МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ
Фізико-математичний факультет
Кафедра інформатики та прикладної математики

«Допущено до захисту»
Завідувач кафедри
_____ Моісеєнко Н.В.
«___» _____ 2024 р.

Реєстраційний № _____
«___» _____ 2024 р.

ВЕБ-СЕРВІС ДЛЯ ОРГАНІЗАЦІЇ НОТАТОК

Кваліфікаційна робота
студента групи І-20
ступінь вищої освіти «бакалавр»
спеціальності
014 Середня освіта (Інформатика)
Яковецького Миколи Едуардовича
Керівник **Моісеєнко М.В.**
к. пед. н., старший викладач

Оцінка:
Національна шкала _____

Шкала ECTS ____ Кількість балів _____

Члени комісії:

_____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

Кривий Ріг – 2024

ЗАПЕВНЕННЯ

Я, Яковецький Микола Едуардович, розумію і підтримую політикум Криворізького державного педагогічного університету з академічної доброчесності. Запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Я не надавав і не одержував недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають покликання на відповідне джерело. Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Криворізького державного педагогічного університету ознайомлений. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

ЗМІСТ

ВСТУП

Актуальність теми. В епоху стрімкого розвитку інформаційних технологій та зростаючих обсягів інформації, ефективне управління завданнями та часом набуває все більшої ваги. Люди постійно стикаються з проблемою відслідковування та виконання різноманітних завдань у роботі та особистому житті.

Згідно дослідження А. Сміта [1] , лише 14% людей ефективно планують та дотримуються запланованих справ. Одним з найпоширеніших способів вирішення цієї проблеми є використання додатків-органайзерів, які дозволяють керувати списком завдань. За прогнозами, ринок подібних додатків досягне $9.7B до 2025 року зі щорічним зростанням у 13.6%.

Крім підвищення особистої продуктивності, використання органайзерів також полегшує управління завданнями та проектами у бізнесі та корпоративному середовищі. За даними досліджень [2-6], компанії що використовують спеціалізовані додатки для планування, звітності та координування завдань, на 20% ефективніше виконують проекти.

Питаннями тайм-менеджменту, розвитком його теорії та практики займались іноземні дослідники: Дж. Клеменс, Далрімпл С.[7], М. Роббінс [8], Б. Трейсі [9] та ін. Серед українських науковців дослідженням ефективного управління часу присвячені роботи А.О. Гаврилової та Л.Л. Калініченко [10], С.Б. Іваницької, Т.О. Галайди та Р.М. Толочій [11], Х.К. Журавльової [12], В.О. Євдокимова [13], Г.І. Євтушенко [14], Н.М. Крукевич [15], Т.В. Лазоренко [16], Л.І. Скібіцька [17], О. Й. Янцаловський [18] та ін.

З огляду на активний розвиток ринку SaaS та хмарних сервісів, створення подібних веб-додатків є перспективним напрямком, що поєднує сучасні технології з ростучим попитом користувачів. Крісс-платформеність та масштабованість також дозволяють охопити велику аудиторію з мінімальними витратами.

Отже, розробка власного веб-додатку для управління списками справ є актуальним завданням. Це дозволить значно полегшити планування та підвищити продуктивність як окремих користувачів, так і цілих організацій. Також, подібні органайзери створюють додаткову цінність для стартапів та компаній, що пропонують хмарні сервіси.

Все вищесказане визначило **мету дослідження:** розробити та реалізувати веб-сервіс для організації нотаток.

Згідно до мети дослідження були поставленні наступні **завдання дослідження:**

1. вивчення літератури на тему тайм-менеджмента та використання електронних органайзерів;
2. вивчення переваг та недоліків популярних застосунків-органайзерів;
3. проєктування та програмна реалізація веб-сервісу для організації нотаток (створення базової структури, моделей та бази даних; створення API на ASP.NET Core; розробка Angular клієнта; взаємодія із задачами; UI/UX оптимізація) та перевірка його працездатності.

Об’єкт дослідження – інформаційні технології в тайм-менеджменті.

Предмет дослідження – розробка веб-сервісу для організації нотаток з використанням технологій ASP.NET Core та Angular.

Практична цінність дослідження полягає в створенні функціонального та ефективного веб-сервісу для організації нотаток та управління списками справ. Основна цінність полягає у впровадженні сучасних технологій та найкращих практик розробки для створення продуктивного та зручного використання веб-сервісу.

Структура та обсяг роботи: робота складається із вступу, трьох розділів, висновків та списку використаних джерел з 21 найменування; містить 32 сторінки тексту, 10 рисунків. Загальний обсяг 34 сторінки.

РОЗДІЛ 1
ТАЙМ-МЕНЕДЖМЕНТ ТА ВИКОРИСТАННЯ В НЬОМУ ЗАСТОСУНКІВ-ОРГАНАЙЗЕРІВ

1.1. Тайм-менеджмент: найрозповсюдженіші проблеми

Тайм-менеджмент з англійської мови перекладається як управління часом. Під цим поняттям в основному розуміють раціональний розподіл та ефективне використання часу, тобто строгий облік і чітке планування часу. На жаль, час є невідновлюваним ресурсом, впливати на час неможливо, але важливо використовувати його раціонально, плануючи і виконуючи плани. Час не можна збільшити в об'ємі, його можна лише використати з максимальною ефективністю, досягаючи поставлених цілей за мінімальний проміжок часу

Важливим завданням є оптимізація будь-якої діяльності шляхом визначення пріоритетів, мінімізації часу на виконання завдань. Перевантаженість та накопичення незавершених справ породжує негативні емоції та нервове напруження, що приводить до стресів та синдрому хронічної втоми. Такий психологічний стан зменшує продуктивність роботи. Тайм-менеджмент – це не тільки правильний розподіл власного часу, а й керування чинниками, які впливають на нього, тобто робоча атмосфера, стосунки з колегами, організація відпочинку, побуту тощо [11]. Однією з ключових проблем тайм-менеджменту є прокрастинація, відкладання виконання будь-яких справ на останній момент, незважаючи на негативні наслідки. «Прокрастинацію можна пояснити різними причинами, такими як лінь чи недостатність мотивації. Це явище також можна пояснити страхом провалу або навіть успіху, оскільки більшість людей важко сприймають зміни в житті, інакше кажучи, страх вийти з так званої «зони комфорту» примушує людей навіть несвідомо саботувати» [19].

В своїй роботі Роббінс М. [8] припускає, що прокрастинація є особливою формою захисту від стресу і тому, якщо людина проявляє ознаки цього явища, то потрібно зрозуміти причини цього, а не звинувачувати себе.

Часто люди відкладають справи через те, що їм важко просто почати виконання задачі. Роббінс М. пропонує для того, щоб цей процес не був такий важкий, створити ритуал, що буде налаштовувати на робочий лад. У своїй книзі [8] вона запропонувала правило «5 секунд», сутність якого полягає в тому, що коли треба почати виконання завдання, необхідно порахувати від 1 до 5 і прийматися за справу. Авторка вважає, що таким чином можна подолати прокрастинацію. Робиться акцент на тому, що головне – це почати виконувати те, що необхідно; не варто ставити перед собою недосяжні цілі, замість цього потрібно зробити якийсь маленький крок.

Переважна більшість досліджень вказують на необхідність і першочерговість планування часу, що є одним із важливих етапів управління ним. Звичайно, планування допомагає краще уявляти обсяги робіт і виконати все необхідне, нічого не упустивши.

Планування є необхідним для будь-кого, оскільки не варто завжди покладатися на свою пам’ять. Саме письмове відображення свого розпорядку дає можливість тримати руку на пульсі всіх поточних справ і приділяти їм увагу у відповідний час.

Планування – одна з функцій менеджменту, яка полягає в тому, щоб знаходити шляхи досягнення цілей та деталізувати їх. Саме менеджер несе відповідальність за результативність діяльності підприємства, оскільки він організовує власний час і час інших працівників колективу.

На жаль, значний відсоток часу в практичній діяльності менеджера витрачається марно, що пояснюється самою природою управлінської діяльності, яка є творчою та безпосередньо пов’язана із взаємовідносинами з людьми.

Варто зауважити, що досвідчені менеджери вказують на першочергову необхідність саме аналізу свого часу з виокремленням непродуктивних втрат і зведенням їх до мінімуму. Планування ж, як вказують практики, має стати наступним етапом організації та використання часу.

1.2. Опис популярних застосунків-органайзерів

На сьогодні існує велика кількість застосунків для управління списками справ та завдань. Розглянемо детальніше декілька найбільш популярних з них.

Todoist – потужний та зручний органайзер із мобільними та веб додатками (todoist.com).

- Рік заснування: 2007;
- Розробник: Doist;
- Кількість активних користувачів: понад 30 млн;
- Особливості: проекти, фільтри, нагадування, календар;
- Технології: власний API Sync, мобільні та веб додатки;
- Інтеграції: пошта, календар, служби файлів;
- Бізнес-модель: freemium, платні плани для бізнесу.

Дозволяє створювати проекти, категоризувати завдання, налаштовувати нагадування. Має інтеграцію з поштовими сервісами та календарем. Підтримує спільну роботу над проектами в командах. Має вбудований календар з drag-and-drop інтерфейсом для планування. Розширені можливості для налаштування нагадувань (дата, час, місце). Підтримує експорт/імпорт даних. Доступні мобільні та десктопні додатки з синхронізацією. Платні плани дозволяють розблокувати додаткові функції для командної роботи. Кількість активних користувачів - понад 30 млн по всьому світу. Заснований в 2007 році, отримав численні нагороди в сфері продуктивності. Використовує власний протокол синхронізації Todoist Sync. Має API для інтеграції з сторонніми сервісами.

- Microsoft To Do – це оновлена версія популярного Wunderlist, яка була викуплена Microsoft. Також доступний у вигляді мобільних та веб додатків. Рік заснування: 2017;
- Розробник: Microsoft Corporation;
- Кількість активних користувачів: понад 10 млн;
- Особливості: списки, нагадування, обмін списками;

- Технології: Azure, Exchange;
- Додатки для OS Інтеграції: Microsoft 365, Outlook, Teams;
- Бізнес-модель: частина платформи Microsoft 365.

Дозволяє створювати завдання, списки, нагадування з геолокацією, а також ділитися списками та поставити завдання іншим користувачам. Має вбудовану підтримку Microsoft Exchange, що дозволяє автоматизувати створення задач на основі email. Надає статистику продуктивності за кількістю виконаних завдань. Дозволяє додавати нотатки, зображення, файли до завдань. Наявна функція автоматичного сортування задач My Day, що виводить за пріоритетом. Наслідує популярний Wunderlist, придбаний Microsoft. Найбільш тісно інтегрований з екосистемою Microsoft - Outlook, Teams, Planner. Використовує хмарні сервіси Azure. Має додатки для iOS, Android та Windows.

Any.do – пропонує чистий дизайн з фокусом на продуктивності та управлінні часом (any.do).

- Рік заснування: 2011
- Розробник: Any.do
- Кількість активних користувачів: понад 10 млн
- Особливості: голосовий ввід, швидке захоплення, нагадування Технології: мобільні та веб додатки
- Інтеграції: Slack, Zapier, Salesforce, Alexa
- Бізнес-модель: freemium

Можливості включають гнучке налаштування нагадувань, підтримку голосового вводу, синхронізацію між пристроями. Any.do також добре оптимізований для швидкого захоплення завдань. Є built-in підтримка для роботи зайнятих людей - диктування задач, нагадування про події в календарі та ін. Платні плани додають можливості автоматизації задач, розширені фільтри та звіти. Доступний на Web, iOS, Android. Понад 100 млн завантажень. Фокус на швидкості та UX. Вбудована підтримка природної мови та голосового вводу завдяки технології розпізнавання мовлення. Інтегрується з

Slack, Zoom, Salesforce та ін. Має безліч шаблонів та підказок для швидкого старту.

Trello – це канбан дошка для організації завдань візуальним способом (trello.com).

- Рік заснування: 2011
- Розробник: Trello, Inc. (Atlassian)
- Кількість активних користувачів: понад 50 млн
- Особливості: канбан-дошки, картки,колонки, чеклисти
- Технології: React, Node.js, MongoDB, мобільні та веб додатки
  Інтеграції: 500+ служб через API, Zapier
- Бізнес-модель: freemium, платні плани для бізнесу

Дозволяє створювати колонки для різних статусів та етапів роботи, переміщати картки між ними, прикріплювати файли, зображення, checklist'и. Підтримує командну роботу. Дозволяє створювати окремі робочі простори, переміщати картки задач між колонками status'ів, призначати виконавців, коментувати. Є можливості фільтрації, групування, сортування карток, налаштування автоматизації. Інтегрується з багатьма зовнішніми сервісами. Доступний безкоштовний обмежений функціонал. Понад 50 млн зареєстрованих користувачів. Популярний в IT та медіа сферах. Використовує React та Node.js + MongoDB в стеку. Має розвинене API та понад 500 інтеграцій зі сторонніми сервісами. Доступний веб-сайт та мобільні додатки.

1.3. Переваги та недоліки популярних органайзерів

Todoist має широкі можливості для структурування та категоризації завдань, гнучкі опції фільтрації та сортування. Проте інтерфейс перевантажений налаштуваннями і додатковим функціоналом, що може відволікати користувача. Також спостерігаються деякі проблеми синхронізації між пристроями.

Microsoft To Do простий у використанні та добре інтегрується з екосистемою Microsoft 365. Втім функціонал обмежено базовими опціями

управління списками та відсутня розширена аналітика. Немає гнучких налаштувань пріоритетів чи контекстів для завдань.

Any.do фокусується на швидкому захопленні завдань та оптимізації інтерфейсу, проте має досить слабку функціональність для структурування складних проектів чи розширеного планування. Також потребує постійного підключення до інтернету для роботи.

Trello чудово підходить для візуалізації робочих процесів, проте не надає зручних інструментів для особистої організації та аналізу власних задач. Складно відстежувати прогрес виконання індивідуальних завдань у часі.

Детальний аналіз можливостей додатків

Todoist:

- Фільтри та категоризація: проекти, теги, дати, пріоритети;
- Сортування: алфавіт, дата, пріоритет;
- Аналітика: звіти продуктивності, графіки, час виконання;
- Командна робота: ролі доступу, управління учасниками;
- Інтеграція: пошта, календар, Google Drive, Dropbox;
- Мобільні додатки: iOS, Android, сповіщення;
- Офлайн режим: доступ до даних без інтернету;
- API: стороння інтеграція через REST API.

Microsoft To Do:

- Фільтри: за датами, пріоритетами, типами
- Сортування: за алфавітом, датою
- Аналітика: базова статистика виконання
- Командна робота: спільний доступ до списків
- Інтеграція: Microsoft 365, Outlook, Teams
- Мобільні додатки: iOS, Android, Windows
- Офлайн режим: часткова підтримка офлайн
- API: обмежене для сторонньої інтеграції

Any.do:

- Фільтри та категоризація: за проектами, тегами, датами
- Сортування: за пріоритетом, датою
- Аналітика: базова статистика виконання
- Командна робота: спільний доступ до списків
- Інтеграція: Slack, Zoom, Salesforce, Alexa, Zapier
- Мобільні додатки: iOS, Android
- Офлайн режим: немає підтримки офлайн
- API: є REST API для інтеграції
- Голосовий ввід: природномовний ввід завдань

Trello:

- Фільтри та категоризація: за дошками, списками, тегами
- Сортування: ручне сортування карток
- Аналітика: обмежена, кількість карток по статусам
- Командна робота: спільний доступ, коментарі
- Інтеграції: 500+ сервісів через Zapier та API
- Мобільні додатки: iOS, Android
- Офлайн режим: обмежена функціональність
- API: розвинене REST API
- Візуалізація: канбан-дошки та картки

Отже, наявні рішення або занадто прості і не надають достатньо контролю та аналітики, або навпаки перевантажені можливостями, що розсіює увагу. Також багато хто покладається на постійне інтернет-з’єднання, що не завжди зручно.

Ідеальний органайзер має поєднувати простоту у використанні, гнучкі можливості структурування та масштабування від особистих до групових проектів. При цьому забезпечувати як продуктивність в роботі онлайн, так і автономний режим без підключення до мережі.

На основі аналізу можна зробити висновок, що існує ніша на ринку для створення органайзера нового покоління, який поєднує в собі переваги і уникає недоліків наявних застосунків.

Проаналізовані додатки-органайзери мають потужний базовий функціонал для управління особистими та груповими задачами. Проте, можна виділити декілька недоліків:

- Обмежені можливості для гнучкої фільтрації та сортування задач за різними критеріями. Бракує розширених опцій аналітики продуктивності.
- Недостатня підтримка офлайн роботи. Деякі додатки потребують постійної наявності інтернет-з’єднання.
- Обмежена інтеграція з популярними сервісами та програмами для гнучкої автоматизації задач.
- Надлишкові або незручні функції в деяких додатках.

На основі цього аналізу можна зробити висновок, що існує ніша для створення більш гнучкого та функціонального органайзеру. А саме, реалізувати:

- Розширені можливості фільтрації, групування та звітності задач
- Повноцінну офлайн синхронізацію та доступність
- Інтеграцію з популярними сервісами через API
- Зручний і оптимізований інтерфейс без зайвого функціоналу

Такий підхід дозволить створити сучасний високотехнологічний органайзер з вдосконаленим користувацьким досвідом.

Висновки до розділу 1

У першому розділі було детально проаналізовано декілька популярних застосунків-органайзерів: Todoist, Microsoft To Do, Any.do та Trello.

Розглянуто основні функціональні можливості цих додатків: створення та управління завданнями, категоризація та фільтрація, встановлення

нагадувань, аналітика та звітність, підтримка зовнішніх інтеграцій, робота в командах.

Виділено ключові переваги проаналізованих рішень: наявність базового набору можливостей, гнучкі опції фільтрації та сортування завдань, часткова підтримка офлайн роботи, інтеграція з іншими сервісами та наявність мобільних додатків.

Разом з тим виявлені певні недоліки: обмежені опції аналітики та звітності, складнощі синхронізації даних між пристроями, недостатня гнучкість в налаштуваннях та автоматизації, перевантаженість деяких інтерфейсів.

Таким чином, є актуальною задача створення вдосконаленого органайзера, який враховуватиме недоліки наявних рішень і реалізує розширені можливості аналітики, синхронізації, автоматизації та зручності користування. Це дозволить задовольнити потреби користувачів та забезпечити ефективне управління завданнями.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ВЕБ-СЕРВІСУ ДЛЯ ОРГАНІЗАЦІЇ НОТАТОК

2.1. Вимоги до функціоналу

Однією з ключових вимог до додатку є можливість створювати, змінювати та видаляти завдання, щоб користувачі могли гнучко управляти своїми списками справ. Також має бути функціонал для позначення статусу завдання - виконано чи ні, щоб відстежувати прогрес.

Ще одним важливим функціоналом є категоризація та організація завдань за допомогою та списків. Це дозволить групувати пов’язані завдання та краще орієнтуватися в інформації. Додаток має надавати зручні засоби для фільтрації та сортування задач.

Для планування також важливо встановлювати нагадування та терміни виконання завдань. Користувачі повинні мати можливість задавати дедлайни та отримувати сповіщення.

Щодо звітності та аналітики, бажано реалізувати функції перегляду статистики виконання задач, продуктивності тощо. Також потрібен експорт даних у зручні формати.

З точки зору якісних показників, пріоритетним є забезпечення високої швидкодії, безперебійної роботи та безпеки даних користувачів. Також проект має бути легко масштабованим та сумісним з різними пристроями і системами.

Поряд з базовою фільтрацією за статусом, типом і терміном, потрібно забезпечити розширений пошук і сортування за датою створення, останніми змінами, автором, пріоритетом. Це дасть змогу аналізувати задачі з різних перспектив.

Для контролю продуктивності доцільно вести статистику щоденно виконаних задач та час витрачений на їх виконання. Такі метрики у комплексі дадуть уявлення про ефективність роботи користувачів в додатку.

Загалом функціональні та нефункціональні вимоги спрямовані на створення продуктивного, інтуїтивного та безпечного додатку для управління особистими та груповими задачами.

2.2. Обґрунтування вибору технологій для клієнту і серверу

Для реалізації серверної частини було обрано платформу ASP.NET Core - кросплатформенний open-source фреймворк від Microsoft, що дозволяє створювати веб-API та веб-застосунки на .NET [20, 21]. Він має вбудовану підтримку Dependency Injection для модульності коду, автоматичне кешування відповідей для підвищення продуктивності, та ряд інших корисних функцій. ASP.NET Core добре масштабується і працює в різних хмарних середовищах.

Для збереження даних використовується технологія Entity Framework. Ця структура значно прискорює розробку, дозволяючи працювати з базою даних на рівні об'єктів, а не SQL-запитів. Це зменшує час, необхідний для написання і підтримки коду. Завдяки автоматичній генерації класів і зв'язків, розробник може зосередитися на бізнес-логіці, а не на написанні SQL-запитів. Вона підтримує різні реляційні бази даних (SQL Server, PostgreSQL, MySQL та ін.), що дозволяє легко змінювати або використовувати кілька баз даних без значних змін у коді застосунку. Також, вона забезпечує механізм міграцій, який дозволяє легко змінювати структуру бази даних та автоматично застосовувати ці зміни у різних середовищах (розробка, тестування, продуктивність).

Для клієнтської частини було обрано фреймворк Angular - потужний інструмент для створення односторінкових застосунків на TypeScript. Він має вбудовану підтримку для форм, маршрутизації, анімацій та багато інших можливостей. Компонентний підхід Angular дозволяє легко масштабувати складні frontend проекти.

Для стилізації елементів використовується популярний CSS фреймворк Bootstrap. Він надає готові класи та компоненти для створення адаптивного інтерфейсу, що значно пришвидшує розробку UI. Також був використаний

Materialize - це CSS і JavaScript фреймворк для швидкого створення адаптивних інтерфейсів на основі правил Material Design від Google.

Він містить готові HTML та CSS компоненти з поліпшеними стилями та анімаціями: кнопки, картки, навігацію, форми, модальні вікна, таблиці та багато іншого. Це значно пришвидшує розробку UI та робить його більш привабливим.

Materialize має вбудовану сітку та систему адаптивності, що дозволяє легко створювати responsive дизайн під різні пристрої та розміри екрану.

Також фреймворк підтримує розширення функціоналу за допомогою JavaScript плагінів - для датапікерів, слайдерів, паралакс ефектів та ін.

2.3. Опис бази даних та структури колекцій

Вибір Entity Framework (EF) як технології для реалізації бази даних у клієнт-серверному веб-застосунку може бути обґрунтований рядом переваг і особливостей цієї ORM (Object-Relational Mapping) технології.

EF є частиною .NET системи, що забезпечує зручну роботу з іншими технологіями і бібліотеками цього середовища. Це особливо корисно для розробників, які вже знайомі з .NET платформою.

Також варто зазначити EF дозволяє використовувати LINQ (Language Integrated Query) для запитів до бази даних, що робить код більш зрозумілим і підтримуваним. Автоматично відстежує зміни в об'єктах і генерує відповідні SQL-запити для оновлення бази даних. Дозволяє легко працювати зі зв'язками між таблицями, використовуючи навігаційні властивості, що спрощує доступ до пов'язаних даних.

EF підтримує різні підходи до розробки бази даних (Code First, Database First, Model First), що дозволяє вибрати найзручніший метод для конкретного проекту.

Ці переваги роблять Entity Framework привабливим вибором для реалізації бази даних у клієнт-серверному веб-застосунку, забезпечуючи баланс між продуктивністю, зручністю та можливостями масштабування.

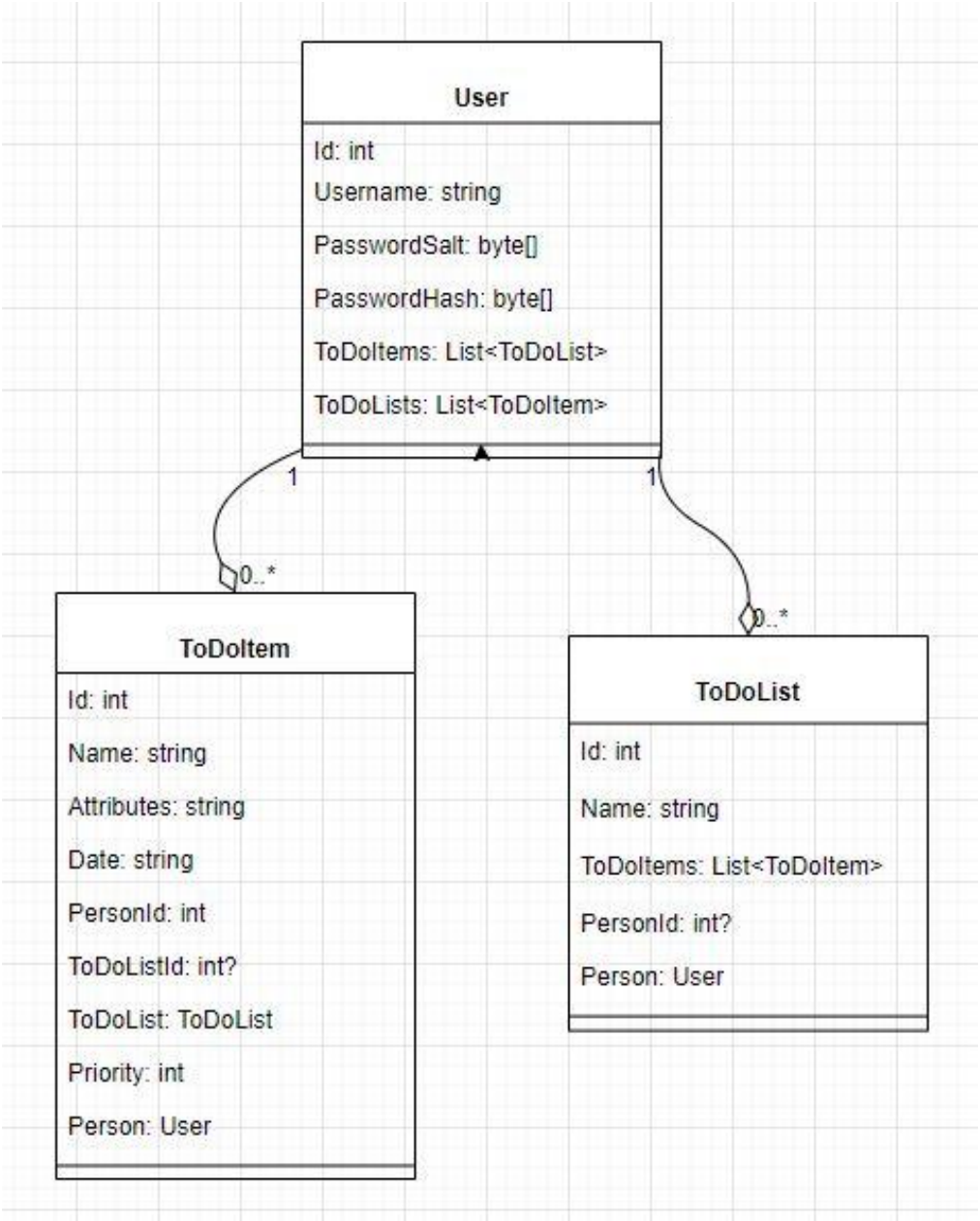

Рис.2.1. Діаграма класів бази даних.

Опис класів бази даних

Клас User (Рис.2.2) - зберігає дані про користувачів системи:

- Id - унікальний ідентифікатор
- Username - ім'я користувача
- PasswordSalt – пароль для шифровки
- PasswordHash - хеш паролю

Проміжні змінні для зв'язку Items та Lists в відношенні багато-до-багатьох:

- ToDoItems - посилання на завдання
- ToDoLists - посилання на список

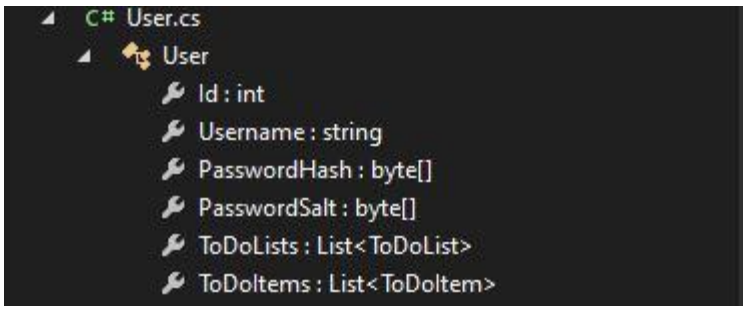

Рис.2.2. Клас User

Клас ToDoItem (Рис.2.3)- зберігає інформацію про завдання:

- Id - унікальний ідентифікатор
- Name - назва завдання
- Attributes - детальний опис
- Date – дата створення
- PersonId - ідентифікатор користувача, що створив завдання
- ToDoListId - ідентифікатор списку до якого прив'язана задача
- Priority - пріоритет (низький, середній, високий

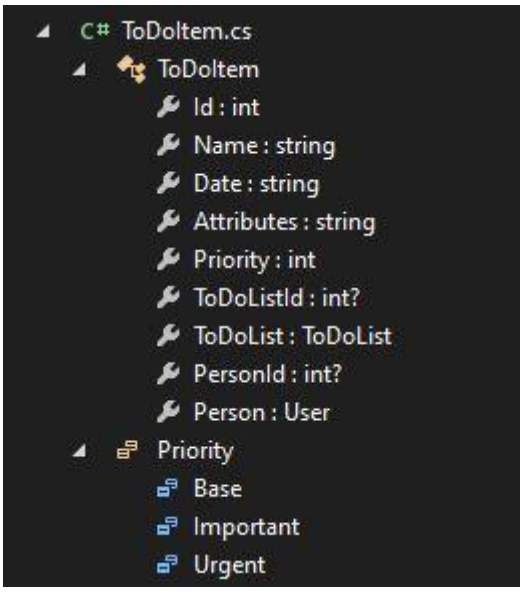

Рис.2.3. Клас ToDoItem.

Клас ToDoList (Рис.2.4) - зберігає дані про списки завдань:

- Id - унікальний ідентифікатор
- Name - назва списку
- PersonId - власник списку
- List<ToDoItem> ToDoItems – список який містить колекцію завдань прив’язаних до списку

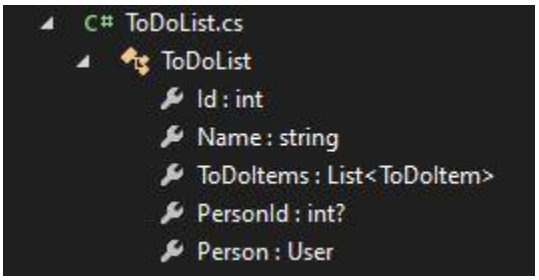

Рис.2.4. Клас ToDoList

Така структура даних дозволяє гнучко зберігати всю необхідну інформацію для організації завдань та списків, при цьому забезпечуючи цілісність та оптимальну швидкодію.

Висновки до розділу 2

У 2 розділі були детально розглянуті ключові аспекти розробки додатку: вимоги до функціоналу, вибір технологій та опис бази даних.

Щодо функціональних вимог, то були виділені такі основні можливості, які має реалізувати додаток: створення, зміна та видалення завдань для гнучкого управління списками справ; позначення статусу завдання, щоб відстежувати прогрес; категоризація та групування завдань за допомогою списків для кращої організації; фільтрація та сортування задач; встановлення нагадувань і термінів виконання; перегляд статистики та звітності про продуктивність.

Для реалізації серверної частини було обрано ASP.NET Core як найкращу cross-platform платформу для створення веб-API на .NET. Це забезпечить гнучкість, модульність та високу продуктивність backend. Для

збереження даних буде використано Entity Framework - структуру об'єктно-реляційного відображення з відкритим кодом для ADO.NET, яка оптимізована для веб-застосунків. На frontend буде застосовано Angular для швидкої розробки односторінкового застосунку з використанням TypeScript. Для стилізації елементів використовуватимуться Bootstrap та Materialize.

Опис структури бази даних містить сутності користувачів, завдань та списків. Між ними налаштовані необхідні зв'язки для реалізації функціоналу додатку. Така структура дозволить надійно зберігати і ефективно обробляти дані в додатку.

Отже, в розділі визначені ключові вимоги до функціоналу, обрані оптимальні технології реалізації та спроектована база даних. Це створює необхідну основу для подальшої розробки гнучкого, масштабованого та продуктивного додатку для організації завдань.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-СЕРВІСУ ДЛЯ ОРГАНІЗАЦІЇ НОТАТОК

3.1. Реалізація серверної частини

Для реалізації серверної частини було обрано ASP.NET Core - кроссплатформений веб-фреймворк від Microsoft на мові програмування C#. Він дозволяє створювати RESTful API з використанням .NET та .NET Core.

Серверна частина реалізована за допомогою архітектурного патерну MVC (Model-View-Controller). Це дозволяє розділити бізнес-логіку, дані та представлення. Моделі описують дані та бізнес-правила, контролери обробляють запити, а представлення відповідають за відображення даних.

Контролери виконують роль посередника між моделями та представленнями. Вони отримують HTTP запити, викликають необхідну бізнес-логіку моделей, формують відповідь та повертають результат у заданому форматі (JSON).

Представлення в даному проекті реалізовані у вигляді серіалізаторів, що конвертують об'єкти моделей в JSON для відправки клієнту. Це дозволяє абстрагуватися від конкретного представлення даних.

Для створення REST API використовується автоматичне генерування маршрутів на основі атрибутів над класами контролерів та методами дій. Це спрощує налаштування routing та дозволяє швидко створювати CRUD операції.

Для роботи з даними використовується ORM фреймворк Entity Framework Core з підтримкою Code First. Він забезпечує зручний об'єктно-орієнтований доступ до бази даних на основі моделей даних в коді.

Також реалізована аутентифікація та авторизація користувачів за допомогою JWT токенів. Це дозволяє безпечно ідентифікувати користувачів та надавати різні рівні доступу до API.

Загалом архітектура backend дозволяє гнучко та надійно реалізувати всі необхідні функції для роботи із завданнями та списками. Використання ASP.NET Core забезпечує швидку розробку та масштабованість.

3.2. Розробка клієнтського інтерфейсу

Для розробки клієнтського інтерфейсу було обрано фреймворк Angular. Він дозволяє створювати односторінкові застосунки з використанням TypeScript.

Інтерфейс реалізований у вигляді набору компонентів, що відповідають за окремі функціональні частини програми. Це дає можливість легко масштабувати та повторно використовувати код.

Основні компоненти: навігація, список завдань, детальна інформація про завдання, форма створення/редагування та ін. Вони організовані ієрархічно та взаємодіють між собою за допомогою служб та інтерфейсів.

Для взаємодії з API використовується Angular HttpClient. Він забезпечує зручний синтаксис для виконання HTTP запитів та отримання відповідей.

Маршрутизація налаштована за допомогою Angular Router. Він дозволяє організувати навігацію в односторінковому застосунку без перезавантаження сторінки.

Для зручної роботи з формами використовується реактивний підхід на основі RxJS. Це спрощує валідацію, відстеження змін та подання даних у формах.

Також застосовані Angular анімації та Materialize дизайн для створення сучасного інтерфейсу користувача з гарною зовнішньою візуалізацією та плавними ефектами.

Використання Angular дозволяє швидко та якісно реалізувати frontend частину веб-застосунку з дотриманням сучасних практик та патернів.

3.3. Опис основних компонентів

Стартова сторінка для реєстрації та входу в систему (автентифікації) веб-додатку має ключове значення, оскільки вона є першим етапом взаємодії користувача з сервісом .

Стартова сторінка (Рис.3.1)**:**

Заголовок який коротко відображає призначення додатку .

Форма Реєстрації яка включає наступні поля:

- Логін
- Пароль.
- Кнопка "Зареєструватися".

Форма Входу яка містить:

- Поле для введення логіну.
- Поле для введення пароля.
- Кнопка "Увійти".

Чистий та зручний дизайн з використанням лаконічних та зрозуміліх елементів.

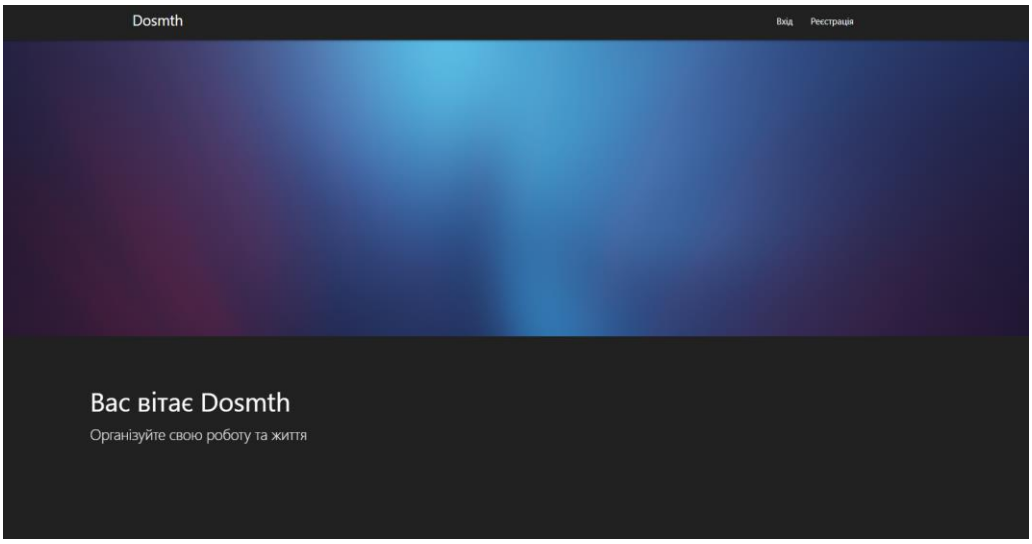

Рис.3.1. Стартова сторінка веб-сервіса

Сторінка реєстрації (Рис.3.2)**:**

Вибір Режиму (Реєстрація або Вхід)

Кнопки "Реєстрація" і "Вхід":

Користувач може обрати, чи хоче він зареєструватися чи увійти в існуючий обліковий запис.

Форма Реєстрації

Форма Входу

Кнопка "Увійти"

Ініціює процес входу після введення коректних облікових даних.

Месседжі про Успіх або Помилку

Месседжі, які вказують на успішну реєстрацію чи вхід, або повідомлення про помилку та інструкції щодо її виправлення.

Навігаційні Посилання - можливість переходу на головну сторінку чи інші розділи додатку.

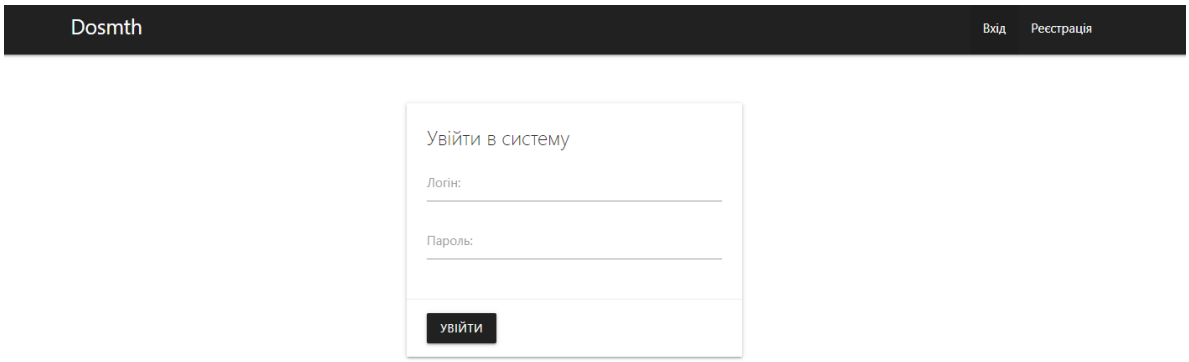

Рис.3.2. Сторінка реєстрації

Головна сторінка веб-застосунку (Рис.3.3**):**

Заголовок та Навігаційні Елементи

Заголовок, що вказує на призначення сторінки (наприклад, "Завдання" або «Списки»).

Можливість навігації до інших розділів додатку.

Відображення Завдань

Список, де кожен рядок представляє окреме завдання.

Відображення основних атрибутів завдань: назва, статус, термін виконання.

Можливість Редагування та Видалення

Для кожного завдання можливість редагувати його деталі чи видалити.

Створення Нового Завдання

Кнопка або інтерфейс для додавання нового завдання.

Відзначення Пріоритетів

Визначення пріоритету завдань (наприклад, за кольорами або іконками).

Відображення Додаткової Інформації

Можливість перегляду додаткової інформації про завдання за допомогою розгортання деталей.

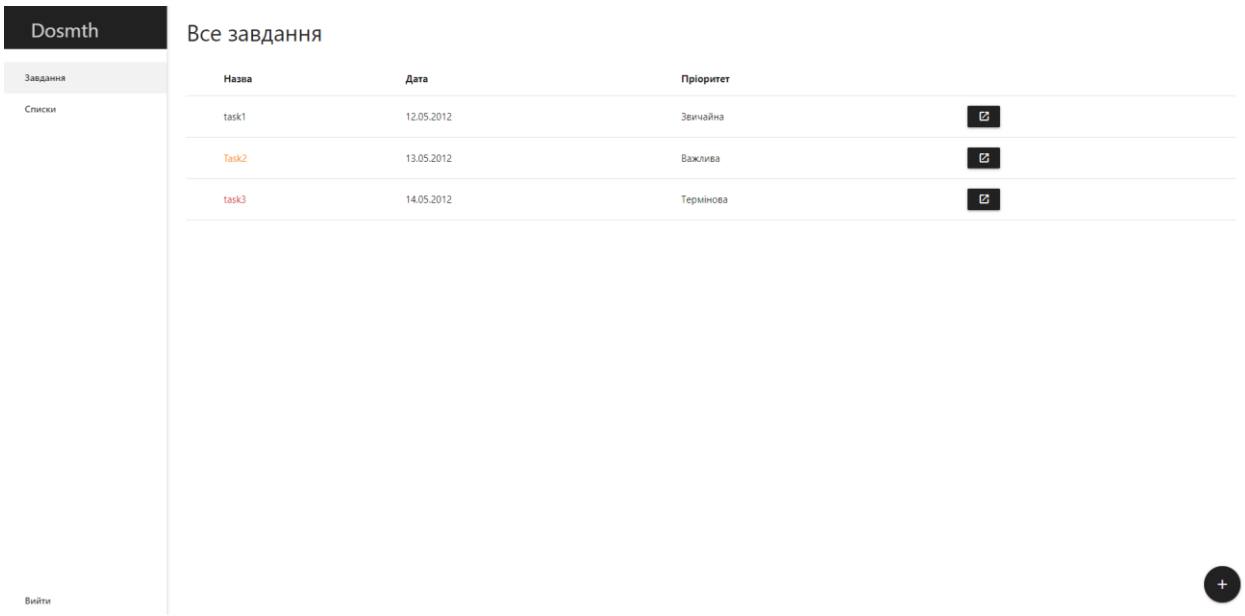

Рис.3.3. Головна сторінка веб-застосунку

Сторінка створення задач (Рис.3.4**):**

Заголовок, наприклад, "Додати завдання".

Можливість навігації назад до списку завдань.

Форма Створення Завдання

Поля для Введення:

- Назва завдання - текстове поле для введення назви.

- Опис - велике текстове поле для введення детальної інформації.
- Термін виконання - поле для введення дати або вибору з календаря.

Вибір Пріоритету:

Можливість вибору рівня пріоритету (високий, середній, низький) або використання інших маркерів (кольорів, іконок).

Кнопка Збереження:

Кнопка для підтвердження та збереження нового завдання.

Вибір до якого списку додати (опціонально)

Можливість вибору категорії або групи, до якої належить завдання.

Відображення Загальної Інформації

Можливість побачити загальну інформацію про інші завдання або важливі деталі.

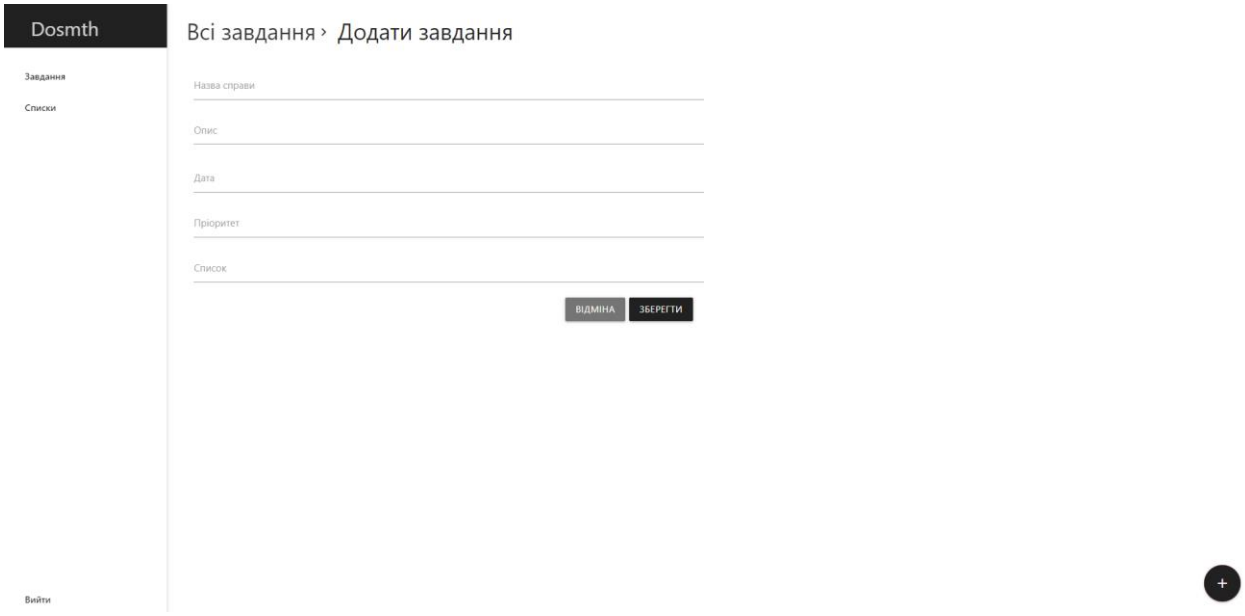

Рис.3.4. Сторінка створення задач

Сторінка детального перегляду завдання (Рис.3.5.**):**

Інформація про Завдання

Назва завдання.

Опис та деталі завдання.

Додаткова Інформація

Дата створення та термін виконання завдання.

Пріоритет, статус та інші атрибути.

Можливість Редагування

Кнопка для редагування параметрів завдання.

Можливість Видалення

Кнопка для видалення завдання.

Кнопка Повернутися Назад

Можливість повернутися до списку завдань чи іншої навігаційної сторінки.

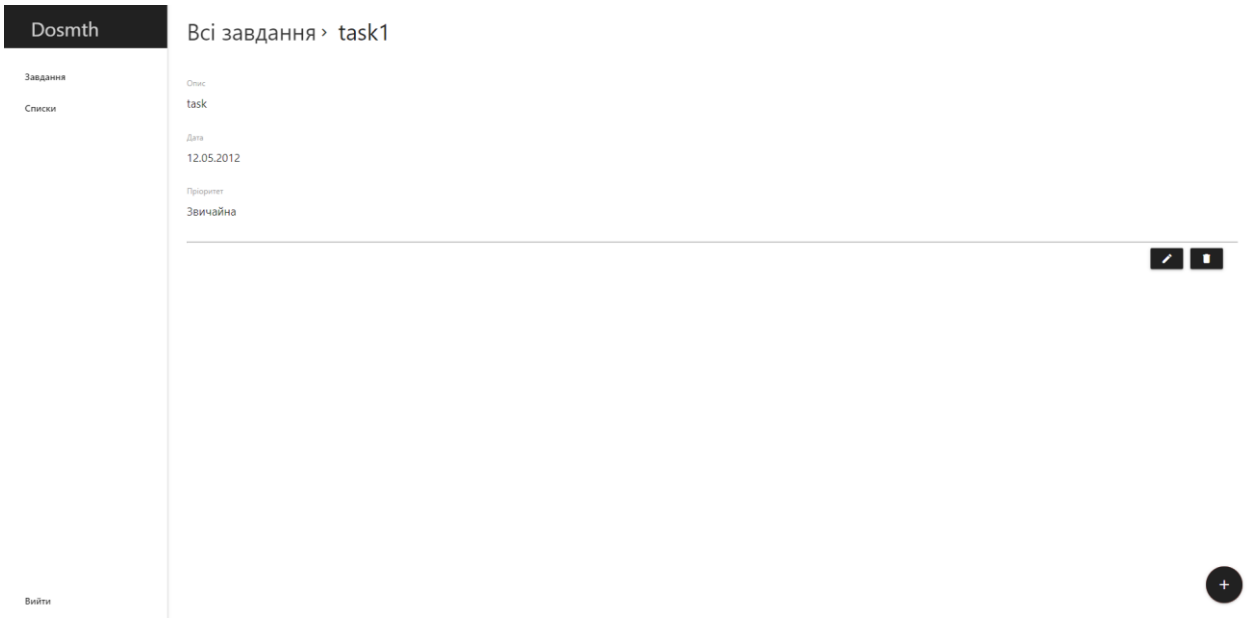

Рис.3.5. Сторінка детального перегляду завдання

Сторінка редагування завдання Рис.3.6**:**

Форма Редагування Завдання

Поля для Введення

Можливість зміни назви, опису, терміну виконання та інших атрибутів завдання.

Зміна Пріоритету

Можливість вибору нового рівня пріоритету або іншого маркера.

Збереження Змін або відміна

Кнопка для підтвердження та збереження внесених змін.

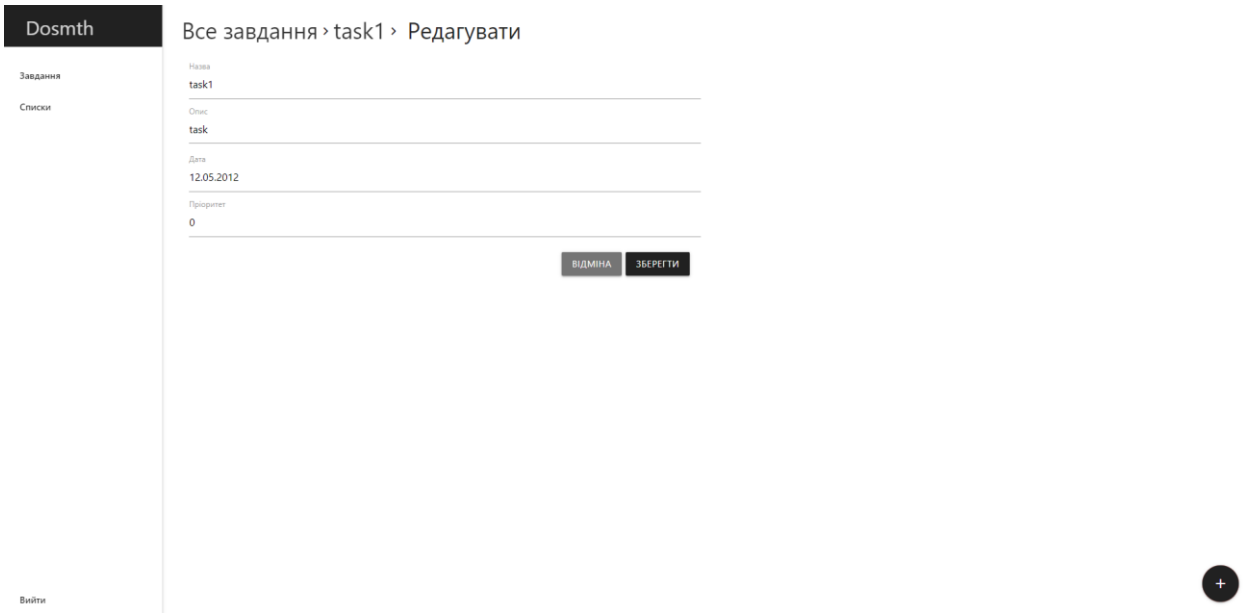

Рис.3.6. Сторінка редагування завдання

Сторінка перегляду всіх списків(Рис.3.7**):**

Перегляд Всіх Списків

Відображення усіх доступних списків або категорій завдань.

Можливість перегляду кожного списку та переходу до його вмісту.

Можливість Створення Нового Списку

Кнопка або інтерфейс для створення нового списку або категорії.

Можливість Редагування та Видалення Списків:

Опції для редагування параметрів кожного списку та їх видалення

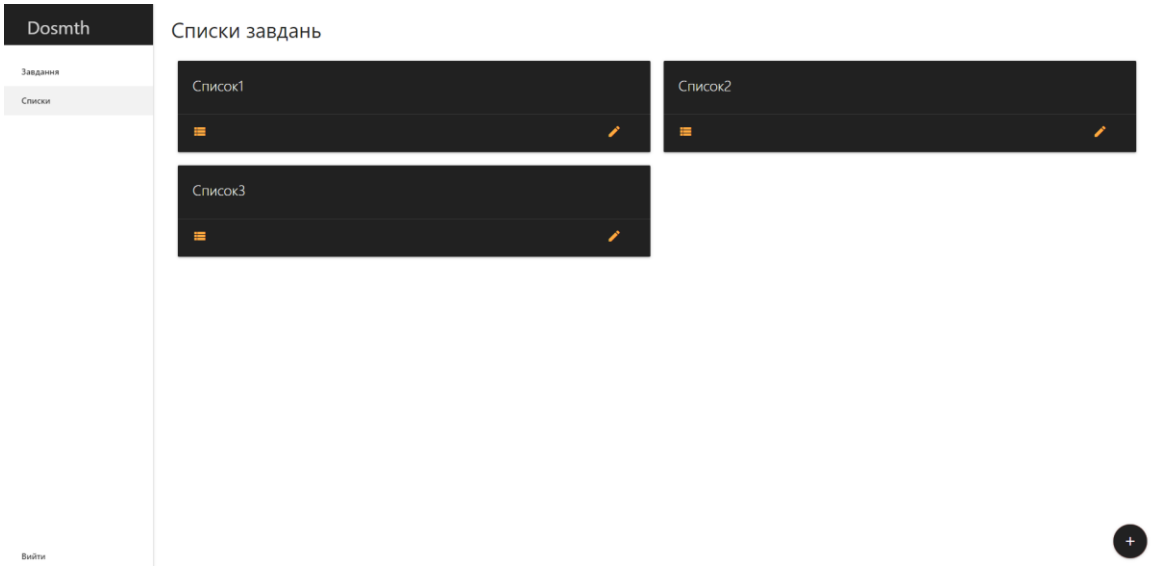

Рис.3.7. Сторінка перегляду всіх списків

Сторінка створення списку (Рис.3.8**):**

Форма Створення Списку

Поля для Введення

Назва списку: текстове поле для введення назви нового списку.

Збереження Нового Списку

Кнопка для підтвердження та збереження нового списку.

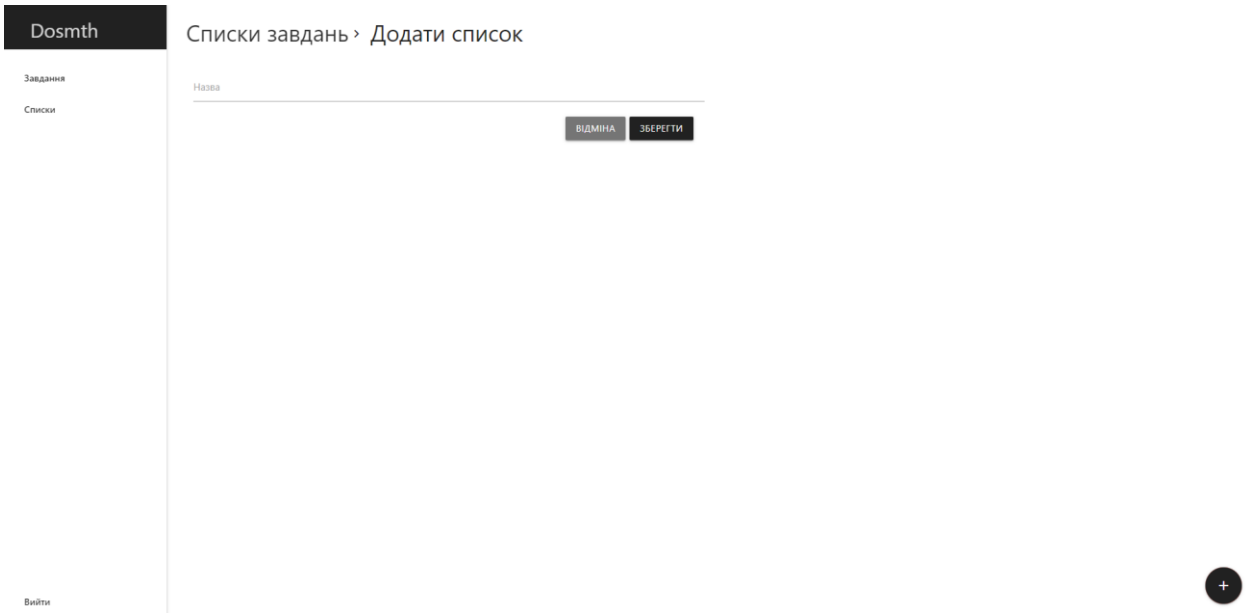

Рис.3.8. Сторінка створення списку

Висновки до розділу 3

Було описано реалізацію серверної частини за допомогою ASP.NET Core з використанням архітектурного патерну MVC та Entity Framework для роботи з даними. Це забезпечило гнучкість, модульність та високу продуктивність бекенду.

Для клієнтської частини використаний Angular фреймворк з компонентним підходом. Це дозволило оптимізувати розробку інтерфейсу та його подальше масштабування.

Основні компоненти реалізовані для аутентифікації, роботи із завданнями, списками, з використанням маршрутизації та реактивних форм. Застосовано сучасні UI компоненти.

Для безпечної аутентифікації користувачів реалізовано можливість реєстрації, входу в систему та авторизації за допомогою JSON Web токенів. Це дає гнучке управління доступом до різних частин додатку.

Для швидкої та якісної розробки інтерфейсу були використані фреймворки Bootstrap та Materialize, що надають готові UI компоненти та полегшують стилізацію. Також застосовано анімаційні ефекти.

Для збереження даних буде використано Entity Framework, структуру об’єктно-реляційного відображення з відкритим кодом для ADO.NET, оптимізовану для веб-застосунків. Це забезпечує надійність, цілісність та оптимальну продуктивність бази даних.

Обрана архітектура та технології дозволили створити функціональний, надійний та масштабований веб-додаток відповідно до поставлених вимог.

ВИСНОВКИ

У роботі розглянуто актуальне питання створення веб-сервісу для організації нотаток і управління списками справ.

Проаналізовано переваги та недоліки популярних органайзерів. Запропоновано вдосконалений функціонал та підходи для нового додатку.

Розроблено архітектуру та спроектовано базу даних з урахуванням сучасних вимог до подібних застосунків. Обґрунтовано вибір технологій для реалізації серверної (ASP.NET Core) та клієнтської (Angular) частин.

Створено бекенд з використанням архітектури MVC та Entity Framework для забезпечення модульності, гнучкості та високої продуктивності. Розроблено функціональний та інтуїтивний інтерфейс користувача з застосуванням сучасних фреймворків та компонентів. Для аутентифікації користувачів обрано сучасний підхід із застосуванням JSON Web Token. Він дозволяє гнучко та надійно управляти доступом і підвищує захищеність даних.

Запропонована архітектура побудована за принципами легкої масштабованості, що дає можливість нарощувати потужності та функціонал системи. Використання хмарних технологій дозволить оптимізувати витрати на експлуатацію та підтримку.

Протестовано роботу основних функцій сервісу, верифіковано відповідність вимогам та виправлено виявлені дефекти. Виконано оптимізацію з точки зору швидкодії, стабільності, безпеки та зручності використання.

В результаті створено цілісне рішення у вигляді веб-сервісу для ефективного планування та відстеження особистих і групових задач. Він поєднує переваги існуючих аналогів, усуваючи їх недоліки на основі сучасних технологій.

Розроблений проект може бути покладений в основу комерційного стартапу або впроваджений у корпоративному середовищі для підвищення продуктивності. Подальший розвиток може включати розширену аналітику, інтеграції та мобільні додатки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Smith A. Planning Fallacy: Only 14% of People Can Accurately Estimate Task Completion Times. *Journal of Experimental Psychology*, 2019. v. 155, issue 3.

2. Buehler R., Griffin D., Peetz J. The Planning Fallacy: Cognitive, Motivational, and Social Origins. *Advances in Experimental Social Psychology*. 2010. V. 43. P. 1-62. DOI: 10.1016/S0065-2601(10)43001-4.

3. ReportLinker. Todo List Software Market - Growth, Trends, COVID-19 Impact, and Forecasts (2021 - 2026).URL: https://www.reportlinker.com/ (дата звернення: 20.11.2023).

4. McKinsey & Company. Improving Project Delivery Through Task Management Software. McKinsey Technology Report. URL: https://www.mckinsey.com (дата звернення: 18.11.2023).

5. Buehler R., Griffin D., Ross M. Inside the planning fallacy: the causes and consequences of optimistic time predictions. *Heuristics and Biases: The Psychology of Intuitive Judgement*. 2002. P. 250-270.

6. Buehler R., Griffin D., Ross M. Exploring the "Planning Fallacy": Why People Underestimate Their Task Completion Times. *Journal of Personality and Social Psychology*. 1994. Vol. 67, №. 3. P. 366-381.

7. Clemens J.K., Dalrymple S. Time Mastery: How Temporal Intelligence Will Make You a Stronger, More Effective Leader. AMACOM, 2005. 210 p.

8. Robbins M. The 5 Second Rule: Transform Your Life, Work, and Confidence with Everyday Courage. Savio Republic, 2017. 241 p.

9. Tracy B. No Excuses! The Power of Self-Discipline. Vanguard Press, 2011. 304 p.

10. Калініченко Л.Л., Гаврилова А.О. Особливості впровадження тайм-менеджменту на підприємстві. *Молодий вчений*. 2017. № 4.4 (44.4). С. 60–63.

11. Іваницька С.Б., Галайда Т.О., Толочій Р.М. Впровадження європейських методик тайм-менеджменту в Україні. *Глобальні та національні проблеми економіки*. 2018. № 21. С. 288–292.

12. Журавльова Х. К. Підвищення ефективності використання робочого часу керівника за допомогою тайм-менеджменту *Управління розвитком*. 2013. № 20. С. 96–98.

13. Євдокимов В.О. Конотопцева Ю.В. Основи планування тайм-менеджменту державного службовця *Теорія та практика державного управління*. 2016. Вип. 2. С. 171–177.

14. Євтушенко Г.І., Дерев'янко В.М. Аналіз стану управління робочим часом та шляхи підвищення ефективності застосування «Тайм-менеджменту» в організації. *Збірник наукових праць Національного університету державної податкової служби України* . 2014. №1. С. 88–96.

15. Крукевич Н. М. Проблеми становлення поняття тайм-менеджмент. *Торгівля, комерція, підприємництво*. 2014. Вип. 17. С. 119–122.

16. Лазоренко Т.В., Дідченко Ю.О. Правила успішного використання тайм-менеджменту. *Молодий вчений*. 2017. № 1(41). С. 632–635.

17. Скібіцька Л.І. Організація праці менеджера : навчальний посібник. К. : Центр учбової літератури, 2010. 360 с.

18. Янцаловський О.Й., Леус О.С. Психологічні особливості тайм-менеджменту студентів. *Вісник Національної академії Державної прикордонної служби України.* 2017. Вип. 4. URL: http://nbuv.gov.ua/UJRN/Vnadpn_2017_4_19.

19. Причепа І. В., Соломонюк І. Л., Лесько Т. В. Тайм-менеджмент як дієвий інструмент ефективного використання часу успішного менеджера за сучасних умов. *Ефективна економіка*. 2018. № 12. DOI: 10.32702/2307-2105-2018.12.

20. De Sanctis V. Building Web APIs with ASP.NET Core. New York: Manning, 2023. 472 p.

21. Lock A. ASP.NET Core in Action. New York: Manning, 2021. 832 p.