МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ
Кафедра комп'ютерних наук та кібербезпеки

ГАЛУШКА ЮРІЙ АНАТОЛІЙОВИЧ

ВЕБ ОРІЄНТОВАНА ПЛАТФОРМА МЕНЕДЖМЕНТУ ПРОЕКТІВ РОЗРОБКИ ПРОГРАМНИХ ПРОДУКТІВ

Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки та інформаційні технології
Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр»

Науковий керівник:
Пастернак Ярослав Михайлович,
доктор фіз.-мат наук, професор
кафедри комп'ютерних наук та
кібербезпеки

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол №_____
засідання кафедри комп'ютерних наук
та кібербезпеки
від _____ 2024 р.
Завідувач кафедри
(_____) Гришанович Т. О.

Луцьк -2024

ЗМІСТ

ВСТУП

У сучасному світі ми можемо помітити як швидко розвиваються технології. Ще нещодавно люди стояли в живих чергах та працювали у великих мануфактурах, виконуючи монотонні дії для автоматизації виготовлення товару. А для того, щоб обговорити якесь питання потрібно було проїхати «7 королівств» тільки для збору групи людей, чи тижнями чекати відповіді на повідомлення.

Проте вже сьогодні технології перевернули все представлення про подібне. Не потрібно стояти в чергах, оскільки вже давно існують онлайн квитки, які дозволяють взяти талончики на зручний час. На заміну людям на великих фабриках та заводах вже працюють роботи та автоматизовані механізми, які облегшують і пришвидшують виконання різних монотонних процесів. Різні сервіси і додатки по типу Telegram, Viber, Zoom, дозволяють не виходячи з дому зібратися для обговорення важливої теми чи питання, відправити повідомлення людині на іншій частині світу за секунду! Але дещо залишилося незмінним – необхідність контролю за процесами та комунікація між членами команди для прискорення і успішного завершення задачі.

Для виконання подібних задач зараз створюються безліч сервісів з необхідним функціоналом. Наприклад у програмістів таким сервісом є середовище розробки GIT, що дозволяє організувати спілкування між розробниками, розподілити завдання на окремі задачі та контролювати весь прогрес різноманітним набором інструментів.

Актуальності набувають веб додатки для ефективного управління проектами, оскільки вони спрощують і оптимізують роботу та надають можливість налагодити співпрацю між членами команди, не залежно від географічного місця розташування. Сервіси дозволяють організувати роботу таким чином, щоб досягти оптимальних результатів за мінімальний час.

Метою роботи є розробити веб застосунок на основі технології Blazor мови програмування C# , що буде мати інструменти для управління задачами під час розробки різних проектів. У додатку буде можливість зареєструватися та набір інструментів для створення і розподілу задач, відслідковування їх статусу чи проценту завершення.

Для реалізації поставленої задачі нам потрібно виконати наступне:

- дослідити поняття проекту та менеджменту проектів;
- ознайомитися з основними методами управління проектами;
- вивчити технології C#, необхідні для реалізації проекту;
- провести аналіз схожих додатків на ринку;
- визначити архітектурні принципи для розробки додатка;
- сформулювати вимоги та функціонал програми;
- провести аналіз сучасних тенденцій у сфері управління проектами та розробки веб застосунку;
- визначити потенційних користувачів серсісу та їх вимоги щодо інструментального набору;
- розробити базовий функціонал, включаючи можливість, аутентифікації;
- застосувати базову верстку для забезпечення естетичного вигляду;
- провести тестування продукту для перевірки його ефективності та стабільності.
- провести аналіз результатів та узагальнити отримані висновки.

Об’єкт дослідження – веб застосунок для менеджменту за проектами з використанням технологій мови програмування C# .

Предмет дослідження – ефективність та налаштування процесів управління та координування між членами команди проекту під час розробки.

Результати роботи були представлені на І Міжнародній науково-практичній конференції «Проблеми комп’ютерних наук, програмного моделювання та безпеки цифрових систем» 13-16 червня 2024 року.

РОЗДІЛ 1. ОСНОВНІ ВИЗНАЧЕННЯ ТА БАЗОВІ КОНЦЕПЦІЇ У МЕНЕДЖЕНТІ ПРОЕТКТУ. НАБІР ІНСТРУМЕНТІВ ДЛЯ ЧІТКОГО УПРАВЛІННЯ ТА ВІДСЛІДКОВУВАННЯ ЗА СТАНОМ ПРОЕТКУ

1.1. Поняття про проект, та його особливості

Взагалі, будь-який проект - це перш за все ідея або задум, лише потім йде план його реалізації і технічне або практичне втілення ідеї на практиці.

Проект (з латини «кинутий уперед задум») – являє собою комплексну систему науково-дослідних, проектно-конструкторських, соціально-економічних, організаційно-економічних та інших заходів, спрямованих на зміну об'єкта управління з метою досягнення поставлених цілей. Ці заходи організовуються з урахуванням ресурсів та термінів і спрямовані на створення та застосування нових технологій і матеріалів з метою підвищення ефективності.

Отож ми можемо виділити основні ознаки проекту (рис 1.1):

- кожен проект має певний термін виконання, включаючи початкову та кінцеву точки;
- реалізація потребує певних ресурсів, але вони є обмежені;
- кожен проект є тимчасовою системою і не може бути повторений після завершення, але можна починати етапи заново;
- результати роботи завжди унікальні і не завжди можуть відповідати очікуванням, навіть для проектів зі схожими цілями;
- наявність соціально значимого завдання;
- планування дій щодо вирішення проблеми;
- пошук та аналіз інформації для подальшого використання;
- створення продукту відповідно до цілей та поставлених завдань;
- презентація отриманого продукту або результату;

Рис. 1.1 Життєві складові проекту

Проект відрізняється від завдання тим, що складається з завчасно відомих процесів, але має високий ступінь невизначеності. KPI проекту розглядаються як трикутник управління проектом приклад рис 1.2, де кожна сторона несе відповідальність за тривалість проекту, обсяг (продуктивність) і вартість. Зміна будь-якого з цих параметрів неминуче вплине на усі інші, ускладнить процес управління проектом і потребує ретельного планування та контролю.

Рис. 1.2 Трикутник проектного менеджменту

1.2. Класифікації проектів

Отже ми розібрали основні поняття, які зустрічаються в проекті, проте ще не визначилися як можна їх класифікувати. Існує безліч критеріїв за якими можна визначати тип і вид проекту, давайте переглянемо основні з них.

Класифікація за розміром відрізняють моно, мульти і мега проекти. Отож давайте розберемо кожен більш детально.

Монопроект – це малий за розміром і за часом виконання проект, типовим прикладом є оновлення існуючого програмного забезпечення або впровадження тестового обладнання. Мало масштабні проекти використовують прості методи проектування та реалізації та можуть швидко закупити технічні ресурси, персонал та обладнання. Однак через обмежений час для виправлення помилок важливо знати характеристики проекту, статус і роботу учасників, а також терміни та інформацію про публікацію і деталі участі.

Для успішної реалізації таких проектів рекомендується:

- призначити одного керівника проекту для координації всіх дій;
- створити гнучку організацію команди проекту, яка допоможе змінювати членів за потреби та налаштовувати координацію між ними;
- використовувати найпростішу форму графіку для проекту, щоб забезпечити ясність і доступність для всіх учасників;
- щоб уникнути непорозумінь і забезпечити ефективну співпрацю потрібно чітко визначити ролі та обсяг роботи для кожного члена команди проекту;
- залучити тих самих розробників, які брали участь у початковому проектуванні, щоб забезпечити узгодженість і безперервність протягом усього процесу реконструкції та змін.

Наступним розглянемо мультипроект – це вже більш складний комплексний проект, який складається з ряду менших за масштабом монопроектів та вже потребує застосування значно більшого управління.

До великих проектів відноситься розробка: бібліотек і фреймворків, великих веб систем чи продуктів, що поділені на під завдання чи модулі. Також сюди відносяться спільні багатопрофільні проекти, де кожен розробник може відповідати за власний під проект, що полегшує співпрацю та розподіл завдань.

Для подібного важливим є узгодженість і стабільність інтерфейсів між різними компонентами системи. У великих проектах, оскільки різні частини системи можуть взаємодіяти одна з одною, важливими є правильне планування, тестування та впровадження змін. Мультипроектне програмування є важливим інструментом для організації та управління складними програмними засобами, чіткого і структурованого розподілу роботи між різними командами та забезпечення ефективної співпраці та менеджменту за проектом.

Останній прикладом поділу за масштабом є мегапроекти. Це складні програмні забезпечення або системи, які поєднують кілька взаємопов’язаних проектів чи компонентів, що мають спільну мету, ресурси та час.

Іншими важливими видами поділу є поділ за складністю, вирізняють прості, організаційно-складні, технічно-складні, ресурсно-складні, комплексно-складні. Наступний клас поділу це тривалість по часу розподіляють на короткострокові, середньострокові та довгострокові.

За вимогами до якості та засобами її забезпечення проекти бувають бездефектними, модульними та стандартними. Також існує варіант розподілу за характером проекту наприклад міжнародний, де залучені команди з різних країн та вітчизняним, що розробляється тільки всередині країни.

Останній спосіб поділу який можна відмітити це за ступенем зв’язку. Існують наступні типи альтернативні існування яких виключає можливість існування інших. У комп’ютерних технологіях суперечливі проекти — це проекти, які не можуть бути реалізовані, якщо інші проекти вирішать їх реалізувати, оскільки вони призначені для вирішення тієї ж потреби. Наприклад, розробка двох конкуруючих програмних продуктів, які обслуговують той самий бізнес-сектор. Ці проекти змагаються за користувачів і ресурси, тому немає сенсу робити те й інше одночасно. Суперечливі елементи

вказують на інші шляхи досягнення такого ж результату в комп'ютерних технологіях. Як правило це проекти-конкуренти, що повинні зайняти однакову нішу на ринку.

Незалежні – відхилення чи прийняття яких не буде впливати на прийняття рішення щодо інших. В комп'ютерних технологіях це наприклад, створення нових програмних продуктів фінансового обліку та створення сайтів для рекламних кампаній. Користувачі програмних продуктів не залежать від доступності веб-сайту, тому їхні потреби та переваги продукту залишаються незмінними незалежно від інших планів.

Далі йдуть взаємопов'язані це проекти, відхилення чи прийняття яких буде залежати від прийняття рішення про інші. Наприклад, розробка програмного забезпечення, яке автоматизує обробку даних, що утворюються під час роботи ТЕС. Необхідність впровадження такого плану розробки програмного забезпечення повністю залежить від правильного рішення щодо впровадження запропонованої технології. Однак, оскільки проект розробки програмного забезпечення може здійснюватися незалежно від будівництва ТЕС, необхідно мати на увазі, що умови цих проектів неоднакові.

Наступним за поділом можуть бути два проекти, робота одного впливає на користь іншого. Наприклад, проект А передбачав розробку складного алгоритму для шифрування даних, тоді як проект В розробив новий криптографічний протокол для захисту інформації. Якщо А буде успішно реалізовано, це може підвищити безпеку передачі даних, але в той же час це може зменшити потребу у використанні нового криптографічного протоколу, представленого В. У цьому випадку обмін є асиметричним, оскільки він використовується правильно. А може зменшити прибуток від товарів В, але прийняття або відмова від товарів В не впливає на прибуток від товарів А.

Останнім видом поділу є синергетичні види. У світі інформаційних технологій проекти зі схожим внеском можна поєднувати для підвищення ефективності та взаємної вигоди. Наприклад, проект розробки нового алгоритму стиснення даних, проект впровадження високошвидкісної системи

передачі даних. Впровадження нових алгоритмів стиснення може зменшити обсяг даних, що надсилаються, що може знизити вартість пропускної здатності мережі та збільшити швидкість передачі даних. З іншого боку, впровадження високошвидкісних мереж полегшує передачу великих обсягів даних, що підвищує ефективність використання нових алгоритмів стиснення. Тому ці елементи разом створюють синергетичний ефект, підвищуючи економічність і ефективність передачі даних.

Більш детально класифікацію можна переглянути на рис. 1.3.

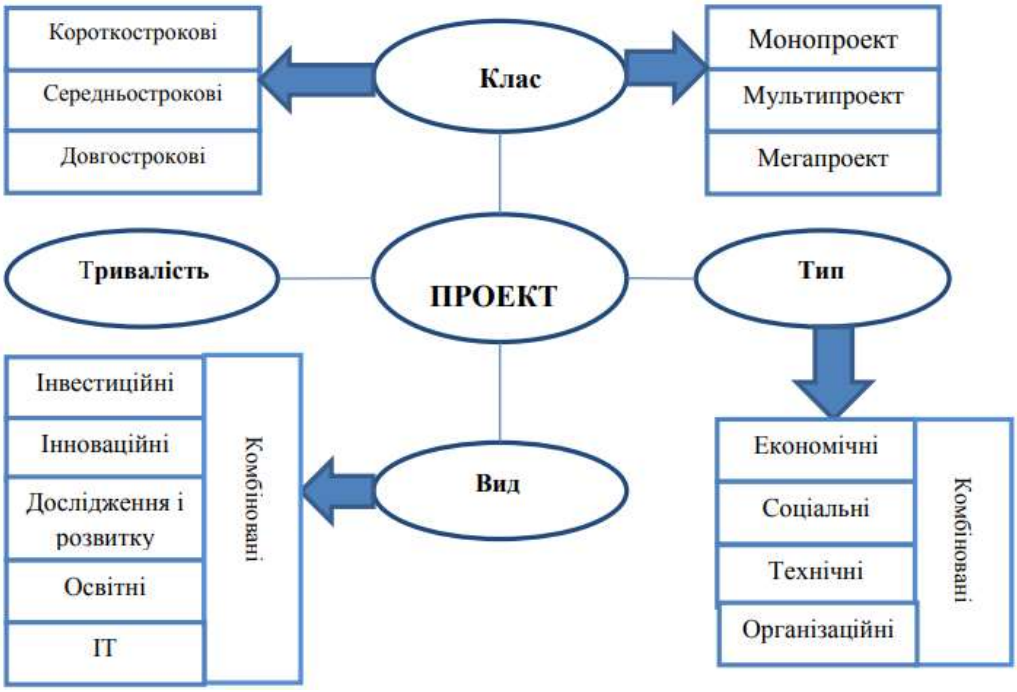

Рис. 1.3 Класифікація проектів за основними критеріями

1.3. Менеджмент проектів

Наступним етапом нашої роботи буде ознайомлення з способами для менеджменту проектів. Власне подібним повинні займатися прожект менеджери або простіше кажучи PM (з англійської – project manager) – це проф. діяльність, яка поєднує найсучасніші наукові знання та технологій з практичними навичками. Він орієнтований на отримання найбільш

ефективного результату в ході реалізації проекту. Він налаштовує і узгоджує всі дії між членами команди, створює необхідні завдання та відслідковує стан їх виконання. На наступному рисунку 1.4 ми можемо помітити всі етапи, що повинен пройти проект для успішного завершення.

Рис. 1.4 Етапи успішної реалізації проекту

Управління проектами — це не просто наука, а методологія, що охоплює організацію, планування та координацію використання людських і фінансових ресурсів у життєвому циклі проекту. Це все дуже важкий і дорогий процес по витратам для деяких компаній, але коли клієнти програмного проекту включають багато зацікавлених сторін, функцій і команд, які впливають на продукт, переваги переважують ризики.

Власне менеджери проектів можуть зменшити накладні витрати, оптимізувавши операції. Вони несуть відповідальність за виконання проектів згідно з бюджетом і графіком, але, що більш важливо, вони можуть покращити відносини між внутрішніми зацікавленими сторонами та клієнтами шляхом покращення комунікації та видимості проекту. Крім того, вони можуть зменшити ризик негативного впливу на бізнес і зменшити проблеми, які заважають команді розробників програмного забезпечення.

Існує багато програмного забезпечення для керування проектами, яке підтримує все, від базових систем до систем преміум класу. Щоб вибрати найбільш підходящий інструмент, спочатку потрібно визначити розмір вашої кампанії, програмний проект, який ви хочете реалізувати, і використовуваний метод розробки. Нижче наведено функції які потрібно застосовувати для успішного слідкувати за кожним етапом SDLC (The software development lifecycle):

- планування та зміна графіку проекту;
- планування та контроль в управлінні ризиками;
- розподіл підлеглих функцій;
- бюджетне управління та прогнозування;
- управління часом і витратами;
- управління та контроль над проектом;
- командна робота та спілкування, включаючи онлайн чат, відео дзвінки та спілкування електронною поштою;
- інструменти візуалізації, такі як дошки Kanban, діаграми Gant та діаграми Pert;
- управління документами та записами;
- налаштування ролей користувачів і дозволів;
- автоматизація робочих процесів;
- комунікаційний менеджмент;
- відстеження та зміна налаштувань
- використання штучного інтелекту;

1.4. Сучасні можливості додатків для управління проектами

Отож розглянемо детальніше, які інструменти використовувати для того, щоб організувати і успішно завершити роботу над новими проектами. Для початку головною фазою проекту є планування, оскільки визначає подальший хід і планування роботи. Першим кроком є чітке визначення цілей

проекту, які мають слідувати технології SMART – бути конкретними, вимірними, досяжними, актуальними та довгостроковими рис 1.5. Далі потрібно створити робочий план Work Breakdown Structure чи WBS — це система яка використовується для поділу роботи на менші частини, що полегшує управління та контроль. Це дозволяє визначити та логічно спланувати всі завдання, необхідні для виконання проекту.

Рис. 1.5 Головні вимоги технологія SMART

Усім необхідним функціоналом ми можемо скористатися використовуючи діаграми GANTT для візуалізації процесу рис. 1.5. У діаграмах відображаються завдання, їх час, розклад і комунікація, що допомагає організувати групову роботу та слідкувати за дедлайнами.

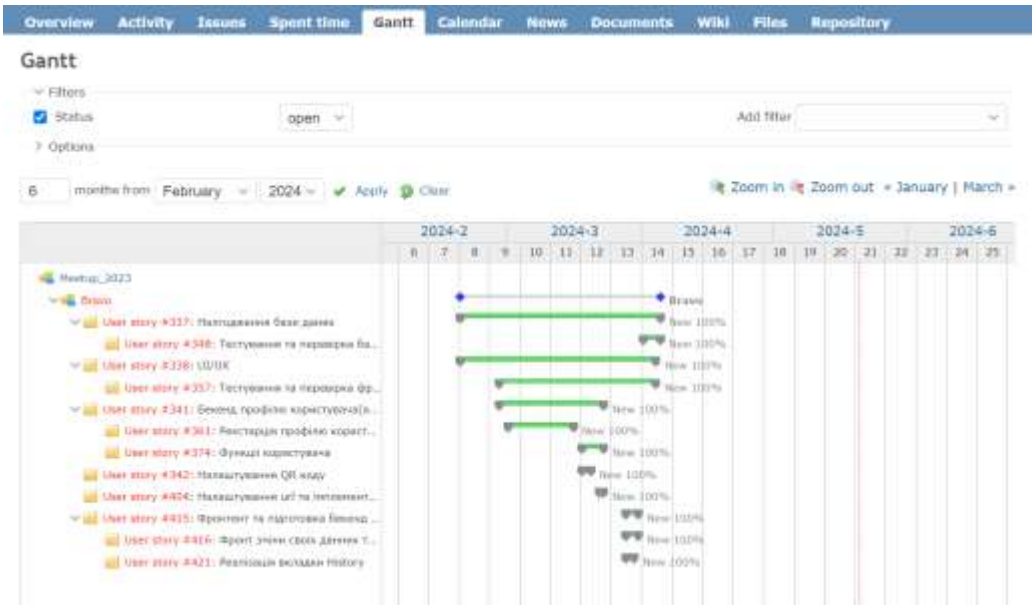

Рис. 1.6 Приклад діаграм GANTT для планування і розподілу задач

1.4.1. Управління і контроль за часом та ресурсами

Ефективне управління часом є критичним для успішного завершення проекту у встановлені терміни. Використання діаграм методу критичного шляху чи по іншому CRM допомагає визначити найдовший шлях у ланцюжку завдань, що визначає мінімальний час, необхідний для виконання завдання. Це дозволяє визначити важливі цілі, які неможливо запланувати, не затримуючи весь проект. Належне управління ресурсами допомагає максимізувати доступні ресурси та зменшити витрати.

Планування та управління ресурсами передбачає визначення того, які ресурси (люди, матеріали, гроші) потрібні для кожного проекту. Оптимізація означає планування спільної роботи, уникнення надмірності та забезпечення їх ефективного використання. Варто організувати розподіліть роботу так, щоб окремі члени команди не були перевантажені та працювали на відповідними завданнями. Це допомагає підтримувати високу продуктивність і запобігає втомі. Ефективне управління командою є важливим для досягнення успіху проекту.

1.5. Аналіз сучасних тенденцій у менеджменті проектів

Сучасна практика управління проектами кардинально змінилася через швидкий технологічний прогрес і зміни в організаційних процесах. Ось основні тренди на які очікуються на зростання у поточному році.

Першим і головним впровадженням яке очікується це звісно штучний інтелект. Він відіграє важливу роль в управлінні проектами: автоматизація роботи, краще планування та прогнозування витрат і термінів. Однак успіх використання штучного інтелекту залежить від навчання та управління даними. Успішні організації, які інвестують у цифрову трансформацію та навчання співробітників, отримують вигоду від впровадження ШІ (Kanban, Agile Manager, SelectHub).

Багато компаній використовують гібридні методи в управління проектами поєднуючи традиційні та гнучкі методи, такі як Scrum і Kanban, щоб підвищити гнучкість і адаптивність. Гібридний підхід забезпечує кращу адаптацію до змін у проекті та потреб різних груп. Наш проект також буде використовувати головним чином цих два підходи тому давайте зупинимось на них більш детально.

Метод Kanban, або сигнальна дошка, часто використовується для невеликих проектів, де не потрібно витрачати багато часу на детальне планування та управління. Це незвичайний метод управління проектами, який передбачає створення завдань і відображення їх на екрані. Усі дії за проектом включено та відзначено їх стан роботи. Такі системи забезпечують високу прозорість, оскільки кожен член команди має доступ до всієї інформації, пов’язаної з роботою та статусом.

Методика Kanban виникла в Японії ще у 1940-х роках як метод управління виробництвом, який допомагає зменшити відходи та підвищити ефективність заводів Toyota. Дошка містить стандартні колонки To Do (зробити завдання), Doing (завдання виконується), Done (завдання завершено). Однак вона є гнучкою з можливістю додати й інші стани та підходить для потреб різних проектів. Завдання переміщуються з одного стану в інший по мірі виконання, що полегшує відстеження прогресу. Дії та статуси бачать усі члени команди, що покращує спілкування та співпрацю. Метод дозволяє швидко адаптуватися до змін проекту без необхідності повністю змінювати плани.

Kanban зазвичай використовується в програмному забезпеченні для відстеження завдань і спринтах. Також його можна застосовувати для особистого планування та керування власними проектами, такими як підготовка до іспитів або планування подорожі. Детальніше з прикладами ми можемо ознайомитися у наступних пунктах роботи.

Рис. 1.7 Приклад інтерфейсу Kanban дошки

Наступним у нашому списку є метод Scrum, модель, показана на картинці 1.8. Це різновид методології Agile, в перекладі з англійської означає «боротьба за м’яч» або «поштовх» у регбі. Ця модель щільно закрита і інтегрує процес управління проектом, що призводить до хорошого результату.

Scrum використовується в проектах, де важлива постійна комунікація з клієнтами. Наприклад, якщо ви хочете перевірити, чи правильно працює проект, Scrum дозволяє швидко виправляти помилки. Таким чином команда може частіше знаходити нові моделі продукції, підтримувати постійний зв’язок із замовниками, прискорювати виконання проекту та підвищувати продуктивність.

Керівник проекту відповідає за встановлення вимог і пріоритетів, забезпечує дотримання процесів і усунення перешкод, а команда керує процесом виконання задач. Scrum включає регулярні, щоденні зустрічі та розпланування спринтів, огляди задач і результатів роботи. Через подібну систему команда завжди в курсі у якому стані знаходиться проект, а також завдяки коротким спринтам (зазвичай 2-4 тижні) команда може швидко реагувати на необхідні зміни та міняти метод робота.

Метод Scrum надає корисний інструмент для ефективного управління проектами, особливо коли йдеться про постійне спілкування з клієнтами та

швидке реагування на важливі зміни. Використання Scrum забезпечує високу чіткість процесу, сприяє ефективній співпраці та дозволяє частіше випускати нові продукти, тим самим підвищуючи продуктивність команди та задоволеність клієнтів.

1.6. Дослідження переваг обраних інструментів та технологій

Розглянемо набір інструментів та технології, які використовуватимуться для створення додатку управління проектами з використанням C#, ASP.NET та Framework Blazor.

На відміну від таких мов, як Python і PHP, що існують уже давно, C# є відносно молода мова програмування. Він був розроблений у 2000 році датським інженером-програмістом Андерсом Хейлсбергом. Зараз він працює в Microsoft старшим розробниом C#. Андерс Хейлсберг відомий як провідний архітектор Delphi і перший автор Turbo Pascal.

Першочергово мова називалася COOL, що означає «C-style Object-Oriented Language». Однак Microsoft не змогла зберегти назву через юридичні обмеження маркетингу. Відтоді мова називається C# ("C-Sharp"). Походить від нотного запису, де символ "#" (дієз) означає підвищення ноти на півтону. Суфікс "sharp" також використовується в інших мовах .NET, включаючи J#, A# і F#. Синтаксично C# схожий на мови виду C, такі як C, C++ і Java. Ця мова програмування відповідає стандарту Common Language Infrastructure (CLI).

В основному розробники віддають перевагу цій мові програмування, оскільки вона чудово продумана та проста у використанні. Крім того ця мова просто необхідна для вивчення людям, що збирають створювати продукти для платформи Microsoft. По перше це універсальна мова, що надає можливість створювати безліч різноманітних продуктів починаючи з динамічних веб сайтів із відкритим кодом закінчуючи різними специфічними додатками і програмами на архітектурі платформи Microsoft.

Ось короткий список переваг:

- мова входить до десятки найбільш затребуваних на ринку;
- низький поріг входження для початківців;
- різні програми дозволяють вибрати потрібний стиль;
- C# підтримує різноманітні інструменти та фреймворки;
- існує велика база даних інструментів, які надають можливість легко зрозуміти та вирішити різноманітні проблеми;
- програма створена, щоб надавати програмісту максимальний контроль над усіма можливими аспектами в програмуванні;
- мова активно і стрімко розвивається, нові програми та фреймворки з'являються швидше, ніж у будь-які іншій мова програмування.

C# підтримує парадигму ООП, що дозволяє створювати модульні та розширювані додатки. Головні відомі принципи, що зустрічаються у ООП: наслідування, інкапсуляція, поліморфізм, також сьогодні вирізняють 4 принцип абстракція. Вони сприяють розділенню функціональності на незалежні компоненти, що збільшує читабельність та повторне використання коду без необхідності дублювати його. Це особливо важливо в управлінні проектами, де складність додатків може швидко зростати.

1.6.1. Технічні особливості .NET Framework

.NET Framework тримає у собі величезну кількість класів та методів для роботи з різними файлами, мережевими ресурсами, базами даних, графічними інтерфейсами та іншими різноманітними компонентами. Це значно спрощує і скорочує час розробки, оскільки розробники можуть використовувати вже готові рішення для стандартних чи монотонних задач. Наприклад, бібліотеки для роботи з XML та JSON спрощують обмін даними між різними частинами системи, а бібліотеки для доступу до баз даних, забезпечують ефективне управління даними.

.NET Framework надає потужні інструменти для захисту програм, включаючи механізми автентифікації та авторизації. Різноманітні вбудовані

механізми безпеки, такі як впровадження SQL і міжсайтовий сценарій (XSS), дозволяють розробникам налаштувати бізнес-логіку, не турбуючись про безпеку програм. У придачу до .NET Framework, Microsoft підтримує .NET Core і його версії .NET /7/8 для Windows, за його допомогою є можливість створювати програми для macOS і Linux. Він дає змогу розробникам створювати кросплатформені додатки і веб застосунки.

Розробка на C# та .NET Framework інтегрована з сучасними інструментами розробки, такими як Visual Studio та Visual Studio Code. Ці середовища розробки, дозволяють писати код, та надають розширені інструменти для тестування та оптимізації. C# і .NET мають велику спільноту розробників і повну документацію з усіма можливими описами вирішення проблем. Відкритість коду сприяє обміну досвідом і доступу до освітніх ресурсів. Через NuGet також доступно багато бібліотек і пакетів, які покращують продуктивність проектів і дозволяють легко інтегрувати інші служби.

1.6.2. Технологія Blazor відповідь Microsoft на можливості JS

Blazor — це сучасний фреймворк від Microsoft створений для розробки інтерактивних веб інтерфейсів використовуючи лише C#. Він дозволяє розробникам використовувати C# замість JavaScript для створення складних інтерфейсів. Це означає, що розробники, знайомі з .NET, можуть легко використовувати його для створення веб додатків. Розділяючи код між серверними та клієнтськими компонентами програми, Blazor скорочує час розробки та витрати ресурсів на професіоналів. Це полегшує зберігання та підтримку коду. Фреймворк дозволяє створювати компоненти, які можна дуже легко використовувати в різних областях програми або в інших проектах. Це покращує ефективність розробки та зменшує зайвий код, забезпечуючи модульність коду.

Blazor працює повністю використовуючи технологію .NET Core, він надає широкий функціонал і високу продуктивність. Швидкий цикл розробки ПЗ, оскільки все відбувається на сервері, оновлення та виправлення можна швидко розгортати без необхідності оновлювати на стороні клієнта. Оскільки код зберігається на сервері це забезпечує безпеку коду, бо він не надсилається клієнту. Також на стороні користувачів завантажується невелика кількість інформації і коду, що дозволяє негайно використовувати продукт.

Хоча існують і певні недоліки, наприклад затримка мережі може мати значний вплив на продуктивність, оскільки кожна взаємодія із програмою потребує підключення до сервера. Через постійний зв'язок з сервером програма не працює без активного підключення до Інтернету. Існує обмеження на «залізо» оскільки сервер повинен мати достатньо ресурсів для обробки всіх запитів клієнтів і надсилання відповідей назад для підтримки зв'язку.

Blazor Server ідеально підходить для внутрішніх бізнес-додатків чи публічних програм з низьким рівнем навантаження. Завдяки високій швидкості розробки та прямому доступу до серверних сервісів ця модель швидко працює, легко створюється та підтримується.

Наступним ми розберемо WebAssembly. Програма працює безпосередньо у браузері, тому навантаження на сервер суттєво зменшується, застосунок може працювати без прямої підтримки сервера .NET.

Оскільки програма може працювати в режимі очікування - тобто існує можливість роботи без підключення до Інтернету. Код надає легкий доступ і обмін до об'єктів C# між клієнтом і сервером. Не дивлячись на переваги існують і свої недоліки. Перший запуск програми може зайняти тривалий час, оскільки початковий розмір завантаження може бути великим (близько 2,4 МБ), особливо при повільному з'єднанні.

Отже Blazer WebAssembly це чудовий вибір для створення просунутих веб додатків, що потребують високої продуктивності та автоматизації. Окрім використання базових технологій для побудови сторінок ви можете поєднати їх наприклад з Angular використовуючи його як заміну традиційним фреймворкам

JavaScript, таким як Vue.js або React, особливо якщо вам потрібні офлайн можливості та хороша реалізація коду для обробки логіки на стороні клієнта та на стороні сервера.

Цей фреймворк забезпечує високий рівень безпеки через влані вбудовані механізми захисту, такі як аутентифікація та авторизація. . Крім того, Blazor WebAssembly виконує код в середовищі пісочниці (sandbox), що обмежує доступ до системних ресурсів та підвищує безпеку додатків. Усі програми розроблені на основі цього фреймворку мають підтримку на різних найпопулярніших сучасних браузерах включаючи Google Chrome, Mozilla Firefox, Microsoft Edge та Safari. Це забезпечує високу сумісність та дозволяє користувачам взаємодіяти з додатком незалежно від обраного браузера.

В загальному Blazer це молодий, перспективний інструмент розробки програм на .NET. Універсальний засіб, що можна використовувати будь-де для створення інтерфейсів користувача, дозволяє розробникам зосередитися на одній мові програмування (C#), зменшуючи складність навчання та розробки. Це відкриває нові можливості для створення інтерактивних, ефективних і надійних веб додатків, які можуть працювати онлайн і офлайн за допомогою сучасних технологій і методів.

1.6.3. Використання ASP.NET для створення серверної частини

ASP.NET потужна веб платформа Microsoft, призначена для створення динамічних веб додатків. Він дозволяє розробникам створювати надійні, масштабовані та безпечні серверні програми, що досягають високого рівня продуктивності та доступності. Такі технології NET, як Blazor і ASP.NET, забезпечують повне рішення для створення сучасних веб додатків.

Давайте розберемо ключові переваги використання ASP.NET:

1. Проста структура розробки з багатьма рішеннями та бібліотеками, недоступними в NuGet. Це дозволяє розробникам додавати нові функції до своїх програм без необхідності створювати програму з нуля.

2. Підтримка модульної архітектури, що забезпечує простоту використання та гнучкість застосування.

3. Компоненти проміжного програмного забезпечення ASP.NET дозволяють керувати конфігураціями програми, додавати нові функції та покращувати продуктивність програми.

4. Вбудовані механізми для захисту ваших програм від поширених загроз, таких як CSRF (підробка запитів), XSS (міжсайтовий сценарій) і впровадження SQL.

5. ASP.NET Identity дозволяє користувачам швидко здійснювати автентифікацію та авторизацію, забезпечуючи при цьому високу безпеку даних;

6. За замовчуванням підтримується HTTPS, забезпечуючи безпечний обмін даними між клієнтом і сервером.

7. Швидка інтеграція з багатьма хмарними провайдерами, включаючи Microsoft Azure, що дозволяє розробникам розгортати, масштабувати та керувати програмами в хмарі. Azure надає багато послуг, таких як бази даних, бази даних, аналітика та машинне навчання, які дозволяють створювати потужні та швидкі веб-програми. Інтеграція Azure DevOps забезпечує безперервну інтеграцію та доставку (CI/CI), масштабованість робочого процесу та легкість розгортання.

8. Остання версія фреймворку, ASP.NET Core, відома своєю продуктивністю та функціоналом. Розроблена для роботи на сучасних серверах і підтримує асинхронне програмування, що дозволяє ефективно обробляти декілька запитів одночасно. В результаті ASP.NET Core може забезпечити більшу гнучкість і гнучкість для складних програм.

9. Підтримка інтеграції з Docker, завдяки якій розробники можуть створювати пакетні програми, прості у використанні та розгортанні в будь-якому середовищі.

10. Останньою перевагою є можливість використовувати сучасні методи та технології, такі як RESTful API, SignalR у реальному часі, WebSockets і GraphQL. Головне ASP.NET підтримує розробку мікросервісів, просто

необхідних у сучасному програмуванні. Вони полегшують створення модульних систем керування.

1.7. Дослідження аналогів додатку

Існує багато додатків, що надають різні набори інструментів для управління проектами, які залежить від специфіки проекту, потреб команди та бюджету. Давайте ознайомимося з подібними програмними продуктами. Наприклад Microsoft Project, підходить для комплексних проектів з великою кількістю ресурсів та завдань. Trello – це чудовий вибір для невеликих команд і простих проектів, що потребують візуалізації завдань. Asana пропонує потужні можливості для управління завданнями та проектами, а Jira є незамінним інструментом для команд розробників програмного забезпечення, що працюють за Agile методологіями. Вибір і використання відповідного інструменту дозволяє значно підвищити ефективність управління проектами та досягти найкращих результатів.

Розглянемо детальніше деякі з найпопулярніших програмних продуктів, які широко застосовуються в проектному менеджменті: Microsoft Project, Trello, Asana та Jira.

Microsoft Project – це один з найбільш відомих і широко використовуваних інструментів для управління проектами, який пропонує комплексні можливості для планування, управління і моніторингу проектів приклад інтерфейсу рис 1.8. Він дозволяє створювати деталізовані плани проектів, включаючи завдання, підзадачі та залежності між ними. Графік можна візуально зобразити за допомогою Gantt діаграм. Інструмент дозволяє ефективно розподілити ресурси (люди, матеріали, обладнання) між завданнями, оптимізувати їх використання і контролювати навантаження. Також є можливість управління фінансовими аспектами проекту, що включає: планування бюджетів, моніторинг витрат і прогнозування фінансових результатів.

Головні переваги:

- потужний набір функцій для повного контролю за процесом управління проектом та інтеграція з іншими продуктами Microsoft такими як, Excel чи SharePoint;

- висока гнучкість і можливості налаштування в залежності від потреб конкретного проекту.

- Серед недоліків можна примітити наступне:

- висока вартість ліцензії, що може обмежити доступ для малих підприємств;

- через великий функціонал і складний інтерфейс програма складна у використанні для новачків.

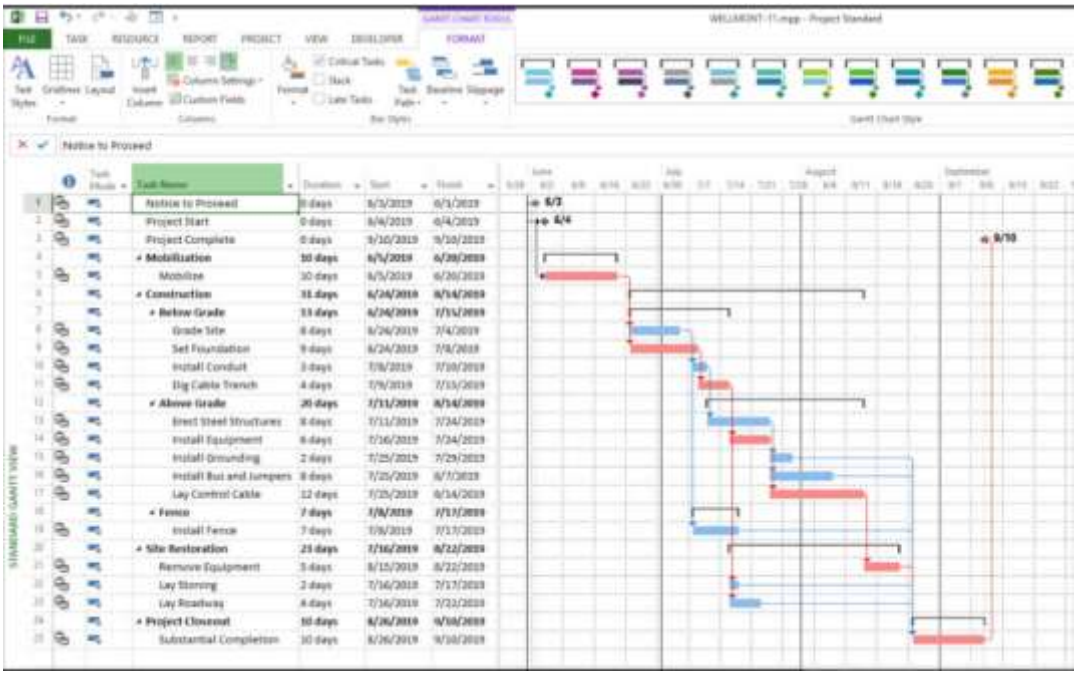

Рис 1.8. Інтерфейс сервісу Microsoft Project

Наступним ми розглянемо інструмент для управління проектами **Trello** рис. 1.9, що базується на концепції Kanban і пропонує візуально зрозумілий і простий спосіб організації завдань.

Користувачі можуть створювати дошки, списки та картки, які представляють завдання і підзадачі. Сервіс дозволяє командам спільно працювати над різними завданнями, додавати коментарі, вкладення та ставити

певні обмеження у часі, дедлайни. Підтримується інтеграції з багатьма іншими популярними сервісами, такими як Slack, Google Drive, Dropbox тощо, що дозволяє розширити функціональність і технічні можливості.

В загальному у програми простий і інтуїтивно зрозумілий інтерфейс, до того ж легкий у використанні. Сам по собі сервіс має безкоштовну версію, що збільшує можливість використання. Також надається гнучкий спосіб налаштування під конкретні потреби команди. Але сервіс не дуже зручний для великих складних проектів з багатьма залежностями між завданнями.

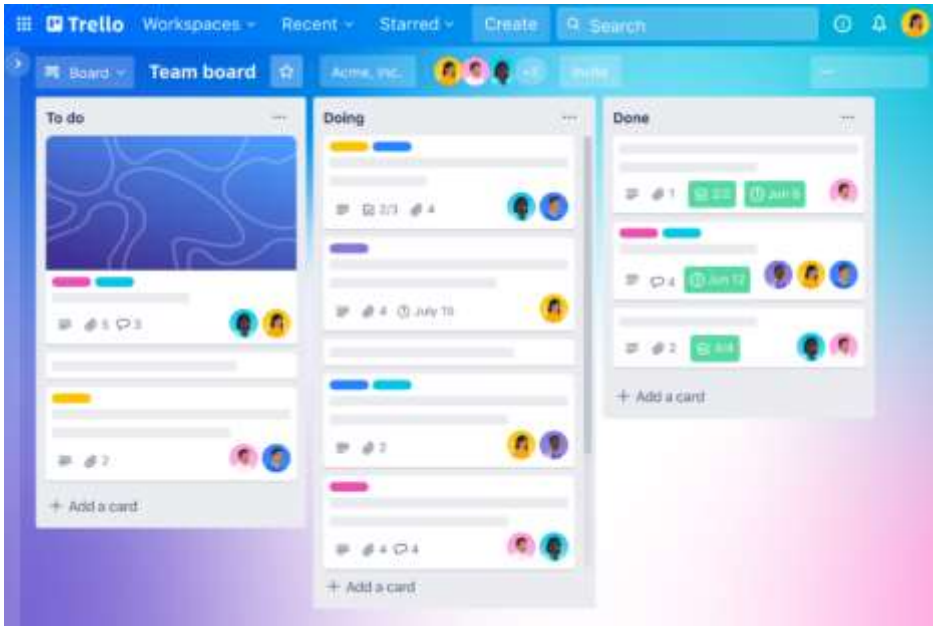

Рис. 1.9. Інтерфейс веб додатку Trello

Ще однією цікавою програмою є **Asana**, яка допомагає командам організовувати роботу та співпрацювати більш ефективніше. Вона дозволяє створювати проекти, завдання, під задачі при чому призначати відповідальних для кожного завдання, встановлювати дедлайни та пріоритети у виконані задач. Користувачі можуть створювати робочі простори для різних груп або проектів, що дозволяє структурувати роботу та підтримувати порядок і простіше відслідковувати результат роботи проекту. Сервіс забезпечує візуалізацію завдань у вигляді календаря та таймлайну рис. 1.10

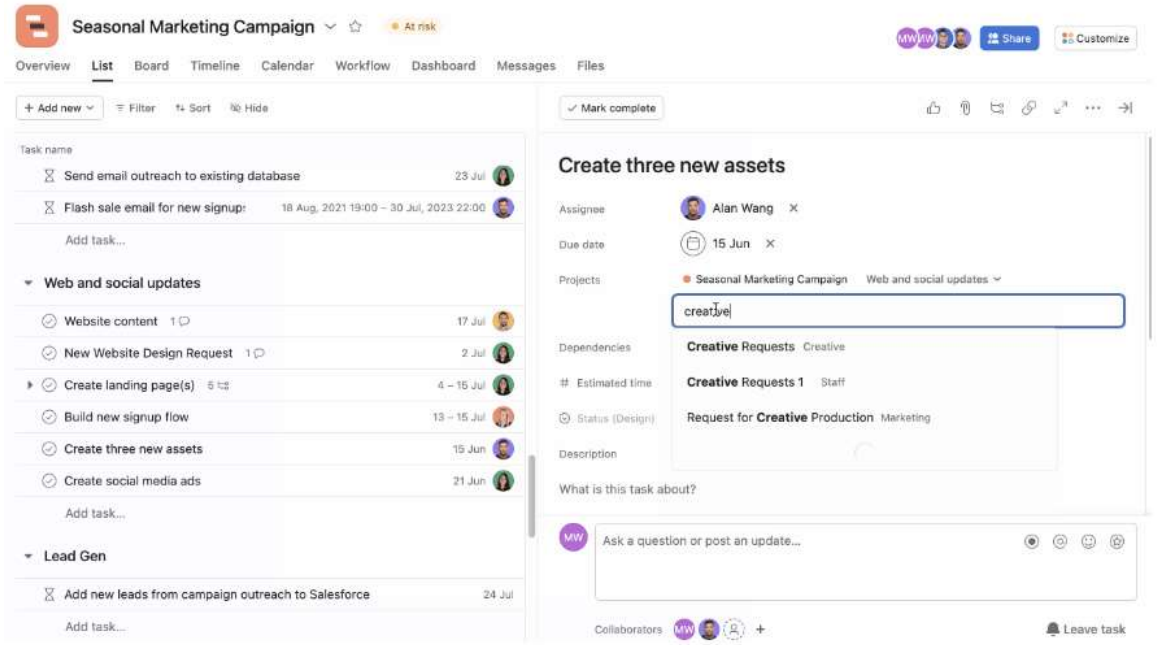

Рис. 1.10. Приклад того як виглядає інтерфейс у програмі Asana

Останнім ми розберемо застосунок **Jira** він підтримує усі гнучкі методології такі як Agile, Scrum, Kanban. Надає можливість створювати завдання, історії користувачів та інші елементи, слідкувати за їх статусом і призначати відповідальних осіб. Підтримка Scrum та Kanban дошки, допомагає командам організовувати роботу в гнучкому середовищі за допомогою зрозумілого візуального відображення. У середовищі є можливість використовувати набір звітів і аналітичних інструментів, які допомагають відстежувати продуктивність команди та прогрес проекту. Веб застосунок взаємодіє з такими сервісами, як Confluence, Bitbucket, GitHub, що дозволяє створити єдину систему для управління проектами. Приклад інтерфейсу і можливостей програми ми можемо переглянути на рис. 1.11.

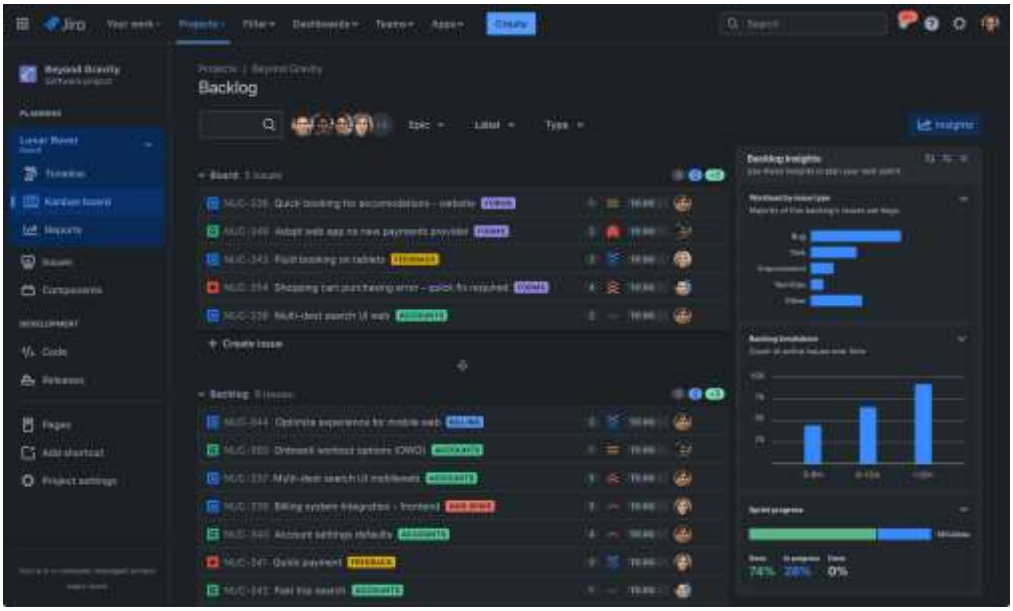

Рис. 1.11 Вигляд інструмент них засобів, що надає програма **Jira**

РОЗДІЛ 2. РОЗРОБКА ТА РЕАЛІЗАЦІЯ ВЕБ-ОРІЄНТОВАНОЇ ПЛАТФОРМИ УПРАВЛІННЯ ПРОЕКТАМИ

2.1. Постановка задачі та визначення вимог до сервісу «SMART»

Для реалізації завдання використовується мова програмування C# та фреймворк написаний для неї Blazor. Метою цього проекту є створення платформи для моніторингу стану роботи та підтримки командної роботи. Після детального аналізу та огляду різних веб застосунків було визначено найважливіші вимоги проекту, давайте розберемо їх детально:

- система повинна дозволяти користувачам автентифікацію та авторизацію. Це найважливіший аспект, необхідний для забезпечення безпеки та конфіденційності інформації та полегшення взаємодії. Користувачі створюють облікові записи і входять на основі своїх зареєстрованих даних;
- користувачі мають мати змогу керувати своїми обліковими записами та змінювати особисту інформацію, таку як ім’я, адреса електронної пошти та пароль. Ці функції забезпечать гнучкість і зручність для користувачі;
- платформа повинна дозволяти створювати нові проекти та приєднуватись до уже існуючих. Це допоможе користувачам налагодити командну роботу та досягти спільної мети;
- важливо переконатися, що рівень обслуговування можна контролювати. Користувача повинні знати, як все працює, хто що робить і яким буде результат. Це допоможе системі бути прозорою та полегшить керування бізнесом;
- користувачі повинні мати можливість роз приділяти завдання між учасниками, що буде гарантувати – робота виконується більш ефективно та у встановлені терміни;
- реалізація можливості залишати коментарі для завдань є дуже важлива для спілкування учасників сервісу. Це дозволить обговорювати деталі, уточнювати вимоги та надавати відгуки;

- спроба реалізувати чати для учасників проекту, щоб налагодити просте спілкування, яке дозволить швидко обмінюватися інформацією, обговорювати актуальні проблеми та негайно вирішувати їх. Це покращить роботу над проектом.

Окрім вимог до проекту також було проаналізовано і описано обмеження без виконання яких, користування платформою стає не можливим.

Вимоги до користувачів:

1. Для початку користувачі повинні створити власний обліковий запис, пройти процес реєстрації. Також потрібно буде ввести свою інформацію та особисту інформацію та налаштувати теми, щоб було комфортно працювати з системою.

2. Після створення облікового запису користувачі уже можуть увійти, використовуючи власні облікові дані.

3. Наступним кроком йде початок роботи. Ви повинні створити новий проект чи доєднатися до наявного. Це дозволяє розпочати працювати та використовувати головні функції платформи для менеджменту.

4. Потрібно створити завдання, виставити пріоритети та часові обмеження, щоб було легше керувати завданнями. Ці дії дозволяють організовувати роботу та встановлювати дедлайни для неї.

5. Коли завдання було створено, потрібно визначити особу, відповідальну за виконання цього завдання.

Ці принципи та процедури були придумані для створення відповідних та ефективних інструментів для командної роботи. Дотримання цих принципів допоможе в організації роботи і досяганню найкращих результатів для цілей у довгостроковій перспективі.

2.2. Вибір моделі розробки програмного засобу

Для розробки програмного продукту SMART було обрано ітераційну гібридну модель. Ця модель поєднує в собі елементи класичної моделі

водоспаду та методології гнучкої розробки, що дозволяє максимально ефективно реагувати на можливі мінливі вимоги та умови під час розробки. Рішення, щодо цього було прийнято у результаті аналізу продукту. Вона надає можливість розпочати роботу з продуктом поетапно поступово розширюючи фунціонал, поєднуючи нові модулі та можливості до сервісу. Поступово вибудовувалися нові вимоги і розроблялися різні модулі, які поєднувалися до коду з кожною новою ітерацією.

Ітераційна модель складається з певних етапів (стадій), які почергово виконуються у кожному новому циклі роботи рис. 2.1.

Рис. 2.1 Принцип роботи ітераційної моделі розробки.

Давайте зупинимось на цьому і розберемо кожний з етапів розробки більш детально, та ознайомимось з особливостями моделей.

На першому етапі, створювалися вимоги до програмного продукту, після відповідного аналізу аналогічних сервісів. Було створено технічне завдання та визначено функціональні та не функціональні вимоги. Згодом стратегія поділяє процес розробки на окремі ітерації, кожна з яких має чітко визначені цілі та завдання. Пріоритети додатково визначаються для кожної окремої ітерації. Наступним кроком для нас є проектування, ми розробляємо загальну архітектуру системи чітко визначаємо модулі та взаємодію між ними. Тут

пояснюється внутрішня структура кожного модуля, вибираються технології та інструменти для реалізації.

Третім етапом ми виконуємо розробку певних окремих модулів системи. Кодування і реалізація використовує елементи водоспадної тобто каскадної моделі рис. 2.2. У цьому випадку код пишеться відповідно до технічних вимог та архітектури системи з використанням сучасних мов програмування та фреймворків. Після завершення кожної ітерації розробки ми проводимо тестування кожного модуля окремо та всієї системи в цілому. Виконуються тести для виявлення можливих помилок та недоліків. Для тестування було обрано регресійний метод, він передбачає повторну перевірку всієї системи після внесення будь яких змін, щоб переконатися, що вони не створюють нових помилок.

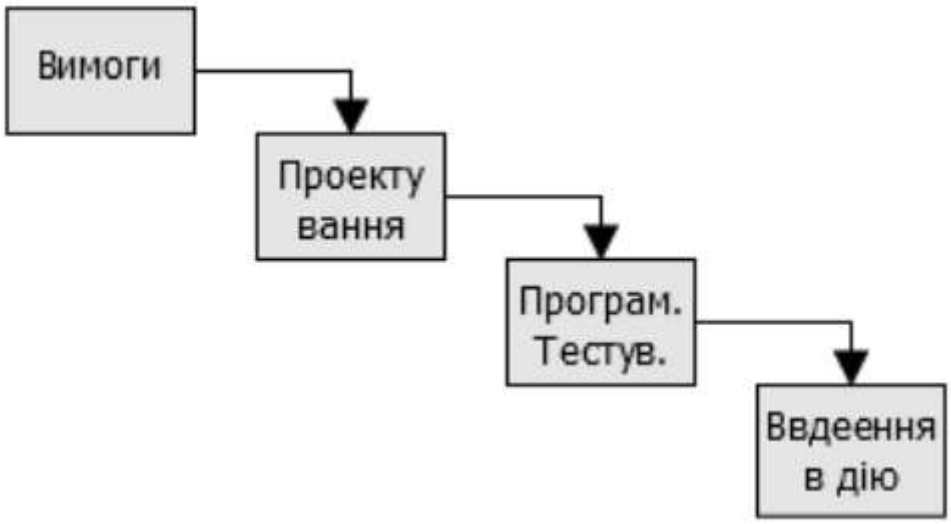

Рис. 2.2 Базові кроки для реалізації каскадної моделі.

На кінець етап впровадження, відбувається забезпечення вимог до інсталяції програмного продукту, таких як конфігурація операційної системи, інсталяція необхідного програмного забезпечення (сервер бази даних, веб сервер, залежності тощо). Відбувається розгортання програми, переміщення файлів програми на цільову платформу, включаючи копіювання файлів, налаштування файлів конфігурації, встановлення баз даних тощо. Також

налаштовується параметри: системи, безпеки, створення користувачів і необхідних прав доступу, визначення параметрів мережі.

Переваги змішаної моделі розробки:

- гнучкість конструкцій дозволяє користувачам швидко адаптуватися до мінливих потреб і умов розробки, зменшуючи ризик невідповідності кінцевого продукту очікуванням;
- безперервне тестування та інтеграція нових функцій забезпечує випуск високоякісного програмного забезпечення та раннє виявлення помилок;
- розподіл процесу розробки на частини дозволяє зосередитися на найважливіших і пріоритетних завданнях, що покращує та прискорює роботу.
- постійний зворотній зв'язок допомагає нам задовольняти усі потреби і функції та підвищити рівень задоволеності у користуванні системою;

Загалом гібридна модель ітераційного типу є ефективним способом розробки програмних продуктів, оскільки поєднує найкращі практики класичних та гнучких методів, що забезпечує високу якість та придатність кінцевого продукту. Модель дозволяє легко швидко реагувати на мінливі потреби та ситуації, забезпечуючи стабільну продуктивність та швидкий хід роботи.

2.3. Загальний опис проекту

Для впровадження продуктів SMART було обрано принцип Clean Architecture (чиста архітектура) з Clien-Server підходом. Цей метод забезпечує високу гнучкість, модульність, масштабованість і стабільність системи, підтримує зміни і нові вимоги, щодо експлуатації.

Чиста архітектура забезпечує чіткий поділ системи на декілька рівнів, що гарантує незалежність у розробці, і відокремлює бізнес логіку від технологічних частин. Основним принципом Clean Architecture є правило, що зовнішні ресурси залежать від внутрішніх, а не навпаки рис 2.3. Це дозволяє

легко змінювати технології та інструменти, не впливаючи на бізнес процеси. Уся система розділена на частини:

- верхній шар – це UI частина, тут доступні інтерфейси користувача, такі як сторінки, API чи інші додатки;
- середній рівень або ж рівень програми, де використовується контекст, який визначає усю бізнес-логіку та порядок виконання дій в системі;
- нижній рівень містить основні бізнес-правила та моделі домену. Цей шар ні на що не посилається і є незалежним від інфраструктури;
- інфраструктурний рівень включає усі зовнішні служби та інструменти, такі як бази даних, веб сервери, фреймворки тощо.

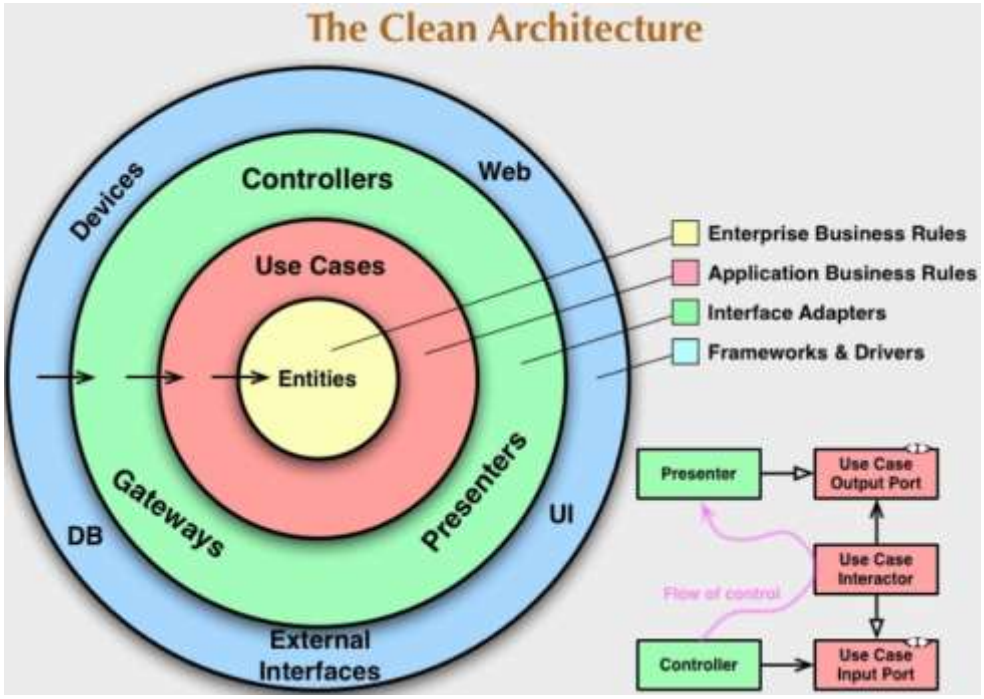

Рис. 2.3 Оригінальна схема Clean Architecture.

Для роботи з даними в програмному продукті SMART використовується патерн репозиторію (Repository Pattern). Він виступає проміжним рівнем між бізнес-логікою та базою даних, та відповідає за відокремлення бізнес процесів від деталей доступу до інформації. Основні переваги патерну:

- ізоляція бізнес логіки, приховує деталі реалізації доступну до даних, що дозволяє зосередитися на бізнес логіці;

- можливість легко змінити репозиторій на рейкову реалізацію програми для модульного тестування;
- загальні операції доступу до даних централізовано в одному місці, що дозволяє повторно використовувати код і уникати дублювання.

Clien-Server підхід, полегшує розподіл системних завдань між клієнтом і сервером, забезпечуючи гнучкість і ефективність роботи. Основними компонентами у ньому є:

1. Сторона клієнта, що відповідає за взаємодію з користувачем. Містить інтерфейси та логіку взаємодії з сервером. Клієнтська сторона може бути реалізована як веб застосунок, мобільна або десктопна програма.

2. Сторона сервера ж виконує усю бізнес логіку та доступ до бази даних для обробки запитів з клієнта. Сервер має свої API, через які клієнт спілкується з сервером.

Загалом, вибір Clean Architecture, Client-Server підходу та Repository Pattern забезпечує високу якість, надійність та гнучкість програмного продукту SMART, що значно спрощує код, полегшує підтримку і розробку сервісу.

2.4. Обґрунтування вибору інструментальних засобів розробки

2.4.1. Вибір Visual Studio для розробки веб орієнтованої платформи управління проектами

Використовувати Visual Studio для створення проекту було обрано з ряду переваг, що надає це IDE. Це повноцінне середовище розробки, яке об’єднує всі інструменти, необхідні для створення сучасних додатків. Завдяки зручному та потужному редактору коду програмувати стає набагато приємніше і простіше. Наявність словникових визначень, автоматичних блокувань та інструментів аналізу коду, допомагають користувачам працювати швидше та ефективніше.

Visual Studio надає прості інструменти виправлення для швидкого пошуку помилок у ваших проектах. Покроковий аналіз коду допомагає розробнику визначити основні причини затримки та покращити продуктивність

виконання програми. Це скорочує час на усунення дефектів і покращує загальну якість продукту.

Однією з найважливіших переваг Visual Studio є наявність засобів автоматизації тестування програмного забезпечення, що дозволяють розробникам легко створювати та запускати тести для процесу аналізу результатів та коректності інтеграції з іншими системами. Це забезпечує можливість слідкувати за кодом та запобігати помилкам на ранніх стадіях розробки. Ще однією вагомою перевагою IDE є підтримка різноманітних систем контролю версій, що дозволяє розробникам ефективно співпрацювати і зберігати прогрес роботи. Інтегрується з Git, Azure DevOps та іншими подібними програмами, щоб краще відстежувати зміни коду та сприяти співпраці команди. Це особливо важливо для великих проектів із кількома зацікавленими сторонами.

Однією з особливостей є використання NuGet для встановлення різних сторонніх бібліотек та компонентів до своїх проектів. Це значно спрощує процес керування залежностями, оскільки він повністю налаштовується цим менеджером, що майже унеможливлює некоректне встановлення і використання сторонніх доповнень. Зручний і простий інтерфейс пошуку дозволяє полегшити знаходження необхідного, зекономити час та зусилля, забезпечуючи доступ до великої кількості готових рішень, що можуть бути використані для розширення функціональності додатку.

Крім усього вищезазначеного Visual Studio пропонує користувачам легко створювати та підтримувати локальні бази даних. За допомогою таких інструментів, як SQL Server LocalDB або SQLite, розробники можуть швидко завантажувати бази даних для тестування та створення без необхідності додаткової інтеграції з іншими серверами. Це забезпечує гнучкість і спрощує процес розробки перед розгортанням продукту уже на робочих серверах

Важливо зазначити, що багато авторитетних компаній використовують Visual Studio як головний інструмент у розробці програмного забезпечення. Знання та вміння використовувати Visual Studio є цінним інструментом для

роботодавців і можуть значно допомогти в процесі пошуку роботи та просування у сфері. Крім того, існує багато навчальних посібників і ресурсів для вивчення Visual Studio для багатьох користувачів. Тому вибір цієї IDE для створення і керування програмним забезпеченням є доцільним і практичним рішенням, яке надає великі переваги, підвищує ефективність і спрощує розробку програмного забезпечення.

2.4.2. Застосування мови C# з фреймворком Blazor для розробки і створення програмного засобу

Використання мови програмування C# обрано з урахуванням її широких можливостей, ефективності та популярності серед розробників програмного забезпечення.

Головні переваги використання C#:

1. Підтримка і використання парадигми об'єктно-орієнтованого програмування під час розробки програмних засобів. ООП дозволяє розробникам моделювати об’єкти та реальні системи, щоб зробити код читабельним, структурованим та масштабованим і придатним для подальшої підтримки. Це особливо важливо для великих, складних проектів, де модульність і повторне використання коду є критичними факторами успіху.

2. C# багата інструментами та бібліотеками, що роблять процес розробки простішим і швидшим. Починаючи від потужних фреймворків, що дозволяють розробляти веб застосунки до більш спеціалізованих бібліотек для роботи з графікою, базами даних, мережами тощо..

3. Підтримка платформи .NET, яка надає користувачам різноманітні інструменти та функції для створення різноманітних програмних продуктів.

4. Мова програмування відома своєю ефективністю та швидкодією, що робить її ідеальним вибором для веб застосунків, які вимагають високої реакції відгуку. Оптимізований компілятор та ефективне керування пам’яттю,

забезпечують високу швидкість виконання коду, що є важливим для додатків із високим навантаженням.

5. C# підтримує інтеграцію з іншими популярними технологіями, допомагаючи створювати потужні та гнучкі додатки. Починаючи від об’єднання з JavaScript та різноманітними бібліотеками фронтенд розробки до співпраці з хмарними службами.

Одним з найцікавіших фреймворків, що створювалися для цієї мови є платформа Blazor, що дозволяє створювати інтерактивні веб програми за допомогою тільки C# без традиційного використання JavaScript. Це забезпечує єдиний мовний стек для серверного та клієнтського коду, що полегшує розробку додатків і подальше керування ними. Також можна виділити й інші головні переваги у використанні цієї платформи:

- Blazer підтримує корпоративні (Blazer Server) і клієнтські (Blazer Web Assembly) програми. Це дозволяє підібрати найкращу архітектуру для конкретної ситуації та проекту;
- забезпечення повної інтеграції з .NET, що дозволяє використовувати усі існуючі бібліотеки та інструменти платформи спрощуючи процес розробки;
- завдяки використанню WebAssembly підвищується продуктивність застосунків, дозволяючи перенести виконання деяких процесів безпосередньо в браузері;
- ці групи технологій мають широкий сектор документації і велику спільноту розробників, що дозволяє швидко знаходити усю необхідну інформацію.

Таким чином, проаналізувавши усі переваги програмних засобів було вирішено використовувати Blazor WebAssembly, для забезпечення виконання деяких функцій безпосередньо на клієнті без необхідності постійної взаємодії з сервером. Середовищем розробки для нас стане IDE Microsoft Visual Studio, що надає має багато функціональних переваг і дозволяє створювати і підтримувати подібні види проектів.

2.4.3. Застосування MSSQL Server для зберігання і розробки бази даних

Вибір MSSQL Server для зберігання та розробки бази даних є логічним і вигідним рішенням. Як одна з найпопулярніших систем керування базами даних вона пропонує широкий спектр інструментів і функцій, які забезпечують надійну обробку і взаємодію з даними.

Вбудовані методи забезпечують високу безпеку у зберіганні даних і найголовніше підтримує масштабованість для обслуговування великих обсягів інформації. Оскільки збереження даних є головним для будь-якого проекту зупинимося тут детальніше. MSSQL Server пропонує потужні функції безпеки, включаючи шифрування, автентифікацію користувачів, контроль доступу на рівні рядків та інтеграцію з Active Directory – це база даних і набір служб, які з'єднують користувачів із мережевими ресурсами, необхідними для виконання роботи [25]. Це дозволяє захистити конфіденційні дані від несанкціонованого доступу та забезпечити дотримання стандартів безпеки.

Самий менеджер надає розробникам і адміністраторам баз даних різноманітні інструменти для створення, керування та оптимізації баз даних. Він забезпечений простим у використанні інтерфейсом для виконання запитів SQL, моніторингу продуктивності та налаштування серверів. Крім того, інтеграційні можливості з Visual Studio дозволяють розробникам працювати з базами даних безпосередньо в середовищі розробки Серед значних переваг системи є підтримка складних запитів та можливість аналізу завдання за допомогою потужного механізму обробки вибірки даних та інструментів оптимізації. Такі функції, як індекси, індексовані подання, збережені процедури, дають змогу швидко й ефективно виконувати складні операції з великими обсягами даних.

Для розробки подібних продуктів дуже важливим є можливість інтегрується з іншими продуктами Microsoft, такими як Azure, Power BI та .NET, тож ви будете мати можливість створювати комплексні рішення для

керування проектами. А взаємодія з хмарними службами Azure дозволяє розширити інфраструктуру бази даних у хмарі, що забезпечує більшу масштабованість і гнучкість. Через це система також забезпечена розширеними можливостями для оптимізації запитів, кешування та ефективного керування ресурсами. Такі функції, як асинхронне виконання запитів та повторне використання методів, а також підтримка механічної пам'яті забезпечують швидку обробку даних навіть у великих і складних базах даних.

Таким чином, MSSQL Server забезпечує надійне, безпечне та ефективне керування даними за допомогою свого функціоналу, надаючи можливість резервного копіювання та оновлення бази даних. Різноманітність функцій і можливостей дозволяє створювати ефективні та масштабовані рішення, які відповідають потребам і вимогам сучасних проектів.

2.5. Особливості програмної реалізації

Розробка подібних проектів вимагає ретельного планування і використання передових технологій у проектуванні та розробці для забезпечення високої продуктивності, розширюваності та безпечності роботи системи. Давайте для початку ознайомимось з архітектурними особливостями, які були використані для реалізації програмного продукту.

Платформа реалізована за допомогою архітектури клієнт-сервер, де інтерфейс і бекенд чітко розділені. Також було дотримано принципи Clean Architecture, щоб забезпечити гнучкість і легку підтримку рис 2.4. Інтерфейс реалізовано за допомогою Blazor, що дозволяє використовувати C# як для серверної, так і для клієнтської сторони. Сервер базується на ASP.NET Core, забезпечуючи високу продуктивність і надійність.

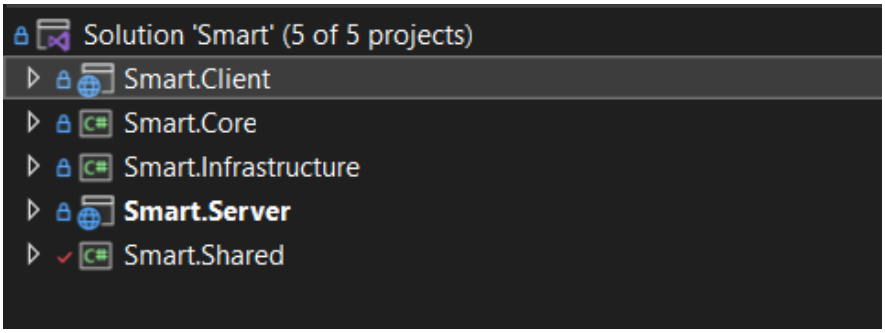

Рис. 2.4 Структура проекту з поділом на різні проекти

Отож детально розглянемо кодову частину нашого проекту. Нижче можемо побачити реалізацію класу ApplicationDbContext, який є основою для роботи з базою даних у нашому проекті. Цей клас унаслідується від IdentityDbContext з моделлю User у якості основи, що дозволяє нам використовувати вбудовані можливості ASP.NET Identity для уже готового налаштування управління користувачами. У конструкторі ми передаємо параметри конфігурації бази даних. Далі ми визначаємо і створюємо DbSet для кожної нашої сутності, що має мати представлення у нашій базі даних рис 2.5. Це дозволяє гнучко налаштовувати підключення до бази даних.

```
11 references
public class ApplicationDbContext : IdentityDbContext<User>
{
    0 references
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
        : base(options)
    {
    }

    //  DbSet для кожної  сутності
    0 references
    public DbSet<Attachment> Attachments { get; set; }
    0 references
    public DbSet<Chat> Chats { get; set; }
    0 references
    public DbSet<ChatMessage> ChatMessages { get; set; }
    0 references
    public DbSet<Comment> Comments { get; set; }
    0 references
    public DbSet<Project> Projects { get; set; }
    0 references
    public DbSet<UserProject> UserProjects { get; set; }
    0 references
    public DbSet<WorkItem> WorkItems { get; set; }
```

Рис. 2.5 Оголошення наших таблиць з використанням Entity Framework

У методі OnModelCreating ми знову викликаємо базову реалізацію та додатково описуємо і налаштовуємо зв’язки між нашими таблицями, прописуючи додаткові обмеження. Для нашої DB створено окремий проект, згідно принципу чистої архітектури. У ньому власне описано повний функціонал нашої бази даних починаючи з опису шаблону, закінчуючи усіма міграціями і змінами.

```
0 references
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);

    //налаштування усіх звязків між таблицями

    modelBuilder.Entity<UserProject>()
        .HasKey(up => new { up.UserId, up.ProjectId });

    modelBuilder.Entity<UserProject>()
        .HasOne(up => up.User)
        .WithMany(u => u.UserProjects)
        .HasForeignKey(up => up.UserId);

    modelBuilder.Entity<UserProject>()
        .HasOne(up => up.Project)
        .WithMany(p => p.UserProjects)
        .HasForeignKey(up => up.ProjectId);

    modelBuilder.Entity<WorkItem>()
        .HasOne(w => w.Project)
        .WithMany(p => p.WorkItems)
        .HasForeignKey(w => w.ProjectId);

    modelBuilder.Entity<WorkItem>()
        .HasOne(w => w.ParentTask)
        .WithMany(w => w.SubTasks)
        .HasForeignKey(w => w.ParentTaskId);

    modelBuilder.Entity<WorkItem>()
        .HasOne(w => w.AssignedUser)
        .WithMany(u => u.AssignedTasks)
        .HasForeignKey(w => w.AssignedUserId);

    modelBuilder.Entity<WorkItem>()
        .HasOne(w => w.CreatedByUser)
        .WithMany()
        .HasForeignKey(w => w.CreatedByUserId);

    modelBuilder.Entity<Chat>()
      .HasMany(c => c.ChatMessages)
      .WithOne(cm => cm.Chat)
      .HasForeignKey(cm => cm.ChatId);

    modelBuilder.Entity<ChatMessage>()
        .HasOne(cm => cm.Chat)
        .WithMany(c => c.ChatMessages)
        .HasForeignKey(cm => cm.ChatId);
```

Рис. 2.6 Налаштування зв’язків між нашими таблицями

Також важливо зазначити що ми використовуємо один із ключових патернів у проектуванні, що спрямований на абстрагування доступу до даних і управління ними. Він допомагає відокремити бізнес-логіку від логіки доступу до даних, роблячи код більш організованим і легко підтримуваним. Це патерн репозиторію, для його опису ми використовуємо інтерфейс рис.2.7. Як бачимо використовувати усі ці методи можуть лише ті класи, що реалізують IBaseEntity. Перевагою такого підходу є уникнення непотрібного дублювання коду для опису взаємодії кожної моделі з базою даних. Подібні репозиторії можуть бути легко змінені чи розширені без впливу на інші частини системи. Це дозволяє змінювати реалізацію методів не порушуючи структури проекту і змін у бізнес логіці.

```
6 references
public interface IRepository<TEntity> where TEntity : IBaseEntity
{
    2 references
    Task<TEntity> AddAsync(TEntity entity);
    1 reference
    Task<IEnumerable<TEntity>> GetAllAsync();
    1 reference
    Task<TEntity> GetByKeyAsync<TKey>(TKey key);
    1 reference
    Task<TEntity> GetByPairOfKeysAsync<TKey, TKey1>(TKey key, TKey1 key1);
    2 references
    Task UpdateAsync(TEntity entity);
    2 references
    Task DeleteAsync(TEntity entity);
    4 references
    Task<int> SaveChangesAsync();
    1 reference
    Task AddRangeAsync(List<TEntity> entities);
    3 references
    Task<TEntity> GetFirstBySpecAsync(ISpecification<TEntity> specification);
    2 references
    Task<IEnumerable<TEntity>> GetListAsync(
        Expression<Func<TEntity, bool>> filter = null,
        Func<IQueryable<TEntity>,
        IOrderedQueryable<TEntity>> orderBy = null,
        string includeProperties = null);
    1 reference
    Task<TEntity> GetEntityAsync(
          Expression<Func<TEntity, bool>> filter = null,
          string includeProperties = null);
}
```

Рис. 2.7 Оголошення методів взаємодії з БД у нашому репозиторію

Оскільки ми використовуємо різні додаткові бібліотеки і сторонні ресурси для розробки нам необхідно їх описати їх у нашому методі

конфігурації, щоб мати можливість використовувати їх у коді в подальшому. Тут прописано використання методу отримування стрічки з’єднання з нашою базою даних. Визначено різні додаткові бібліотеки типу Fluent Validation , Auto Mapper та оголошено усі сервіси рис. 2.8.

Рис. 2.8 Налаштування усіх необхідних бібліотек і додаткових сервісів

Для кожного набору завдань у нашому проекті ми створюємо окремий інтерфейс, що містить перелік методів, необхідних для коректної роботи. Кожний інтерфейс реалізується власним сервісом, в якому уже описана логіка роботи всіх функцій. Усі ці класи зберігаються в ядрі програми оскільки відносяться до бізнес логіки. Для виклику цих методів на стороні клієнта використовуються так звані контролери. Розглянемо приклад функції автентифікації за прикладі інтерфейсу IAuthenticationService рис. 2.9. Тут перечисленні усі методи необхідні для реєстрації і авторизації користувачів. Їх логіка описана у відповідному сервісі, методи якого викликаються власним контролером, що забезпечує взаємодію з клієнтом.

Рис. 2.9 Реалізація інтерфейсу для аутентифікації користувача

Проглянемо як виглядає наш метод реєстрації у контролері рис. 2.10. Цей метод позначений як HTTP POST, що означає – метод отримує дані від клієнта для подальшої обробки, також існують позначення GET – передає інформацію на сторінки, PUT – використовуються для редагування даних зміни інформації та DELETE служить для видалення. Як можна побачити сама модель User не використовуються для обміну інформації між клієнтом і сервером, оскільки це не дуже вигідно і ресурси затратно. Тому для цього ми використовуємо DTO (Data Transfer Objects) які містять тільки необхідні для передачі чи отримання дані. Усі вони розміщуються у спільному проекті Shared, де зберігаються ресурси, що використовуються як на сервері, так і на клієнті. Кожний метод контролера може повертати різні статуси виконання завдання, такі як успішно (**200)** чи не знайдено (**404)**, а також і сам результат запиту. Обмін інформацією відбувається через конвертацію даних формат JSON.

```
[HttpPost]
[Route("registration")]
0 references
public async Task<IActionResult> RegistrationAsync([FromBody] UserRegistrationDTO registrationDTO)
{
    var user = new User()
    {
        UserName = registrationDTO.Email,
        Lastname = registrationDTO.Lastname,
        Firstname = registrationDTO.Firstname,
        Email = registrationDTO.Email,
        BirthDate = registrationDTO.BirthDay,
        ImageUrl = "userBase.png"
    };
    await _authenticationService.RegistrationAsync(user, registrationDTO.Password, SystemRoles.User);

    return Ok();
}
```

Рис. 2.11 Опис методу реєстрації користувача

На клієнтські частині для побудови сторінок було використано відкриту бібліотеку Radzen для Blazor, що надає широкий спектр різноманітних компонентів, стилів та додаткових функцій типу сповіщень чи контекстних меню. Це все дозволяє прискорити і покращити якість сторінок. Для прикладу розберемо сторінку реєстрації рис. 2.12. Кожна сторінка починається з визначення маршруту за яким вона буде доступна для цього використовується атрибут @page "/ register " проте головна сторінка зажди завантажується першою, тому має пустий маршрут. Далі ми маємо ввести необхідні простори імен і визначити залежності. У цій формі нам потрібно заповнити поля такі як: ім’я, прізвище, адреса електронної пошти, пароль і дата народження. Компоненти Radzen використовуються для кожного поля, наприклад RadzenTextBox, RadzenPassword і RadzenDatePicker, щоб забезпечити зручний інтерфейс. Кнопка реєстрації створена на основі компонента RadzenButton, а кнопка сторінки входу — на основі компонента RadzenLink.

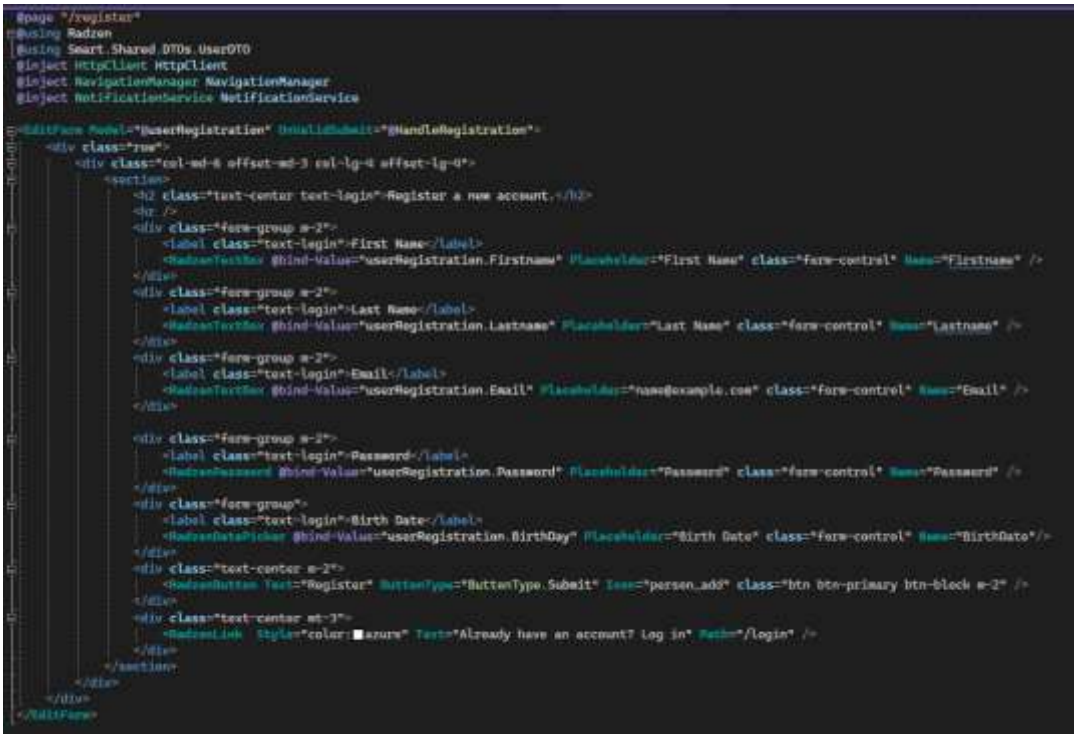

Рис. 2.12. Програмний код сторінки для реєстрації користувача.

Метод HandleRegistration відповідає за обробку і подання форми. Він виконується, коли користувач натискає кнопку реєстрації. У цьому методі ми надсилаємо HTTP POST запит до API для передачі даних на сервер де вони будуть протестовані та за успішності відбудеться реєстрація нового користувача. Метод спочатку надсилає введені користувачем дані на сервер за допомогою запиту api/authentication/registration. Якщо сервер успішно обробляє запит і повертає статус успіху, користувач перенаправляється на сторінку входу. Якщо станеться помилка, користувач отримає сповіщення від служби сповіщень, яка повідомить про невдалу спробу реєстрації та надасть детальну інформацію про помилку.

2.6. Організація тестування та налагодження програмного засобу

Наш сайт призначений для надання інструментальних засобів в управлінні проектом. Він надає можливість створювати вати проекти та завдання у них. Для початку варто зазначити, що першочергово відбувалося

тестування наших методів контролера через Swagger UI. Після створення кожного методу контролера було проведено перевірку на коректність роботи. Основним методом зв’язку бекенду з клієнтом є використання HTTP запитів для виклику різних серверних API.

Фронтенд частину була побудована з використання бібліотеки Radzen яка надавала готові компоненти та стилі для елементів інтерфейсу. Також було використано систему власних css стилів та базову html розмітку для налагодження коректного відображення картинки.

Продукт було протестовано на коректність виконання головних функцій:

- реєстрація, логінізація;
- створення проектів та до єднання до них;
- створення завдань та відображення їх статусу на сторінці;
- редагування видалення даних;
- можливість комуні кування через простий функціонал чату.

Результати роботи програмного продукту, можна побачити на рис. 2.13 – 2.16.

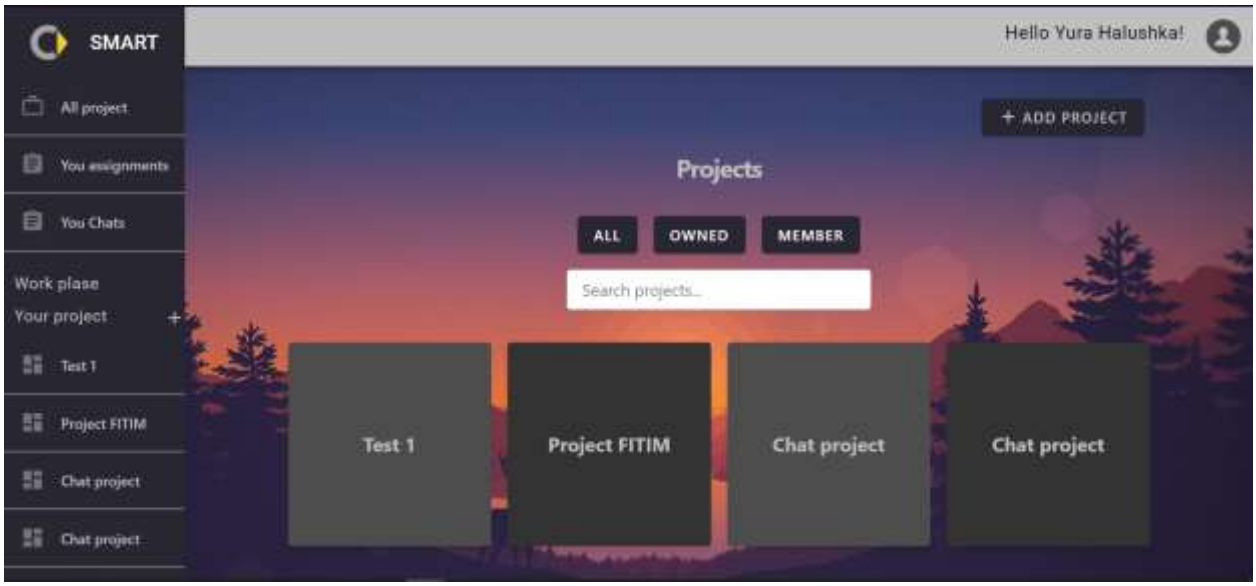

Рис. 2.13 Вигляд сторінки з усіма доступними проектами

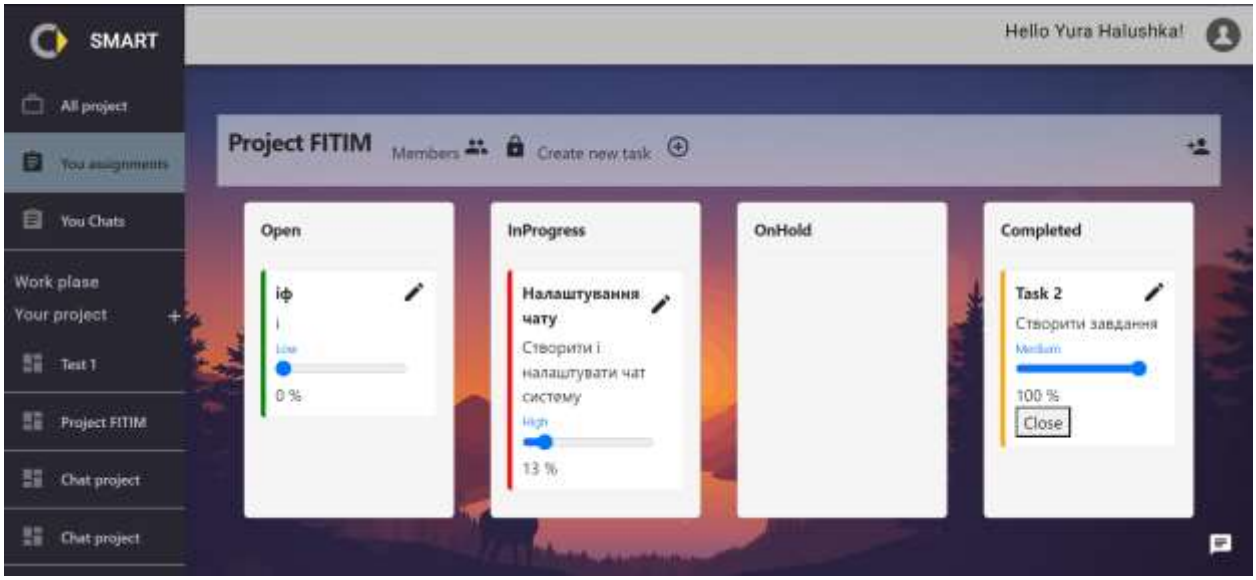

Рис. 2.14 Сторінка проекту з функціоналом відображення і створення завдань

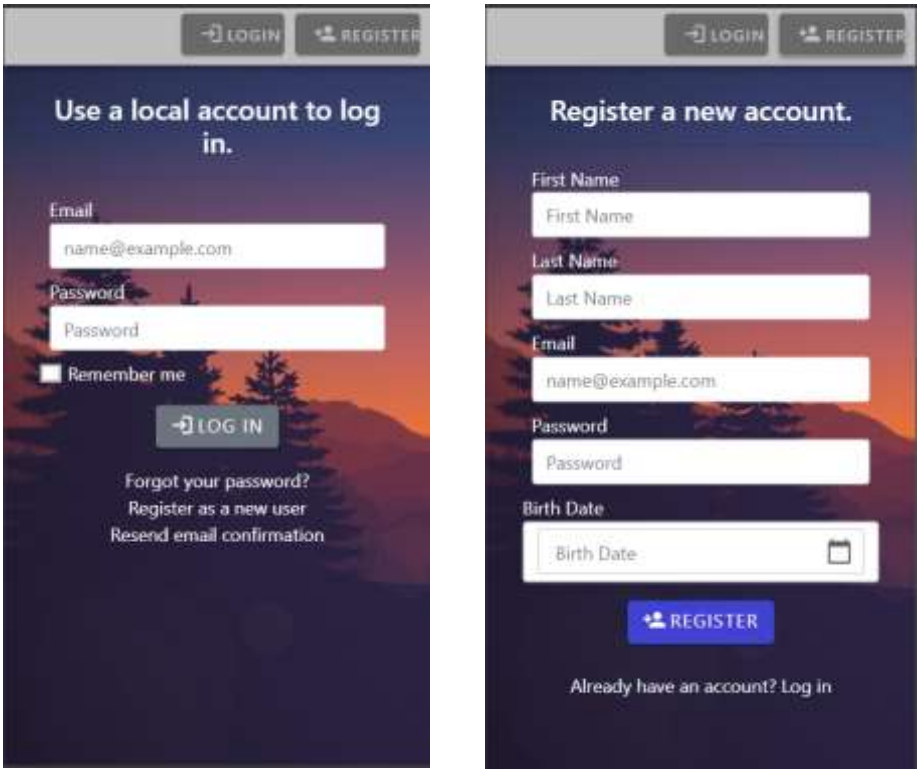

Рис. 2.15 Вигляд сторінок для входу та реєстрації облікового запису

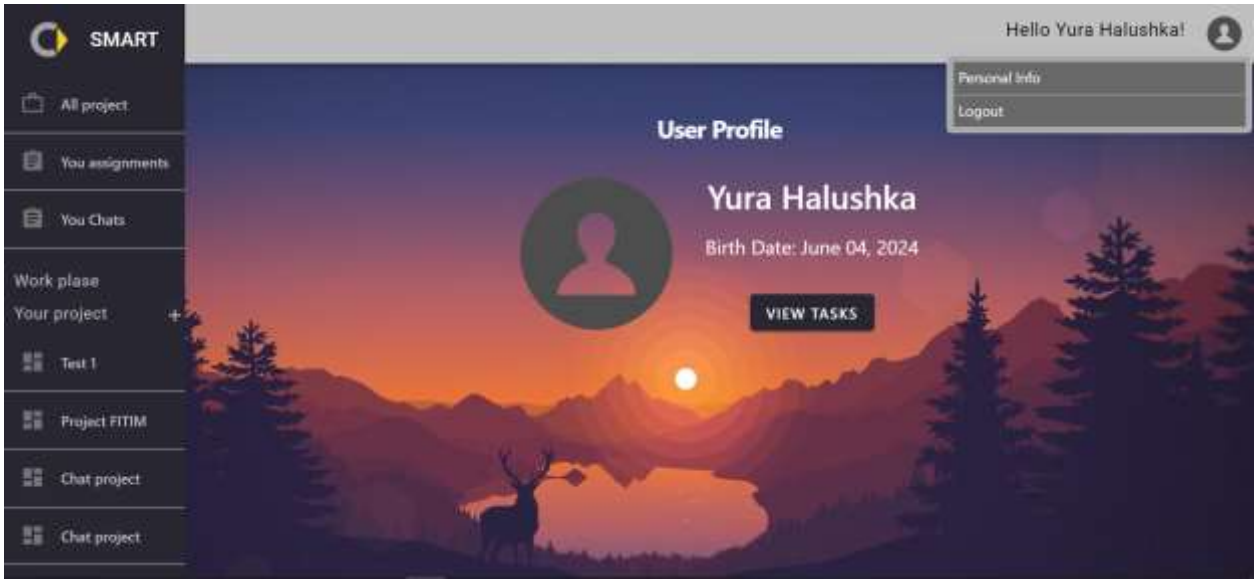

Рис. 2.16 Головна сторінка профілю користувача

2.7. Рекомендації по використанню та впровадженню програмного засобу

Програмний засіб розроблений з використанням технології ASP.NET, Blazor. Продукт надає можливість розподіляти задачі і керувати станом їх виконання у межах конкретного проекту, надає можливість комунікації. Для коректної роботи функціоналу сайту потрібно:

1. Мати стабільне підключення до Інтернету.
2. Створити акунт та увійти до нього.
3. Приєднатися до чужого проекту чи створити власний.
4. Розмістити завдання на дошках, та відслідковувати процес їх виконання, редагувати дані за потреби.

Для запуску програми на новому пристрої потрібно:

- Microsoft Visual Studio;
- Asp.Net and web development;
- .NET 7.0;
- MSSQL server;
- Запуск можна виконати на Windows чи macOS.

ВИСНОВКИ

У ході роботи ми ознайомилися з різними найпопулярнішими методами управління проектами та проаналізували різні моделі розробки проектів, що дозволило визначити найбільш оптимальні. Одним із таких способів став вибір фреймворку Blazor для створення веб застосунку. Він дозволяє писати повнофункціональні сайти за допомогою мови програмування C# без необхідності використання JavaScript.

Було розроблено програмний продукт, який надає засоби управління проектами, розподілу задач. Застосунок дозволяє створювати облікові записи, відстежувати статус завдань, редагувати та видаляти інформацію, а також спілкуватися в рамках проекту. Програмний продукт значно спрощує процес планування та управляння проектом, надаючи користувачам інтуїтивно зрозумілий інтерфейс і ефективні інструменти для спільної роботи. Використання Blazor надало високу продуктивність і надійність програмного забезпечення, що сприяє підвищенню ефективності у роботі і досягненню поставлених цілей в заданих умовах.

Під час тестування були виявлені певні недоліки технології Blazor. Незважаючи на те, що він забезпечує досить високу швидкість роботи при дуже великому навантаженні, дані завантажуються значно повільніше. Хоча це відкритий продукт і він має дуже велику аудиторію користувачів, через перевантаження інформації, її пошук стає дещо важчим. Це ускладнило роботу з деякими методами і розбиратися з новими та застосуватися їх на практиці виявилось трішки складно. В результаті застосунок вийшов не ідеальним, іноді трапляється збій у його роботі. Однак незважаючи на це існує великий потенціал для подальшого розвитку продукту, оскільки є багато можливостей для вдосконалення, а сама технологія Blazor дозволяє такі розширення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Життєві цикли розробки ПЗ. *Онлайн-курси від компанії QATestLab | Головна сторінка*. URL: https://training.qatestlab.com/blog/technical-articles/popular-software-development-life-cycles/ (дата звернення: 10.06.2024).

2. Інкрементна модель у SDLC: використання, переваги та недоліки. *Guru99*. URL: https://www.guru99.com/uk/what-is-incremental-model-in-sdlc-advantages-disadvantages.html#:~:text=Що%20таке%20інкрементальна%20модель?,%20тестування/перевірки,%20обслуговування. (дата звернення: 10.06.2024).

3. Онлайн-курс: розвиток креативного мислення: від теорії до практики. *Освітній проект «На Урок» для вчителів*. URL: https://naurok.com.ua/learn/rozvitok-kreativnogo-mislennya-vid-teori-do-praktiki-45 (дата звернення: 10.06.2024).

4. Основные преимущества языка программирования C# - Бізнес новини Звягеля. *04141.com.ua - Сайт міста Звягель*. URL: https://www.04141.com.ua/list/326508 (дата звернення: 10.06.2024).

5. Що таке C#? Кому підходить програмування на сі шарп? Beetroot academy. *https://beetroot.academy/blog/shcho-take-c-chi-pidhodit-meni-cya-mova-programuvannya-chomu-vona-kruta*. URL: https://beetroot.academy/blog/shcho-take-c-chi-pidhodit-meni-cya-mova-programuvannya-chomu-vona-kruta (дата звернення: 10.06.2024).

6. поняття та класифікація проектів - бібліотека buklib.net. *Головна - Бібліотека BukLib.net*. URL: https://buklib.net/books/22265/ (дата звернення: 10.06.2024).

7. 2.2. види проектів та їхня класифікація - бібліотека buklib.net. *Головна - Бібліотека BukLib.net*. URL: https://buklib.net/books/34064/ (дата звернення: 10.06.2024).

8. 2.2. Основні ознаки проекту. *StudFiles*. URL: https://studfile.net/preview/4495008/page:4/ (дата звернення: 10.06.2024).

9. 6 основних етапів SDLC. *QualityAssuranceGroup*. URL: https://qagroup.com.ua/publications/6-osnovnykh-etapiv-sdlc/ (дата звернення: 10.06.2024).

10. 9 причин изучить язык C# - FoxmindEd. *FoxmindEd*. URL: https://foxminded.ua/ru/9-prichin-vivchiti-movu-c/ (дата звернення: 10.06.2024).

11. ASP.NET core blazor. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-8.0 (date of access: 10.06.2024).

12. Blazor university - introduction. *Blazor University*. URL: https://blazor-university.com/ (date of access: 10.06.2024).

13. Displaying the task path in microsoft project. *Ten Six Consulting*. URL: https://tensix.com/displaying-the-task-path-in-microsoft-project/ (date of access: 10.06.2024).

14. Google авторизація з identity. *Dijix - Блог*. URL: https://dijix.com.ua/blog/uk/google-avtorizacziya-s-identity/#googlecreate (дата звернення: 10.06.2024).

15. Guide to software project management | smartsheet. *Smartsheet*. URL: https://www.smartsheet.com/content/software-project-management (date of access: 10.06.2024).

16. ProgIngContrSystems. *ProgIngContrSystems*. URL: https://pupenasan.github.io/ProgIngContrSystems/Лекц/17_lyfecycle.html (date of access: 10.06.2024).

17. Tkachuk I. Переваги та недоліки .NET: швидкий розвиток, велика поширеність і середні зарплати. URL: https://dou.ua/lenta/articles/pros-and-cons-of-dotnet/?hl=ua (дата звернення: 11.06.2024).

18. What is SDLC? - software development lifecycle explained - AWS. *Amazon Web Services, Inc.* URL: https://aws.amazon.com/what-is/sdlc/#:~:text=The%20software%20development%20lifecycle%20(SDLC,expectations%20during%20production%20and%20beyond. (date of access: 10.06.2024).

19. What is SDLC? - software development lifecycle explained - AWS. *Amazon Web Services, Inc.* URL: https://aws.amazon.com/what-is/sdlc/#:~:text=The%20software%20development%20lifecycle%20(SDLC,expectations%20during%20production%20and%20beyond. (date of access: 10.06.2024).

20. What's behind the hype about Blazor? - Stack Overflow. *The Stack Overflow Blog - Stack Overflow*. URL: https://stackoverflow.blog/2020/02/26/whats-behind-the-hype-about-blazor/ (date of access: 10.06.2024).

21. Що таке регресійне тестування? Визначення, переваги та найкращі інструменти - рішення Visure. Visure Solutions. URL: https://visuresolutions.com/uk/blog/what-is-regression-testing-definition-and-top-tools/ (дата звернення: 10.06.2024).

22. 10 найкращих програм для управління проектами. doola: Start your dream US business and keep it 100% compliant. URL: https://www.doola.com/uk/blog/best-software-for-project-management/ (дата звернення: 10.06.2024).

23. ASP.NET core blazor. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/uk-ua/aspnet/core/blazor/?view=aspnetcore-8.0&WT.mc_id=dotnet-35129-website (date of access: 10.06.2024).

24. Clean coder blog. Clean Coder Blog. URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html (date of access: 10.06.2024).

25. Contributors to Wikimedia projects. Active directory - wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Active_Directory (date of access: 10.06.2024).

26. Patrick God. Build a chat with signalr & blazor webassembly in .NET 8 □, 2024. YouTube. URL: https://www.youtube.com/watch?v=ikfygSrR8H0 (date of access: 10.06.2024).

27. ProgIngContrSystems. ProgIngContrSystems. URL: https://pupenasan.github.io/ProgIngContrSystems/Лекц/17_lyfecycle.html (date of access: 10.06.2024).

ДОДАТКИ

Додаток А

Технічне завдання

1. Вступ

Технічне завдання описує сервіс для планування і розподілу завдань, менеджменту за проектами SMART. Цей проект це вебсайт, що дозволяє зручно розподіляти завдання між учасниками проекту та вести моніторинг за станом роботи.

2. Підстави для розробки

Розробка проводиться на основі індивідуального завдання, поставленого керівником бакалаврської роботи для спеціальності 122 “Комп’ютерні науки”

3. Призначення розробки

Функціональне призначення: програмний продукт Smart призначений щоб надати інструменти для моніторингу та розподілу завдань, командного спілкування та налагодження ефективності роботи в максимально зрозумілій, гнучкій та регульованій формі.

Експлуатаційне призначення: призначений для використання у групових та індивідуальних проектах задля підвищення ефективності управління та налагодження командної співпраці.

4. Вимоги до програми чи програмного продукту

4.1. Вимоги до функціональних характеристик

1. Система повинна надавати реєстрацію та автентифікацію користувачів, дозволяти новим користувачам створювати облікові записи і забезпечувати зареєстрованим користувачам вхід в систему за допомогою своїх облікових даних.

2. Повинна існувати система для забезпечення створення нових проектів, зміну та видалення існуючих. Користувачі повинні мати можливість визначати

параметри проекту, встановлювати терміни, призначати завдання та відстежувати прогрес.

3. Власник проекту повинен мати права на керування командою проекту, включаючи додавання нових учасників та видалення їх з проекту за потреби.

4. Повинен міститися функціонал для створення та розсилки запрошень для участі в проектах. Це може включати можливість створити спеціальне посилання для запрошення для нових учасників приєднатися до проекту.

5. Учасники повинні мати можливість залишати коментарі під завданнями. Система повинна дозволяти спілкування в чаті. Це забезпечує ефективну комунікацію між проектом і зацікавленими сторонами, забезпечуючи обговорення, переговори та вирішення проблем у реальному часі.

6. Користувач повинен мати можливість видаляти свій обліковий запис.

4.2. Вимоги до надійності

1. Система повинна бути спроектована та налаштована для ефективної роботи під високим навантаженням, здатна обробляти одночасні запити багатьох користувачів без значного зниження продуктивності.

2. Сервіс має відповідати на запити користувачів швидко та уникати системних збоїв та перебоїв у роботі.

3. Сайт повинен бути стабільним та надійним, забезпечуючи безперебійний доступ до функціоналу для користувачів.

4. Сайт повинен пройти комплексне тестування перед випуском, щоб виявити та виправити помилки і вразливості.

5. Система повинна бути постійно моніторена для виявлення проблем та своєчасного реагування на них.

6. Система повинна мати ефективні механізми захисту, такі як аутентифікація та авторизація, для запобігання несанкціонованому доступу до даних та функцій системи.

4.3. Умови експлуатації

1. Веб застосунок має бути доступним для користувачів 24/7, за винятком проведення планових технічних робіт.

2. Повинні бути надані інструкції з використання та експлуатації системи.

4.4. Вимоги до складу і параметрів технічних засобів

Для запуску клінської частини підійде будь-який комп’ютер з встановленим браузером з підтримкою сучасних веб стандартів та доступом до інтернету.

4.5. Вимоги до інформаційної і програмної сумісності

1. Серсів повинне бути сумісним з різними веб переглядачами і працювати на різних операційних системах.

2. Продукт повинен працювати в режимі онлайн і бути доступним для користувача на пристрої із стабільним інтернет підключенням.

4.6. Вимоги до транспортування і збереження

Всі дані сайту повинні зберігатися у надійному сервері з відповідними стандартами безпеки.

5. Вимоги до програмної документації

Документація повинна містити:

- опис всіх функцій;
- інструкцію для користувачів;
- опис алгоритмів роботи, а також опис архітектури та дизайну сервісу.

6. Техніко-економічні показники

Техніко-економічні показники мають включати оцінку витрат на розробку, впровадження та на підтримку сервісу SMART. Також надавати

оцінку економічної ефективності використання сервісу та очікуваних результатів і користі для організації.

7. Стадії і етапи розробки

7.1. Передбачені стадії розробки

1. Визначення всіх головних та аналіз можливих функціональних та не функціональних вимог до системи.
2. Проектування та розробка архітектурної частини проекту та дизайну інтерфейсу для користувача.
3. Розробка та імплементація функціональних модулів:
   - розробка і налаштування бази даних для зберігання і отримання інформації про наявних користувачів, їх проекти, завдання, коментарі;
   - інтеграція з системами авторизації;
   - створення і опис необхідних сторінок та елементів на сайті;
4. Тестування, від лагодження та оптимізація сервісу.
5. Впровадження та налаштування системи.
6. Подальша підтримка для коректної експлуатації системи.
7. Проведення регулярних заходів з оновлення і розширення, щоб врахувати нові вимоги користувачів або вдосконалювати наявний функціонал.
8. Проводити моніторинг продуктивності системи, здійснювати процеси оптимізацію для забезпечення її найкращої роботи.

7.2. Необхідні сторінки та структурні елементи на сайт

Назва елементу	Опис

MainLayaut	Головний шаблон сайту, на якому мають бути розміщені - шапка, з логотипом та назвою сервісу, а також активна навігаційна панель, де розміщений перелік існуючих проектів користувача та кнопка для створення нового; - головне тіло сайту, де буде розміщуватися індивідуальний контент для кожної з сторінок сервісу; - панель заголовок, де розміщена інформація про користувача і кнопки авторизації.
Home	Домашня сторінка міститиме: - тематичні заголовки інформацію та зображення; - буде розміщено карусель з текстом описом сайту; - кнопка, для початку роботи програми.
Privacy	Сторінка, з описом політики конфіденційності.
NotFound	Шаблон, який показується у випадках, коли немає існуючої сторінки
Register	Сторінка реєстрації, серед полів для реєстрації повинні бути Firstname, Lastname, Email, Password, Confirm password, Birthdate також кнопка підтвердження реєстрації – Submit. Всі поля повинні мати валідацію вхідних значень з перевіркою як на сервері так і на клієнті.
Login	Опис сторінки входу в особистий акаунт сайту. Щоб увійти, користувач має коректно ввести інформацію email та password і натиснути кнопку підтвердження.

My Account	Головний профіль користувача з інформацією про поточний акаунт: First Name, Last Name, Email, Birth Date, а також кнопки зміни та видалення даних.
Account Edit	Вікно, де користувач може міняти деякі дані свого профілю, такі як First Name, Last Name та зберегти зміни.
Project Create	Компонент, що слугує для створення нового проекту. У вікні розміщене поле Name необхідне для створення проекту. Після заповнення потрібно, натиснувши на кнопку Create нас перекине на сторінку проекту.
Project Edit	Вікно, що дозволяє редагувати назву проекту чи видалити його та всю інформацію, що розміщена в ньому.
Project	Основна сторінка конкретного проекту, що доступна тільки для його учасників, там відображена вся інформація про проект розміщена кнопка створення завдання. Також тут відображається усі існуючі завдання з відмітками. Для власника проекту доступні наступні можливості: - копіювання запрошувального посилання для нових учасників. - швидкий доступ до сторінки редагування проекту за допомогою відповідної іконки.
Task Create	Вікно у якому учасник проекту може створювати нове завдання чи під завдання призначати терміни, виставляти пріоритет та часові рамки.

Task Edit	Вікно з можливістю редагувати стан роботи, його процент виконання та за необхідності редагувати дані чи видаляти завдання.
Invite	Сторінка запрошення до команди, на якій буде вказано основну інформацію щодо проекту, таку як назва, учасники, власник буде можливість проігнорувати запрошення, нажавши на Ignore та приєднатися до проекту, нажавши кнопку Apply.
Chat	Вікно яке буде відкриватися при натисканні іконки чату у відповідному проекті. Учасники будуть мати можливість надсилати повідомлення та обмінюватися інформацією.

8. Порядок контролю і приймання

Після закінчення кожного етапу роботи необхідно проводити тестувати сервісу наступним чином:

- тестування на коректність роботи з БД, а саме відправка і отримання даних, коректне встановлення і оновлення інформації у таблицях;
- провести перевірку системи на наявність потенційно можливих вразливостей, забезпечення захисту даних користувачів, перевірку коректності аутентифікації та авторизації, а також виявлення інших можливих атак чи порушень безпеки;
- зробити перевірку компонентів системи на коректну взаємодію, інтегрування з базою даних та додатковими сервісами;
- виконати функціональне тестування для перевірки працездатності всі наявних функції програми згідно з зазначеними вимогами. Впевнитися, що програма працює з усіма даними коректно;
- провести вимірювання часу відгуку системи на різні можливі запити, спробувати виявити можливі буттєві точки та провести оптимізацію

продуктивності сервісу;

- зробити перевірку роботи сервісу на різних доступних операційних системах, веб браузерах та моделях пристроїв;

- підготувати документацію, яка буде описувати результати тестування, знайдені помилки, збої в роботі та надаватиме опис засобів їх виправлення;

- здійснювати моніторинг у роботі системи та складати відповідні звіти щодо її продуктивності та стабільності та доступності для роботи.

9. Додатки до технічного завдання

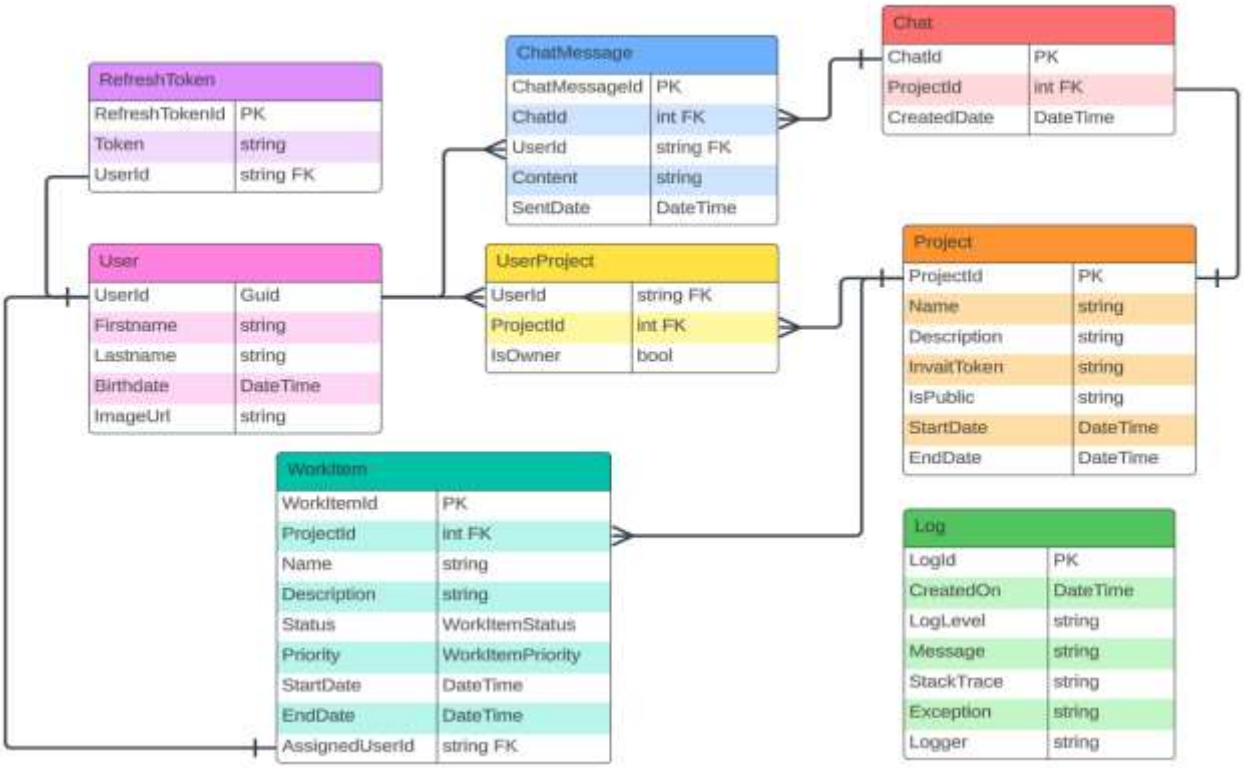

Рис.1. Схема структури реляційної бази даних

Додаток Б

Інструкція користувача

Для коректної роботи застосунку SMART та початку користування інструментами, що забезпечують управляння проектами, потрібно першочергово виконати вхід у акант чи за необхідності створити його.

Після виконання цього, вам буде надано доступ до можливостей платформи. На панелі ви зможете перейти на відповідну сторінку з проектами доступними вам. Також буде надана можливість створити новий проект натиснувши на значок додати. Після створення чи приєднання ви зможете перейти на сторінку проекту та розпочати роботу, додавши нові завдання, які будуть відображатися у відповідних станах виконання. Перейшовши на деталі завдання, вам буде надано детальну інформацію, щодо нього та ви зможете змінити стан його виконання.

Учасники можуть покинути проект за бажанням, також вам доступний чат у межах проекту, перейти до нього можна натиснувши відповідний значок. Після виконання задач проект можна закрити, видаливши всю інформацію, щодо нього.

АНОТАЦІЯ

Галушка Ю. А. – **Веб орієнтована платформа менеджменту проектів розробки програмних продуктів**.

Кваліфікаційна робота за спеціальністю 122 Комп'ютерні науки. – Волинський національний університет імені Лесі Українки, Луцьк. – 2024р.

Робота присвячена розробці веб застосунку Smart, який надаватиме інструменти для ефективного управління проектами. Основна мета – створити платформу, що полегшує організацією роботи, незалежно від місця розташування команди. Застосунок розроблено на основі технології Blazor та мови програмування C#. Платформа Smart містить основні функції, такі як реєстрація користувачів, авторизація та аутентифікація, створення та розподіл задач, відстеження статусу та проценту завершення завдань. Особлива увага приділяється інтерфейсу і простоті використання. Робота охоплює всі етапи розробки застосунку від проектування до програмної реалізації та тестування.

Ключові слова: веб застосунок, керування сервісом, Blazor, C#, Asp.Net Framwork, методи менеджменту, патерни програмування, Clean Architecture, клієнт серевер, JWT токени.