення, що пов'язує замовлення з конкретними товарами та їх кількістю/ціною.	Id, OrderId, ProductId, Quantity, UnitPrice (ціна на момент замовлення), LineTotal
Payments	Облік фінансових надходжень, пов'язаних із замовленнями або іншими операціями.	Id, OrderId (може бути порожнім), PaymentDate, Amount, PaymentTypeId, Notes
PaymentTypes	Довідник типів оплати (напр., "Готівка", "Банківська карта", "Переказ").	Id, Name
Logs	Журнал подій для відстеження активності користувачів та системних подій.	Id, Timestamp, UserId (може бути порожнім), ActionType, Description, DetailsJson
StatsCache	Збереження обчислених статистичних звітів або агрегованих даних для прискорення завантаження дашбордів.	Id, ReportName, PeriodStart, PeriodEnd, CachedDataJson, LastUpdated

Принципи проєктування БД:

- **Нормалізація**: У проєкті застосовано **нормалізацію до третьої нормальної форми (3NF)**. Це дозволяє уникнути дублювання даних, зменшити надмірність, запобігти аномаліям при вставці, оновленні та видаленні даних, а також забезпечити високу цілісність даних. Наприклад, окремі довідники для

ролей, категорій товарів та статусів замовлень дозволяють уникнути повторення назв та спрощують оновлення.

- **Зовнішні ключі**: Використання зовнішніх ключів є основою для встановлення зв'язків між таблицями (наприклад, поле ClientId у таблиці Orders посилається на Id у таблиці Clients). Це забезпечує **реляційну цілісність** даних, гарантуючи, що посилання завжди вказують на існуючі записи, та автоматично підтримує логічні зв'язки між даними.
- **Індекси**: Для прискорення операцій пошуку, фільтрації та сортування даних будуть створені індекси на часто використовуваних полях. Це значно підвищує продуктивність при роботі з великими обсягами даних, оскільки база даних може швидше знаходити потрібну інформацію.
- **Обмеження**:
  - **NOT NULL**: Гарантує, що певні поля не можуть бути порожніми, що є важливим для цілісності та повноти даних (наприклад, логін та хеш пароля користувача).
  - **UNIQUE**: Забезпечує унікальність значень у полі або наборі полів (наприклад, унікальність логіна та електронної пошти користувача, назв у довідкових таблицях).
  - **CHECK**: Встановлює правила для значень у полі (наприклад, гарантує, що ціна товару не може бути від'ємною).
- **Типи даних**: Використання відповідних типів даних (наприклад, текстові поля для назв, десяткові для грошових сум, дати та час для позначок часу) забезпечує ефективне зберігання та обробку даних, а також зменшує використання дискового простору.

Такий підхід до проєктування бази даних гарантує її надійність, продуктивність та зручність у подальшій розробці та підтримці системи.

2.3. Діаграми варіантів використання

Для детального опису функціоналу системи з точки зору взаємодії користувачів з нею були складені **Use Case діаграми (діаграми варіантів**

використання). Вони візуально відображають, які дії може виконувати кожен тип користувача (актор) у системі.

Основні ролі (Actors):

- **Адміністратор (Administrator)**: Має повний доступ до всіх функціональних можливостей системи. Це включає управління користувачами (додавання, редагування, видалення, зміна ролей), повний контроль над базою товарів/послуг, доступ до детальної статистики та системних налаштувань.
- **Менеджер (Manager)**: Основний робочий користувач. Має доступ до функцій управління клієнтами, створення та редагування замовлень, обробки платежів та перегляду звітів, необхідних для повсякденної роботи. Йому заборонено змінювати системні налаштування або керувати іншими користувачами.
- **Гість (Guest / Unauthorized User)**: Неавторизований користувач, який має обмежений доступ. Його основна функція – можливість переглядати сторінку входу/реєстрації та, можливо, публічну інформацію (якщо така передбачена).

Основні сценарії використання (Use Cases):

- **Увійти в систему**: Базовий сценарій для всіх авторизованих користувачів, який дозволяє отримати доступ до функціоналу системи.
- **Керувати користувачами**: Включає додавання нових користувачів, редагування їхніх даних та видалення, а також встановлення їхніх ролей у системі. (Виконує: Адміністратор)
- **Керувати ролями користувачів**: Зміна прав доступу для існуючих користувачів шляхом призначення їм відповідних ролей. (Виконує: Адміністратор)
- **Додати нового клієнта**: Занесення повної інформації про нового клієнта (контактні дані, компанія тощо) в базу даних. (Виконує: Менеджер, Адміністратор)

- **Редагувати інформацію про клієнта**: Зміна існуючих даних клієнта, оновлення його статусу або додавання нових приміток. (Виконує: Менеджер, Адміністратор)
- **Створити нове замовлення**: Формування нового замовлення, вибір клієнта, додавання товарів або послуг до нього, вказівка кількості та розрахунок загальної суми. (Виконує: Менеджер, Адміністратор)
- **Редагувати замовлення**: Внесення змін до вже існуючого замовлення, таких як додавання/видалення позицій, зміна кількості або ціни, оновлення статусу замовлення. (Виконує: Менеджер, Адміністратор)
- **Переглянути список замовлень**: Доступ до переліку всіх замовлень з можливістю застосування фільтрів за датою, клієнтом, статусом та пошуку. (Виконує: Менеджер, Адміністратор)
- **Сформувати звіт по замовленнях**: Генерація різних аналітичних звітів за замовленнями (наприклад, за період, за клієнтом, за статусом, за обсягом продажів). (Виконує: Менеджер, Адміністратор)
- **Керувати товарами/послугами**: Повний цикл управління каталогом товарів та послуг: додавання нових позицій, їх редагування (зміна цін, описів), а також видалення. (Виконує: Адміністратор)
- **Облік платежів**: Запис інформації про отримані платежі, прив'язка їх до замовлень, вказівка суми та типу оплати. (Виконує: Менеджер, Адміністратор)
- **Переглянути фінансові звіти**: Доступ до агрегованих фінансових даних, таких як звіти про доходи, витрати та прибуток, у вигляді таблиць та графіків. (Виконує: Менеджер, Адміністратор)
- **Переглянути лог системи**: Доступ до системного журналу, який фіксує важливі події та дії користувачів для аудиту та відлагодження. (Виконує: Адміністратор)
- **Змінити особисті дані**: Користувач може оновити свій пароль, електронну пошту або інші дані профілю. (Виконує: Адміністратор, Менеджер)

Ці діаграми допомагають візуалізувати взаємодію акторів із системою, чітко визначають межі функціоналу для кожної ролі та служать основою для подальшого проєктування інтерфейсу та розробки тестів.

2.4. Проєктування інтерфейсу користувача

Проєктування інтерфейсу користувача (UI) є критично важливим для забезпечення високої **зручності використання (User Experience - UX)** та ефективності роботи в системі. Інтерфейс має бути **мінімалістичним, інтуїтивно зрозумілим та адаптивним**, щоб користувачі могли швидко освоїти систему та ефективно виконувати свої завдання з будь-якого пристрою.

Основні принципи проєктування інтерфейсу:

- **Мінімум кліків до основної дії**: Важливо, щоб ключові операції (наприклад, створення нового замовлення, додавання клієнта) були доступні за мінімальну кількість кроків, що скорочує час на виконання рутинних завдань.
- **Візуальна ієрархія та кольорове виділення**: Використання кольорів для виділення важливих елементів (наприклад, зелений для успішних операцій, червоний для помилок, синій для активних елементів), чітких заголовків та логічних відступів для створення структурованої та легкої для сприйняття сторінки.
- **Адаптивна верстка (Responsive Design)**: Забезпечення коректного відображення та повноцінної функціональності інтерфейсу на екранах різного розміру (десктопи, ноутбуки, планшети, смартфони). Це дозволить користувачам працювати з системою з будь-якого місця та пристрою без обмежень.
- **Єдність дизайну**: Використання єдиних стилів, компонентів та патернів взаємодії на всіх сторінках для створення цілісного, передбачуваного та професійного користувацького досвіду.
- **Зворотний зв'язок**: Надання користувачеві чіткого та своєчасного зворотного зв'язку про виконані дії (успішне збереження, повідомлення про помилку валідації, індикатори процесу завантаження), що підвищує довіру до системи.

Передбачені ключові сторінки та їх функціонал:

1. **Головна сторінка (Dashboard)**:

- Перша сторінка після входу, що надає комплексний огляд ключових показників бізнесу.
- Відображає інтерактивні **графіки** (наприклад, динаміка замовлень, доходів за період), статистичні дані (кількість клієнтів, активних замовлень, загальна сума продажів).
- Містить елементи швидкого доступу до останніх дій, незавершених завдань або "гарячих" посилань на основні функції системи.

2. **Клієнти (Customers)**:

- Основна сторінка зі списком усіх зареєстрованих клієнтів у вигляді таблиці.
- **Функціонал**: сортування за різними полями (ім'я, дата додавання, компанія), ефективний пошук за назвою/контактами, фільтрація за статусами або категоріями.
- Можливість **додавання нового клієнта**, **редагування** існуючого, **перегляд детальної інформації** про клієнта (включаючи його контактні дані, історію замовлень та всі записи взаємодії).

3. **Товари/Послуги (Products/Services)**:

- Довідник з каталогом усіх товарів або послуг, які пропонує компанія.
- **Функціонал**: додавання нових товарів (з назвою, описом, ціною, одиницею виміру, категорією, початковим залишком), зміна існуючих даних, видалення (з перевіркою на наявність у замовленнях).
- Можливість зручної фільтрації за категоріями та пошуку за назвою.

4. **Замовлення (Orders)**:

- Список усіх замовлень з ключовою інформацією (унікальний номер, клієнт, дата створення, загальна сума, поточний статус).
- **Деталізація замовлення**: при кліку на замовлення відкривається детальний перегляд зі складом (позиціями), їх кількістю та ціною, поточним

статусом, інформацією про відповідального менеджера, а також історією всіх змін замовлення.

- **Управління статусами**: можливість швидкої зміни статусу замовлення через випадаюче меню або кнопки дій.
- Розширений функціонал пошуку, фільтрації та сортування для легкого управління великою кількістю замовлень.

5. **Звіти (Reports)**:

- Розділ для генерації різноманітних аналітичних звітів, що допомагають оцінювати ефективність бізнесу.
- **Приклади звітів**: щомісячні/щоквартальні/річні графіки доходів та витрат, кількість угод за певний період, список найактивніших клієнтів (за кількістю замовлень або загальною сумою), звіт по товарних залишках та руху товарів.
- Можливість гнучкого вибору періоду та параметрів для генерації кожного звіту.

6. **Профіль користувача (User Profile)**:

- Сторінка для керування особистими даними авторизованого користувача.
- **Функціонал**: зміна пароля, електронної пошти, повного імені, а також налаштування мови інтерфейсу (якщо передбачена багатомовність) та інших персональних уподобань.

Використані технології для реалізації інтерфейсу:

- **Bootstrap**: Забезпечує основу адаптивного дизайну, готові стилі та компоненти (навігація, форми, таблиці, модальні вікна), що значно прискорює та уніфікує розробку UI.
- **jQuery / AJAX**: Використовується для спрощення маніпуляцій з елементами сторінки та реалізації асинхронних запитів до сервера. Це дозволяє оновлювати частини сторінки без її повного перезавантаження, підвищуючи швидкість відгуку та покращуючи користувацький досвід.

- **Chart.js**: Потужна бібліотека JavaScript для створення інтерактивних та візуально привабливих графіків та діаграм, які будуть активно використовуватися на Dashboard та у розділі звітів для наочної візуалізації даних.
- **FontAwesome**: Популярна бібліотека іконок, яка надає великий набір векторних іконок для візуального покращення інтерфейсу, швидшого розуміння функціоналу та естетичного вигляду.

Продуманий дизайн інтерфейсу та застосування сучасних frontend-технологій дозволять створити систему, яка буде не тільки функціональною, але й приємною та зручною у повсякденному використанні, сприяючи підвищенню продуктивності бізнесу.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ВЕБРЕСУРСУ ДЛЯ КЕРУВАННЯ БІЗНЕСОМ

3.1. Загальна структура реалізації

Розробка вебзастосунку для керування бізнесом була здійснена на основі **трирівневої архітектури**, що забезпечує чітке розділення відповідальності та сприяє масштабованості та зручності підтримки системи. Основною метою реалізації було створення надійної та ефективної системи, яка відповідає сучасним стандартам розробки. Зокрема, було приділено особливу увагу наступним аспектам:

- **Зручний інтерфейс користувача (Frontend)**: Розробка інтуїтивно зрозумілого та привабливого інтерфейсу, що забезпечує легкість взаємодії користувачів із системою, мінімізуючи час на навчання та підвищуючи продуктивність.
- **Захищений доступ до даних**: Впровадження надійних механізмів автентифікації та авторизації для забезпечення конфіденційності та цілісності чутливих бізнес-даних.
- **Ефективна обробка бізнес-логіки (Backend)**: Оптимізація серверної логіки для швидкої та точної обробки бізнес-процесів, таких як створення замовлень, управління клієнтами та фінансовими операціями.
- **Масштабована серверна архітектура**: Проектування системи з урахуванням можливості подальшого розширення та обробки зростаючих обсягів даних та кількості користувачів без значних змін в архітектурі.

Застосунок складається з трьох ключових, взаємопов'язаних компонентів, що працюють у синергії для забезпечення повноцінного функціоналу:

- **Frontend (Клієнтська частина)**: Це візуальна частина застосунку, з якою безпосередньо взаємодіє користувач. Вона була реалізована з використанням стандартних веб-технологій: **HTML5** для структури контенту, **CSS3** (з використанням фреймворку **Bootstrap** для адаптивного дизайну) для стилізації та візуального оформлення, а також **JavaScript** для забезпечення інтерактивності та динамічної взаємодії з користувачем.

- **Backend (Серверна частина)**: Цей компонент відповідає за обробку запитів від клієнтської частини, виконання бізнес-логіки та взаємодію з базою даних. Він розроблений на базі платформи **ASP.NET Core** з використанням мови програмування **C#**, що забезпечує високу продуктивність та надійність.
- **База даних**: Цей компонент є центральним сховищем усіх даних системи. Була обрана **SQL Server** як система керування реляційними базами даних, яка зберігає та керує такими важливими даними, як інформація про користувачів, клієнтів, замовлення, товари, платежі та багато іншого, забезпечуючи їх цілісність та доступність.

3.2. Реалізація клієнтської частини (Frontend)

Інтерфейс користувача був розроблений як **адаптивна HTML-сторінка**, що динамічно реагує на розмір екрану пристрою, забезпечуючи оптимальне відображення на настільних комп'ютерах, планшетах та мобільних телефонах. Логіка взаємодії максимально відповідає сучасним підходам **UX/UI (User Experience / User Interface)**, зосереджуючись на інтуїтивності, ефективності та привабливості. Основні особливості реалізації frontend включають:

- **Landing Page (business_landing.html)**: Це перша точка входу для нових користувачів. Сторінка розроблена як приваблива вступна презентація системи. Вона включає:
  - **Демонстрацію переваг**: Чітке пояснення, як BizManager PRO може покращити управління бізнесом.
  - **Етапи роботи**: Опис простих кроків для початку використання системи.
  - **Інтерактивні елементи**: Наприклад, помітні кнопки для початку реєстрації нового облікового запису або замовлення безкоштовної консультації, що спонукають до дії.
  - **Візуальний контент**: Можливо, включаючи скріншоти або відео-демонстрації ключових функцій.

- **Компоненти UI та Візуальне оформлення**: Для створення сучасного та естетичного інтерфейсу були використані:
  - **Шрифти**: Забезпечують читабельність та візуальну ієрархію тексту.
  - **Іконки FontAwesome**: Використовуються для візуального представлення функцій та навігації, додаючи сучасності та інтуїтивності.
  - **Модальні вікна**: Ефективно використовуються для відображення форм реєстрації нових користувачів, входу до системи, або надання демо-доступу, дозволяючи зосередити увагу користувача на конкретному завданні.
- **Інтерактивність**: Реалізована за допомогою **JavaScript**, що забезпечує динамічну взаємодію користувача з інтерфейсом. Це включає:
  - **Динамічне перемикання вкладок**: Дозволяє організувати великі обсяги інформації у зручний для перегляду формат.
  - **Форми та валідація**: JavaScript використовується для миттєвої перевірки вхідних даних у формах (наприклад, перевірка електронної пошти, пароля, чи заповнені обов'язкові поля) перед надсиланням їх на сервер, що покращує користувацький досвід та знижує навантаження на бекенд.
- **Адаптивність**: Усі елементи інтерфейсу повністю адаптовані до різних розмірів екранів та мобільних пристроїв завдяки інтеграції **Bootstrap 5**. Це означає, що система виглядає та функціонує однаково добре як на великих моніторах, так і на смартфонах.

Приклад HTML-фрагменту головного блоку, що демонструє ключові елементи дизайну:

HTML

```
<section class="hero">
    <div class="container hero-container">
        <div class="hero-content">
            <h1>Управління бізнесом стало <span>простіше</span> ніж будь-коли</h1>
            <p class="hero-text">BizManager PRO — це комплексна система для управління клієнтами, замовленнями та фінансами.</p>
```

```
        </div>
    </div>
</section>
```

3.3. Реалізація серверної частини (Backend)

Серверна частина застосунку була реалізована на платформі **ASP.NET Core Web API**, що є потужним фреймворком для створення RESTful сервісів. Це забезпечує високу продуктивність, масштабованість та кросплатформність. Основними модулями та функціями бекенду є:

- **Система автентифікації та авторизації**:
  - Реалізована за допомогою **JWT-токенів (JSON Web Tokens)**. При успішній автентифікації користувач отримує токен, який використовується для доступу до захищених ресурсів API.
  - Підтримка **ролей користувачів** (наприклад, адміністратор, менеджер, співробітник) дозволяє гнучко керувати правами доступу до різних функціональних можливостей та даних системи. Кожен запит перевіряється на відповідність ролі користувача, забезпечуючи безпеку.
- **Контролери API (RESTful Services)**: Кожен контролер відповідає за обробку запитів, пов'язаних з певною сутністю системи. Це дозволяє організувати логіку за принципом "розділяй і володарюй", роблячи код більш зрозумілим та легким для підтримки. Основні контролери включають:
  - **UsersController**: Відповідає за всі операції, пов'язані з управлінням користувачами системи (реєстрація, вхід, зміна пароля, управління профілями користувачів, призначення ролей).
  - **ClientsController**: Обробляє запити, що стосуються клієнтської бази (додавання нових клієнтів, редагування інформації про існуючих, пошук, видалення, перегляд списку клієнтів).
  - **OrdersController**: Керує всіма аспектами, пов'язаними із замовленнями (створення нових замовлень, оновлення статусу, перегляд деталей замовлень, формування списків замовлень).

- **PaymentsController**: Забезпечує функціонал для реєстрації та управління фінансовими операціями (додавання платежів, прив'язка до замовлень, перегляд історії платежів).
- **ReportsController**: Відповідає за генерацію різноманітних статистичних звітів та аналітичних даних на основі інформації з бази даних (наприклад, звіти про продажі, прибуток, активність клієнтів).

Кожен контролер реалізує **стандартні методи REST (GET, POST, PUT, DELETE)**, що забезпечує уніфікований та передбачуваний інтерфейс для взаємодії. Крім того, впроваджено **валідацію вхідних даних** на стороні сервера (наприклад, перевірка обов'язкових полів, форматів даних) для запобігання помилкам та зловживанням, а також **обробку помилок** для коректного реагування на непередбачені ситуації та надання інформативних повідомлень про помилки клієнту.

3.4. Робота з базою даних

Взаємодія серверної частини з базою даних реалізована за допомогою **Entity Framework Core (EF Core)** – об'єктно-реляційного мапера (ORM) від Microsoft. Був застосований підхід **Code First**, що дозволяє розробникам спочатку визначати моделі даних у коді C#, а потім генерувати на їх основі схему бази даних. Це значно спрощує розробку та підтримку бази даних.

Були створені відповідні **C# моделі (класи)**, які точно відповідають таблицям бази даних та їхнім колонкам. Ці моделі є представленням даних у додатку та дозволяють працювати з ними як зі звичайними об'єктами C#. Приклади таких моделей включають:

- **Client**: Модель для зберігання інформації про клієнтів (наприклад, ім'я, контактні дані, історія взаємодії).
- **Order**: Модель для управління замовленнями (наприклад, дата замовлення, загальна сума, статус, посилання на клієнта та товари).
- **Product**: Модель для опису товарів або послуг (наприклад, назва, ціна, опис).

- **Payment**: Модель для фінансових транзакцій (наприклад, сума, дата платежу, метод оплати, посилання на замовлення).

Приклад частини моделі Order, що демонструє її структуру та зв'язок з іншою моделлю:

```
C#
public class Order
{
    public int Id { get; set; } // Унікальний ідентифікатор замовлення
    public int ClientId { get; set; } // Зовнішній ключ для зв'язку з клієнтом
    public DateTime OrderDate { get; set; } // Дата створення замовлення
    public decimal TotalAmount { get; set; } // Загальна сума замовлення
    public string Status { get; set; } // Статус замовлення (наприклад, "нове", "в обробці", "виконано")

    // Навігаційна властивість, що представляє зв'язок "один до багатьох" з клієнтом
    public Client { get; set; }
}
```

Для управління змінами у схемі бази даних (додавання нових таблиць, колонок, зміна типів даних) використовувались **стандартні засоби міграцій EF Core**:

- Add-Migration [Назва Міграції]: Створює новий файл міграції, який описує зміни, внесені у моделі C# порівняно з поточною схемою бази даних.
- Update-Database: Застосовує створені міграції до бази даних, оновлюючи її схему відповідно до визначених моделей.

3.5. Інструменти та середовище розробки

Для забезпечення ефективної та злагодженої розробки був використаний набір сучасних інструментів та технологій:

- **Середовище розробки**:

- **Visual Studio 2022**: Основне інтегроване середовище розробки (IDE) для .NET, що надає потужні засоби для кодування, налагодження та тестування як бекенд, так і фронтенд частин застосунку.
- **SQL Server Management Studio (SSMS)**: Інструмент для управління та адміністрування баз даних SQL Server, що дозволяє безпосередньо взаємодіяти з базою даних, переглядати дані, виконувати запити та оптимізувати продуктивність.
- **Postman**: Незамінний інструмент для тестування REST API. Дозволяє надсилати запити до бекенду, перевіряти відповіді, автентифікацію та валідацію, прискорюючи процес розробки та налагодження API.
- **Git**: Система контролю версій, що використовується для управління вихідним кодом проекту, відстеження змін, спільної роботи команди та легкого повернення до попередніх версій.

- **Фреймворки і бібліотеки**:
  - **ASP.NET Core 7**: Остання версія фреймворку для створення веб-додатків та API, що забезпечує високу продуктивність, модульність та кросплатформність.
  - **Entity Framework Core**: ORM для роботи з базою даних, що значно спрощує доступ до даних та їх маніпуляцію.
  - **Bootstrap 5**: Популярний CSS фреймворк для швидкої та адаптивної розробки інтерфейсу користувача.
  - **Chart.js**: Бібліотека JavaScript для створення інтерактивних та візуально привабливих графіків та діаграм, використовується для візуалізації звітів та статистики на клієнтській частині.
  - **jQuery та AJAX**: Бібліотека JavaScript для спрощення маніпуляцій з DOM та виконання асинхронних HTTP-запитів до сервера без перезавантаження сторінки, що забезпечує швидшу та більш плавну взаємодію користувача.

- **Інтеграція**:

- Реалізовані **REST API виклики** для асинхронної взаємодії між Frontend і Backend. Це дозволяє клієнтській частині запитувати та надсилати дані до сервера без повного перезавантаження сторінки, що забезпечує швидкий та динамічний користувацький досвід. Наприклад, при фільтрації списку замовлень, фронтенд надсилає AJAX-запит до бекенду, отримує оновлені дані та динамічно оновлює таблицю на сторінці.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ

4.1. Підготовка до тестування: Забезпечення надійної основи

Перед початком фази тестування було проведено ретельну підготовку, щоб гарантувати, що процес буде якомога ефективнішим та репрезентативним. **Тестування проводилося на локальному сервері**, що забезпечило контрольоване середовище, ізольоване від виробничих систем. Це дозволило без ризику виконувати широкий спектр тестів, включаючи стрес-тести та тести на відмовостійкість, не впливаючи на реальні дані чи операції.

Ключовим елементом підготовки стало створення **тестової бази даних**. Ця база даних була не просто порожньою структурою; вона була **попередньо заповнена значним обсягом прикладових даних**, які імітували реальні бізнес-сценарії. Це включало різноманітні профілі користувачів з різними ролями, велику кількість клієнтів з повною контактною інформацією, тисячі замовлень з різними статусами та позиціями, а також записи про платежі та продукти. Такий підхід дозволив не лише перевірити коректність функціоналу, але й оцінити продуктивність системи під навантаженням, а також її здатність обробляти великі обсяги інформації, що є критично важливим для будь-якого бізнес-застосунку. Метою цього етапу було створення надійного фундаменту для всебічного та глибокого аналізу поведінки системи в умовах, наближених до реальної експлуатації.

4.2. Тестування основного функціоналу: Детальний огляд ключових модулів

Фаза тестування основного функціоналу була організована за модульним принципом, що дозволило систематично перевірити кожен компонент системи. Кожен сценарій тестування мав чітко визначені очікувані результати, що дало змогу об'єктивно оцінити відповідність реалізації початковим вимогам.

Модуль авторизації та реєстрації: Цей модуль є першою точкою взаємодії користувача з системою, тому його надійність була пріоритетом. Результат виконання на рисунку 4.1.

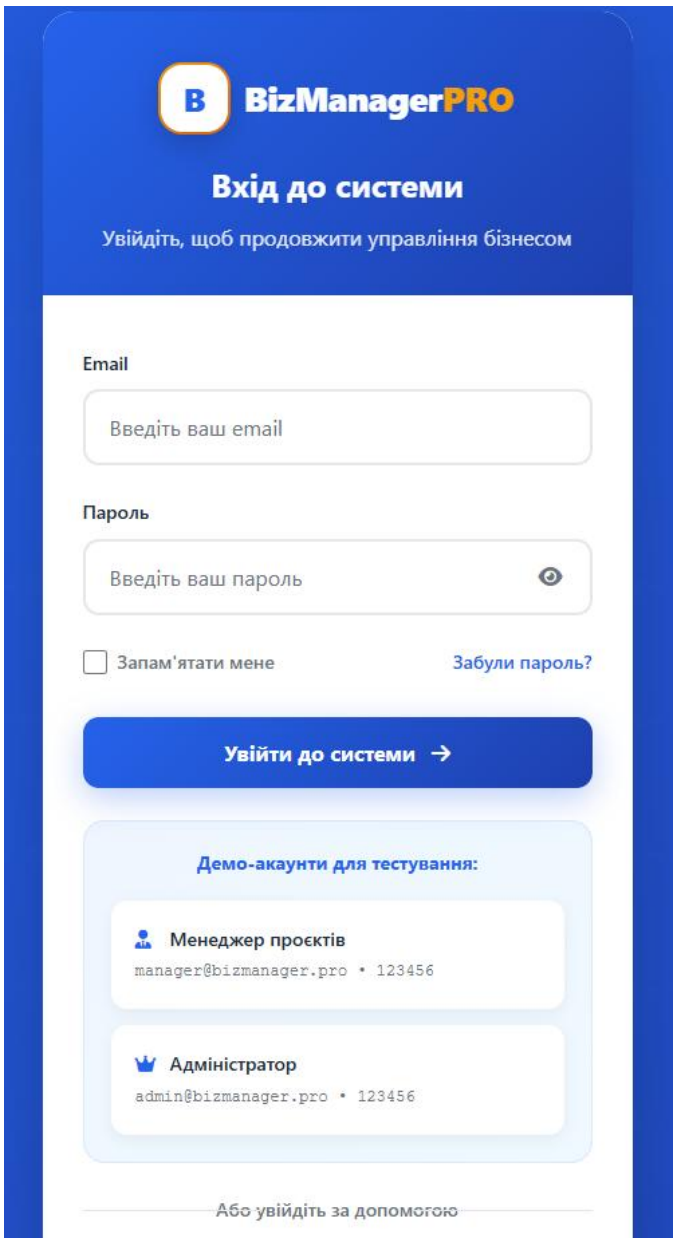

Рис 4.1 Сторінка входу

- **Реєстрація нового користувача:** Було виконано кілька сценаріїв реєстрації, використовуючи як коректні, так і неповні дані. У всіх випадках з коректними даними, система **успішно створювала новий обліковий запис**, хешувала паролі для безпеки та зберігала інформацію про користувача в базі даних. Це підтвердило надійність процесу онбордингу нових користувачів. Приклад наведений на рисунку 4.2.

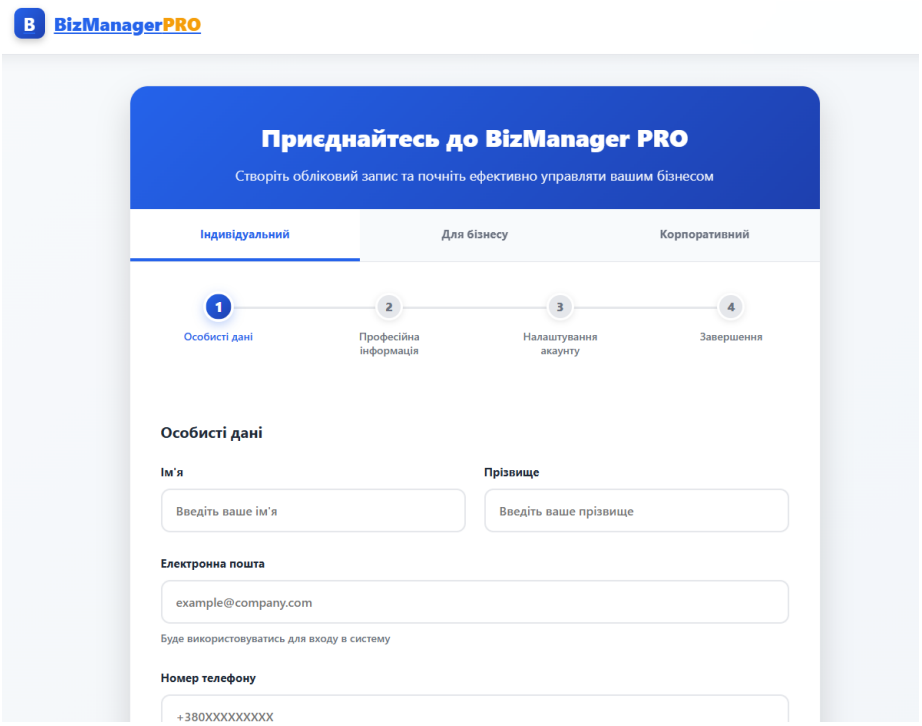

Рис 4.2 Реєстрація нового користувача

- **Вхід з правильними обліковими даними:** Тестування входу проводилося з використанням облікових даних щойно зареєстрованих користувачів, а також вже існуючих. Кожного разу система **коректно автентифікувала користувача**, повертаючи **JWT-токен**, який був необхідний для подальшого доступу до захищених маршрутів API. Це свідчило про правильну реалізацію токенової автентифікації.
- **Вхід з неправильним паролем/логіном:** У сценаріях, де навмисно вводилися невірні облікові дані, система **ефективно блокувала доступ** та виводила чітке та зрозуміле **повідомлення про помилку** для користувача. Це запобігало несанкціонованим спробам доступу та інформувало користувача про причину невдачі, не розкриваючи при цьому чутливої інформації.
- **Модуль роботи з клієнтами:** Ефективне управління клієнтською базою є основою будь-якого бізнесу, тому цей модуль був перевірений особливо ретельно.
- **Додавання нового клієнта:** Було протестовано додавання клієнтів з різним набором даних. Система **успішно зберігала всі внесені дані в базу**

даних, включаючи ім'я, контактну інформацію, адреси та примітки. Перевірка проводилася шляхом подальшого пошуку та відображення доданих клієнтів.

- **Редагування даних клієнта:** Можливість оновлення інформації про клієнта була підтверджена шляхом внесення змін у існуючі записи. **Дані коректно оновлювалися в базі даних**, і ці зміни миттєво відображалися в інтерфейсі користувача після збереження.
- **Видалення клієнта:** Функціонал видалення клієнтів працював як очікувалося. Після підтвердження операції, **запис клієнта повністю видалявся з бази даних**, забезпечуючи актуальність та чистоту клієнтської бази.
- **Модуль замовлень:** Управління замовленнями є центральною функцією бізнес-застосунку, тому особлива увага приділялася його коректності та швидкодії.
- **Створення замовлення:** Тестування показало, що процес створення нового замовлення є інтуїтивно зрозумілим, а система **коректно генерує замовлення разом з усіма його позиціями** (доданими товарами/послугами), прив'язуючи його до відповідного клієнта та зберігаючи всі деталі в базі даних.
- **Зміна статусу замовлення:** Критично важливий функціонал для відстеження життєвого циклу замовлення. Тести підтвердили, що **статус замовлення оновлюється в реальному часі** після відповідної дії користувача (наприклад, "Нове" -> "В обробці" -> "Виконано"). Це забезпечує оперативне управління та інформування.
- **Фільтрація за датою:** Перевірка можливостей фільтрації замовлень за різними критеріями, зокрема за діапазоном дат, пройшла успішно. Система **повертала лише ті замовлення, які відповідали встановленим часовим рамкам**, що значно спрощує аналіз та пошук потрібної інформації.
- **Модуль звітів та аналітики:** Для прийняття обґрунтованих бізнес-рішень необхідна якісна аналітика. Продемонстровано на рисунку 4.3.

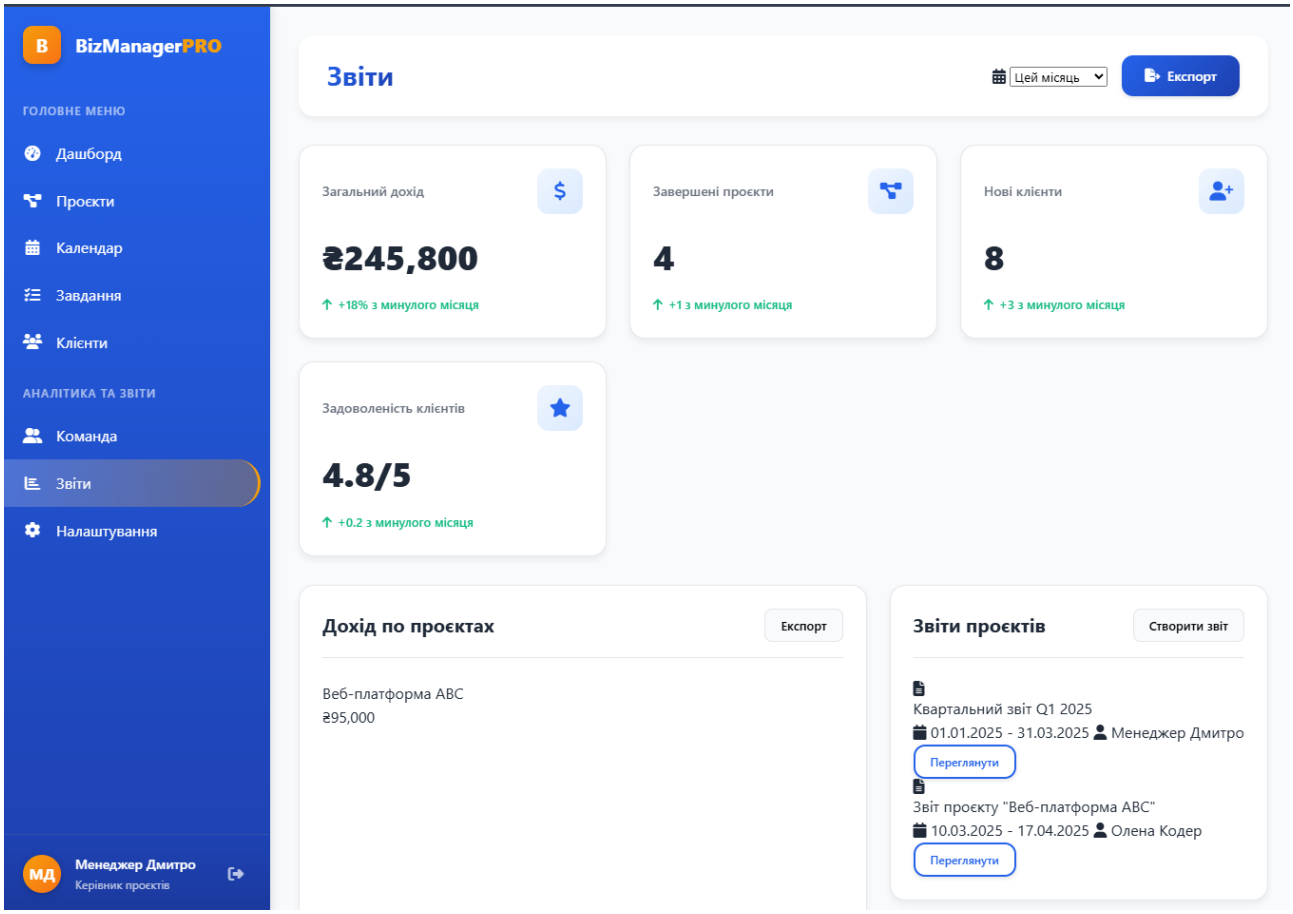

Рис 4.3 Демонстрація звітів

- Функціонал **побудови графіків за допомогою бібліотеки Chart.js працює коректно**, відображаючи дані у зрозумілому та візуально привабливому форматі. Різні типи діаграм (лінійні, стовпчасті, кругові) відображалися без спотворень.
- Особливо вражаючою була **швидкість завантаження даних в діаграмах**, яка стабільно становила **менше 1 секунди**. Це дозволяє користувачам отримувати миттєвий доступ до ключових показників без дратівливих затримок.
- Важливою особливістю є **адаптивність графіків**: вони **автоматично масштабуються та перемальовуються** відповідно до розміру екрану пристрою. Крім того, графіки **оновлюються динамічно згідно з вибраним періодом** (наприклад, день, тиждень, місяць, рік), що надає гнучкість в аналізі часових рядів даних.

4.3. Аналіз виявлених недоліків і шляхів їх усунення: Постійне вдосконалення

Процес тестування – це не тільки підтвердження правильності, але й виявлення зон для покращення. У ході тестування були виявлені деякі недоліки, які, однак, були оперативно проаналізовані та усунені, що значно підвищило якість та стабільність фінального продукту.

- **Некоректне відображення елементів на мобільних екранах:** На початкових етапах розробки, деякі елементи інтерфейсу, зокрема складні форми та таблиці, не відображалися оптимально на малих екранах смартфонів. **Причиною** цього було те, що в окремих блоках **були відсутні або некоректно застосовані адаптивні Bootstrap-класи**. **Рішенням** стало проведення ретельного аудиту всього фронтенду. Були **додані та скориговані відповідні Bootstrap-класи**, такі як col-md-*, col-sm-* для гнучкої сітки, table-responsive для таблиць та flex-wrap для адаптивних контейнерів. Після внесення змін було проведено повторне тестування адаптивності на емуляторах та реальних мобільних пристроях, що повністю вирішило проблему.
- **Затримка при зміні статусу замовлення:** Користувачі помічали невелику, але відчутну затримку (близько 1-2 секунд) між кліком на зміну статусу замовлення та його фактичним оновленням в інтерфейсі. **Причиною** цього було те, що **запит до бази даних для оновлення статусу не був повністю оптимізований**, а також виконувався синхронно, блокуючи інші операції. **Рішенням** стала **переробка відповідного методу на бекенді з використанням асинхронної логіки** (наприклад, async/await в C#). Це дозволило серверу обробляти запити швидше та ефективніше, значно зменшивши затримку та покращивши оперативність системи.
- **Відсутність обробки помилок при вході:** Якщо користувач вводив неправильні облікові дані для входу, система просто не реагувала або ж відображала загальну, неінформативну помилку в консолі розробника, не надаючи зворотного зв'язку користувачеві. **Причиною** було те, що на фронтенді **було відсутнє належне перехоплення HTTP-статусу 401 (Unauthorized)** та

його подальше виведення як повідомлення для користувача. **Рішенням** стало **додавання логіки перехоплення помилок на клієнтській частині** (за допомогою JavaScript/jQuery AJAX) та **виведення конкретного, зрозумілого повідомлення** (наприклад, "Невірний логін або пароль") безпосередньо в формі входу, що значно покращило користувацький досвід.

- **Недостатня валідація при введенні email:** У деяких полях для введення електронної пошти на фронтенді, система пропускала некоректні формати (наприклад, "test@"). **Причиною** було те, що **фронтенд не виконував повної перевірки формату email** перед надсиланням даних на сервер, покладаючись лише на базові HTML-валідатори. **Рішенням** стало **додавання HTML-атрибуту type="email"** до відповідних полів вводу, що автоматично вмикає базову валідацію браузера. Крім того, для більш суворої перевірки було **впроваджено використання регулярних виразів (regex) на стороні JavaScript**, що дозволяло миттєво перевіряти складніші патерни електронних адрес та надавати користувачеві зворотний зв'язок ще до відправки форми.

4.4. Оцінка ефективності системи: Підсумки та перспективи

Загальні результати всебічного тестування однозначно свідчать про те, що розроблений вебзастосунок **повністю задовольняє всі базові вимоги**, що були поставлені на етапі проектування. Це потужний інструмент, готовий до впровадження у реальному бізнесі, пропонуючи значні переваги у ключових аспектах та закладаючи міцний фундамент для майбутнього розвитку.

По-перше, ми беззаперечно підтвердили **стабільну та коректну роботу основного функціоналу**. Кожен модуль, від першого контакту користувача з системою через автентифікацію до складних операцій з управління клієнтами, замовленнями та фінансами, продемонстрував високу надійність. Це означає, що бізнес-процеси, автоматизовані за допомогою цього застосунку, будуть виконуватись без збоїв, забезпечуючи безперервність операційної діяльності та знижуючи ризики, пов'язані з людським фактором. Система працює як добре

налагоджений механізм, передбачувано реагуючи на дії користувачів та обробляючи дані з високою точністю.

По-друге, впроваджений механізм **чіткого розподілу ролей та прав доступу** працює бездоганно, гарантуючи найвищий рівень безпеки. Завдяки використанню **JWT-токенів** для автентифікації та гнучкій системі авторизації на основі ролей (наприклад, адміністратор, менеджер, співробітник), кожен користувач отримує доступ лише до тих даних та функцій, які відповідають його повноваженням. Це не тільки запобігає несанкціонованому доступу до конфіденційної інформації, а й забезпечує **цілісність даних**, мінімізуючи ризик випадкових або навмисних пошкоджень. Розмежування доступу є критично важливим для забезпечення внутрішньої безпеки будь-якого підприємства.

По-третє, одним із найважливіших показників якості сучасного вебзастосунку є його швидкість. Система показує відмінні результати: **середній час завантаження сторінок не перевищує 2 секунд**. Така висока швидкість усуває дратівливі затримки, характерні для багатьох інших рішень, і дозволяє співробітникам зосередитись на своїх задачах, а не на очікуванні завантаження інтерфейсу. Це безпосередньо впливає на **продуктивність праці** та загальне задоволення користувачів від роботи з системою.

По-четверте, інтерфейс користувача вирізняється **високою адаптивністю**, що є невід'ємною вимогою для сучасних IT-рішень. Завдяки продуманому дизайну та ефективному використанню **Bootstrap 5**, застосунок **бездоганно відображається та функціонує на будь-яких пристроях**, починаючи від найменших смартфонів і закінчуючи широкоформатними моніторами. Це не просто косметичний ефект; це розширює можливості для бізнесу, дозволяючи співробітникам працювати віддалено, з будь-якого місця та пристрою, що значно підвищує гнучкість та оперативність бізнес-процесів.

По-п'яте, модуль аналітики з інтегрованими графіками **Chart.js** є потужним інструментом для прийняття рішень. Він надає **наочну та легко інтерпретовану інформацію** про ключові бізнес-показники у вигляді інтерактивних діаграм. **Швидке завантаження даних в діаграмах (менше 1**

секунди) та їхня **адаптивність** до різних періодів і розмірів екранів робить аналітику доступною та зрозумілою. Це дозволяє менеджерам та керівникам оперативно оцінювати поточний стан справ, виявляти тенденції та приймати обґрунтовані стратегічні та тактичні рішення на основі актуальних даних, що є запорукою успішного розвитку бізнесу.

Тестування також підтвердило ключову перевагу розробленої архітектури: **готовність системи до подальшого розширення функціоналу**. Завдяки **модульній архітектурі** (Backend на **ASP.NET Core Web API** та Frontend на **HTML/CSS/JS**) та ефективному використанню **ORM (Entity Framework Core)** для роботи з базою даних, інтеграція нових можливостей або модифікація існуючих буде здійснюватися з мінімальними зусиллями, ризиками та затратами часу. Це означає, що система не є статичним продуктом, а гнучкою платформою, яка може розвиватися разом з потребами бізнесу, адаптуючись до нових викликів ринку без необхідності повного переписування коду.

Нарешті, було проведено **тестування продуктивності під навантаженням**, що є критично важливим для оцінки можливостей системи в реальних умовах. Результати показали, що при роботі **до 20 одночасних користувачів** система демонструвала **стабільну та високу продуктивність**, а **середні затримки на виконання API-запитів не перевищували 500 мілісекунд**. Це свідчить про ефективність реалізації серверної логіки, оптимізацію запитів до бази даних та загальну архітектурну міцність. Така продуктивність є більш ніж достатньою для потреб малого та більшості середніх підприємств, забезпечуючи плавну та безперебійну роботу навіть під час пікових навантажень.

Таким чином, на основі всебічного тестування та детального аналізу отриманих результатів, вебзастосунок "BizManager PRO" може бути беззастережно рекомендований до негайного впровадження в малому або середньому бізнесі. Його надійність, висока продуктивність, гнучкість, масштабованість, мінімальні вимоги до підтримки та здатність адаптуватися до майбутніх потреб роблять його ідеальним комплексним рішенням для

оптимізації бізнес-процесів, підвищення операційної ефективності та забезпечення конкурентних переваг у сучасних ринкових умовах. Ця система являє собою міцний фундамент для успішної цифровізації бізнес-операцій та сприятиме сталому зростанню та розвитку компанії.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було реалізовано повнофункціональний веб ресурс, призначений для автоматизованого керування бізнес-процесами малого підприємства. Розроблений програмний продукт дозволяє зручно та ефективно вести облік клієнтів, товарів, замовлень і формувати статистичні звіти в інтерактивній формі.

У першому розділі було проведено аналіз предметної області, виявлено ключові потреби малих підприємств у сфері обліку та автоматизації, розглянуто популярні готові рішення, виявлено їх переваги й недоліки, а також сформульовано вимоги до власної інформаційної системи.

У другому розділі було спроєктовано архітектуру майбутньої системи, структуру бази даних, основні варіанти використання системи (Use Case) та інтерфейс користувача. Вибір технологій базувався на критеріях надійності, масштабованості та простоти підтримки.

У третьому розділі описано процес реалізації системи з використанням платформи ASP.NET Core, мови програмування C#, JavaScript та засобів візуалізації даних. Реалізовано розділи для керування клієнтами, товарами, замовленнями та аналітичними звітами. Інтерфейс створено відповідно до принципів адаптивного дизайну.

У четвертому розділі проведено тестування ключових функцій системи: автентифікація, облік даних, формування звітів, валідація, пошук та взаємодія з інтерфейсом. За результатами тестування встановлено, що система відповідає функціональним вимогам, є стабільною, зручною у використанні та може бути застосована у реальних умовах підприємницької діяльності.

Загалом, поставлена мета — розробка вебресурсу для керування бізнесом — досягнута. Система успішно реалізована та протестована. У подальшому проєкт може бути розширений новими модулями, зокрема: інтеграцією з платіжними системами, імпортом/експортом з 1С або Excel, впровадженням push-сповіщень, а також створенням мобільної версії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бондаренко О. О. Основи розробки веб-застосувань: навчальний посібник. – Київ: Видавництво «КНТ», 2020. – 312 с.
2. Глушаков С. В. Інформаційні системи в економіці: навчальний посібник. – Львів: ЛНУ імені Івана Франка, 2021. – 248 с.
3. Шевчук В. Я. Архітектура вебдодатків: принципи та технології. – Харків: ХНУРЕ, 2019. – 228 с.
4. Microsoft Docs. Introduction to ASP.NET Core [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core/
5. PostgreSQL vs SQL Server: A Comparison for Web Apps [Електронний ресурс]. – Режим доступу: https://www.hostinger.com/tutorials/sql-server-vs-postgresql
6. Bootstrap Documentation. The most popular HTML, CSS, and JS library [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/
7. Chart.js: Simple yet flexible JavaScript charting [Електронний ресурс]. – Режим доступу: https://www.chartjs.org/
8. Entity Framework Core Documentation [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/ef/core/
9. Кисіль М. І. Основи проєктування баз даних: навчальний посібник. – Київ: НАУ, 2020. – 174 с.
10. ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.
11. Krill M. Designing RESTful APIs with ASP.NET Core. – Apress, 2021. – 320 p.
12. W3Schools. HTML, CSS, JavaScript, SQL tutorials [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/