МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ Аліна САВЧЕНКО

"____"_____2021 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИЦІ ОСВІТНЬОГО СТУПЕНЯ
"МАГІСТР"

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

Тема: "Web-сервіс для планування та управління проектами"

Виконавиця: Ковальчук Анна Юріївна

Керівник: к.т.н., доцент Колісник Олена Василівна

Нормоконтролер: _____Ігор РАЙЧЕВ

Київ – 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології" .

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Аліна САВЧЕНКО
«____»_____2021р.

ЗАВДАННЯ

на виконання дипломної роботи студентки

Ковальчук Анни Юріївни
(прізвище, ім'я, по батькові)

1. **Тема роботи:** «Web-сервіс для планування та управління проєктами» затверджена наказом ректора від 12.10.2021 р. за № 2228/ст.
2. **Термін виконання роботи:** з 12.10.2021 р. по 31.12.2021 р.
3. **Вихідні дані до роботи:** функціональні вимоги до веб-сервісу, база даних Firebase, фреймфорки: AngularJS, ExpressJS, програмне середовище Visual Studio Code.
4. **Зміст пояснювальної записки:** аналіз існуючих веб-сервісів управління проєктами, аналіз та порівняння обраних інструментів і технологій для веб-сервісу, реалізація веб-сервісу для планування та управління проєктами.
5. **Перелік обов'язкового ілюстративного матеріалу:** Слайди презентації MS Powerpoint: популярні веб-сервіси, структура програмного продукту, скріншоти користувацького інтерфейсу.

6. Календарний план-графік

№ з/п	*Завдання*	*Термін виконання*	*Підпис керівника*
1.	Дослідження та аналіз предметної області використання	12.10.2021– 15.10.2021	
2	Опрацювання інформації за тематикою дипломного проекту	18.10.2021– 20.10.2021	
3	Розробка стилізації web-сервісу	20.10.2021– 22.10.2021	
4	Розробка програмної частини сервісу	25.10.2021– 03.11.2021	
5	Розробка та впровадженння системи	04.11.2021 – 17.11.2020	
6	Написання пояснювальної записки дипломного проекту	17.11.2021– 30.11.2020	
7	Оформлення та друк пояснювальної записки.	02.12.2021 – 11.12.2021	
8	Підготовка демонстраційного матеріалу та доповіді	12.12.2021 – 20.12.2021	

7. Дата видачі завдання: 12.10.21 р.

Керівник дипломної роботи _____Олена КОЛІСНИК
(підпис керівника)

Завдання прийняла до виконання _____Анна КОВАЛЬЧУК
(підпис випускниці)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи “Web-сервіс для планування та управління проєктами” складається із вступу, трьох розділів, загальних висновків, списку використаних джерел і містить 84 сторінок тексту, 39 рисунків та 6 таблиць. Список використаних джерел містить 14 найменувань.

Метою дипломної роботи є аналіз теоретичних та практичних матеріалів для створення веб-сервісу для управління та планування проєктами, який в подальшому буде використовуватися для застосування користувачами відповідно до їх потреб.

Предметом дослідження є розроблення інформаційної системи, яка має набір модулів, що проходять попереднє порівняння та оцінення для визначення альтернативних засобів для використання для системи.

Об’єктом дослідження є структура веб-сервісу управління проєктами, який включає в себе множину обробників подій, компоненти дизайну та структура візуалізації.

Ключові слова: ВЕБ-СЕРВІС, ПРОЄКТ, СИСТЕМА УПРАВЛІННЯ, БАЗА ДАНИХ, КОРИСТУВАЧ, АДМІНІСТРАТОР.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – база даних

ПК – персональний комп’ютер

ПрО – предметна область

OCL - Object Constraint Language

UML - Unified Modelling Language

PDDL - Planning Domain Definition Language

WSDL - Web Services Description Language

URI - Uniform Resource Identifier

BPEL - Business Process Execution Language

HTTP - HyperText Transfer Protocol

XML - Extensible Markup Language

SOAP - Simple Object Access Protocol

UDDI - Universal Description Discovery & Integration

DAML-S - DARPA Agent Markup Language for Web services

OWL - Ontology Web Language

XML-RPC - Extensible Markup Language Remote Procedure Call

REST - Representational State Transfer

RMI - Remote Method Invocation

CORBA - Common Object Request Broker Architecture

DCOM - Distributed Component Object Model

BPEL4WS - Web Services Business Process Execution Language

CRM - Customer relationship management

OTRS - Open source Ticket Request System

HTML - HyperText Markup Language

CSS - Cascading Style Sheets

API - Application Programming Interface

MIT - Massachusetts Institute of Technology

SDK - Software Development Kit

SQL - Structured Query Language

SGML - Standard Generalized Markup Language

ВСТУП

Дана дипломна робота присвячена вирішенню актуального практичного завдання, що полягає у розробці веб-сервісу для організації налагодженого життєвого циклу над проектами.

Актуальність. Актуальність дипломної роботи полягає в тому, що використання веб-сервісів для управління проектами значно спрощує роботу працівників, допомагаючи ефективно скоординувати процеси та виділити пріоритети задач всередині колективу. Всі процеси контролюються за допомогою веб-сервісу гарантуючи, що вся інформація буде збережена спочатку і до кінця. Простий і зрозумілий інтерфейс спрощує взаємодію та внаслідок чого робота виконується ефективніше. Кожен користувач зможе налаштувати функції веб-сервісу відповідно до своїх потреб скористувавшись інструментами даного сервісу. Функціональною перевагою є те що, всі члени команди відповідальні за розробку та управління проекту будуть отримувати відповідне повідомлення до задач з якими вони працюють, якщо ці задачі змінилися. Користувачі зможуть створювати обговорення та задавати питання.

На сьогоднішній день управління проектами неможливе без використання сучасних програмних засобів, адже зростають розміри проектів, частота використання, обсяг оброблюваної інформації. Серед систем управління проектами є як безкоштовні так і платні варіанти. Більшим попитом користуються безкоштовні системи, які не поступаються у функціональних параметрах платним відповідникам. Для стартапів та компаній, які не націлені на складні та масштабні проекти можна використовувати безкоштовні системи управління проектами. В інших випадках краще обрати системи з розширеним функціоналом.

По набору функцій системи управління проєктами можна розділити на два види: з простим функціоналом та зі спеціальним утилітами та модулями. Перший вид систем, це програми, які розподіляють завдання на працівників та команди загалом, складати звітності та графіки, планувати затрати, формувати звітності для визначення

швидкості виконання поставлених задач. Ці системи розраховані на малогабаритні проєкти, адже такі системи можуть не витримати великого навантаження та наслідком цього можуть бути тривалі затримки в роботі веб-сервісу чи навіть збій системи. Другий вид систем, це професіональні програми, які складаються з додаткових утиліт та модулів, призначених для вирішення різних специфічних задач. До таких задач можна віднести аналізування та конструювання ризиків, відокремлення працівників, які мають вищий рейтинг по успішності виконання задач в проєктах, визначення термінів виконання проєктів відповідно до фінансових затрат.

В кожного програмного засобу є свої переваги та недоліки. Головним недоліком веб-сервісів для управління проектами є складність налаштування та висока вартість. Наслідком чого є витрата часу та сил для освоєння нового продукту, а можливо і не розібравшись в інструкції по експлуатації самостійно залучення спеціалістів до налаштування системи, що потребує додаткових витрат.

Для створення веб-сервісу планування та управління проектами необхідно вирішити наступні задачі:

- розробити ядро веб-сервісу;
- створити обробники подій для управління веб-сервісу;
- налаштування серверної частини розробки та маршрутизації;
- створення шаблону для оформлення панелі задач.

Метою дослідження дипломної роботи магістра є створення веб-сервісу для управління та планування проєктами.

Об’єктом дослідження є створення нових проєктів та управління ними.

Предметом дослідження є створення web-сервісу.

Веб-сервіс сприятиме вирішення таких важливих проблем як отримання продуктивних результатів за мінімальний проміжок часу, створення налагодженої роботи в колективі, мінімізація форс-мажорних ситуацій(наприклад: працівник

захворів, пішов у відпустку, а терміни виконання підходять до кінця), збої у роботі бази даних, як наслідок втрата всіх даних.

Для вирішення даних проблем буде застосовано таких функціонал як контроль виконання поставлених задач для панелі адміністратора, пропорційне розподілення задач на працівників, а також можливість працювати декільком працівникам над однією задачею для швидкого виконання, можливість взаємозаміняти працівників у разі виникнення проблем, створення хмарного середовища збереження даних.

Веб-сервіси для управління проєктами охоплюють великий спектр сфер діяльності в яких вони можуть бути застосовані, серед них можна виділити такі як, органи виконавчої влади, бізнес, підприємницька діяльність, органи громадського порядку та управління, інформаційні технології.

Веб-сервіси можуть мати такі застосування:

- у сфері місцевого самоврядування: веб-сервіси для розвитку громадян та інших інструментів (місто в смартфоні), аналітично-комунікаційна система управління звернень громадян, інформаційно-аналітичні системи для звітності по діяльності керівництва органів влади;
- сфера бізнесу: ведення обліку витрат для виявлення сторонніх відрахувань, сервіс реєстру підписаних договорів із керуючими компаніями міста, залучення іноземних інвестицій;
- сфера медичних послуг: сервіс контролю дотримання карантинних обмедень, веб-сервіси медичної допомоги учасникам ООС та переселенцям;
- сфера сільського господарства: інформаційно-аналітична система для допомоги фермерам та аграріям, веб-сервіс обліку висаджених дерев, системи обліку територій та природничих зон;
- сфера комунальних послуг: сервіси автоматизації витрат енергоресурсів, газу, води, тепла, сервіси електронних диспечерських для виклику при аварійних ситуаціях;

- сфера інформаційних технологій: веб-сервіси для розгортання проєктів (створення веб-сайтів, облікових систем, мобільних додатків);
- сфера фінансів: сервіси для ведення бухгалтерської звітності по підприємству/особі-підприємцю, сервіси для електронної сплати податків, державних внесків .

Веб-сервіс складатиметься як із панелі адміністратора так і з користувацької панелі. З боку адміністратора будуть виконуватися дії для налаштування процесів роботи з проєктом, розподілення основних задач, вирішення користувацьких потреб. З боку користувача, буду доступне переглядання панелі назначених задач, перегляд проєктів в яких залучений користувач, налаштування користувацького інтерфейсу.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ ВЕБ-СЕРВІСІВ УПРАВЛІННЯ ПРОЕКТАМИ

1.1. Основні поняття

Веб-сервіс, веб-служба (web service) - інформаційна система, що викладають в Інтернет-мережу, яка може містити набір простого функціоналу чи розширені властивості користувацького функціоналу та може бути інтегрована з різноманітними додатками і мережевими протоколами (SOAP, XML-RPC, REST тощо). Він має інтерфейс, що описаний в машинно-обчислювальному форматі (WSDL). SOAP-повідомлення використовують у випадках, коли системи, що взаємодіють з веб-сервісом з визначеним описом та передають веб стандарти завдяки протоколу HTTP з XML форматом.

Серед найбільш поширених веб-сервісів виділяють:

- пошукові системи;
- веб-хостинги;
- веб-пошта;
- веб-архіви.

Основною властивістю веб-вервісу є те, що незалежно від провайдера, комп'ютера чи браузера є можливість працювати з даними в будь-якій точці світу, де є доступ до інтернету.

1.1.1. Архітектура та протоколи веб-сервісів

Задача реалізації веб-сервісів досить проста при використанні сучасного програмного забезпечення, яке дозволяє зосередитися на розробці функціональності самого веб-сервісу, а не турбуватися щодо реалізації мережевих протоколів.

Сьогодні для реалізації веб-сервісів використовують наступні стандарти:

- XML. Це мова розмітки, її використовують для запису і відправки структурованої інформації.
- SOAP. Даний протокол дозволяє відправляти і отримувати повідомлення, використовуючи XML в якості основи.
- WSDL. Це мова, яка служить для опису сторонніх інтерфейсів на базі XML.
- UDDI. Це інтерфейс відноситься до універсальних інструментів і служить для опису та інтеграції.

Дані технології можуть використовуватися в будь-яких операційних системах, мовах програмування, серверах додатків та інше.

Переваги використання стандартів:

- незалежно від платформи створення відповідних умов для взаємодії з програмними елементами;
- використовуючи XML веб-сервіси забезпечуються простотою формування та налаштування;
- завдяки використанню протоколу HTTP, веб-сервіси можуть взаємодіяти за допомогою мережевого доступу.

Недоліками стандартів є:

- завдяки використанню XML-повідомлень знижується продуктивність і збільшується об'єм трафіку на відміну від систем RMI, CORBA, DCOM;

- безпека сервісів - забезпечення безпеки має проводитися за допомогою кодування та обов'язкової авторизації користувача в системі, для вирішення цієї проблеми потрібно застосовувати більш надійні протоколи, такі як HTTPS, XML Encryption тощо.

1.1.2. Композиція веб-сервісу

Досягнення функціональних показників несе в собі об’єднання та координацію веб-сервісів, що є основою для композиції веб-сервісу. Ці показники не можуть бути застосовані завдяки існуючим веб-сервісам.

Основними підходами до композиції веб-сервісів є статичний і динамічний. Статичний підхід являє собою ручне налаштування сервісу, яке виконується на етапі проектування, коли планується архітектура і проект системи. Даний підхід добре працює до того моменту поки компоненти сервісу не змінюють. Прикладом статичного підходу є Biztalk від Microsoft, WebLogic від Bea.

Динамічний підхід змінює програму під час її виконання як шляхом прямого людського втручання засобами адаптації, так і автоматично, при якому зміна програми може бути виконана системою безпосередньо. Прикладом динамічного підходу є Web Services Composition Platform фірми Sun, засоби e-flow фірми HP та інші.

Виділяють п’ять категорій композицій:

- орієнтовані на модель;
- орієнтовані на бізнес-правила;
- декларативна;
- автоматизована;
- композиції семантичних веб-сервісів.

Композиції сервісів, орієнтовані на моделях управління

Створення даного сервісу ділиться на 4 етапи: визначення сервісу, планування, конструювання, виконання. Абстрактна модель є визначенням сервісу, яка аналізує компоненти сервісу та взаємозв'язки між ними. Цілью даної моделі є вибір елементів сервісу та правил композиції.

Композиція сервісів, яка заснована на бізнес-правилах

Композиція визначається шляхом додавання бізнес-правил, що регламентують: перед- та після- умови; опис подій, що мали місце при виконанні процесів композиції сервісу; потоку процесів; визначення та блокування операцій; визначення повідомлень, що містять вхідну та вихідну інформацію; опис ролей у процесі композиції сервісів.

Правила композиції сервісів можливо моделювати з використанням мови OCL. Бізнес-правила можна класифікувати таким чином:

- структурні правила для формування процесів та структури;
- правила для планування пріоритету процесів у композиції;
- правила управління даними та повідомленнями;
- правила поведінки для управління процесами та забезпечення цілісності композитного процесу;
- правила використання ресурсів, вибору сервісів, провайдерів;
- правила для визначення виключень у поведінці сервісів;
- правила ролей для управління учасниками, залученими у сервісі;
- правила повідомлення для регулювання використовування інформації;
- правила для управління поведінкою композиції сервісу відповідно щодо очікуваних або непередбачених подій та інші.

Для опису абстракції високого рівня використовують мову UML, яка забезпечує опис бізнес-процесів, засоби підключення до інших стандартів, таких як BPEL4WS. Для формування бізнес-правил та опису потоку процесів

використовується мова OCL, за допомогою якої описуються умови вибору сервісів, структури веб-сервісів та розробка плану розгортання сервісу.

Декларативна композиція сервісів

Більшість підходів засновані на тому, що спочатку мають бути створені бізнес-процеси. Декларативний підхід кардинально відрізняється від підходів, що засновані на бізнес-правилах та розділяється на дві фази: перша фаза полягає у визначенні цілей композиції та конструювання планів для кожної цілі. Наступна фаза полягає у тому, що для кожного плану здійснюється пошук відповідних сервісів та будується схема потоку робіт. Перша фаза реалізується на основі PDDL та засобів XML Web services Request Language, які мають забезпечити машино-прочитувані об’єкти та специфічність поведінки сервісу. Наступна фаза може бути реалізована на основі існуючих мовних засобів моделювання таких як BPEL. В декларативному підході архітектура має бути покращена, а саме опис копцепції сервісів та реєстр. Наприклад, засоби WSDL не забезпечують опис необхідних атрибутів сервісу, а засоби UDDI не містять інформацію про те, що робить сервіс. Декларативний підхід виконується завдяки мові композитивного опису веб-сервісу. Ця мова показує записи користувачів, нові типи джерел та функцій на основі унікальних імен URI. Для розгортання даного підходу потрібно мати модель координованих сервісів та універсальні протоколи.

Автоматична композиція сервісів

На відміну від ручної композиції сервісів, автоматична композиція потребує використання онтологій. Відповідно щодо сервісів, онтологія визначається, як множина об’єктів, які розділяють ту ж саму предметну область, та правила, за якими сервіси можуть бути описані та доступні. Для забезпечення загальної структури та опису значень для властивостей складних класів використують такі онтологічні мови як RDF, DAML-S, DAML+OIL, або OWL тощо. Засоби DAML-S забезпечують механізм для онтологічної організації сервіса. В якості типів онтологій, необхідних для композиції сервісів є такі: метрик, одиниць виміру, властивостей та інші.

Композиції семантичних веб-сервісів

Завдяки назвам та загальним властивостям, що містять вхідні та вихідні повідомлення можна визначити операції сервісів. Для цього вводять синтаксичні та семантичні правила. Синтаксичні правила визначають спосіб композиції сервісу. Вважають, що два сервіси є придатними до композиції, коли, з одного боку, кожен клієнт або сервер на яких розміщені сервіси, мають бути зв'язані каналами зв'язку, а з другого боку сервіси мають бути погоджені з відповідним протоколом транспортування мережевих повідомлень. Семантичні правила виконують такі цілі як структурність повідомлень, семантичність структури, надійність.

Повідомлення, що приймаються від клієнта умовно поділяються на різні типи, що визначаються при запуску сервісу та мають бути сумісними. Сумісніть повідомлення - коли вони сумісні по типам даних. Операційна композиційність має місце тоді, коли предметні області інтересів сумісні або синонімічні і, якщо операції забезпечують ті ж самі характеристики. Якісна композиційність має місце коли сервіси мають ті ж самі якісні характеристики, наприклад, безпеки або прав доступу до ресурсів. Для композиції сервісів вводять поняття схеми композиції, яка забезпечує композиційну надійність. Композиційна схема зв'язана з композиційним сервісом та використовується для порівняння додаткових значень. Наприклад, існує два відокремлені сервіси, такі як замовлення для видачі книжок та замовлення авіаквитків, коли автори роблять замовлення квитків для перельоту в те місце де замовили книжки. Здійснюється вибір найбільш відповідних композиційних планів, визначаються обмеження. Далі генерується результуючий композиційний сервіс.

1.1.3. Вимоги до композиції веб-сервісів

Щоб процес композиції був повноцінним і точним він має підпорядковуватися певним вимогам, які вирішують проблеми, що можуть виникнути в процесі експлуатації.

Автоматизація

Завдяки автоматизації можна задовольнити вимоги користувачів, для цього потрібно зменшити взаємодію користувача з сервісом та пришвидшити процес композитивного сервісу. Вона зменшує затрати часу, в порівнянні з часом, при створенні схеми композицій людиною, відкидається людський фактор та мінімізується загальна вартість процесу. Тому автоматизація реалізує успішний підхід до веб-сервісів.

Динамічність

Динамічність надає композиції веб-сервісів такі властивості як, сумісність та існування сервісу з першого релізу. Вирішуються такі питання як, зниження попиту на користування веб-сервісом, а згодом і втрата технічної підтримки, зміна старих сервісів на нові.

Семантичність

Семантично багаті описи сервісів і цілей композиції можуть бути використані додатками або іншими сервісами без участі людини. Таким чином, ефективні підходи композиції сервісу повинні використовувати семантичні описи, щоб реалізувати свої цілі.

QoS-поінформованість

До QoS-підходів належать функціональні та не функціональні характеристики до яких відносять якість, час, ціну тощо. Важливим фактором QoS є задоволення функціональними вимогами більше ніж одного вервісу.

Масштабованість

Кожен композиційний підхід може показати себе з різних сторін з тим чи іншим набором сервісів, взаємодіяти успішно чи мати проблеми в реалізації. Тому для визначення продуктивності роботи сервіси перевіряються на великих реєстрах сервісів.

Коректність

Коректність композиції потрібна, коли ми хочемо перевірити правильність введення даних, наявність синтаксичних та орфографічних помилок. Коректність встановлюється через методи верифікації.

Адаптивність

Адаптивність дає веб-сервісу можливість бути стійким до змін. Адаптивність може бути активною коли зміни застосувалися до того як станеться збій під час робочого сеансу і до того, як таке відхилення може привести до проблем, або реактивною обробка помилок і відновлення після порушень, повідомлених під час виконання.

1.2. Огляд ринку веб-сервісів управління проектами

Існує багато веб-сервісів для управління проектами, в залежності від поставлених цілей та задач кожен користувач може обрати для себе відповідний сервіс.

Серед існуючих сервісів виділяють:

- *Стаціонарне ПЗ.* Такі програми зазвичай дозволяють зберігати інформацію в файл, який надалі може бути викладений в загальний доступ для інших користувачів або ж дані зберігаються в центральній базі даних. Як приклад, можна навести такі програми: Cerebro, GanttProject, KPlato, Microsoft Project, OpenProj, Open Workbench;
- *Веб-додатки* : Trello, Basecamp, Pivotal Tracker, Bontq;
- *Для одного користувача.* Системи для одного користувача можуть використовуватися як персональні або для керування невеликими компаніями;

- *Для великої кількості користувачів.* Призначені для координації дій декількох десятків або сотень користувачів. Зазвичай будуються за технологією клієнт-сервер та мають веб-інтерфейс. До систем, що розраховані на велику кількість користувачів, відносяться: Jira, Bitbucket, Launchpad, OTRS, TeamLab та інші.

Розглянемо детальніше деякі з них.

Basecamp – це одна з найпопулярніших CRM (система управління взаємовідносинами з клієнтами). Вона дозволяє спілкуватися з командою, ділитися документами, створювати to-do списки, призначати відповідальних, додавати коментарі та багато іншого. Цей сервіс існує вже 14 років, за цей період засновники додали чимало корисних інструментів, створили додатки для розширення можливостей. А ще, під час розробки Basecamp, народився фреймворк Ruby on Rails.

Рис. 1.1. Логотип Basecamp

Переваги:

- Зручний, інтуїтивний інтерфейс;
- Програма надсилає миттєві оповіщення щодо оновлень;
- Необмежена кількість користувачів, не потрібно додатково платити за кожного співробітника;
- Програма пропонує учасникам записати над чим вони працювали протягом дня, чим займаються щотижня та щомісяця. А отже, менеджеру не потрібно постійно нагадувати та просити своїх працівників звітуватись;
- Кожен член проєкту чи команди може завантажувати файли та має доступ до усіх документів, що стосуються його проєкту;
- Можна приймати імейли та відповідати на них прямо в системі;
- Вбудований чат, з назвою “Campfire”, допомагає підтримувати невимушену атмосферу;
- Є можливість кастомізувати робочий простір відповідно до вашого бренду, додати логотип, змінювати кольори;
- Доступний місяць безкоштовного пробного користування.

Недоліки:

- Коли задаєш дедлайн для проекту, то у календарі утворюється кольорова смуга від дати початку проєкту до кінцевої дати. Якщо проектів небагато, то це виглядає нормально, але якщо їх більше 5, то календар стає схожим на веселку;
- При плануванні великих проектів необхідно створити безліч to-do списків, в яких потім легко заплутатись.

Jira – це складна та багатогранна система, яка слугує помічником, як для програмістів, так і для менеджерів. Вона ідеально підходить для роботи над проектами та їх аналітикою.

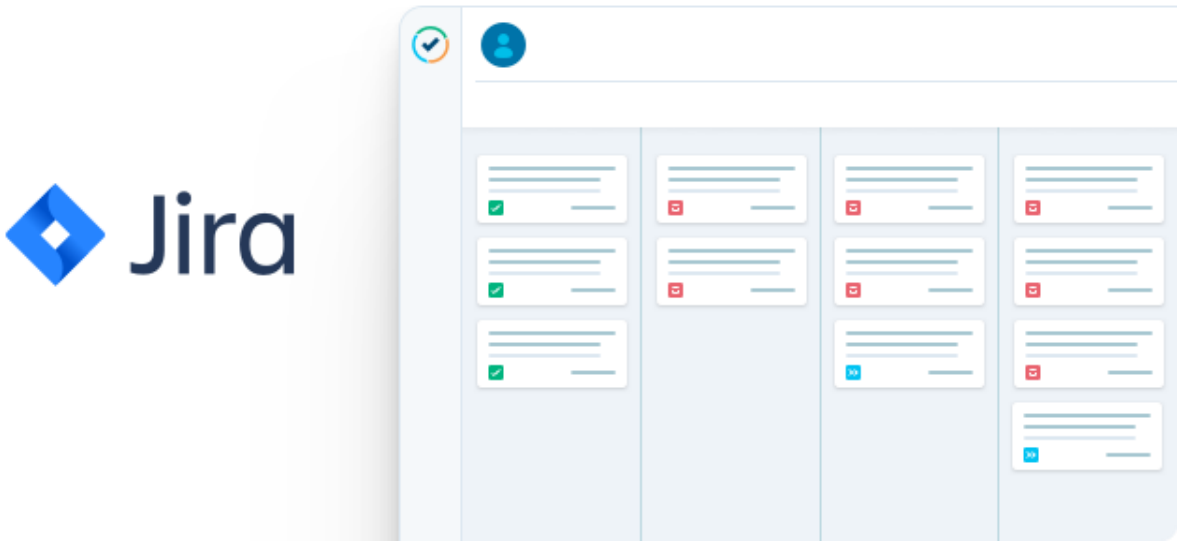

Рис. 1.2. Логотип Jira

Переваги Jira:

- Легко інтегрується з іншими інструментами (Salesforce, Zendesk, Bitbucket);
- Є вбудована система трекінгу часу;
- Створено безліч плагінів для будь-яких потреб;
- Можливість імпортувати звіти в Excel;
- Є інструменти, які допомагають досить точно оцінити скільки часу піде на виконання задачі;
- Дозволяє виставляти пріоритети задач;
- Прозорість системи: кожен виконавець може відслідковувати проблеми проекту і стежити за ходом виконання кожного завдання;
- Різноманітні діаграми для наочної звітності та аналізу;
- Безкоштовний місяць для тестування.

Недоліки Jira:

- Ціна. Особливо дорого для компаній, у яких більше 10 працівників, адже необхідно платити за кожного члена;
- Досить складні налаштування. Щоб усе працювало як треба, необхідно витратити чимало часу та сил. На офіційному сайті є спеціальний курс відеолекцій з тестами для підготовки майбутніх користувачів. Якщо туторіали вам не допомогли, то можна знайти спеціалістів, які налаштують Jira для вашої компанії за гроші.

Trello - дозволяє створювати дошки, на яких розміщуються списки, що складаються з карток. На картках ми зазвичай записуємо завдання, які далі обговорюємо у коментарях, додаємо термін виконання та прикріплюємо вкладення. Також є можливість додати чек-лист, мітки різних кольорів, учасників. Картки можна переміщувати по дошці, вносячи до іншого списку.

Рис. 1.3. Логотип Trello

Переваги Trello:

- Простий, інтуїтивний інтерфейс;
- Легко включитись до роботи через візуальне сприйняття обсягу завдань;
- Якщо ваше ім’я було вказане у картці, то вам на емейл буде надіслане це повідомлення;
- По праву сторону дошки у меню відображаються усі останні дії, що дозволяє стежити за взаємодією учасників;
- Можна змінювати фон дошки;
- Платна версія інтегрується з Jira, Bitbucket, Evernote, Google Hangouts, Mailchimp, Salesforce, Slack та іншими сервісами;
- Безкоштовна версія надає усі основні можливості, для невеликих проєктів її достатньо.

Кілька недоліків:

- Немає можливості видаляти картки, тільки архівувати;
- Для проведення аналітики потрібно підключати додаткові сервіси;
- Недостатньо функціоналу для повноцінної роботи над великими проєктами.

Asana - дуже привабливий та зручний веб-додаток, який розробили засновники Facebook. Починаючи працювати в середовищі Asana, у першу чергу слід створити робочий простір (workspace), тобто окремий розділ для певного проєкту чи компанії, та додати співробітників, які будуть над нам працювати. Далі ви можете створювати задачі з описом, коментарями, тегами та підзадачами у вигляді чек-листа. Якщо необхідно, то можна розбити підзадачі на більш дрібні підзадачі. При створенні задачі пропонується обрати кінцевий термін виконання, призначити виконавця, додати фоловерів, які будуть отримувати сповіщення щодо процесу роботи. Також можна користуватись спільним чатом та груповим календарем.

Рис. 1.4. Логотип Asana

Переваги Asana:

- Немає необхідності користуватись електронною поштою, адже можна спілкуватись та обмінюватись файлами у рамках системи;
- Одна задача може відноситись до кількох проєктів одночасно, що допомагає зберегти час;
- Дозволяє провести аналіз ефективності;
- Завдяки можливості додавати підзадачі на будь-якому рівні, легко деталізувати кожен етап роботи;
- Можна прикріплювати файли зі свого комп'ютера, Google Drive чи Dropbox;
- Візуально приємний інтерфейс;
- Після успішного виконання задач вас вітають чарівні істоти (єдиноріг, єті, нарвал та жар-птиця);
- Інтеграція з популярними сервісами (Evernote, Dropbox, MailChimp, Slack, Instagantt та іншими);
- Розроблено також мобільні додатки (Android, iOS);
- Базова версія є безкоштовною для команди менше 15 людей.

Недоліки Asana:

- Преміальна версія з розширеними можливостями коштує досить дорого, слід платити окремо за кожного співробітника;
- Мобільна версія не завжди працює добре.

Smartsheet – це онлайн-інструмент, завдяки якому клієнти та менеджери можуть легко спостерігати за процесом виконання завдань. Якщо ви раніше користувались Excel, то легко зорієнтуєтесь у цій програмі. Smartsheet допомагає планувати завдання, організуючи їх у таблиці, призначати виконавця та терміни виконання. Також можна додавати коментарі та відмічати стан виконання завдань, а ще спілкуватись у вбудованому чаті. Найкраще рішення для людей, які надають перевагу роботі з графіками, таблицями чи діаграмами.

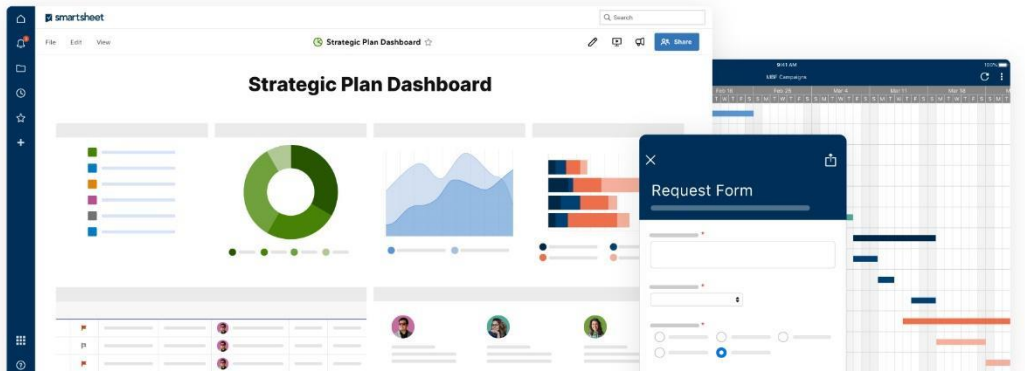

Рис. 1.5. Логотип Smartsheet

Переваги Smartsheet :

- Має інтерактивний календар, що синхронізується з Google та iCal;
- Дозволяє будувати графіки Ганта, за допомогою яких усі члени команди можуть спостерігати за просуванням проекту;
- Легко інтегрується з Google G-Suite, Microsoft Office 365, Salesforce, Jira та іншими сервісами;
- Завчасно нагадує про терміни виконання задач;
- Дозволяє додавати файли з Google Drive, OneDrive, Dropbox та інших додатків;
- Дає кожному користувачу можливість встановити оповіщення про важливі зміни чи оновлення;
- Зберігає усі дії у журналі, що дозволяє переглядати відомості про внесені зміни;
- Дозволяє змінювати колір таблиць та додати логотип своєї компанії

Недоліки Smartsheet:

- Висока ціна, слід платити за кожного користувача окремо;
- Спочатку було важно розібратись у налаштуваннях, адже програма має багато функцій.

Проведемо аналіз даних сервісів за функціональними характеристикам, наведено на Таблиця 1.1.

Таблиця 1.1.

Порівняльна таблиця веб-сервісів

Програмне забезпечення	*Співпраця з ПЗ*	*Відстеження помилок*	*Планування*	*Управління портфелем проектів*	*Управління ресурсами*	*Управління документами*	*Управління робочим процесом*	*Звітність*
Asana	+	-	+	+	+	-	+	+
Basecamp	+	-	-	-	+	+	-	-
Jira	+	+	+	-	-	-	+	+
Smartsheet	+	+	+	-	+	+	-	+
Trello	+	+	+	-	-	-	-	-

В більшості веб-сервісів налагоджено відстеження помилок у проектах, планування та взаємодія із програмним забезпеченням, що дає їм перевагу серед інших сервісів. Серед недоліків даних сервісів можна виділити управління робочим процесом, портфелями проектів та ресурсами. Управління робочим процесом є головною задачею веб-сервісу, тому вона має забезпечувати безвідмовну роботу та контроль можливих проблем. Для команди працівників, які взаємодіють з веб-сервісом, потрібно чітко розуміти свої задачі та термін здачі виконаного завдання. Управління портфелями проектів дає користувачам можливість працювати з декількома проектами, тому веб-сервіс має налагоджено працювати без збоїв. Адже проектів може бути багато, відповідно через велику завантаженість та інформаційну наповненість, можуть виникнути технічні відхилення. Управління ресурсами є важливим процесом сервісу, в який можуть внеситися зміни, додаватися нові дані, використовувати існуючу інформацію та інше. Цей процес має бути забезпечений захистом від шкідливих носіїв та сумнівних джерел інформації, для цього дані, що вносяться мають фільтруватися. Отже, у власному веб-сервісі є можливість

виправити недоліки аби задовольнити користувачів та залучити використовувати новий веб-сервіс.

Серед даних веб-сервісів є як безкоштовні варіанти так і платні, проведемо аналіз по характеристиками обліку витрат, часу та коштів наведено на Таблиці 1.2.

Таблиця 1.2.

Порівняльна таблиця веб-сервісів

Програмне забезпечення	*Облік витрат*	*Відстеження часу*	*Витрата коштів*
Asana	-	+	-
Basecamp	-	-	-
Jira	-	+	-
Smartsheet	-	-	-
Trello	-	-	-

По даним характеристикам веб-сервіси показали себе не з кращої сторони лише деякі можна відзначити за відстеження часу такі як Asana та Jira. Щодо витрати коштів дані сервіси за свій функціонал, просять внести фіксовану суму, яка досить зависока для звичайного користувача.

Є і безкоштовні відповідники, що надають схожі функції. Звідси слідує, що користувачам потрібен сервіс, що буде надавати послуги за доступною ціною, а можливо і мати безкоштовний відповідник. При цьому бути простим у використанні, не гаючи часу на освоєння сервісу.

ВИСНОВОК ДО РОЗДІЛУ 1

На сьогоднішній день існує велика кількість веб-сервісів залежно від потреб та функціональності даного програмного забезпечення. Кожен користувач обирає свого фаворита в сегменті представлених систем. В кожній такій системі є свої недоліки так і переваги за якими складається загальний рейтинг завантажуваності та використання.

У сфері інформаційних технологій сервіси управління та планування проєктами набувають стрімкої популярності, а відповідно і попиту. Тому розробка веб-сервісу є актуальним завданням для даної сфери і не тільки її.

Провівши аналіз вимог користувачів до веб-сервісів управління проєктами можна зробити висновок, що розроблене програмне середовище має бути легким для розуміння (від бухгалтера до програміста), доступним, орієнтуватися на різних по забезпеченню користувачів, а то й взагалі мати безкоштовну версію, можливість встановлення на будь-яку операційну систему.

РОЗДІЛ 2

АНАЛІЗ ТА ПОРІВНЯННЯ ОБРАНИХ ІНСТРУМЕНТІВ І ТЕХНОЛОГІЙ ДЛЯ ВЕБ-СЕРВІСУ

2.1. HTML

HTML - це мова, яка має текстовий підхід до опису структури вмісту, що міститься у файлі з розширенням .htm. Розмітка даної мови повідомляє браузеру, в якому вигляді відтворювати текст, картинки та інші формати мультимедійних файлів на сторінці.

HTML як кожна мова розробки, має певний синтаксис, файлову структуру та правила іменування, що відображає вміст файлу і має прочитуватися як всі інші файли, ці дані надаються веб-серверу та комп'ютеру. Застосувуючи перелічені властивості для HTML, можна створити веб-сторінку з текстового файлу в будь-якому текстовому редакторі і відтворити її в Інтернеті.

Основною з умов написання структури файлу HTML є пропис оголошення типу документа на початку текстового файлу. Завдяки даному оголошенню пристрій розуміє, що це саме файл HTML. Зазвичай заголовок документа виглядає так: <!DOCTYPE html>. В його написанні не має бути якогось вмісту чи розриву. Дані, що будуть вказані попереду цього скрипту не будуть розпізнані комп'ютером.

Типи документів може використовувати не лише HTML, а також застосовуватися для створення будь-якого документа, який використовує SGML. SGML - це стандарт для визначення конкретної мови розмітки, яка використовується. Серед мов до яких застосовують оголошенння SGML і doctype.

Ще одною важливою умовою для коректного створення файлу HTML є збереження документу із розширенням .htm чи .html, наведено на Рис. 2.1.

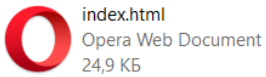

Рис. 2.1. Файловий вигляд документу HTML

Вказання розширення документу та прописування в файлі текстового типу дає комп'ютеру важливу інформацію для обробки. В процесі обробки дані перевіряються та готуються до завантаження до Інтернету, веб-сервер зчитує тип текстового розширення та після узгодження відправляє файл на клієнтський комп’ютер для прочитання внутрішнього вмісту.

В основному структура HTML-документа складається з тегів, за допомогою яких Інтернет-браузери розпізнаються вміст наповненої інформаціїї та розшифровують їх подання. Кожен тег має початкове оголошення і кінцеве оголошення (із направляючим слешем). Щоб розробник міг легко орієнтуватися в написанні базової структури коду HTML-документу, було введено найменування тегів, що несуть пряму інформацію по їх імені, що саме має містити цей так званий "контейнер". Приклад коду :

```
<div filter="moc" class="col-xl-4 col-md-6">
    <img src="img/item2.png" alt="">
    <div class="text">
        <h5>One Day Pass</h5>
        <p>Mockups</p>
    </div>
</div>
```

Рис. 2.2. Використання тегів HTML

Особливості HTML:

- мова легка для розуміння, проста в написанні;
- забезпечує гнучку структуру веб-сторінок для оформлення дизайну;
- не має прив'язки до операційних систем, а отже може відображатися на будь-які платформі від Linux до Windows;
- не є чутливою до регістру, тому написання тегів може бути як у нижньому так і у верхньому регістрі.

Плюси HTML:

- має велику кількість доступних ресурсів для вивчення;
- запускається в любому браузері;
- вивчається досить легко;
- має чистий та структурований вихідний код;
- можливість інтерграції з бекендовими мовами такими, як PHP.

Мінуси HTML:

- використовується для статичних веб-сторінок і немає динамічного спрямування;
- компоненти мають створюватися окремо не залежно від того чи використовують вони схожі елементи;
- не всі браузери можуть підтримувати нові версії мови.

2.2. CSS

CSS - це мова для опису представлення веб-сторінок, що включає в себе макет, шрифти, палітру кольорів тощо. Це дозволяє оформити зовнішній вигляд сторінок, адаптувати до різних типів пристроїв від браузерної версії до мобільної. Завдяки CSS можна керувати кольором тексту, стилем шрифтів, фоновими зображеннями,

створювати різноманітні ефекти тощо. CSS легкий у вивченні та розумінні, він має великий контроль над поданням HTML-документа.

Для того щоб комп'ютер розпізнав файл зі стилями, головною умовою при створенні документу є вказання розширення .css (Рис. 2.3.).

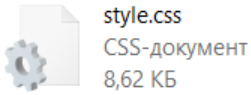

Рис. 2.3. Файловий вигляд документу CSS

Щоб до відповідної сторінки застосувалися стилі, потрібно підключити файл-CSS всередині файлу-HTML за допомогою тегу link у блоці head і ніде більше.

```
<link rel="stylesheet" href="css/style.css">
```

Рис. 2.4. Спосіб підключення файлу CSS

Структуру CSS можна охарактеризувати простими словами як набір правил, що описують, як має виглядати елемент. Правило складається з селектора та блоку оголошень.

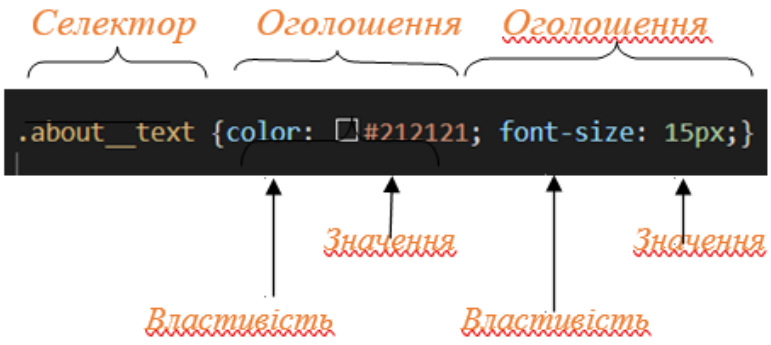

Рис. 2.5. Структура CSS

Описувати стилізацію сторінки можна у файлі відділеному від HTML, або ж прописувати стилі безпосередньо в структурі HTML. Для уникнення помилок та довгочасного відлагодження помилок більшість розробників розділяє вміст дизайну та структуру веб-сторінки. При цьому для звернення до елементів структури, CSS використовує назви заголовків, наведемо приклад:

```
.menu__item a {
    font-family: 'Open Sans Semibold';
    font-size: 20px;
    color: #212121;
}
```

Рис. 2.6. Застосування стилів в CSS

Переваги CSS:

- економія часу (можна один раз написати код, а потім повторно його використовувати для різних елементів);
- швидке завантаження сторінок;
- простота використання (можна внести зміни в одному місці і всі елементи зміняться автоматично);
- має великий набір стилів;
- сумісність для різних типів пристроїв.

Недоліки CSS:

- не всі браузери можуть підтримувати CSS;
- вибір версії (часто призводить до путаниці серед розробників і користувачів);
- не має безпеки для коду (будь-який користувач може внести зміни до коду, а можливо і не санкціоновано проникнути до сайту).

2.3. JAVASCRIPT

JavaScript - це мова програмування, яка використовується для створення та керування динамічним вмістом веб-сайту, тобто будь-яким, що переміщується, оновлюється або іншим чином змінюється на вашому екрані, не вимагаючи від вас перезавантажувати веб-сторінку вручну.

JavaScript застосовується при виконанні таких функцій як: анімована графіка, слайд-шоу з фотографій, текстові пропозиції автозаповнення, інтерактивні форми. Поширені приклади застосування мови JavaScript, ми може спостерігати щоденно, а саме, стрічка з підсумком новин, оновлення стрічки у Twitter, оновлення годинникової шкали на екрані вашого пристороюю чи девайсу, коли Google пропонує пошукові терміни на основі кількох літер, які ви почали вводити.

Основним способом застосування JavaScript є розробка веб-додатків та сайтів. За межами браузерів ця мова знайшла використання в програмному забезпеченні, обслуговування елементів апаратного керування та серверах. Розглянемо основні напрямки, де застосовується JavaScript:

1. Інтерактивні веб-сторінки

JavaScript дає змогу користувачам робити різні маніпуляції без перешкод на веб-сторінках ресурсів, ось кілька прикладів:

- Показати або приховати додаткову інформацію одним натисканням кнопки;
- Змінити колір кнопки, коли миша наведе на неї курсор;
- Прокрутити карусель із зображеннями на домашній сторінці;
- Збільшення або зменшення масштабу зображення;
- Відображення таймера або зворотного відліку на веб-сайті;
- Відтворення аудіо та відео на веб-сторінці;
- Відображення анімації.

2. Створення веб та мобільних додатків

Для полегшення розробки веб та мобільних додатків в JavaScript є різноманітний вибір фреймворків. Популярні інтерфейсні фреймворки JavaScript включають такі як, React, React Native, Angular і Vue. Багато компаній використовують Node.js, середовище виконання JavaScript, побудоване на движку JavaScript V8 Google Chrome. Кілька відомих прикладів розроблених на цій мові програмування Paypal, LinkedIn, Netflix та Uber.

3. Створення веб-серверів і розробка серверних додатків

В асортименті програмних продуктів, які взаємодіють з JavaScript є безпосередньо веб-сервер Node.js, що був розроблений компанією, що випустила JavaScript.

4. Розробка ігор

Можна розробляти як браузерні ігри, на зразок аркад, а також ігри для ПК.

Для програмування на JavaScript достатньо обрати будь-який текстовий редактор на вибір користувача, серед них розрізняють такі як - Atom, Sublime Text, Coda, Visual Studio Code, Notepad++, TextPad и Xcode. В кожному з цих редакторів є виділення синаксису коду різними кольорами, це дає змогу швидше знайти помилки в коді.

Не зважаючи на те, що для кодування мовою JavaScript потрібно лише текстовий редактор та браузер, також для розробки програмних продуктів використовують такі інструменти як:

- Git - цей інструмент бере на себе контроль версій, який допомагає управляти проєктами в залежності від масштабності та налагоджувати роботу над одним проєктом багатьох розробників;

- Node - забезпечую віддалений доступ скриптів поза браузером;

- Gulp - інструмент для конструювання розробки;

- Babel - перетворює ES5 на ES6;

- ESLint - це аналізатор, допомагає уникнути допускання частих помилок при написанні коду.

Переваги:

- підтримується більшістю браузерами, взаємодіючи із таблицями стилів та серверною частиною;
- при обробці веб-сторінок на користувацьких ПК даний движок не робить запитів до сервера, завдяки цьому знижується трафік та навантаження на сервер;
- велика різноманітність плагінів для взаємодії з JavaScript, що полегшує роботу для розробників;
- простий та компактний у використанні;
- інтерфейси, а саме наведення курсору, заповнення форм, активація кнопок є зручною у використанні;
- в порівнянні з іншими мовами програмування JavaScript є одним із легкодоступних мов.

Недоліки:

- обмеженість в завантаженні файлів та їх прочитання є причиною для збереження безпеки;
- має місце недоліки в перетворенні даних це проявляється саме в процесі роботи;
- немає віддаленого доступу до розроблювальних продуктів;
- відкритий код відкриває можливість зловмисникам нанести непоправну шкоду.

2.4. Фреймворки JS

Фреймворки JS - це бібліотеки програмування JavaScript, які мають попередньо заготовлені скрипти для використання у стандартних задачах та функціях.

Головною задачею фреймворків є те, що вони суттєво полегшують процес кодування.

Переваги фреймворків:

- безкоштовні у використанні;
- мають відкритий вихідний код, що мінімізує всі ризики;
- підвищують продуктивність (готові скрипти дають можливість меньше витрачати час на написання коду вручну, код стає "чистіший", налагодження помилок відбувається швидше);
- модульність додатку, що дає можливість працювати декільком працівникам над додатком одночасно.

Одним із недоліків фреймворків є неповноцінна підтримка пошуковими браузерами. Хоча навіть цей недолік вирішили передові пошукові системи, такі як Google.

Розглянемо використані фреймворки в дипломній роботі.

Angular - це JavaScript-фреймворк, що дозволяє створювати динамічні веб-додатки. Це середовище дозволяє розробнику поєднувати такі інструменти як, JavaScript, HTML та CSS. Angular має багато готових рішень, а також в ньому добре розроблена структура зберігання даних та система компонування складових проєкту, тому він підходить для розгортання великих проєктів із складною структурою. Популярним застосуванням Angular є створення односторінкових сайтів, тобто SPA. Сутью таких сторінок - користувач переходячи на псевдосторінку не завантажує нову інформацію, а оновлюються тільки динамічні дані. Завдяки такій властивості, сторінка кожен раз не оновлюється як це відбувається на стандартних сайтах, коли кожна нова сторінка витягується із сховища і ми можемо спостерігати полотно білого екрану. Важливим фактором є те, що це впливає на контурентоспроможність сайту, коли між користувачем постає вибір зайти на сайт із швидким оновленням даних та сайтом із затримкою завантаження даних до 10 сек, відповідно користувач обере перший варіант.

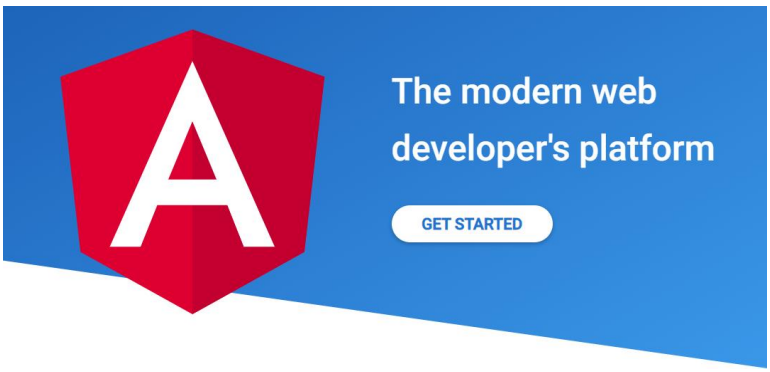

Рис. 2.7. Середовище Angular

AngularJS є досить використовуваним продуктом, який застосований на таким відомих сайтах як:

- сайт відеоконтенту YouTube;

- сайт побутової техніки Bosch;

- біржа фрілансерів freelancer.com;

- сайт виробника багатьох видів техніки General Electrics.

Переваги AngularJS:

- використання декларативного програмування дозволяє покращити тестувальність та масштабність додатків; завдяки визначенню залежностей, компоненти можна відділити від їх залежності та повторно використовувати;
- двостороннє прив'язування даних, коли дані змінюються, змінюється і подання; двостороннє прив'язування даних покращило роботу розробникам, а саме скоротило час кодування, оскільки не потрібно писати додатковий код для забезпечення постійної синхронізації моделі та представлення;
- підтримка технологій та інструментів таких як - Ajax, DOM, шаблони, анімації, маршрутизація та багато інших;
- комунікативність, завдяки високому попиту на використання AngularJS, з'явилось багато технічної літератури, спільнот для обговорення питань по застосуванню скриптів, рішення для вирішення кожної проблеми, що виникає.

Мінуси AngularJS:

- продуктивність, динамічні програми спочатку не працювали так ефективно як на теперішній час, наприклад, односторінкові сторінки, можуть бути не зручними у використанні через свої розміри;
- різноманітність способів виконання завдань, викликає путаницю між розробників, але завдяки великій кількості навчальних посібників та обговорень можна вирішити більшість проблем;
- конфлік версій - в нових версіях середовищах, ті ж функції перероблюються в кожній версії, що викликає непорозуміння.

ExpressJS - це гнучкий фреймворк веб-додатку Node.js, який використовується для веб та мобільних додатків. Даний фреймворк має відкритий вихідний код та підтримується фондом NodeJS, мінімальний інтерфейс для створення програм. ExpressJS, крім того, дає нам інструменти, необхідні для створення програми. ExpressJS також є гнучким, оскільки існують різні модулі, які доступні в консолі і які можна безпосередньо підключити до нього, тобто Express.

ExpressJS також називають "фреймворком для інших фреймворків", бо існує велика кількість інших фреймворків, які побудовані з використанням ExpressJS. За допомогою ExpressJS створені такі сайти як Coursera, IBM, Nike, Paypal, Twitter, Uber.

Рис. 2.8. Лого ExpressJS

Переваги ExpressJS:

- робить розробку веб-додатків Node.js швидкою та легкою;
- легко установити та налаштувати;
- дозволяє визначати маршрути вашої програми на основі методів HTTP та URL-адрес;
- включає різні модулі проміжного програмного забезпечення, які можна використовувати для виконання додаткових завдань для запитів і відповідей;
- легко інтегрувати з різними механізмами шаблонів, такими як Jade, Vash, EJS тощо;
- дозволяє визначити проміжне програмне забезпечення обробки помилок;
- легко обслуговувати статичні файли та ресурси вашої програми;
- дозволяє створити сервер REST API;
- легко підключатися до таких баз даних, як MongoDB, Redis, MySQL.

Недоліки ExpressJS:

- немає готових рішень для забезпечення безпеки;
- трохи плаваюча структура.

Node.js - це кросплатформне середовище виконання з відкритим вихідним кодом для розробки серверних і мережевих додатків. Програми Node.js написані на JavaScript і можуть запускатися в середовищі виконання Node.js в OS X, Microsoft Windows і Linux.

Node.js також надає багату бібліотеку різноманітних модулів JavaScript, що значною мірою спрощує розробку веб-додатків за допомогою Node.js.

Основна ціль Node.js: використовувати не заблоковане орієнтоване введення/виведення даних, щоб залишатися легким за об'ємом та ефективним при зверненні до додатків, обробляючими великі об'єми даних в реальному часі і функціонуючими на розподільних пристроях.

Рис. 2.9. Лого Node.js

Особливості Node.js:

1. Асинхронний і керований подіями — усі API бібліотеки Node.js асинхронні. По суті, це означає, що сервер на основі Node.js ніколи не чекає, поки API поверне дані. Сервер переходить до наступного API після його виклику, а механізм сповіщень про події Node.js допомагає серверу отримати відповідь від попереднього виклику API.
2. Дуже швидкий. Бібліотека Node.js, побудована на движку JavaScript V8 від Google Chrome, дуже швидко виконує код.
3. Однопотоковий, але високомасштабований — Node.js використовує однопотокову модель із циклічним циклом подій. Механізм подій допомагає серверу відповідати неблокуючим способом і робить сервер дуже масштабованим на відміну від традиційних серверів, які створюють обмежені потоки для обробки запитів. Node.js використовує одну потокову програму, і та сама програма може надавати обслуговування набагато більшої кількості запитів, ніж традиційні сервери, такі як Apache HTTP Server.
4. Без буферизації — програми Node.js ніколи не буферизують будь-які дані. Ці програми просто виводять дані фрагментами.
5. Ліцензія — Node.js випущено за ліцензією MIT.

Node.js показує себе з кращої сторони при створенні швидких масштабних мережевих додатків, оскільки дозволяє одночасно обробляти велику кількість з'єднань з великою пропускною спроможністю.

Прикладом невдалого використання Node.js є операції, які навантажують процесор та важкі обчислення, навантажують середовище, що приводить до збоїв в роботі.

Переваги:

- легкість і швидкість написання коду;
- кросплатформеність;
- різноманітність безкоштовних інструментів;
- можливість сумісного і повторного використання;
- швидкість роботи додатків.

Недоліки:

- гнучкість та часте оновлення програмних продуктів, несе в собі і мінуси, адже потрібно постійно спостерігати за оновленням, бо деякі продукти виходять недостатньо протестованими;
- при видаленні бібліотеки із npm і більшість додатків використовуваних нею перестали працювати.

2.5. База даних Firebase

Firebase - це платформа для розробки, відома своєю базою даних у реальному часі, яка оптимізована для синхронізації даних та центральним сховищем даних. Ця платформа розроблена для того, щоб полегшити життя розробникам, завдяки обробленню більшої частини даних для передавання та витягування. Дані, що будуть записані узгоджуються у всій системі.

Головним інструментом для хмарної розробки Google є Firebase. У 2014 році Google викупила продукт в одноіменної компанії. Firebase доступний через Google. Тим часом доступні бібліотеки та інструменти з відкритим кодом, які взаємодіють із Firebase.

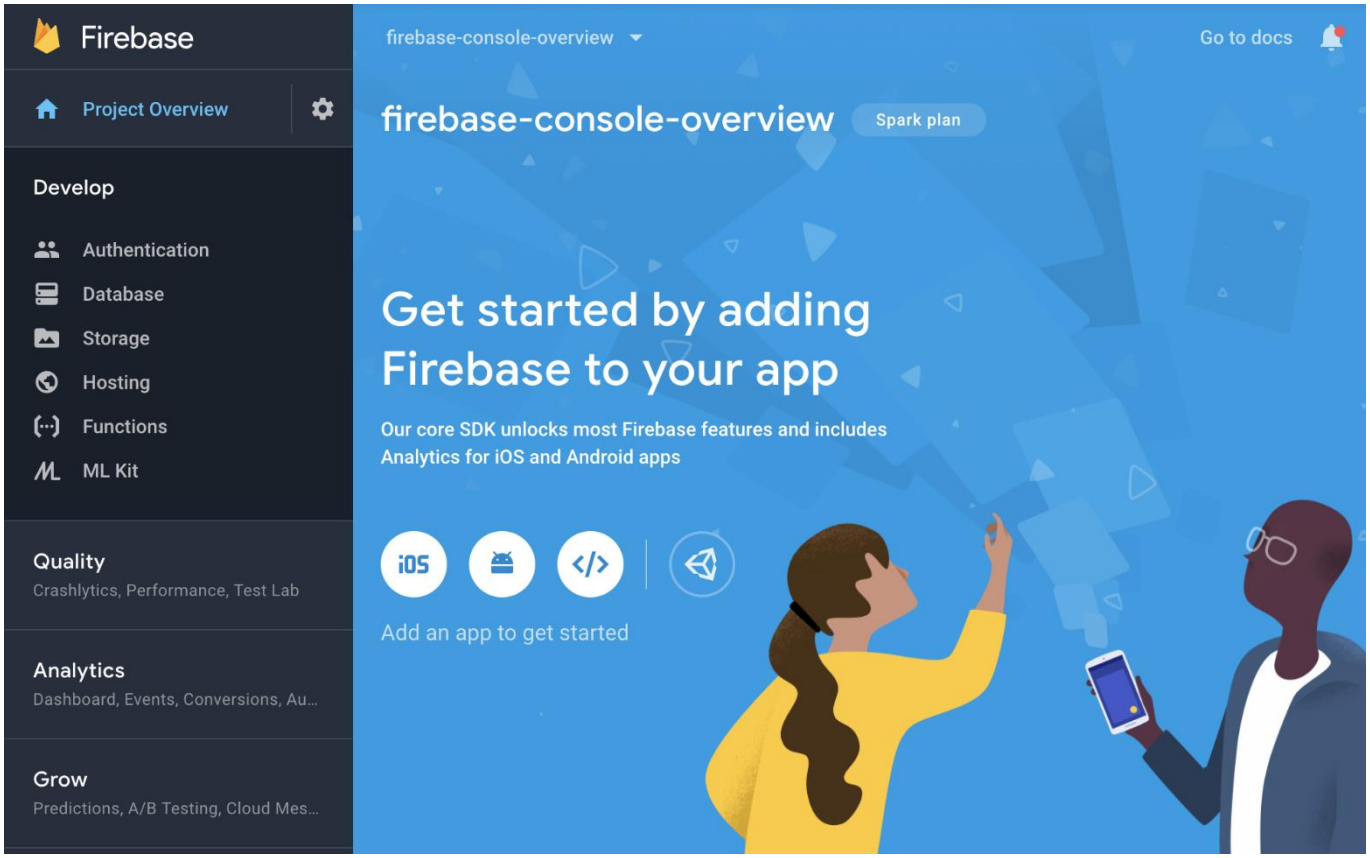

Рис. 2.10. Головна сторінка Firebase

Більшість користувачів обирає Firebase через, те, що платформа може постійно поширювати та синхронізувати зміни між локальними копіями інформації, що зберігаються в хмарі. Платформа не обмежується одним комп'ютером, вона дозволяє розділяти навантаження між кількома машинами розділяючи набори даних. База даних може розподілятися на всю мережу доступних пристроїв для цього використовується узгодженість між центрами обробки даних, по суті кожен задіяний пристрій є частиною хмари. Розробникам не потрібно турбуватися про небезпеку невідповідності між даними, скажімо, на телефоні користувача та центральній базі

даних. Після того, як дані зберігаються локально, Firebase відправляє копії на хмарні сервери, щоб обидві версії були узгоджені. Передача також працює в іншому напрямку, оскільки зміни, внесені в хмару, вони реплікуються локально. Розробники на стороні сервера можуть спілкуватися з клієнтським програмним забезпеченням, яке працює, просто записуючи дані в хмару Firebase.

Існує також безсервісний варіант хмарних функцій, що можна інтегрувати з Firebase, щоб нові дані могли запускати функції. Коли користувач вперше входить в систему або щоразу, коли база даних змінюється, буде викликатися функція, яка потім може ініціювати інші події або функції в хмарі Google або в іншому місці. Ці функції можна використовувати для простої обробки зображень, очищення тексту або забезпечення узгодженості даних.

Основні характеристики Firebase:

1. *Аутентифікація*

   Він підтримує аутентифікацію за допомогою паролів, номерів телефонів, Google, Facebook, Twitter тощо. Аутентифікацію Firebase (SDK) можна використовувати, щоб вручну інтегрувати один або кілька методів входу в програму.
2. *База даних реального часу*

   Дані синхронізуються між усіма клієнтами в режимі реального часу і залишаються доступними, навіть коли програма переходить в автономний режим.
3. *Хостинг*

   Firebase Hosting забезпечує швидкий хостинг для веб-програми; вміст кешується в мережах доставки вмісту по всьому світу.
4. *Випробувальна лабораторія*

   Додаток тестується на віртуальних і фізичних пристроях, розташованих у центрах обробки даних Google.

5. *Повідомлення*

   Повідомлення можна надсилати за допомогою Firebase без додаткового кодування.

Переваги Firebase:

- Електронна адреса та пароль, аутентифікація Google, Facebook та Github;
- Дані в реальному часі;
- Готовий API;
- Вбудований захист на рівні вузла даних;
- Сховище файлів із підтримкою Google Cloud Storage;
- Статичний файлообмінник;
- Змінювана інфраструктура.

Недоліки Firebase:

- Обмежені можливості запитів через модель потоку даних Firebase;
- Традиційні реляційні моделі даних не застосовуються до NoSQL, отже, ваші фрагменти SQL не передадуться.

ВИСНОВОК ДО РОЗДІЛУ 2

Даний розділ присвячений інструментам та технологія, які будуть використовуватися в розроблювальному продукті "Web-сервіс для планування та управління проектами ". Серед зазначених вище інструментів були описані такі, як HTML, CSS, JavaScript, фреймворки JS та віртуальна база даних Firebase. Кожен інструмент було детально проаналізовано та описано, відзначено особливості кожного, переваги та недоліки. Обрані інструменти та технології було обрані попередньо вивчивши доцільність використання, актуальність інструментів, зрозумілість та легкість експлуатації, аби в майбутньому не виникало складнощів у налаштуванні та відлагодженні.

РОЗДІЛ 3

РЕАЛІЗАЦІЯ WEB-СЕРВІСУ ДЛЯ ПЛАНУВАННЯ ТА УПРАВЛІННЯ ПРОЄКТАМИ

3.1. Постановка задачі

Для реалізації веб-сервісу потрібно створити інтерфейс для розподілення задач над проєктом, в якому можна адміністратору/куратору проєкту створювати нові задачі для розробників відповідно у створеному раніше проєкті, що будуть містити в собі такі дані для заповнення: тип завдання, назва завдання, примітки до назначеної задачі, на кого розподілена задача, дата призначення, статут завдання. При створенні нового проєкту адміністратор має коректно заповнити дані, такі як назва проєкту та примітки до проєкту. Форма для створення проєкту має перевіряти чи адміністратор заповнив поля і не залишив їх порожніми, а у разі відсутності даних форма має відобразити повідомлення, що поля мають бути заповнені. Тільки у випадку заповнених полів, адміністратору запропонують створити новий проєкт. Відповідальний за проєкт матиме можливість повернутися з вікна створення проєкту до головного меню.

Адміністратору проєкту після створення задач для працівників має бути можливість переглядати розподілені задачі, вносити зміни чи видаляти їх. Також йому мають відкриватися дані по проєктам, відповідно який проєкт на якій стадії розробки(тільки створений проєкт, в процесі розробки, готовий до тестування та налагодження, реалізований). Тільки адміністратор проєкту може розподілити задачі та вносити різного виду зміни до проєкту чи відповідальна особа якій призначено статус адміністратора.

Перед початком роботи з веб-сервісом, кожен новий користувач має проходити процедуру реєстрації, яка включатиме в себе внесення даних по електронній пошті, імені користувача та створений користувачем пароль. Також користувач, зможе зайти у систему скориставшись існуючим обліковим записом в Google.

У кожного користувача при вході в систему має відображатися панель з проєктами до яких він залучений та які на нього розподілені завдання. В процесі виконання поставлених задач, член команди на якого призначена задача зможе коригувати статус поставленої задачі, тобто в процесі, готове до тестування, виконано.

Для кожного проєкту має бути своя сторінка для відображення даних. На цій сторінці мають міститися записи про команду розробників (прізвище, ім'я, електронна пошта, посада, контакти), розподілені задачі (на кого призначена, статус задачі, тип задачі, термін виконання, примітка, назва завдання). Для кожної задачі має бути окрема сторінка де має відображатися для адміністратора – всі завдання, які було створено ним, а для розробника – перелік призначених на нього завдань.

3.1.1. Інформаційні потоки користувацького інтерфейсу

У користувацькому інтерфейсі застосовані такі інформаційні потоки:

- персональні дані (спосіб індентифікаціїї користувача в системі з поміж інших, джерело інформації - користувач);
- планувальник задач (задачі для окремого проєкту, внесення змін, видалення задач, терміни
- опрацювання, джерело інформації - адміністратор);
- компонувальник проєктів (назва проєкту, основні події, група працівників, цілі проєкту, термін виконання, джерело інформації - адміністратор);
- рейтинг виконуваності поставлених задач.

Також до інформаційних потоків можна віднести форми введення та виведення даних, модальні вікна. До таких форм належать: форма коригування даних проєкту/задач, форма видалення проєкту/задач, форма реєстрації нового користувача.

3.1.2. Розробка структури програмного проєкту

Визначемо основні абстакції системи

Проаналізувавши постановку задач для реалізації в розроблюваній системі, можна виділити основні абстракції предметної області, представлені в таблиці.

Таблиця 3.1

Абстракції інформаційної системи

Абстракція	*Опис*
Користувач	Користувач має можливість переглядати розподілені на нього задачі, бачити перелік проєктів до яких він приєднаний, коригувати стан виконання задач, користуватися особистим кабінетом.
Адміністратор	Адміністратор наділений правами облаштування проєкту та задач.
Проєкт	Це група задач, яка поєднує в собі реалізацію одного вихідного продукту.

Абстракція	*Опис*
Задача	Це проблемна ситуація, яка потребує вирішення попередньо проаналізувавши кроки її вирішення.
Модальне вікно	Це вікно, яке блокує роботу користувача до того моменту поки його не закриють.
Примітки	Стислі коментарі для полегшення роботи з тим чи іншим проєктом/задачею.
Документ	Це файл, який містить інформацію для ознайомлення групи користувачів.

Дані абстракціїї має такі поведінки, наведені в таблиці 3.2.

Таблиця 3.2.

Опис поведінки абстакції

Абстракція	*Поведінка*	*Опис поведінки*
Користувач	Вхід/реєстрація до веб-сервісу. Користування особистим кабінетом, внесення змін до виконаних задач.	Користувач здійснює вхід/реєструється до веб-сервісу, коригує особистий кабінет під свої потреби, управляє статусом задач.
Адміністратор	Створює нові проєкти та розподіляє задачі.	Адміністратор може вносити зміни, видаляти задачі/проєкти.
Проєкт	Містить набір задач.	Об’єднує задачі в один програмний продукт.
Задача	Містить опис потребуючого представлення готового вирішення задачі.	Надається користувачу для вирішення.
Модальне вікно	Вікно із результатом виконаної дії.	Відображається при внесенні змін/видаленні проєкту/задачі.
Примітки	Коментарі для представлення опису проєкту/задач.	Полегшують роботу працівникам.
Документ	Файли для ознайомлення користувачам, може мати різні розширення.	Зберігається в веб-сервісі, доступний для користувачів веб-сервісу.

Побудуємо діаграму бізнес-об'єктів на якій відображено взаємодію об'єктів між собою (Рис. 3.1).

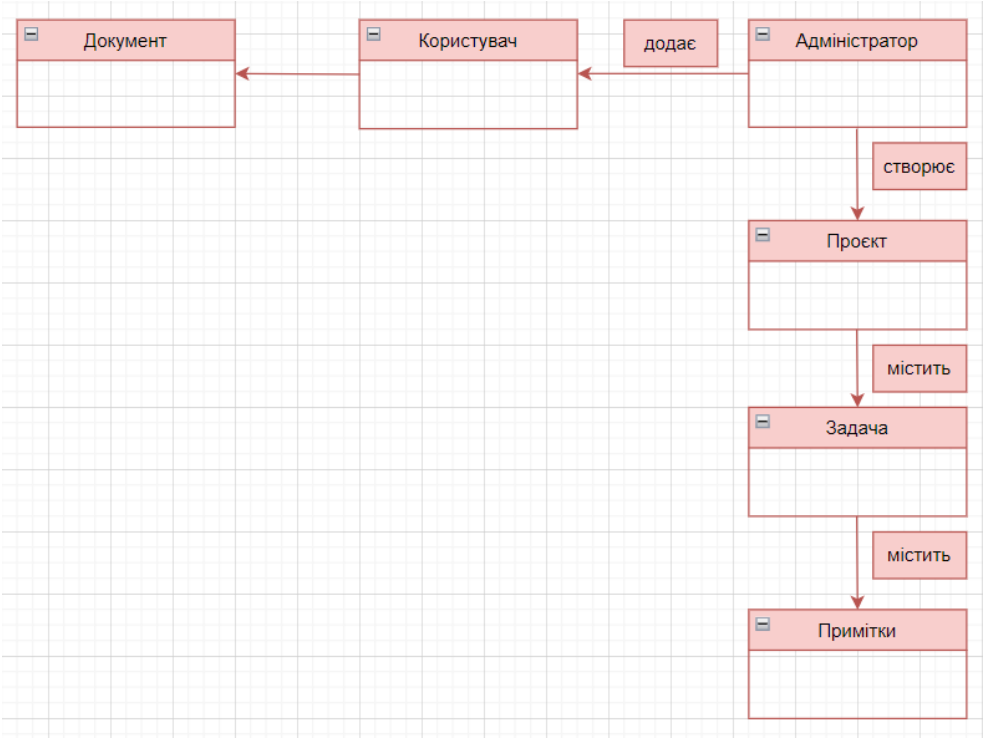

Рис. 3.1. Діаграма бізнес-об'єктів

Розробка діаграми варіантів використання системи

Системи має виконувати наступні вимоги:

- у системі має бути розділено права управління веб-сервісом;
- саме адміністратор може створювати/ видаляти/змінювати проєкти;
- із панелі адміністратора має бути можливість розподіляти задачі на групу працівників;
- при зміні/видаленні/додаванні даних до проєкту/задачі має відображатися модальне вікно із підтвердженням виконаної дії;
- адміністратор має мати панель для відображення створених проєктів/задач для контролювання виконання поставлених задач;
- адміністратор має можливість додавати/видаляти користувачів до проєктів;

- користувач має можливість проглядати дані по проєкту/задачам;
- зареєстрований користувач має можливість авторизуватися у системі за допомогою існуючого облікового запису Google.

Була побудована діаграма варіантів використання, яка предсталена на Рис. 3.2.

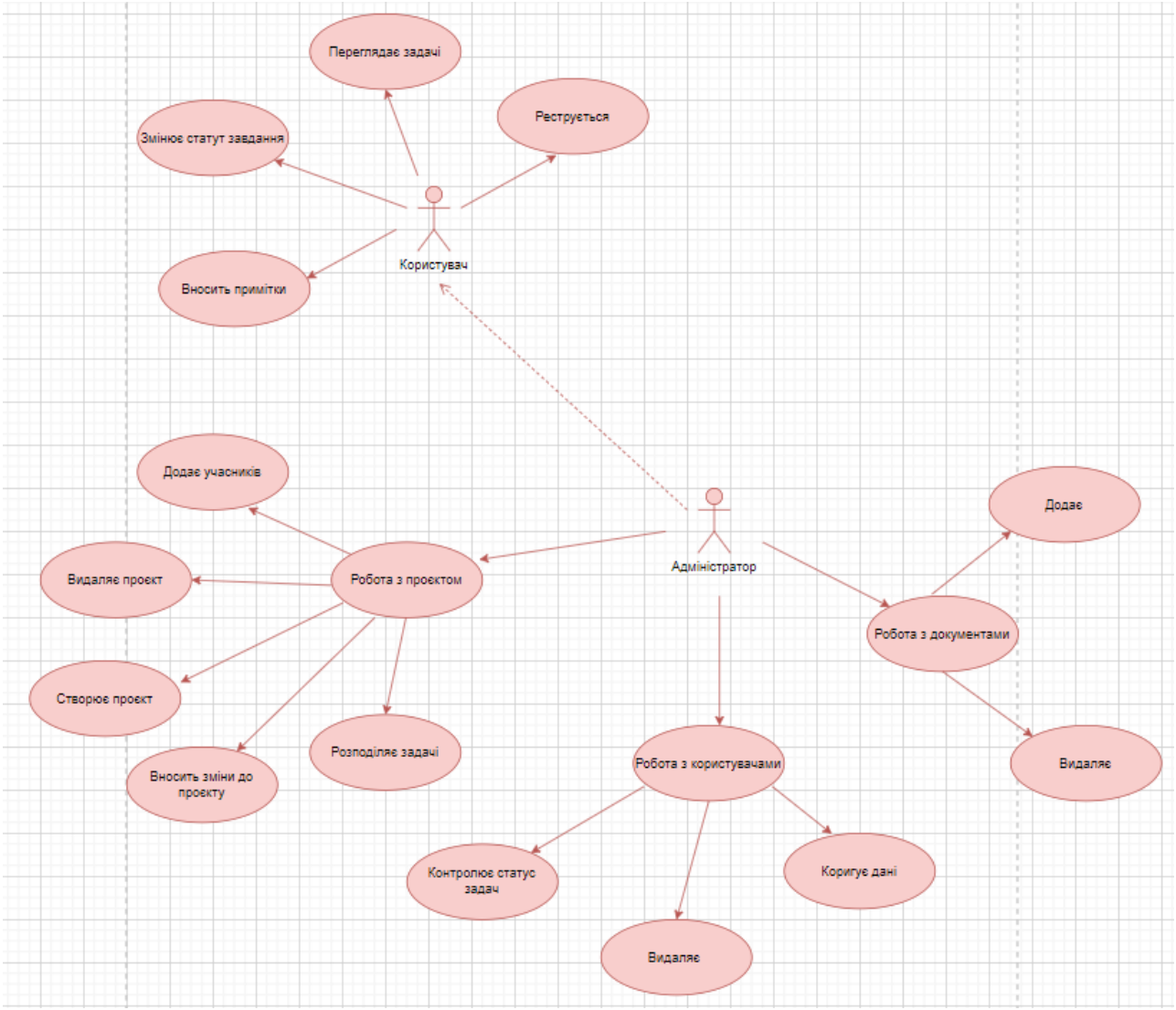

Рис. 3.2. Діаграма варіантів використання

Опишемо дійових осіб в діаграмі варіантів використання (Таблиця 3.3).

Таблиця 3.3.

Опис дійових осіб

Дійова особа	Посада	Варіанти використання
Користувач	Працівник проєкту	Реєструється для входу
		Переглядає задачі
		Змінює статус завдання
		Вносить примітки
Адміністратор	Керівник проєкту	Створює проєкт
		Видаляє проєкт
		Вносить зміни
		Додає учасників
		Розподіляє задачі
		Видаляє користувачів з проєкту
		Вносить зміни по працівникам
		Контролює статус виконання задач
		Додає документи
		Видаляє документи

В таблиці 3.4 наведено опис варіантів використання.

Таблиця 3.4.

Опис варіантів використання

Назва	Опис
Реєстрація для входу	Внесення користувачем персональних даних для подальшої індентифікації.
Перегляд задач	Відображення панелі задач, де вказано задачі, які розподілені на конкретного працівника.
Зміна статусу завдання	Після завершення виконання задачі, працівник відмічає готовність завдання, а також може поставити на паузу/не розпочате.
Вносити примітки	Користувач може додавати примітки до задачі, аби іншим працівникам було простіше зрозуміти хід виконання чи у випадку розподілення задачі на двох працівників, один працівник проглянув, які вимоги виконані чи є якісь проблеми/неточності, що треба виправити.
Створення проєкту	Адміністратор вводить у форму назву проєкту, опис проєкту, перелік працівників, термін виконання.
Видалення проєкту	Адміністратор заходить у вже створений проєкт і видаляє його за необхідності.
Внесення змін	Адміністратор натискає кнопки Змінити та вносить корегування по тим даним, які йому потрібно змінити.
Додання учасників	При створенні проєкту адміністратор обирає працівників, які будуть залучені до проєкту.

Розподілення задач	До кожної задачі, яку потрібно виконати адміністратор додає відповідного працівника чи працівників залежно від об’єму виконуваної роботи.
Видалення користувачів з проєкту	В залежності від обставин, які виникли, адміністратор може видалити одного працівника замінивши іншим.
Внесення змін по працівникам	Якщо дані працівників потрібно підкорегувати, адміністратор заходить по профілю користувача і вносить зміни, а саме ті, що стосуються проєкту.
Контроль статусу виконання задач	Адміністратор може переглядати панель створених проєктів та панель виконуваних задач, контролюючи цим терміни здачі робіт.
Додання документів	Адміністратор може додавати файли різних розширень, що матимуть інформативність для користувачів.
Видалення документів	Адміністратор може видаляти файли, наприклад: у разі некоректного відкриття чи втрати актуальності цим чистячи реєстр.

3.2. Підготовка до розробки веб-сервісу

3.2.1 Опис програмних модулів

Розроблений веб-сервіс має таку файлову структуру:

- e2e
    - src
    - protractor.conf.js – налаштування локального сервера;
    - tsconfig.json - файл конфігурацій;
- src
    - app
        - component – компоненти веб-сервісу;
        - guards – сценарії;
        - services – сервіси;
        - shared – посилання;
        - app-routing.module.ts – налаштування модуля маршрутизації;
        - app.component.html – структура маршрутизатора;
        - app.component.scss – стилізація;
        - app.component.spec.ts – створення компонентів;
        - app.component.ts – авторизація користувача;
        - app.module.ts – модулі;
        - interfaces.ts – дані інтерфейсу;
    - assets

        - img – картинки/іконки;
    - environments - специфічні компоненти середовища;
    - index.html – файл підключення інструментів;
    - main.ts – налаштування середовища;
    - polyfills.ts – системний файл;
    - styles.scss – стилізація;
    - test.ts – тестування середовища;
- .browserslistrc – браузер-лист;
- .editorconfig – зміна конфігурації;
- .gitignore – ігноровані файли з git;
- angular.json – файл конфінурації компонувальних складових сервісу;
- karma.conf.js – підключення модулів;
- package-lock.json – технічний файл;
- package.json – технічний файл;
- README.md – текстовий файл, що містить дані про файли;
- server.js – сервер;
- tsconfig.app.json – конфігураційний файл для Angular;
- tsconfig.json – загальний файл, що містить typescript;
- tsconfig.spec.json – файл конфігурацій typescript для тестування додатків;
- tslint.json – параметри інтерфейсу.

Приклад файлової структури представлено на Рис. 3.3.

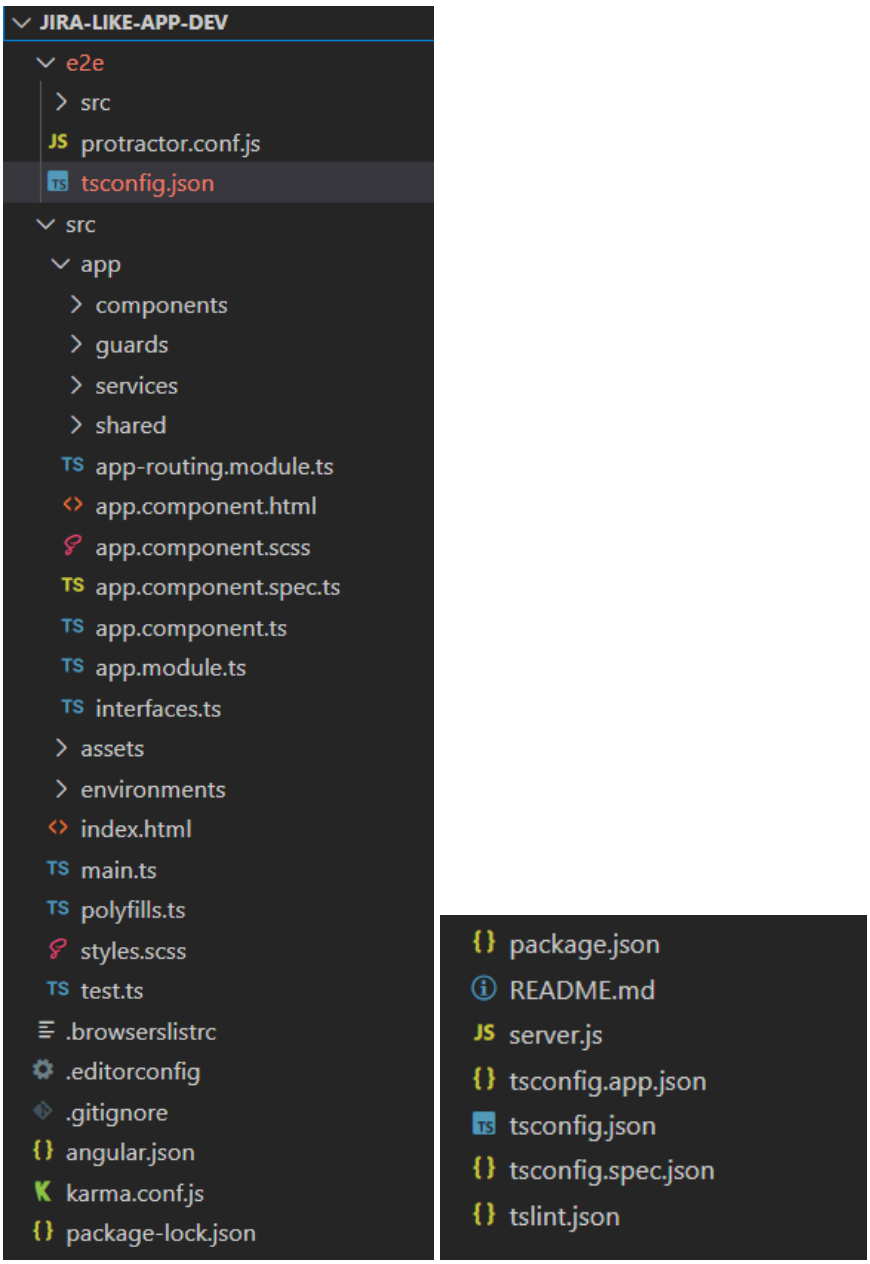

Рис. 3.3. Файлова структура веб-сервісу

3.2.2. Розгортання та налаштування проєкту

Для початку розгортання проекту запустимо середовище розробки в даному випадку – Visual Studio Code. Попередньо створимо папки для розподілу компонентів та зручності використання.

Створимо файл «package.json», що буде описувати створений проєкт загалом. Для його створення використаємо вбудований термінал в середовищі розробки. Даний файл містить назву проєкту, версію, скрипти та підключені бібліотеки (Рис. 3.4).

```
{
  "name": "jira-like-app",
  "version": "0.0.0",
  ▷ Debug
  "scripts": {
    "ng": "ng",
    "start": "ng build & node server.js",
    "build": "ng build",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "~10.2.0",
```

Рис. 3.4. Створення проєкту

Далі запустимо локального сервер для розгортання проєкту (додамо бібліотеки, відслідкування url-адрес, дані з яких будуть записуватися у відповідний текстовий файл, створемо статичну папку, де буде міститися різні .html, .css та img файли). Для цього створимо файл «server.js» та проведемо ініціалізацію.

```
JS server.js > ...
1   const express = require('express');
2   const path = require('path');
3   const app = express();
4
5   app.use(express.static(__dirname + '/dist/jira-like-app'));
6
7   app.get('*', function(req, res) {
8       res.sendFile(path.join(__dirname + '/dist/jira-like-app/index.html'));
9   });
10
11  app.listen(process.env.PORT || 8080);
12
```

Рис. 3.5. Підключення сервера

Програмний продукт містить базу даних та підключений локальний сервер. У базу даних будуть відправлятися дані по користувачам веб-сервісу, інформацію з бази зможе переглядати/змінювати/видаляти дані за необхідності системний адміністратор.

Для підключення бази даних створимо файл "environment.ts", де пропишемо скрипт для з'єднання з обраною хмарною БД – Firebase. Платформа Firebase завдяки тому, що знаходиться в хмарі дає можливість не турбуватися щодо виникненняконфузів та спокійно виконувати розширення розроблювальних програмних продуктів.

```
const firebaseConfig = {
  apiKey: "AIzaSyCKK4MD8Mpb8ThbmuHUoDdg5tHk6sd-zfM",
  authDomain: "jira-7afb4.firebaseapp.com",
  projectId: "jira-7afb4",
  storageBucket: "jira-7afb4.appspot.com",
  messagingSenderId: "1019039171160",
  appId: "1:1019039171160:web:a44cdda04ef4ade0ac402d",
  measurementId: "G-EBCXNHYDH0"
};
```

Рис. 3.6. Підключення БД

3.2.3. Опис основних логічних процесів

Розпочнемо описувати логіку для програмного продукту. Для початку створимо функцію ініціалізації користувача для входу в систему (зареєстрованого/не зареєстрованого).

Приклад коду для ініціалізації зареєстрованого користувача наведено на Рис.3.7.

```
  signIn() {
    if (this.email && this.password) {
      this.snackBar.openSnackBarAfterSuccess(
        this.authService.signIn(this.email, this.password),
        SNACKBAR_MESSAGES.login_success
      )
    } else {
      this.validSignIn = false;
    }
  }

  setUserDataForAuth(event) {
    this.email = event.email;
    this.password = event.password;
  }
}
```

Рис. 3.7. Код верифікації зареєстрованого користувача

Приклад коду для ініціалізації не зареєстрованого користувача наведено на Рис. 3.8.

```
  signIn() {
    if (this.email && this.password) {
      this.snackBar.openSnackBarAfterSuccess(
        this.authService.signIn(this.email, this.password),
        SNACKBAR_MESSAGES.login_success
      )
    } else {
      this.validSignIn = false;
    }
  }
```

Рис. 3.8. Код верифікації не зареєстрованого користувача

Коли користувач відкриває веб-сервіс, він йому відкривається сторінка входу до системи, яка пропонує користувачу обрати спосіб входу:

- Для зареєстрованого користувача:
  - Внести дані логіна та пароля;
  - Зайти до системи за допомогою облікового запису Google.

- Для нового користувача:

    - пройти реєстрацію.

Щоб система мала можливість взаємодіяти з обліковими записами Google створимо функцію loginViaGoogle().

```
loginViaGoogle() {
  this.authService.loginViaGoogle().subscribe(data => {
    const checkForDuplicate = this.listOfUsers?.filter(item => item.email === data.user.email);
    if (checkForDuplicate.length === 0) {
      this.authService.listOfUsers.push({
        email: data.user.email,
        name: data.user.displayName,
        photoURL: data.user.photoURL
      })
    }
  })
  }
}
```

Рис. 3.9. Вхід через Google

Розробимо функції управління проєктами для адміністратора, а саме створення нового проєкту та створення задач.

У функції addProject() опишемо вікно створення нового проєкту наведено на Рис. 3.10.

```
addProject() {
  this.snackBarService.openSnackBarAfterSuccess(
    this.projectService.addProject({
      ...this.newProject.value,
      team: this.teamMembers,
      createdBy: this.userEmail
    }),
    SNACKBAR_MESSAGES.update_success
  )
  }
}
```

Рис. 3.10. Створення нового проєкту

У функції addTask() опишемо вікно створення нових задач наведено на Рис. 3.11.

```
addTask() {
  const date: string = this.newTask.value.dueDate;
  const email: string = this.newTask.value.assignedTo;

  this.snackBarService.openSnackBarAfterSuccess(
    this.taskService.addTask(
      {
        ...this.newTask.value,
        createdBy: this.userEmail,
        assignedTo: email || DEFAULT_ASSIGNEE,
        dueDate: date.toString()
      }),
    SNACKBAR_MESSAGES.create_success
  );
}
```

Рис. 3.11. Створення нової задачі

Для внесення змін до створеної задачі створимо функцію editOrAddTask(), наведено на Рис. 3.12.

```
editOrAddTask() {
  if (this.isEdit) {
    this.snackBarService.openSnackBarAfterSuccess(
      this.taskService.updateTask(this.newTask.value),
      SNACKBAR_MESSAGES.update_success
    );
    this.current.setCurrentProjectOrTask('currentTask', this.current.getTaskKey())
  } else {
    this.addTask()
  }
}
}
```

Рис. 3.12. Створення нової задачі

3.3. Проектування інтерфейсу

3.3.1. Аналіз функцій інтерфейсу

Інтерфейс веб-сервісу містить єдине зовнішнє оформлення для кожної сторінки додатку, які включають в себе (палітку кольорів, структурне оформлення навігаційних елементів, форму реєстрації нових користувачів, панель задач, адміністративну частину управління).

Логічна комплектуюча веб-сервісу включає в себе розроблювані проєкти та структурний розподіл задач на команду майбутніх розробників, а також налаштування інтерфейсу для користувачів.

3.3.2. Вибір способу організації і опис реалізації інтерфейсу

Для розробки інтерфейсу веб-сервісу було обрано програмне середовище Visual Studio Code IDE. Visual Studio Code – це середовище в якому можна розробляти та редагувати вебзастосунки на таких мовах програмування як JavaScript, HTML, Java, CSS, TypeScript. Даний програмний продукт підтримується таким операційними системами як Windows, Linux і OS X в різних версіях та є безкоштовним у користуванні, що дає можливість залучати більше користувачів.

Розроблювальним продуктом є інтерфейс веб-сервісу. На сьогоднішній день більшість розробників використовують HTML із застосуванням CSS і JavaScript. В даній веб-сервісі буде застосовано саме цей набір технологій для розробки інтерфейсу, а також додатково фреймворки JavaScript, такі як ExpressJS та AngularJS.

Малюнки, які будуть відображені нижче, показують архітектуру веб-сервісу, взаємодію користувача з інтерфейсом(внесення персональних даних, редагування та в кінцевому варіанті відображення на екрані пристрою).

Даний веб-сервіс дозволить користувачам слідкувати за проектними роботами до який вони залучені, отримувати задачі та редагувати статус робіт по мірі виконання. Веб-сервіс буде доступний та легкий для розуміння кожній людині, тобто для вивчення функціоналу не потрібно буде залучати спеціалістів.

Головна сторінка для входу в систему, якщо у випадку не зареєстрованого користувача, Рис. 3.13.

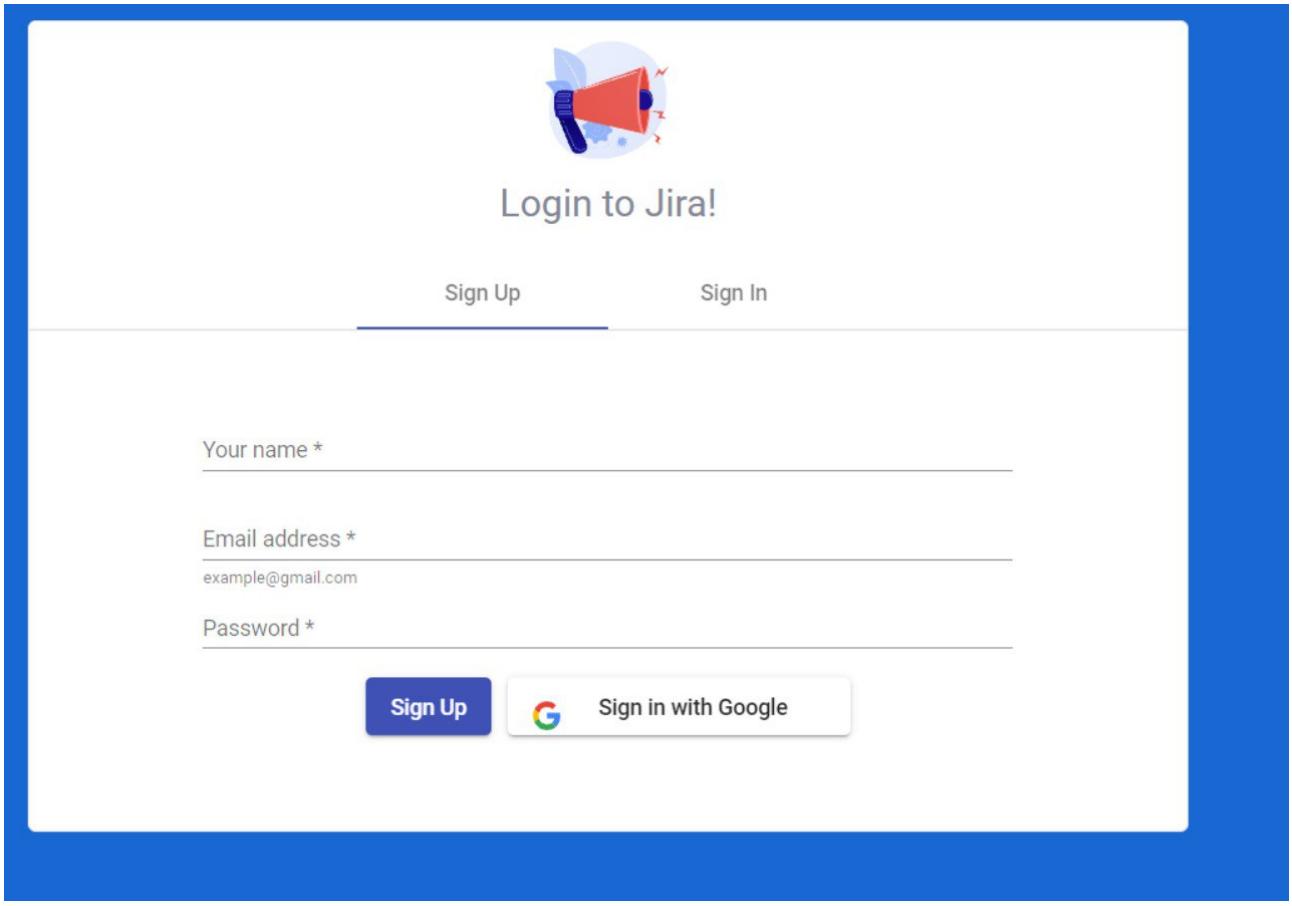

Рис. 3.13. Основна сторінка, ящо користувач не зареєстрований в системі

Зареєстрований користувач відповідно може зайти в обліковий запис заповнивши поля "Email" та "Password" чи скориставшись існуючим обліковим записом в мережі Google.

Користувач системи може бачити інформацію по створеним проєктам до яких він безпосередньо доданий, список задач, які призначені на нього.

Сторінка "Your project" містить перелік проєктів до яких залучена відповідна особа, а також назва проєкту, примітки до проєкту, клавіши для переходу до таблиці прогресу виконання проєкту та команда, яка відповідальна за опрацювання проєкту.

Основна сторінка після авторизації в системі показано на Рис. 3.14.

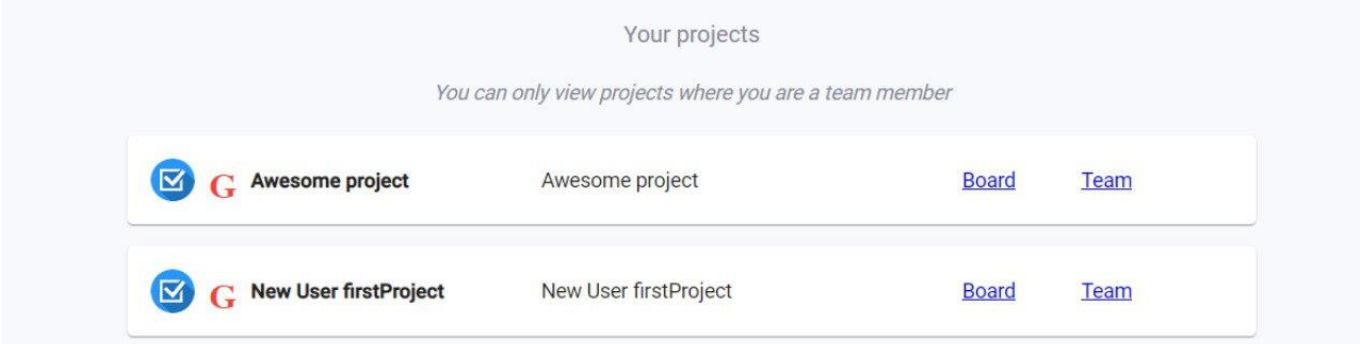

Рис. 3.14. Основна сторінка, ящо користувач зареєстрований в системі

Сторінка "Your task" містить перелік задач призначені для члена команди, а також назва задачі, примітки до задачі, клавіши для переходу до таблиці прогресу виконання задачі, термін здачі завдання, на кого призначена задача. статус завдання, тип завдання.

Сторінка перегляду отриманих задач наведено на Рис. 3.15.

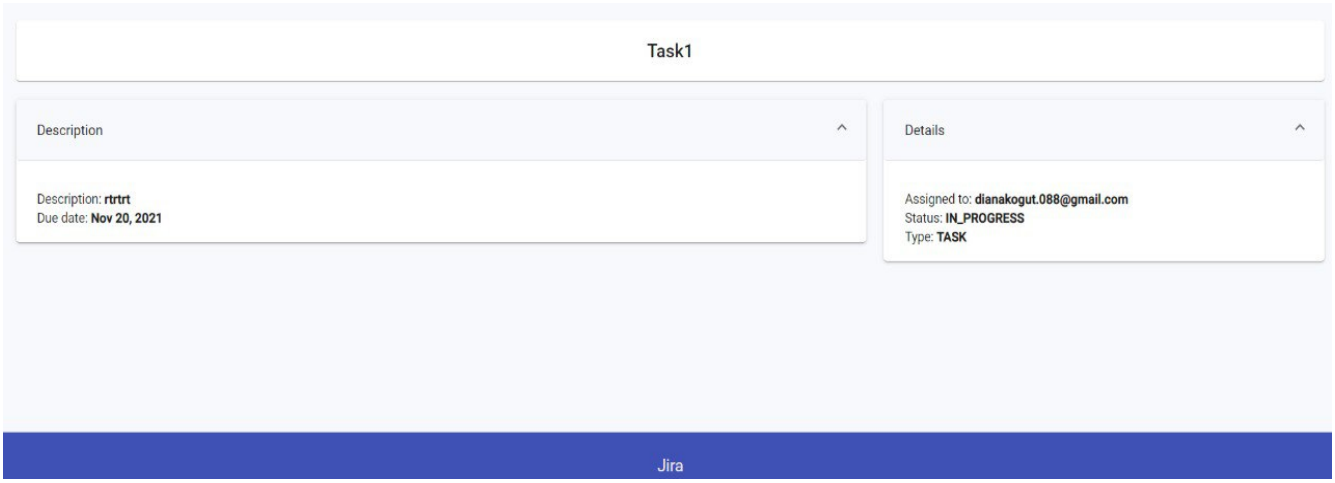

Рис. 3.15. Отримані завдання для працівника

Обліковий запис адміністратора містить більший функціонал ніж звичайний користувач. До цього функціоналу відноситься створення проєкту, його управління та створення задач.

Основні сторінки, які відкриваються при вході до облікового запису адміністратора показано на Рис. 3.16 та Рис. 3.17.

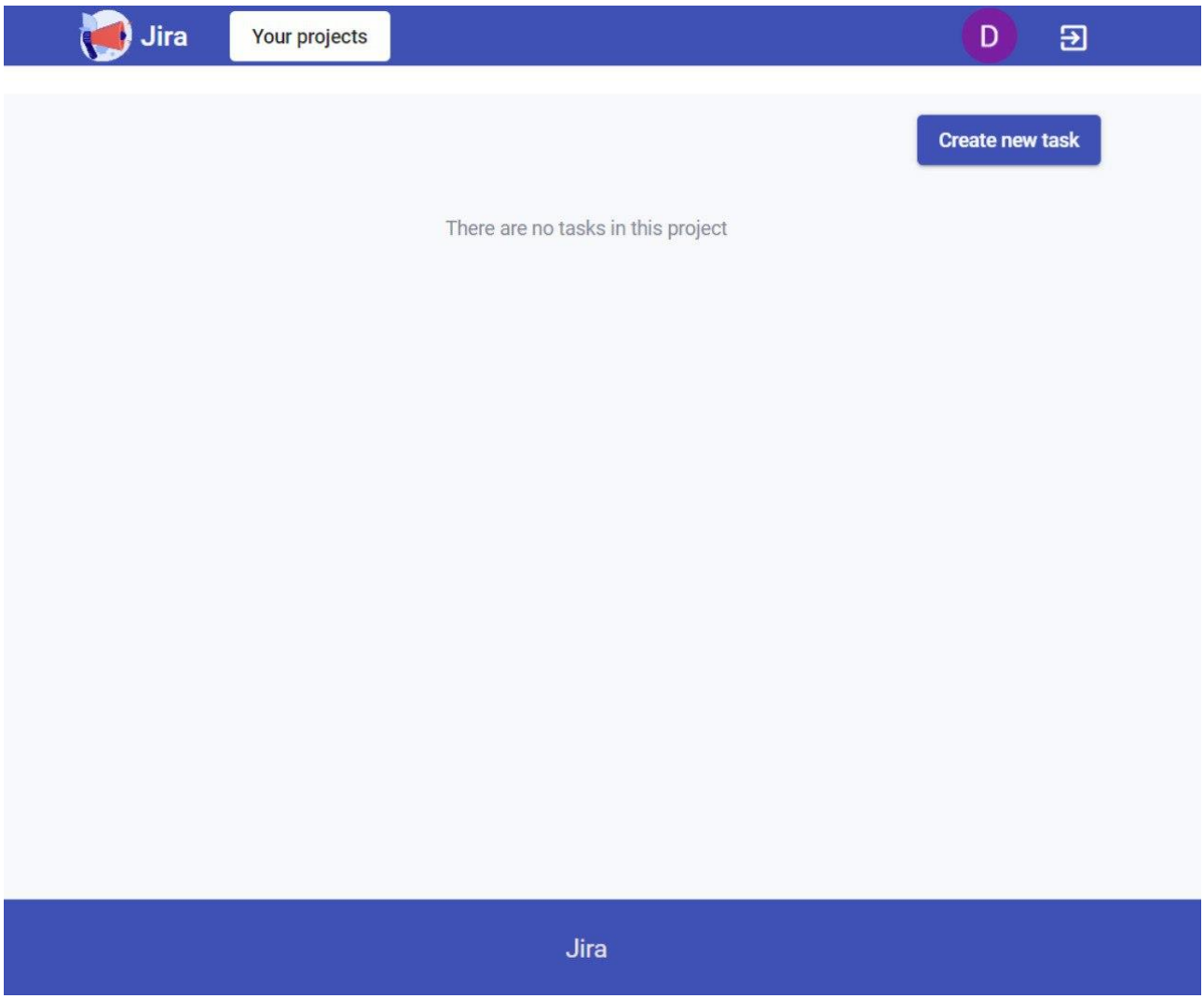

Рис. 3.16. Сторінка відображення задач по проєктам

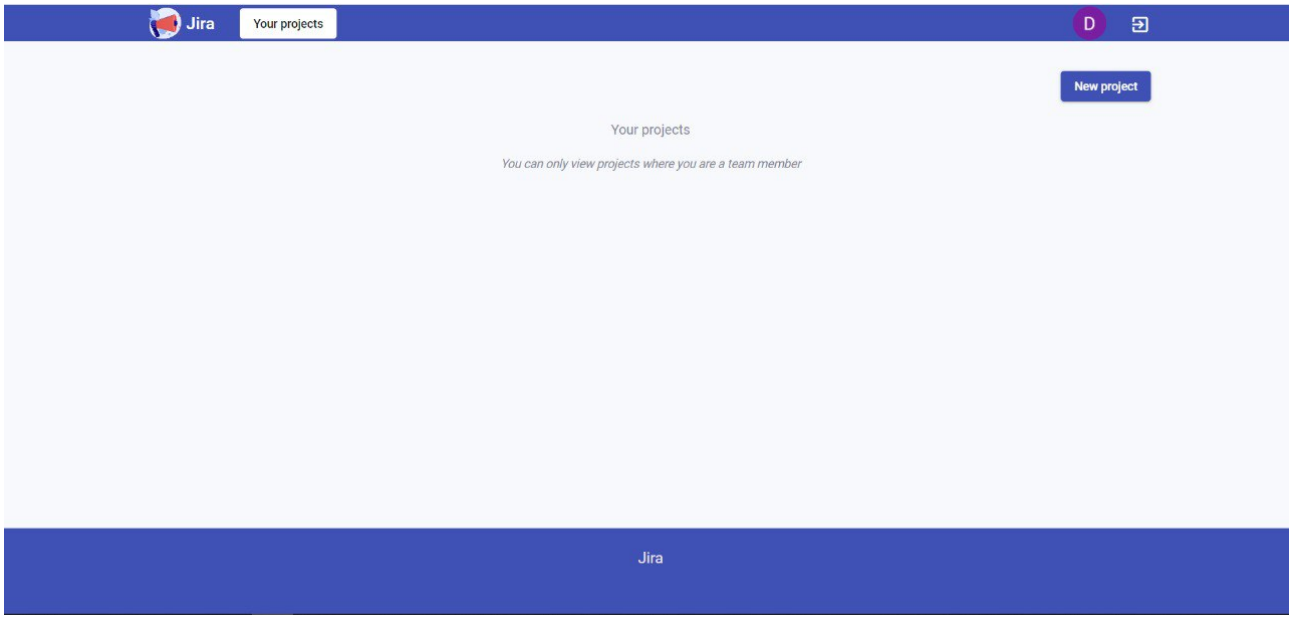

Рис. 3.17. Сторінка відображення проєктів

Адміністратор може створити новий проєкт у вікні "Create new project" та нову задачу у вікні "Create new task".

При створенні нового проєкту у вікні "Create new project" адміністратор має заповнити всі поля позначеня знаком "*", до цим полів відносяться такі як: назва(заголовок) проєкту, призначення(визначення) проєкту. Якщо дані поля не будуть заповнені у вікні "Create new project" відтвориться попереджувальне повідомлення: "Enter all required fields" та попередньо внесені дані будуть стерті (Рис. 3.18).

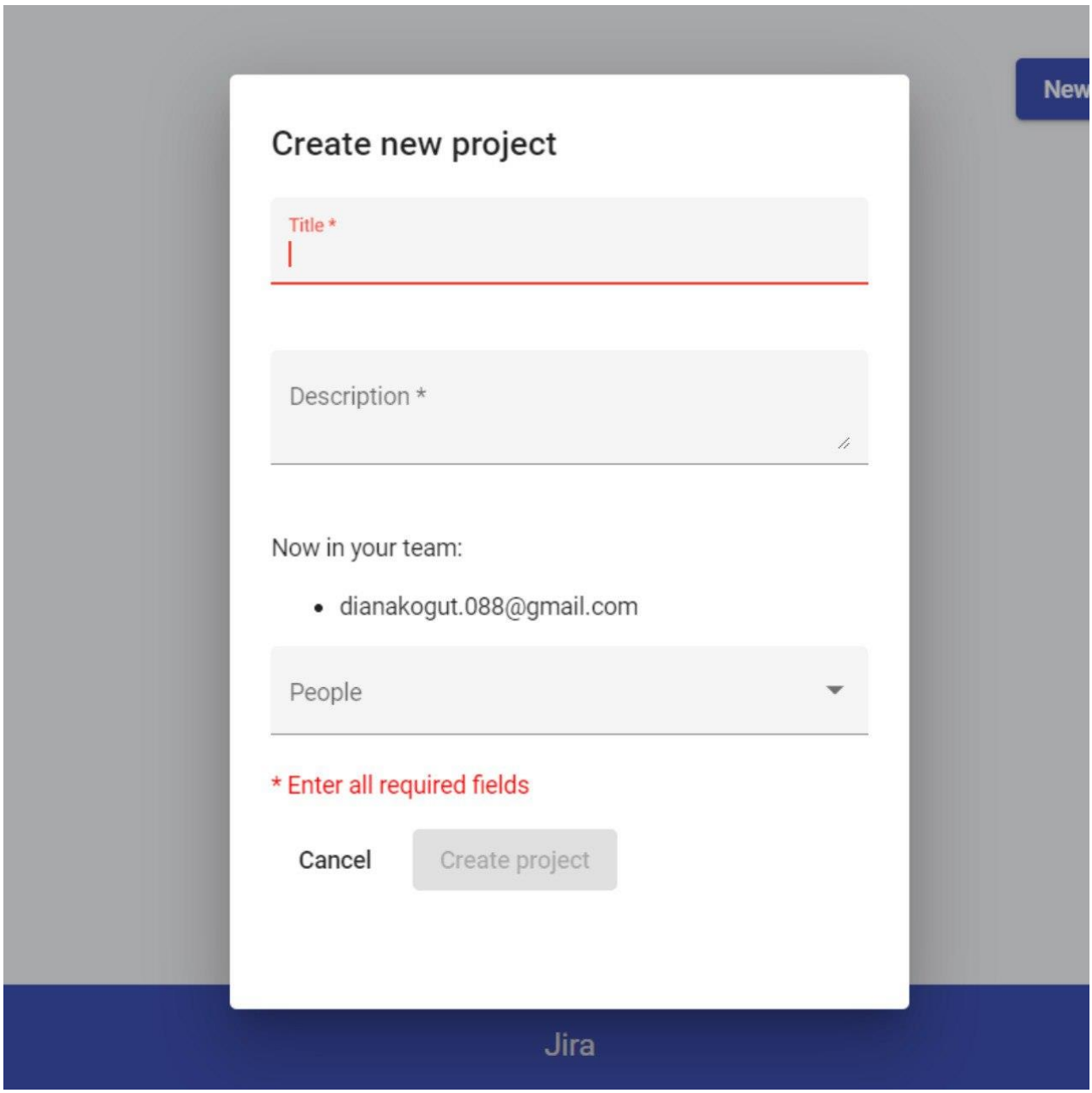

Рис. 3.18. Верифікація форми створення проєкту

Заповнивши дані у потрібні поля адміністратор зможе створити проєкт натиснувши на підсвічену кнопку "Create project".

При створенні нової задачі у вікні "Create new task" адміністратор має можливість обрати тип завдання (Рис. 3.19).

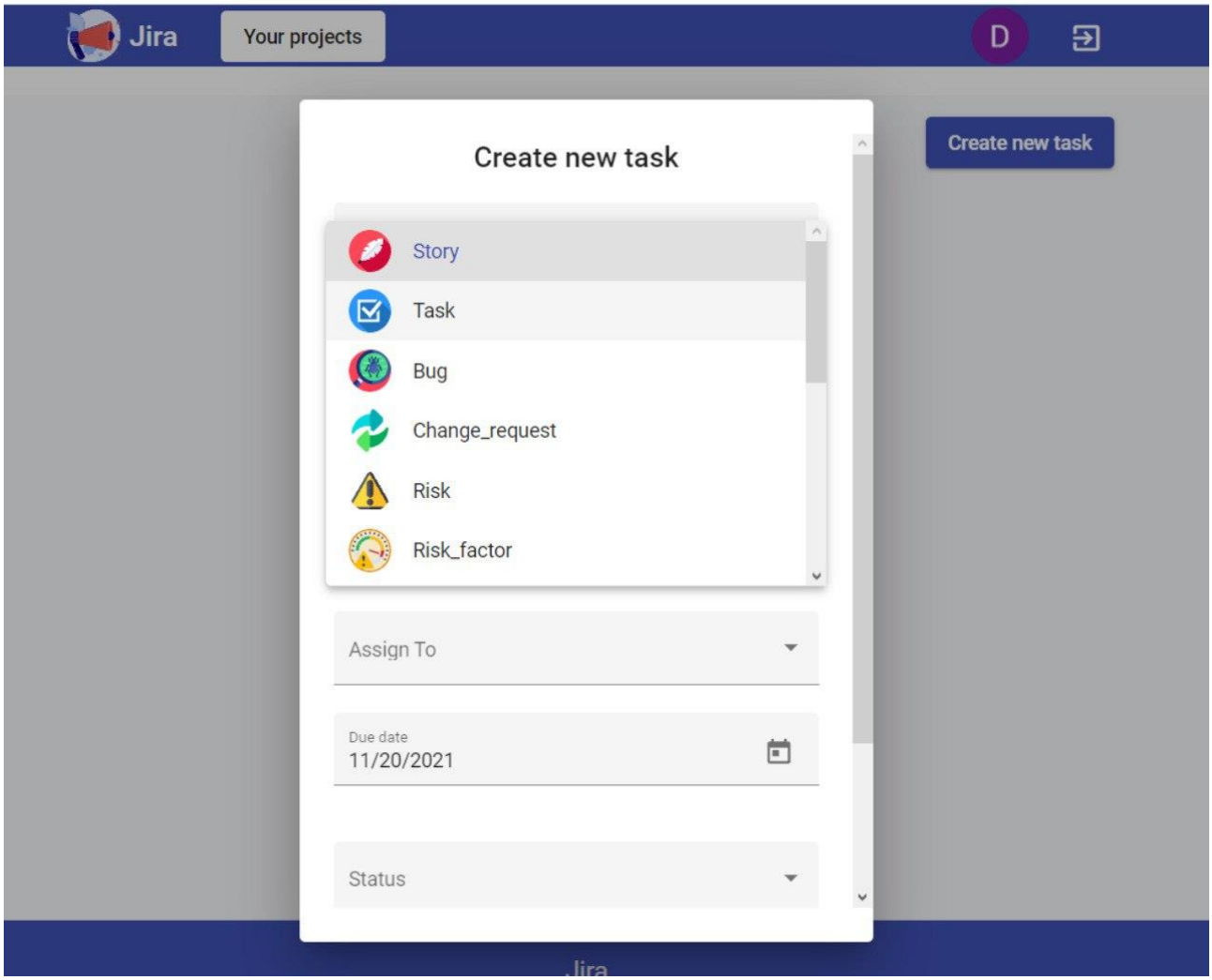

Рис. 3.19. Вибір типів завдань

Заповнюючи поля для нової задачі адміністратор має внести дані в такі поля як тип завдання, назва завдання, примітки до завдання, призначення на відповідну особу, дата виконання, статус завдання після чого може створити завдання (Рис. 3.20).

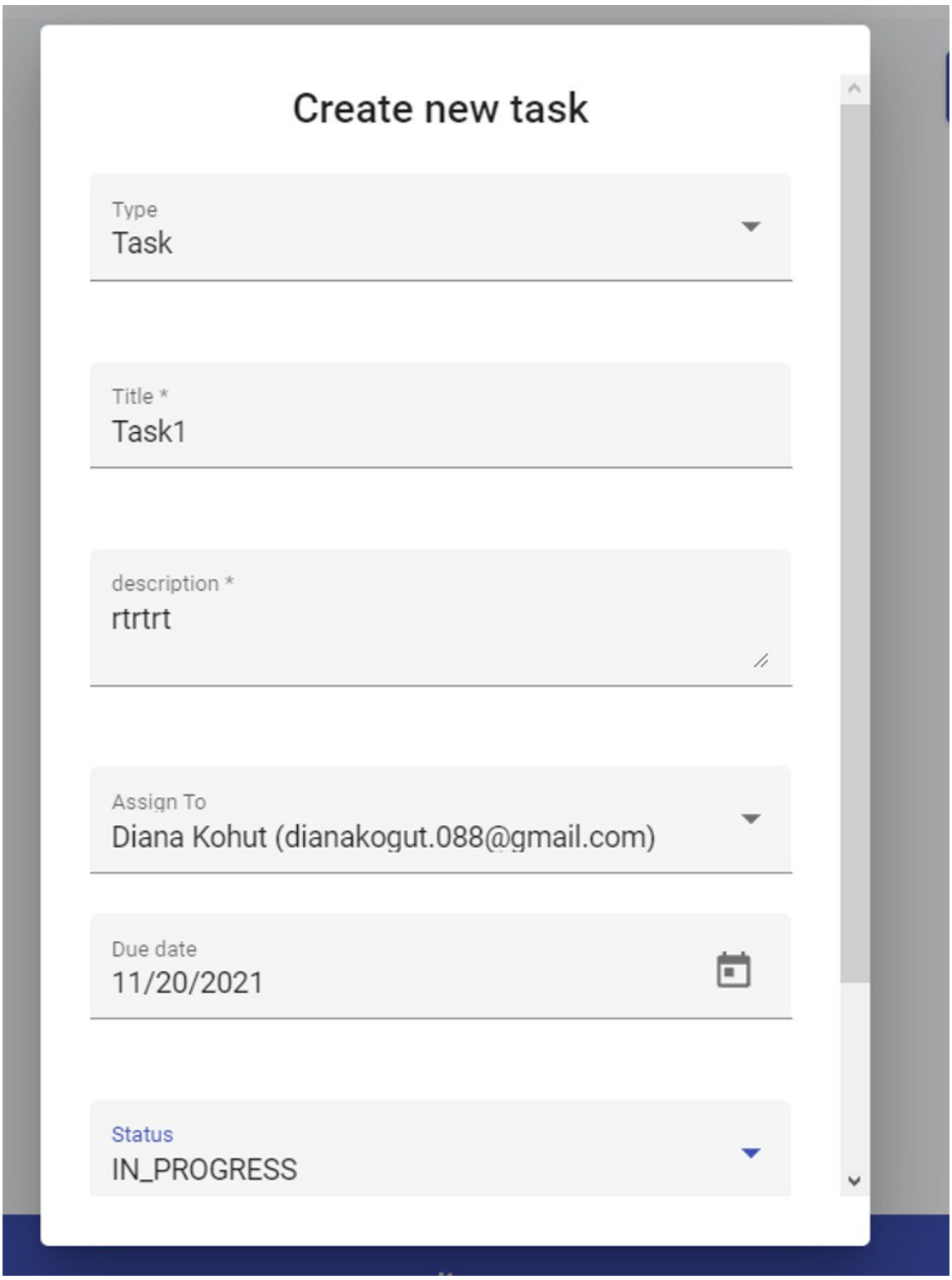

Рис. 3.20. Заповнена форма для нового завдання

На сторінці "Your project" відображається запущенні проєкти, зміни до яких керівник проєкту може внести навіть після розділення задач або ж видалити проєкт в цілому (Рис. 3.21).

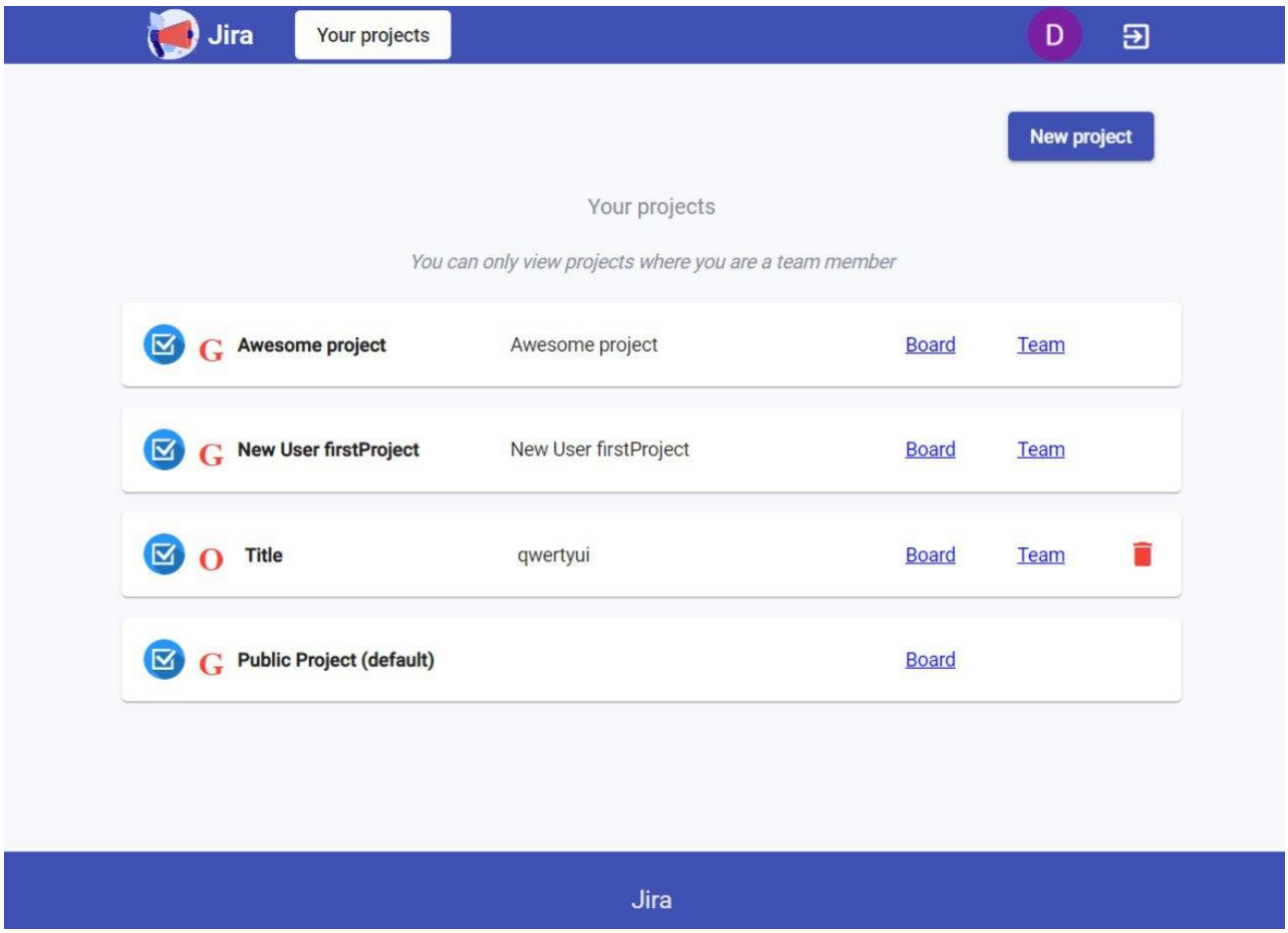

Рис. 3.21. Перелік створених проєктів

Також на сторінці "Your project" можна переглянути створені завдання, вносити зміни, видаляти вже створені задачі та повертатися до переліку проєктів (Рис. 3.22).

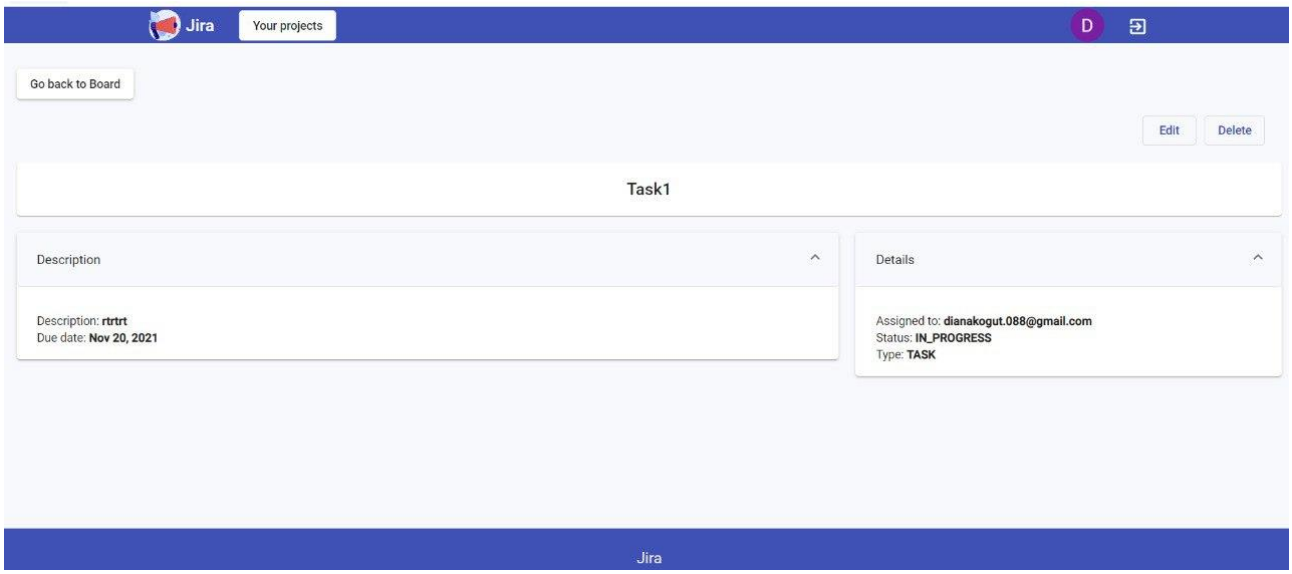

Рис. 3.22. Сторінка створеної задачі

Після додавання/коригування/видалення записів на головній панелі задач зміни застосовуються після оновлення сторінки. Вже оновлена сторінка відображає актуальні записи на сторінці. При додаванні/коригуванні/видаленні записів після кожної виконаної дії відображається модальне вікно в нижній частині сторінки посередині з надписом "Successfully updated", яке можна закрити натиснувши "Close" (Рис. 3.23).

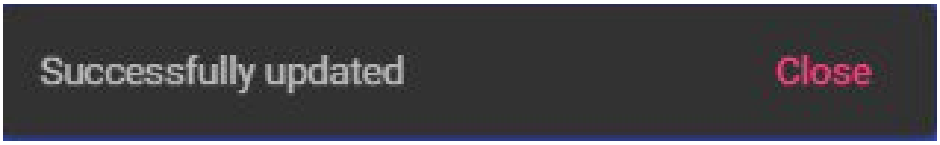

Рис. 3.23. Модальне вікно відповіді

Адміністратор може контролювати процес виконання проєкту, статус виконання відображається у вигляді колонок, які поділяєються на :

- Backlog(Резерви/Невиконані проєкти);
- To do(Можна виконувати);

- In progress(В процесі);
- Ready for testing(Готовий до тестування);
- Done(Виконано).

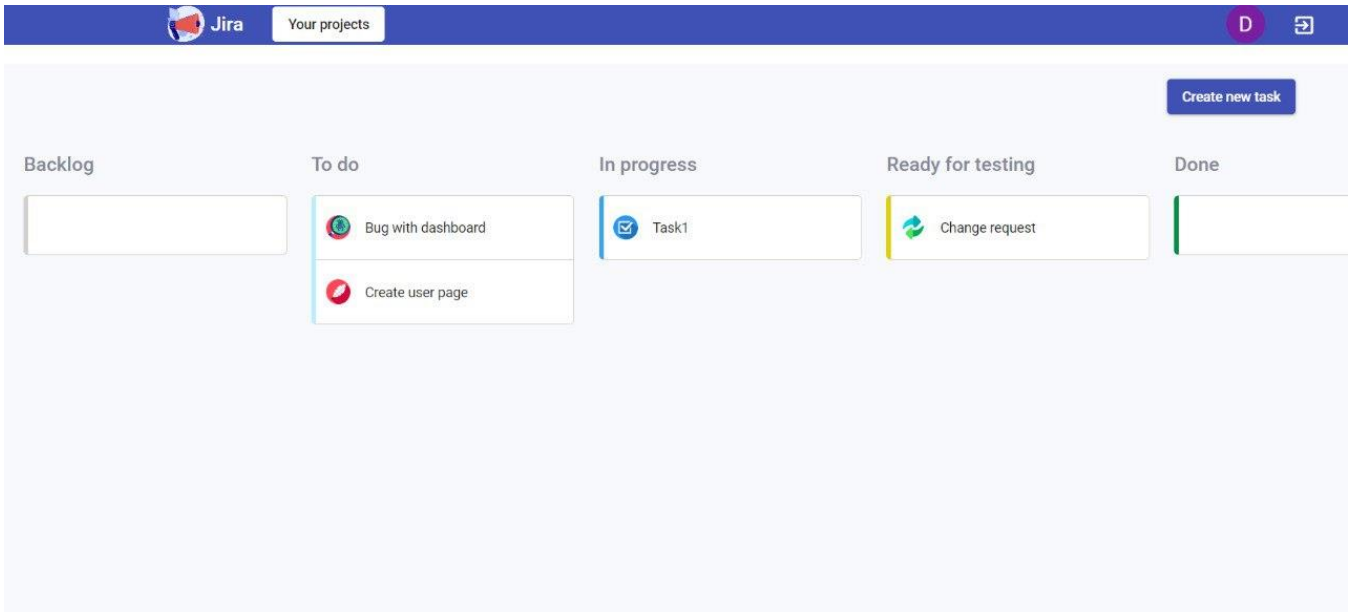

Рис. 3.24. Статус проєкту

ВИСНОВОК ДО РОЗДІЛУ 3

В даному розділу описано реалізація програмного продукту у вигляді веб-сервісу. Для розгортання веб-сервісу було застосовано безліч інструментів та технологій, а саме гіпертекстову мову розмітку HTML, каскадна таблиця стилів CSS, фреймворки: AngularJS, ExpressJS, хмарну базу даних Firebase, середовище розробки Visual Studio Code. Задачі, що були поставлені для реалізації в проекті були втілені у веб-сервісі завдяки аналізу користувацьких вимог.

Веб-сервіс для управління та планування проєктів орієнтований на всі сфери діяльності в яких потрібне використання обліка виконання справ, розробки проєктів, які потребують розподілу та контролю задач. Даний веб-сервіс дає можливість продуктивно виконувати як маштабні так і маленькі проєкти. Програмний продукт є доступний та легкий в налаштуванні, має простий та приємний для очей інтерфейс, який підійде для різних категорій користувачів.

ВИСНОВКИ

У ході виконання дипломної роботи був розроблений веб-сервіс для управління та планування проєктами. Система має два інтерфейси - користувача та адміністратора. Кожен інтерфейс має окремі функції та властивості. Панель адміністратора наділена такими функціями як створення нового проєкту, видалення проєкту, додавання/корегування/видалення задачі, контроль виконуваності завдань.

Панель користувача має такі функції як перегляд призначених завдань/проєктів, коригування статусу при виконанні завдання, перегляд особистих даних, коригування персональних даних. Всі дані по існуючим користувачам системи зберігаються у хмарній базі даних – Firebase, яка працює в реальному часі та кожен підключений користувач невід'ємно взаємодіє з нею за допомогою своїх девайсів. Головою перевагою бази даних є зберігання даних в хмарному середовищі, що забезпечує безпеку даних системи від втрати у випадку виходу з ладу операційного забезпечення пристроїв зберігання бази даних.

У процесі виконання дипломної роботи проведений аналіз різноманітних технологій та інструментів для розгортання веб-сервісу. В результаті було обрано інструменти, що мають гнучкий функціонал та легкі в налаштуванні.

В результаті виконання дипломної роботи реалізовані наступні цілі:

- за допомогою Visual Studio Code було розроблено інтерфейс середовища - адміністратора і користувача;
- за допомогою Angular було розроблено основні обробники подій;
- за допомогою Angular було реалізовано розмежування функціоналу для адміністратора і користувача.

Веб-сервіс був відлагоджений, сервіс добре функціонує та може бути застосований в багатьох сферах діяльності. Наприклад, в сфері інформаційних технологій використовуватися для розробки програмних продуктів; організовувати тімбілдінги,

свята - розподіляючи кожен клаптик роботи на відповідного працівника. Чи технічна підтримка банку - кожному працівникові розподіляються вхідні дзвінки відповідно до профільованості спеціаліста.

Даний проект в подальшому буде розвиватися та покращуватися, відповідно до потреб замовників інтегрувати фінкціонал, покрашувати візуалізацію, впроваджувати нові технології та інстументи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Найкращі сервіси для управління проектами. – режим доступу: https://bestwebsoft.com.ua/bestproject-management-software/?__cf_chl_managed_tk__=pmd_UY9pu3lXk3vJzio._ucaK6HVKc3Apauz3T3rtujEO5Q-1631728475-0-gqNtZGzNAtCjcnBszRE9
2. Jira. – режим доступу: https://ua.softlist.com.ua/articles/chto-takoe-jira/
3. Аналіз сучасних систем управління проектами. – режим доступу: http://apgs.kdu.edu.ua/statti/ANALYSIS%20OF%20MODERN%20PROJECT%20MANAGEMENT%20SYSTEMS.doc
4. Системи управління проектами. – режим доступу: https://xn--90aamhd6acpq0s.xn--j1amh/teoriya/sistemi-upravl-nnya-proektami/
5. Поняття веб-сервіс. – режим доступу: https://semantica.in/blog/chto-takoe-veb-servis.html
6. Веб-сервіси. – режим доступу: https://yiiframework.com.ua/uk/doc/guide/topics.webservice/
7. Веб-сервіси. – режим доступу: https://www.seonews.ru/glossary/veb-servis/
8. Підходи до композиції сервісів в семантичному web-середовищі. – режим доступу: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/290/%D0%94%D0%B5%D1%80%D0%B5%D1%86%D0%BA%D0%B8%D0%B9_%231.pdf?sequence=1
9. Переваги та недоліки веб-сервісів. – режим доступу: https://passportbdd.ru/windows/kakie-byvayut-web-services-c-chto-takoe-veb-servis-kak-eto-vse-rabotaet/
10. HTML. – режим доступу: https://developer.mozilla.org/ru/docs/Web/HTML
11. БД Firebase-. – режим доступу: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0
12. Node.js і Express. – режим доступу: https://habr.com/ru/company/piter/blog/265649/

13. AngularJS. – режим доступу: https://metanit.com/web/angular/1.1.php
14. Плюсы и минусы разработки веб-приложений на Node.js. – режим доступу: https://codernet.ru/articles/web/plyusyi_i_minusyi_razrabotki_veb-prilozhenij_na_nodejs/