МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____

“____”_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСНИКА ОСВІТНЬОГО СТУПЕНЯ
“МАГІСТР”

Тема: “Застосунок системи організації управління робіт ФОП”

Виконавець: Поліщук Костянтин Аркадійович

Керівник: д.т.н., професор Пархомей Ігор Ростиславович

Нормоконтролер: к.ф.-м.н. Гололобов Дмитро Олександрович

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра інженерії програмного забезпечення
Освітній ступінь магістр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Програмне забезпечення систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
"___" _____ 2023 р

ЗАВДАННЯ
на виконання дипломного проекту студента
Поліщука Костянтина Аркадійовича

1. Тема проекту: «Застосунок системи організації управління робіт ФОП» затверджена наказом ректора від 29.09.2023 р. №1994/ст.

2. Термін виконання проекту: з 02.10.2023 р. до 31.12.2023 р.

3. Вихідні данні до роботи: програмний продукт розробити мовою JavaScript з використанням HTML, CSS та Node.js.

4. Зміст пояснювальної записки:
    1. Аналіз предметної області
    2. Опис вимог до системи
    3. Структура системи
    4. Реалізація системи

5. Перелік обов'язкових слайдів презентації:
    1. Актуальність теми.
    2. Потенційні користувачі.
    3. Вимоги до системи.
    4. Архітектура системи.
    5. Робота системи.
    6. Майбутній розвиток.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1.	Розробка плану роботи, назви розділів ПЗ та затвердження їх керівником	02.10.2023 – 08.10.2023	
2.	Підготовка та написання 1 розділу. Відсилка керівнику	09.10.2023– 15.10.2023	
3.	Підготовка та написання 2 розділу. Відсилка керівнику	16.10.2023– 22.10.2023	
4.	Підготовка та написання 3 розділу. Відсилка керівнику	23.10.2023– 29.10.2023	
5.	Підготовка та написання 4 розділу. Відсилка керівнику	30.10.2023– 05.11.2023	
6.	Редагування та друк пояснювальної записки, графічного матеріалу	06.11.2023– 19.11.2023	
7.	Загальне редагування та друк пояснювальної записки, графічного матеріалу. Отримання відгуку керівника, рецензії та заповнення листа про доброчесність. Розробка доповіді	20.11.2023– 03.12.2023	
8.	Завершення написання ПЗ. Проходження нормо-контролю. Друк ПЗ. Отримання відгуку керівника. Підготовка презентації та доповіді на перед захист.	04.12.2023– 10.12.2023	
9.	Передзахист каліф. Роботи. Отримання рецензії	11.12.2023– 17.12.2023	
10.	Здати секретарю ДЕК: ПЗ, ГМ, CD-R з електронними версіями ПЗ, ГМ, презентацію, відгук керівника, рецензію, довідку про успішність, 2 папки, 2 конверта)	18.12.2023– 24.12.2023	
11.	Захист дипломної роботи перед ЕК.	25.12.2023– 31.12.2023	

Дата видачі завдання: 02.10.2023 р.

Керівник дипломної роботи: д.т.н. професор Ігор ПАРХОМЕЙ

Завдання прийняв до виконання: Костянтин ПОЛІЩУК

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Застосунок системи організації управління робіт ФОП»: 106 с., 14 рис., 0 табл., 28 інформаційних джерел.

УПРАВЛІННЯ РОБІТ, ОРГАНІЗАЦІЙНІ СЕРВІСИ, ПОДАТКИ, ЗВІТНІСТЬ, ПОСЛУГИ, ПІДПРИЄМЕЦЬ

Об’єкт дослідження – системи організації управління робіт ФОП.

Мета дипломної роботи – полегшення роботи ФОП що може служити для ефективного планування, оптимізації використання ресурсів та забезпечення фінансового контролю. Окрім того, вона сприяє покращенню комунікації та відповідності законодавчим вимогам, сприяючи підвищенню продуктивності та результативності бізнесу ФОП.

Метод дослідження – використання комбінації аналітичних та емпіричних методів. Аналіз існуючих практик та літературних джерел дозволить оцінити ефективність систем управління. Результати дослідження і аналіз переваг та недоліків можуть слугувати основою для рекомендацій щодо оптимізації та покращення подібних систем.

Під час виконання проекту був проведений глибокий аналіз сучасних архітектурних підходів до розробки програмного забезпечення, і як результат, було розроблено програмний продукт для системи організації управління роботою фізичних осіб-підприємців.

Результати цієї роботи можуть бути використані при розробці програмного забезпечення для системи управління роботою фізичних осіб-підприємців.

Процес розробки та дослідження виконувався під управлінням операційної системи Windows, а програма розроблялася в середовищі Visual Studio Code з використанням мови програмування JavaScript.

.

ABSTRACT

Explanatory note to the thesis " Application of the system of organizations of management performance works ": 106 pp., 14 figures, 0 tables, 28 information sources.

WORK MANAGEMENT, ORGANIZATIONAL SERVICES, TAXES, REPORTING, SERVICES, INDIVIDUAL PROPRIETOR

The object of research is the system of organization of the management of works of the FOP.

The purpose of thesis is to facilitate the work of the FOP, which can serve for effective planning, optimization of resource use, and financial control. In addition, it contributes to the improvement of communication and compliance with legal requirements, helping to increase the productivity and effectiveness of the business of the FOP.

Research method is the use of a combination of analytical and empirical methods. Analysis of existing practices and literary sources will allow to assess the effectiveness of management systems. Research results and analysis of advantages and disadvantages can serve as a basis for recommendations on optimization and improvement of such systems.

During the implementation of the project, an in-depth analysis of modern architectural approaches to software development was carried out, and as a result, a software product was developed for the system of organizing the work of individual entrepreneurs.

The results of this work can be used in the development of software for the work management system of individual entrepreneurs.

The development and research process was carried out under the control of the Windows operating system, and the program was developed in the Visual Studio Code environment using the JavaScript programming language.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

ФОП – Фізична особа-підприємець

API – Application Programming Interface

DOM - Document Object Model

IDE – Integrated Development Environment

UI – User Interface

UX – User Experience

SQL – Structured Query Language

CRM – Customer Relationship Management

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

URL – Uniform Resource Locator

ВСТУП

У сучасному економічному просторі, де фізичні особи-підприємці стають каталізаторами інновацій та розвитку, важливо докладати зусиль для створення оптимальних умов для їхнього успішного функціонування. Дипломна робота "Організація управління робіт ФОП" визначає основні виклики, перед якими стоїть цей клас підприємців, та спрямована на розробку практичних рішень, що сприяють підвищенню ефективності їхньої діяльності.

У глибокому аналізі сучасних тенденцій управління роботою ФОП, наша робота визначає ключові напрямки, які варто враховувати при розробці та впровадженні систем управління. Взаємодія з питаннями фінансового контролю, планування робіт та використання ресурсів стають основними фокусами для досягнення оптимальних результатів у сучасному підприємницькому середовищі.

У цій роботі ми звертаємо увагу на ключові аспекти систем управління, які можуть допомогти ФОП у вирішенні викликів сучасного бізнесу. Ми розглядаємо питання планування робіт, розподілу ресурсів, ведення фінансового обліку та інші аспекти, які визначають ефективність управління роботою фізичних осіб-підприємців.

Дипломна робота містить не лише огляд теоретичних аспектів управління, але й конкретні рекомендації та інструменти, які можуть стати цінним внеском для практичного застосування в сфері діяльності фізичних осіб-підприємців. Наша мета — надати підприємцям конкретні інструменти та підходи, які підтримають їхню успішну діяльність, сприяючи розвитку та стабільності в умовах сучасного ринкового середовища.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Уявімо умовного підприємця, який володіє сервісним центром та отримав велике замовлення від місцевої компанії на обслуговування їхнього парку комп'ютерів. Замовлення було великим кроком вперед для його бізнесу, але також стало випробуванням його систем управління роботою.

Швидко зрозумівши, що відсутність ефективної організації управління роботою ФОП ускладнює розподіл завдань та контроль над технічним обслуговуванням. Потрібно було визначити, який технік працює над кожним комп'ютером, відстежувати терміни виконання та мати чіткий облік витрат.

Впровадження системи організації управління роботою дозволило ефективно розподіляти завдання серед свого персоналу, відстежувати кожен етап робіт та автоматизувати облік використаних матеріалів. Після цього стало не лише зручно виконувати замовлення вчасно, але й оптимізувати робочий час, зменшуючи можливі помилки та покращуючи задоволення клієнтів.

Завдяки цьому, підприємець не тільки успішно може впоратися з великим замовленням, але й створити стійку основу для подальшого розвитку свого бізнесу.

1.1 Актуальність дослідження

Актуальність дослідження організації управління робіт ФОП визначається необхідністю вдосконалення та оптимізації бізнес-процесів в умовах сучасного підприємницького середовища. Дослідження організації управління робіт ФОП актуальне не лише як відповідь на сучасні виклики, але й як важлива передумова для їхнього стійкого успіху та розвитку в умовах сучасного підприємницького середовища.

Актуальність досліджень насамперед виявляється через кілька ключових аспектів:

1. Зростання кількості ФОП. У багатьох країнах кількість фізичних осіб-підприємців зростає. ФОП мають можливість швидко реагувати на ринкові зміни та запроваджувати нові бізнес-ідеї без великих адміністративних обмежень. Статус ФОП може стати альтернативою традиційному зайняттю на постійній роботі. Це може бути актуальним у ситуаціях, коли споживачі шукають додатковий дохід або можливість самостійно керувати своєю кар'єрою. У періоди економічної нестабільності та кризи багато людей можуть обирати статус ФОП як спосіб забезпечення власного доходу та фінансової стабільності. Зростання кількості ФОП може сприяти формуванню нових ринків та галузей, особливо у сферах, де інновації та нові ідеї швидко втілюються в життя. Ця тенденція має значний вплив на економіку, ринок праці та бізнес-середовище і вимагає розвитку та адаптації інструментів та послуг для підтримки ФОП у їхній діяльності. [8]

2. Потреба в ефективному управлінні завданнями. ФОП повинні ефективно планувати та виконувати роботи, вести облік фінансів та контролювати робочий процес. Системи управління завданнями та проектами можуть значно спростити цей процес.

3. Розвиток інформаційних технологій. Завдяки росту технологічних можливостей і доступності мобільних додатків, ФОП мають доступ до широкого спектру інструментів для автоматизації і оптимізації своєї роботи. Це відкриває нові можливості для управління роботами. Завдяки Інтернету та соціальним мережам, ФОП можуть легко створити цифрову присутність для свого бізнесу, яка дозволяє залучати клієнтів, вести маркетингову кампанію та збільшувати обсяги продажів. ФОП можуть легко створювати та управляти онлайн-магазинами, що дозволяє їм продавати товари та послуги через Інтернет та розширювати аудиторію клієнтів. Використання хмарних технологій дозволяє зберігати дані та документацію в безпечному та доступному місці, а

також отримувати доступ до програм та ресурсів без необхідності власного інфраструктурного обладнання. Розробка та використання мобільних додатків дозволяє покращити комунікацію з клієнтами, спростити процеси обліку та управління роботами. Використання IoT-технологій може допомогти в моніторингу та управлінні об’єктами чи процесами, що стосуються їхньої діяльності (наприклад, в галузі логістики або сільського господарства). Використання аналітики та інструментів штучного інтелекту допомагає аналізувати дані, прогнозувати попит, оптимізувати процеси та приймати кращі управлінські рішення. У зв’язку з розвитком IT, багато ФОП можуть виконувати свою роботу дистанційно, що дозволяє залучати фахівців з усього світу та оптимізувати робочі процеси.

4. Конкуренція. У бізнес-середовищі конкуренція постійно зростає. ФОП, які володіють ефективними інструментами для управління роботами, можуть бути більш конкурентоспроможними та здатними задовольняти вимоги своїх клієнтів.

5. Податкова та бухгалтерська звітність. ФОП повинні вести облік доходів і витрат, а також подавати звіти до податкових органів. Ефективні системи управління можуть спростити цей процес та забезпечити точність податкової та бухгалтерської звітності.

6. Цифрова трансформація. Використання інформаційних технологій для оптимізації бізнес-процесів стає необхідністю в епоху цифрової трансформації. ФОП не можуть відстати від цього тренду.

Таким чином, дослідження та розробка систем організації управління виконанням робіт ФОП є актуальними та важливими завданнями, оскільки вони сприяють підвищенню ефективності діяльності, їхній конкурентоспроможності та розвитку, а також спрощують багато процесів, пов’язаних з підприємництвом.

1.2 Аналіз предметної області

Аналіз предметної області щодо застосунку системи організації управління виконанням робіт фізичних осіб-підприємців (ФОП) може бути розширений та розглянутий з різних точок зору. Ось деякі ключові аспекти для дослідження:

1. Стан ринку ФОП. Розглянемо стан ринку на прикладі України. "Державна податкова служба навела статистику щодо кількості зареєстрованих фізичних осіб - підприємців та стану сплати ними податків до державного бюджету." [6]

Так, статистичні дані свідчить, що малий бізнес працює, а кількість зареєстрованих ФОП в Україні за останні 5 років коливається в межах 2 млн осіб. Станом на 01.06.2023 року кількість ФОП, які перебувають на обліку в органах ДПС складала 2008,2 тис. осіб; у 2022 році їх було 1975,7 тис. осіб; у 2021 році - 1974,4 тис. осіб; у 2020 році - 1901,2 тис. осіб; у 2019 році - 1885,9 тис. осіб. Платниками єдиного податку, при цьому, є 76% від загальної кількості зареєстрованих підприємців є. Податковим кодексом визначені річні ліміти доходу, що прив’язані до розміру мінімальної зарплати, які має кожна з груп фізичних осіб-підприємців. Отже, ліміти доходу для:

І групи – 1 млн 118 тис. 900 грн (тобто 167 розмірів мінімальної зарплати);

ІІ групи – 5 млн 587 тис. 800 грн (834 розміри мінімальної зарплати);

ІІІ групи – 7 млн 818 тис. 900 грн (1167 розмірів мінімальної зарплати).

Підприємець буде переведений на загальну систему оподаткування у разі перевищення цих лімітів, що може призвести до збільшення податків.

2. Інструменти та підходи для управління виконанням робіт ФОП. Фізичні особи-підприємці (ФОП) використовують різні інструменти, методи та системи організації для планування, виконання та контролю своїх завдань і проектів. Наприклад:

- Методика SMART. SMART є абревіатурою від Specific (конкретний), Measurable (вимірюваний), Achievable (досяжний), Relevant (відповідний або той, що підходить) і Time-bound (обмежений в часі). Вона допомагає ставити конкретні та реальні цілі і завдання.

- Планування Ганта. Діаграма Ганта (Gantt chart) використовується для графічного відображення завдань та строків їх виконання. Вона допомагає визначити послідовність та тривалість робіт.

- Методології управління проектами. ФОП можуть використовувати методології, такі як Agile, Scrum, Kanban або Waterfall [7], для кращого організації та керування проектами.

Рис. 1.1. Життєвий цикл проекту по Scrum

- Електронні таблиці. Програми для створення електронних таблиць, наприклад, Microsoft Excel або Google Sheets, використовуються для створення списків завдань, графіків та обліку ресурсів.
- Системи управління завданнями. Платформи, такі як Asana, Trello, Todoist, Wrike тощо, допомагають ФОП створювати, призначати, відстежувати та спільно працювати над завданнями та проектами.

- Системи управління відносинами з клієнтами (CRM). Вони дозволяють вести облік клієнтів, історії взаємодії, робити планування контактів та робити маркетинговий аналіз.
- Електронні засоби комунікації. Використання електронної пошти, месенджерів, відеоконференцій та спеціалізованих комунікаційних інструментів для взаємодії з клієнтами, партнерами та співробітниками.
- Системи бухгалтерського обліку. Використання програм для обліку доходів, витрат, виписування рахунків та оподаткування.
- Електронні підписи. Використання електронного підпису для підпису договорів та документів в електронній формі.
- Аналітика і звіти. Використання інструментів аналізу даних для відстеження результатів роботи та формування звітів для прийняття управлінських рішень.

Ці інструменти та методи допомагають ФОП краще організовувати свою роботу, виконувати завдання вчасно та ефективно, а також контролювати процеси виконання завдань і проектів. Вибір конкретних інструментів залежить від виду діяльності ФОП та їхніх потреб.

3. Технологічний аспект. Фізичні особи-підприємці (ФОП) можуть використовувати різні технології та інструменти для полегшення управління виконанням робіт та покращення ефективності своєї діяльності. Ось деякі з них:

- Мобільні додатки. ФОП можуть використовувати мобільні додатки для управління своєю діяльністю. Наприклад, додатки для управління завданнями (як Todoist, Trello), додатки для фінансового обліку (як QuickBooks, Xero), додатки для електронного підпису (як Adobe Sign) і багато інших.

- Хмарні сервіси. Використання хмарних сервісів для збереження та синхронізації даних. Наприклад, Google Drive або Dropbox можуть використовуватися для зберігання та обміну документами та іншою інформацією.

- Електронні таблиці. Використання програм для створення електронних таблиць, таких як Microsoft Excel або Google Sheets, для ведення обліку фінансів, створення розрахунків та статистичного аналізу.
- Електронна пошта. Використання електронної пошти для комунікації з клієнтами, постачальниками та іншими стейкхолдерами.
- Електронна бухгалтерія. Використання спеціалізованих програм для ведення бухгалтерського обліку та формування фінансової звітності.
- Проектний менеджмент. Використання інструментів для керування проектами, таких як діаграми Ганта або системи управління завданнями, для планування та виконання робочих завдань.
- CRM-системи. Використання систем управління відносинами з клієнтами (CRM) для збереження та обробки інформації про клієнтів, ведення історії взаємодії та планування маркетингових заходів.
- Електронний облік. Використання спеціалізованих програм для ведення обліку доходів та витрат, оподаткування та інших фінансових аспектів діяльності ФОП.
- Електронний підпис. Використання електронного підпису для підписання документів та угод в електронній формі.
- Аналітичні інструменти. Використання інструментів аналітики для відстеження результатів роботи та формування звітів для прийняття управлінських рішень.

Використання таких технологій може значно полегшити роботу ФОП, зменшити адміністративні та фінансові труднощі та покращити ефективність управління. Вибір конкретних інструментів залежить від типу діяльності ФОП і їхніх індивідуальних потреб.

4. Фінансовий аспект. Системи управління виконанням робіт можуть значно впливати на фінансове становище та дохідність фізичних осіб-підприємців (ФОП), а також сприяти оптимізації витрат. Ось як це може відбуватися:

- Ефективніше використання часу. Системи управління завданнями та проектами допомагають ФОП планувати та розподіляти свій час ефективніше. Це дозволяє збільшити продуктивність та виконувати більше завдань за той же період часу.

- Кращий контроль над строками. Системи управління проектами допомагають слідкувати за строками виконання робіт і завдань. Це дозволяє уникнути затримок та пеналізацій за прострочені проекти і завдання, що може позитивно вплинути на фінансове становище.

- Підвищення якості робіт. Кращий контроль та спостереження за виконанням робіт дозволяють уникнути помилок і дефектів, що може зменшити витрати на виправлення помилок після завершення проекту.

- Планування ресурсів. Системи управління можуть допомогти ефективно розподіляти ресурси, включаючи людські ресурси та матеріали, що може зменшити зайві витрати.

- Контроль витрат. За допомогою систем управління можна відстежувати витрати на проекти та завдання, що дозволяє аналізувати, де можна зекономити гроші та виправдовувати витрати.

- Збільшення клієнтської задоволеності. Ефективне управління проектами та завданнями може покращити якість обслуговування клієнтів, що може призвести до збільшення задоволеності клієнтів та залучення нових клієнтів.

- Оптимізація робочих процесів. Системи управління можуть допомогти виявити та усунути надмірність у робочих процесах, що дозволяє покращити їх ефективність та ефективність.

- Збільшення конкурентоспроможності. ФОП, які використовують системи управління, можуть бути більш конкурентоспроможними на ринку, оскільки вони здатні виконувати завдання швидше, якісніше та ефективніше.

Узагальнюючи, системи управління виконанням робіт допомагають ФОП краще організовувати свою діяльність, покращувати якість робіт, виявляти ефективність та оптимізувати витрати, що сприяє покращенню їхнього фінансового стану та доходності.

5. Завдання та виклики ФОП. У фізичних осіб-підприємців (ФОП) є ряд завдань та викликів у процесі управління виконанням робіт, і системи організації можуть допомогти в їхньому вирішенні. Ось деякі з цих завдань та викликів і способи їхнього вирішення:

1. Планування завдань і проектів:

- Завдання. ФОП повинні створювати чіткі плани робіт та проектів, визначати їх пріоритети та строків виконання.

- Рішення. Використання систем управління завданнями і проектами для створення списків завдань, надання пріоритетів і визначення строків виконання.

2. Розподіл ресурсів:

- Завдання. Ефективне розподілення людських, фінансових та матеріальних ресурсів для виконання робіт.

- Рішення. Використання систем управління ресурсами для планування та визначення, які ресурси потрібні для кожного завдання або проекту.

3. Відстеження прогресу:

- Завдання. Відстежувати, наскільки вдало ФОП виконують завдання та проекти і як швидко вони наближаються до досягнення мети.

- Рішення. Використання систем управління виконанням робіт, які надають можливість відстежувати прогрес, оновлювати статуси завдань та отримувати повідомлення про досягнення «майлстоунів».

4. Комунікація і співпраця:

- Завдання. Взаємодія з клієнтами, партнерами та співробітниками, обмін інформацією і документами.

- Рішення. Використання спеціалізованих систем управління завданнями, де можливо вести обговорення, обмінюватися файлами та спільно працювати над завданнями.

5. Уникнення затримок і конфліктів:

- Завдання. Уникнення прострочення завдань, затримок у проектах і виникнення конфліктів.

- Рішення. Використання систем управління строками та конфліктами, які допомагають вчасно визначати та вирішувати проблеми.

6. Ефективне використання робочого часу:

- Завдання. Забезпечення того, щоб робочий час ФОП був максимально продуктивним та не розсіювався.

- Рішення. Використання методів та інструментів для планування та управління робочим часом, таких як методика Pomodoro або програми для відстеження робочого часу.

7. Аналіз та оптимізація процесів:

- Завдання. Пошук способів покращення робочих процесів та оптимізація витрат.

- Рішення. Використання аналітичних інструментів та систем управління якістю процесів для аналізу і оптимізації бізнес-процесів.

8. Збереження документації та архівування:

- Завдання. Збереження і ведення документації щодо завдань та проектів для майбутнього використання та архівування.

- Рішення. Використання електронних систем для збереження та архівування документів та інформації.

Системи організації допомагають ФОП краще управляти всіма цими завданнями та викликами, полегшуючи їм планування, співпрацю, відстеження та аналіз їхньої діяльності.

6. Законодавчий та правовий аспект**.** Регуляторні рамки та законодавство, які стосуються фізичних осіб-підприємців (ФОП) у контексті управління роботами, можуть варіюватися в різних країнах і регіонах. Ось загальний огляд того, як це може впливати на їхню діяльність:

Реєстрація та ліцензування. У багатьох країнах ФОП повинні зареєструвати свою діяльність та отримати необхідні ліцензії для виконання певних видів робіт або послуг. Незаконна діяльність може призвести до адміністративних штрафів та судового переслідування.

- Оподаткування. ФОП повинні дотримуватися податкових обов'язків та правил оподаткування для своєї діяльності. Це включає в себе подачу податкових звітів та сплату податків на прибуток, ПДВ, соціальні внески та інші податки.
- Працівники і трудове законодавство. Якщо ФОП мають співробітників, вони повинні дотримуватися відповідних норм трудового законодавства, таких як виплата заробітної плати, забезпечення безпеки праці та інші аспекти відносин між роботодавцем і працівниками.
- Захист персональних даних. Якщо ФОП збирають та обробляють персональні дані клієнтів або співробітників, вони повинні дотримуватися вимог щодо захисту цих даних відповідно до законодавства про захист персональних даних.
- Легальність контрактів та угод. ФОП повинні укладати юридично обов'язкові контракти та угоди з клієнтами, постачальниками та іншими

сторонами. Неправильні або неправомірні угоди можуть призвести до правових проблем.

- Споживчий захист. Якщо ФОП надають послуги або продукти споживачам, вони можуть бути піддані правилам і законам, які захищають права споживачів. Невідповідність може призвести до скарг споживачів і штрафів.
- Екологічні норми. Якщо діяльність ФОП включає в себе використання ресурсів або вплив на навколишнє середовище, вони повинні дотримуватися екологічних норм і стандартів, встановлених відповідними органами.
- Боротьба з корупцією. Деякі країни мають законодавство, що стосується боротьби з корупцією в бізнесі. ФОП повинні виконувати вимоги щодо антикорупційної діяльності та дотримання етичних стандартів.

Важливо для ФОП вивчати та дотримуватися всіх відповідних законодавчих та регуляторних вимог, що стосуються їхньої діяльності. Невиконання цих вимог може призвести до серйозних правових наслідків, штрафів та інших санкцій, які можуть вплинути на їхню діяльність і фінансове становище.

7. Кейси успіху та приклади впровадження. Системи організації управління виконанням робіт можуть бути корисними для фізичних осіб-підприємців (ФОП) в різних галузях та ділових сферах. Ось кілька реальних випадків їх використання:

1. Інформаційні технології (IT). ФОП-програмісти та веб-розробники можуть використовувати системи управління завданнями для планування і відстеження проектів, управління строками та ресурсами, а також спільно працювати над програмними продуктами.

2. Будівництво. ФОП-будівельники можуть використовувати системи для управління проектами та ресурсами на будівельних об'єктах. Це допомагає збільшити продуктивність та відстежувати прогрес будівництва.

3. Фріланс і креативні послуги. Дизайнери, копірайтери та інші ФОП у сферах творчості можуть використовувати системи для керування клієнтськими проектами, відстеження часу та виставлення рахунків.

4. Маленький бізнес і торгівля. Власники магазинів та малого бізнесу можуть використовувати системи для управління запасами, замовленнями та обліку продажів.

5. Медицина. Медичні фахівці, такі як лікарі, психологи або фізіотерапевти, можуть використовувати системи для планування прийому пацієнтів, ведення медичної історії та обліку послуг.

6. Ресторани та гастрономія. Власники ресторанів можуть використовувати системи для управління резерваціями, обліку складових страв та контролю над обігом грошей.

7. Освіта. Вчителі-ФОП можуть використовувати системи для планування навчальних програм, виставлення оцінок та спілкування з учнями та батьками.

8. Логістика та транспорт. ФОП, які працюють у галузі логістики та транспорту, можуть використовувати системи для відстеження вантажів, планування маршрутів та керування флотом автотранспорту.

Ці приклади показують, що системи організації управління виконанням робіт можуть бути корисними для ФОП у різних галузях, допомагаючи їм ефективніше управляти своєю діяльністю, планувати роботу та збільшувати продуктивність.

8. Тенденції та майбутні перспективи. Прогнози щодо розвитку систем організації управління виконанням робіт для фізичних осіб-підприємців (ФОП) в майбутньому показують низку тенденцій та можливих напрямків розвитку:

1. Зростання попиту на рішення для ФОП. Спостерігається зростаючий попит на рішення для ФОП з усіх сфер діяльності, оскільки більше осіб обирають підприємницьку діяльність. Це призведе до розвитку нових програм та сервісів, призначених саме для цієї категорії бізнесу.

2. Розширення функціональності. Системи управління виконанням робіт для ФОП стануть більш різноманітними і функціональними. Вони будуть включати не тільки управління завданнями, але й фінансовий облік, обробку документів, аналіз даних і звітність.

3. Хмарні технології. Зростає популярність хмарних рішень, оскільки вони дозволяють ФОП отримувати доступ до своїх даних і ресурсів з будь-якого пристрою та забезпечувати безпеку даних та резервне копіювання.

4. Інтеграція зі сторонніми додатками. Важливим аспектом розвитку буде інтеграція систем організації з іншими корисними додатками, такими як банківські системи, соціальні мережі, платіжні системи тощо.

5. Збільшена автоматизація. У майбутньому системи будуть надавати більше можливостей для автоматизації рутинних завдань, що допоможе ФОП зосередитися на стратегічних аспектах бізнесу.

6. Розширення мобільних можливостей. Застосунки для смартфонів та планшетів стануть більш доступними і зручними для ФОП, що дозволить їм управляти бізнесом під час переміщень.

7. Зростання кількості конкурентів. Більше компаній входитимуть на ринок систем організації для ФОП, що створить більше варіантів для вибору і може підштовхнути до зниження цін та покращення якості продуктів.

8. Більша безпека та захист даних. Оскільки кількість кіберзагроз та порушень безпеки зростає, системи організації будуть надавати більше можливостей для захисту даних та конфіденційності клієнтів.

9. Персоналізація. Застосунки та рішення будуть більше персоналізовані, враховуючи специфічні потреби і бажання ФОП. [25]

10. Сприяння стартапам та малому бізнесу. Багато розвинених країн активно підтримують малі підприємства та стартапи шляхом надання їм доступу до спеціальних програм та інструментів для управління бізнесом.

Загалом, розвиток систем організації управління виконанням робіт для ФОП буде відображати загальні тенденції в сфері інформаційних технологій, такі як зростання автоматизації, підвищена безпека та доступність хмарних рішень. ФОП, які вміло використовують ці інструменти, зможуть покращити ефективність своєї діяльності, пристосовуватися до змін на ринку та задовольняти потреби своїх клієнтів.

1.3 Порівняльний аналіз існуючих інструментів для управління роботою ФОП

Дослідження існуючих інструментів для управління роботою фізичних осіб-підприємців (ФОП) може бути корисним для знаходження оптимального рішення, яке відповідає конкретним потребам та специфіці діяльності ФОП. Ось деякі з інструментів та підходів, які можна дослідити:

1. Системи управління завданнями. Системи управління завданнями (Task Management Systems або Task Management Software) - це програми або платформи, які допомагають організовувати, відстежувати та керувати завданнями та проектами. Основна мета таких систем - полегшити спрощення та покращення організації робочих завдань для досягнення більшої продуктивності та ефективності в особистій та професійній діяльності. Деякі популярні системи для організації завдань та проектів включають Asana, Todoist, Trello, Wrike, і множину інших. Дослідження цих систем допоможе зрозуміти, яка з них найкраще відповідає вашим потребам.

2. Системи обліку та фінансів. Системи обліку та фінансів - це програми або онлайн-платформи, які допомагають у веденні фінансового обліку, контролю витрат, створенні фінансових звітів та плануванні бюджету. Такі системи розроблені для покращення фінансової дисципліни, збереження часу та ресурсів, а також для забезпечення точності та достовірності фінансової інформації. Для ведення обліку доходів і витрат ФОП може розглядати програми, такі як QuickBooks, FreshBooks, Xero, або використовувати хмарні сервіси для обліку та фінансів.

3. Електронні календарі та планувальники. Це інструменти, які допомагають організовувати та планувати ваш час, події, зустрічі, завдання та інші події. Вони роблять це за допомогою електронного інтерфейсу, який дозволяє легко створювати, редагувати та відстежувати різні події та завдання. Використання електронних календарів, таких як Google Календар або Microsoft Outlook, допоможе планувати зустрічі, дедлайни та події.

4. Системи управління клієнтами (CRM). Це програми та платформи, призначені для зберігання, організації та управління інформацією про клієнтів і взаємодію з ними.

Рис. 1.2. Цикл інформаційних процесів в CRM.

Головною метою CRM-систем є покращення взаємодії з клієнтами, збільшення задоволеності клієнтів та оптимізація бізнес-процесів. Якщо ви працюєте з клієнтами, системи CRM, такі як HubSpot CRM, Salesforce, або Zoho CRM, можуть бути корисними для ведення бази даних клієнтів та взаємодії з ними.

5. Системи для автоматизації завдань. Це програми та платформи, призначені для автоматизації рутинних завдань і процесів у бізнесі та особистому житті. Ці інструменти дозволяють автоматизувати дії, які раніше вимагали б великої кількості часу та зусиль, що допомагає підвищити продуктивність, зменшити помилки та оптимізувати робочий процес. Інструменти, як Zapier або Integromat, допоможуть автоматизувати багато рутинних операцій та інтегрувати різні додатки.

Під час дослідження інструментів важливо враховувати ваші конкретні потреби, обсяг роботи та бюджет. Також важливо звернути увагу на можливості інтеграції між різними інструментами для забезпечення ефективної роботи та обміну даними.

Системи управління завданнями та проектами можуть бути корисними для фізичних осіб-підприємців (ФОП), допомагаючи їм ефективно організовувати свою роботу та досягати поставлених цілей. Далі ми розглянемо популярні системи управління завданнями та проектами, які можуть використовувати ФОП:

1.3.1 Todoist

Todoist - це веб- та мобільний додаток для управління завданнями та списками справ. Він дозволяє користувачам створювати, впорядковувати та відстежувати свої завдання, а також спільно працювати над ними в реальному часі. Todoist дозволяє створювати проекти, надавати завданням терміни

виконання, встановлювати пріоритети та використовувати інші функції для ефективного управління робочими процесами та особистими завданнями. [18]

Цей популярний сервіс для управління завданнями, був запущений в 2007 році його засновником Ієндо Кармелло. Ідея Todoist виникла під час навчання в університеті, коли Ієндо стикнувся з важкістю збереження та відстеження своїх задач.

Починаючи як особистий проект, Todoist швидко став популярним серед користувачів завдяки своїй простоті, функціональності та доступності на різних платформах. Спростити управління завданнями та зробити його більш ефективним - це була головна мета Todoist.

Особливості Todoist:

- Дозволяє організовувати завдання в проекти та використовувати мітки для визначення категорій чи пріоритетів. Це дозволяє користувачам ефективно впоратися з різними аспектами їхнього життя та роботи.
- Дозволяє встановлювати терміни виконання завдань та надсилати нагадування, щоб користувачі завжди були в курсі важливих справ. Це особливо корисно для підтримки організованості та вчасного виконання завдань.
- Підтримує численні інтеграції з іншими популярними інструментами та сервісами, такими як Google Calendar, Dropbox, Slack, і багатьма іншими. Це забезпечує більш гладку і синергетичну роботу з іншими інструментами.

Переваги Todoist:

- Інтуїтивний та простий інтерфейс: Todoist відомий своєю легкістю використання та зрозумілим інтерфейсом, що дозволяє користувачам швидко розпочати роботу без довгого вивчення.

- Можливість працювати на різних платформах: Todoist доступний як веб-версія, так і мобільні додатки для різних операційних систем, що робить його дуже зручним для користувачів, які працюють на різних пристроях.
- Широкі можливості організації завдань: Todoist дозволяє організовувати завдання в проекти, використовувати мітки та створювати «підзадачі», надаючи високий рівень гнучкості при управлінні завданнями.
- Ефективна система нагадувань та термінів виконання: Todoist надає можливість встановлювати терміни виконання та нагадування, що робить його ефективним інструментом для контролю за часовими обмеженнями.
- Спільна робота та обмін завданнями: Todoist дозволяє створювати спільні проекти та ділитися завданнями, що робить його ідеальним для командної роботи та спільних проектів.

Недоліки Todoist:

- Обмежені можливості безкоштовної версії: Деякі розширені функції та інтеграції доступні тільки у платних версіях Todoist, що може бути обмеженням для користувачів, які використовують безкоштовну версію.
- Відсутність гант-діаграм чи календарних переглядів: Todoist не надає деяких розширених видів перегляду завдань, таких як графічні гант-діаграми або календарний перегляд, що може бути важливим для користувачів із певними потребами.
- Обмеженість у роботі з прикріплюваними файлами : У порівнянні з деякими конкурентами Todoist може виглядати менш зручно у роботі з прикріплюваними файлами до завдань, інтеграції із хмарами та зберіганням файлів.
- Залежність від Інтернет-з'єднання: Використання Todoist може бути ускладненим у випадках відсутності Інтернет-з'єднання, оскільки деякі функції вимагають доступу до мережі.

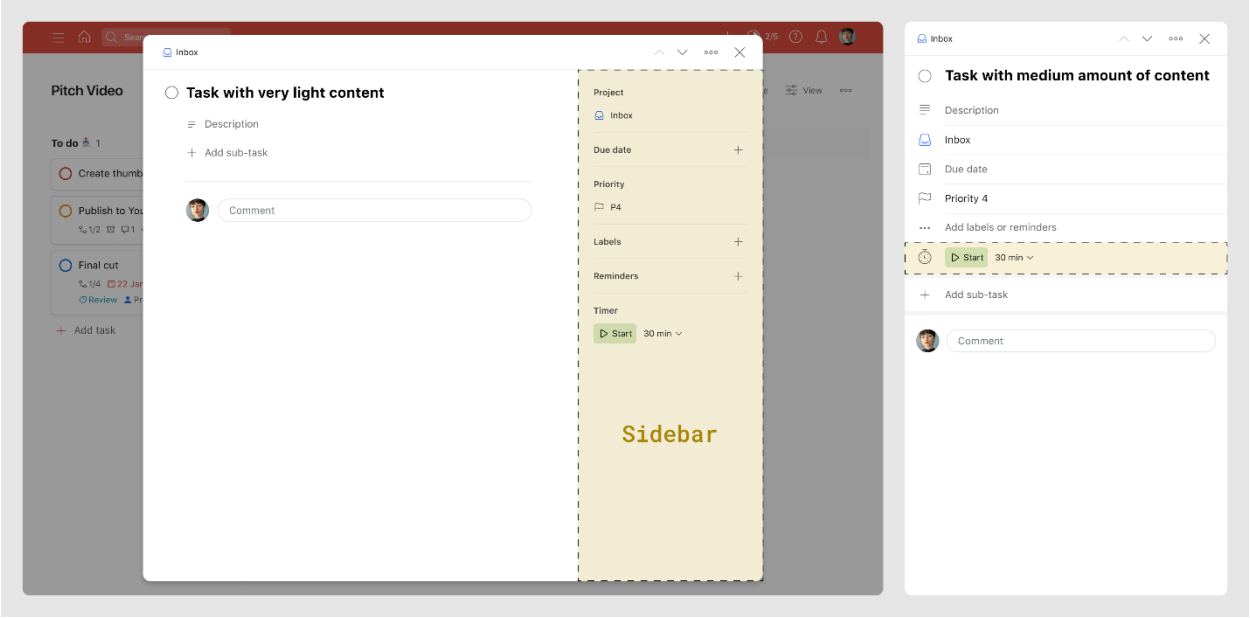


Рис 1.3. Інтерфейс додатку Todoist

Todoist найкраще підходить для:

Todoist підходить для широкого кола користувачів та може бути ефективним інструментом для управління завданнями для різних категорій людей. Нижче наведені кілька категорій користувачів, для яких Todoist може бути особливо корисним:

- Професіонали та Керівники Проектів: Todoist дозволяє організовувати завдання у вигляді проектів, встановлювати терміни виконання, надавати пріоритети, що робить його ідеальним для менеджерів проектів та осіб, які ведуть багатозадачну роботу.
- Студенти та Викладачі: Для студентів та викладачів, які мають багато завдань та термінів, Todoist допомагає впоратися з графіком, встановлювати нагадування для важливих дат, а також організовувати завдання за предметами чи проектами.
- Фрілансери та Самозайняті Особи: Для фрілансерів та самозайнятих осіб, які мають численні завдання від різних клієнтів, Todoist допомагає організувати та відстежувати роботу над проектами, встановлювати терміни та ефективно керувати робочим часом.

- Особи, які прагнуть до особистої продуктивності: Todoist надає ряд функцій для пріоритетів, фільтрів та організації завдань, що робить його ідеальним для тих, хто прагне до систематичного та ефективного підходу до своїх завдань.

Todoist є досить гнучким інструментом, який може адаптуватися до різних потреб користувачів, незалежно від їхнього типу діяльності чи області застосування.

1.3.2 Trello

Trello - це інтерактивна онлайн-платформа для управління завданнями та проектами, що використовує систему дошок та карток для організації робочих процесів. Платформа була створена в 2011 році фахівцями з програмування Майклом Прайором, Джоелем Спольські та Деніелем Варендорфом. Початково вона розроблялася як проект в межах компанії Fog Creek Software та отримала значний успіх завдяки своїй простоті та високій функціональності. У 2014 році Trello стало самостійним продуктом під керівництвом новоствореної компанії Trello, Inc., а пізніше, в 2017 році, було придбано такою відомою компанією як Atlassian.[19]

Особливості Trello:

- Система дошок та карток: Trello базується на концепції дошок, на яких користувачі створюють картки для представлення завдань чи ідей. Ця система візуально подає процес роботи та дозволяє легко орієнтуватися в завданнях.
- Комунікація через коментарі: Trello дозволяє користувачам коментувати кожну картку, що дозволяє ефективно обговорювати деталі завдань та обмінюватися інформацією прямо в контексті проекту.

- Контроль статусів завдань: Кожна картка в Trello може мати свою власну статусну колонку (наприклад, "В роботі", "На перевірці", "Завершено"), що дозволяє відстежувати поточний статус завдань та їхній перехід між різними етапами.

Переваги Trello:

- Візуальна Орієнтація: Одна з найбільших переваг Trello - це візуальний підхід до управління завданнями через дошки та картки. Це особливо корисно для тих, хто краще орієнтується в інформації в зручній візуальній формі.
- Гнучкість та Кастомізація: Trello надає велику гнучкість у створенні власних робочих процесів. Користувачі можуть налаштовувати дошки, статуси завдань та інші параметри, щоб вони відповідали конкретним потребам.
- Комунікація та Коментарі: Trello дозволяє ефективну комунікацію між учасниками проекту, завдяки можливості залишати коментарі на кожній картці. Це сприяє обговоренню та уточненню деталей завдань.
- Гнучкість у Спільній Роботі: Можливість створювати спільні дошки та ділитися завданнями робить Trello ефективним інструментом для командної роботи, особливо у тих випадках, коли важлива взаємодія та співпраця.
- Зручність для Проектів з Великою Кількістю Етапів: Trello особливо ефективний для проектів з багатьма етапами або завданнями, оскільки кожна картка може представляти окремий етап чи завдання, а колонки дошки - різні фази процесу.

Недоліки Trello:

- Обмеженість Управління Завданнями: У порівнянні з більш розширеними системами управління завданнями, Trello може виглядати

менш функціональним у плані деяких продуктивних інструментів та функцій.

- Відсутність Календарного Перегляду: Trello не надає календарного перегляду, що може бути недоліком для тих, хто звик до графічного представлення завдань на календарі.
- Спрощені Можливості Списків Завдань: Деякі користувачі можуть знайти обмеженість у спрощених можливостях роботи зі списками завдань, особливо у порівнянні з більш розвинутими інструментами.
- Залежність від Візуальної Орієнтації: Для тих, хто більше прагне до текстового або спискового представлення завдань, система дошок та карток в Trello може виявитися менш зручною або менш ефективною.

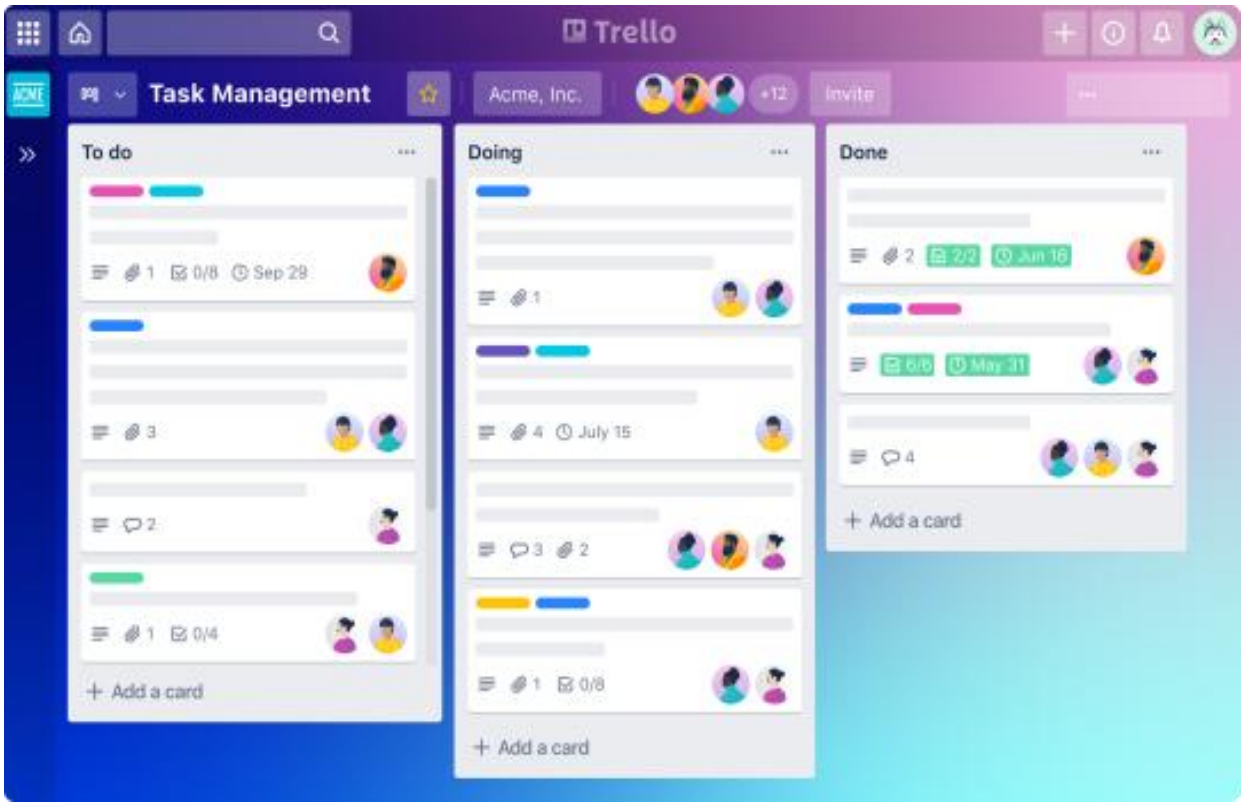


Рис 1.4. Інтерфейс додатку Trello

Trello найкраще підходить для:

Trello найкраще підходить для управління виконаними задачами для осіб та груп, які цінують візуальний підхід до організації робочих процесів та комунікації у вигляді дошок та карток.

- Креативні Команди: Креативні групи та агентства можуть використовувати Trello для візуального відстеження проектів, ідей та завдань. Дошки та картки створюють зручні простори для спільної роботи та творчого процесу.
- Викладачі та Навчальні Групи: Для викладачів та студентів Trello може бути ефективним інструментом для відстеження роботи над уроками, проектами та спільною роботою у групах, сприяючи активній комунікації та обміну ідеями.
- Проекти та Заходи: Для організаторів подій та проектів Trello може слугувати зручним інструментом для планування, відстеження прогресу та координації різних етапів, дозволяючи команді спільно працювати над реалізацією ідей.

1.3.3 Asana

Asana - це онлайн-інструмент для управління проектами та завданнями, який дозволяє командам ефективно співпрацювати та відстежувати прогрес роботи. [12]

Щодо історії Asana, цей сервіс був заснований Дастіном Московіцем та Джастіном Розенстоком у 2008 році. Обидва засновники раніше працювали у компанії Facebook і створили Asana з метою полегшення спільної роботи та кращого управління завданнями у бізнес-середовищі. Основні принципи Asana полягають у створенні зручного інтерфейсу для планування, відстеження та виконання роботи, спрощуючи процеси управління проектами та комунікації всередині команд.

Asana зростає в численних сферах бізнесу та використовується як маленькими підприємцями, так і великими корпораціями для оптимізації робочих процесів та підвищення ефективності командної роботи.

Особливості Asana:

- Метаробочі Простори: Asana надає можливість створювати "Метаробочі Простори", що дозволяють групувати пов'язані проекти та завдання для організації робочого простору команди або відділу.
- Гантівські Діаграми та Відгуки: Asana має вбудовану функціональність для створення гантівських діаграм, що дозволяє візуально відстежувати та планувати проекти за допомогою графічного представлення термінів та залежностей. Також, можливість додавати коментарі та відгуки до завдань полегшує комунікацію в команді.
- Секції та Колонки для Сортування: Asana дозволяє використовувати секції та колонки для легкої сортування та групування завдань, що робить відстеження робочих процесів більш організованим.

Переваги Asana:

- Гнучкість та Комплексність: Asana володіє великою гнучкістю та широким набором функцій, що дозволяє впоратися з різними видами проектів та завдань, забезпечуючи широкий функціонал для команд різного розміру.
- Метаробочі Простори: Можливість створення метаробочих просторів дозволяє організувати проекти та завдання на рівні команди чи відділу, що полегшує спільну роботу та координацію великих команд.
- Шаблони Проектів: Asana надає можливість використовувати шаблони проектів, що спрощує процес створення нових завдань та забезпечує стандартизацію проектних процесів.
- Завдання з Багаторівневою Деталізацією: Asana дозволяє створювати завдання з докладною деталізацією, включаючи вкладені підзавдання та коментарі, що полегшує роботу з складними проектами.
- Співпраця та Комунікація: Інтеграції з різними засобами комунікації, такими як Slack чи Microsoft Teams, дозволяють зручно обговорювати завдання та спільно працювати над проектами в одній системі.

Недоліки Asana:

- Складність для Новачків: У порівнянні з іншими сервісами, Asana може виглядати складним для новачків через велику кількість функцій та можливостей, що може вимагати тривалого вивчення.
- Відсутність Безкоштовної Версії: Asana обмежує функціональність безкоштовної версії, що може бути невигідним для невеликих команд чи фрілансерів, які шукають безкоштовні інструменти управління завданнями.
- Обмежена Взаємодія з Деякими Іншими Інструментами: Інтеграція Asana із деякими іншими інструментами може бути обмеженою порівняно із конкуруючими сервісами, що може вплинути на сумісність та обмін даними.

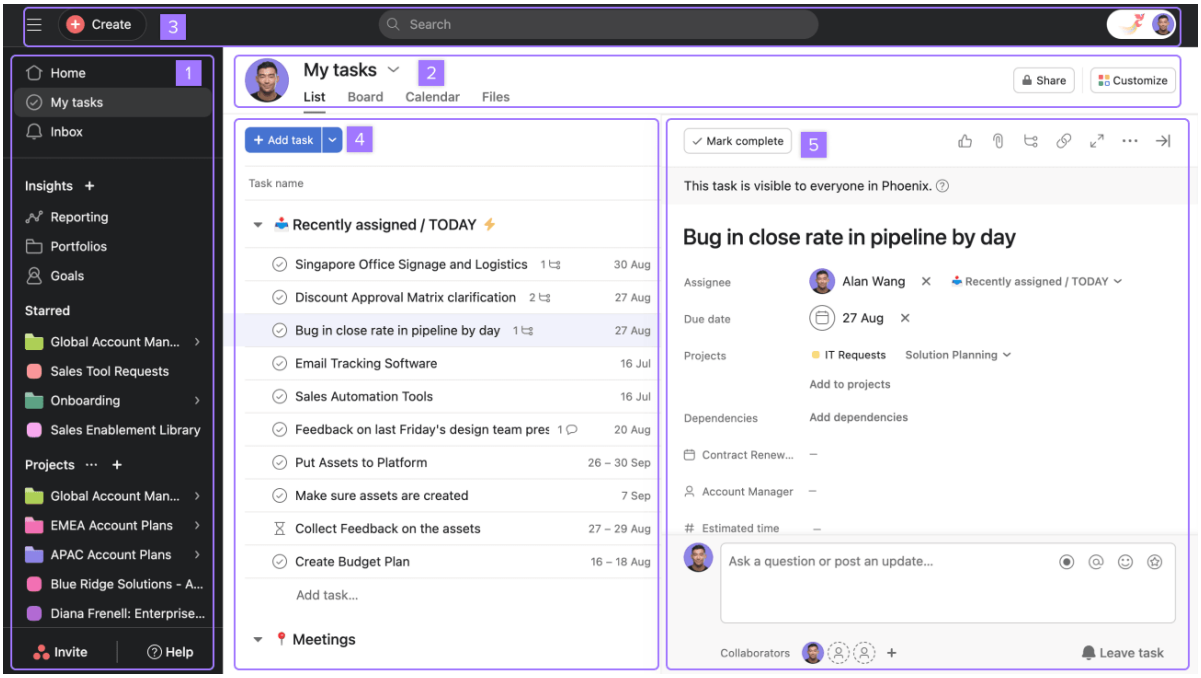


Рис 1.5. Інтерфейс додатку Asana

Asana найкраще підходить для:

- Проекти з Складною Структурою: Для тих, хто веде проекти зі складними структурами, Asana дозволяє деталізувати завдання та

використовувати багаторівневі підзавдання для більшої деталізації та контролю.

- Проекти з Великою Кількістю Взаємозв'язків: Для тих, хто веде проекти з великою кількістю взаємозв'язків між завданнями та командами, Asana надає можливість чітко визначати залежності та забезпечує зручний контроль над взаємодією елементів проекту.

1.3.4 Google Tasks

Google Tasks - це безкоштовний сервіс для управління завданнями від Google, який дозволяє користувачам створювати, відстежувати та організовувати свої завдання. Сервіс був представлений у 2008 році разом із оновленням Gmail та з тих пір інтегрується з різними іншими Google-продуктами, надаючи користувачам простий інструмент для управління своїми завданнями в екосистемі Google. [13]

Особливості Google Tasks:

- Інтеграція з Google Продуктами: Google Tasks має сильну інтеграцію з іншими сервісами Google, такими як Gmail, Google Календар та Google Диск. Це дозволяє користувачам легко переходити між різними інструментами та синхронізувати свої завдання.
- Взаємодія із Завданнями в Gmail: Google Tasks вбудований безпосередньо в інтерфейс Gmail, дозволяючи користувачам швидко створювати та відстежувати завдання прямо з електронної пошти. Це полегшує обробку завдань, пов'язаних з електронними листами.

Переваги Google Tasks:

- Простота та Мінімалізм: Google Tasks славиться своєю простотою та мінімалізмом, що дозволяє швидко створювати та відстежувати завдання без зайвих ускладнень і навіть для новачків.

- Безкоштовність: Google Tasks є безкоштовним сервісом, що робить його доступним для широкого кола користувачів, особливо тих, хто вже використовує інші продукти Google.
- Інтеграція із Завданнями в Gmail: Інтеграція Google Tasks з Gmail дозволяє зручно організовувати завдання прямо з електронної пошти, що полегшує обробку завдань, пов'язаних з листами.

Недоліки Google Tasks:

- Брак Розширених Функцій: Google Tasks може бути занадто мінімалістичним для користувачів, які шукають розширених функцій та більше можливостей управління завданнями.
- Обмежені Функції Перегляду та Сортування: Google Tasks може бути менш гнучким у плануванні та сортуванні завдань порівняно з іншими сервісами, що пропонують різноманіття переглядів та фільтрів.
- Відсутність Автономності: Google Tasks не завжди надає повноцінну автономність роботи в офлайн-режимі, що може бути невигідним для користувачів, які часто використовують сервіс без доступу до Інтернету.

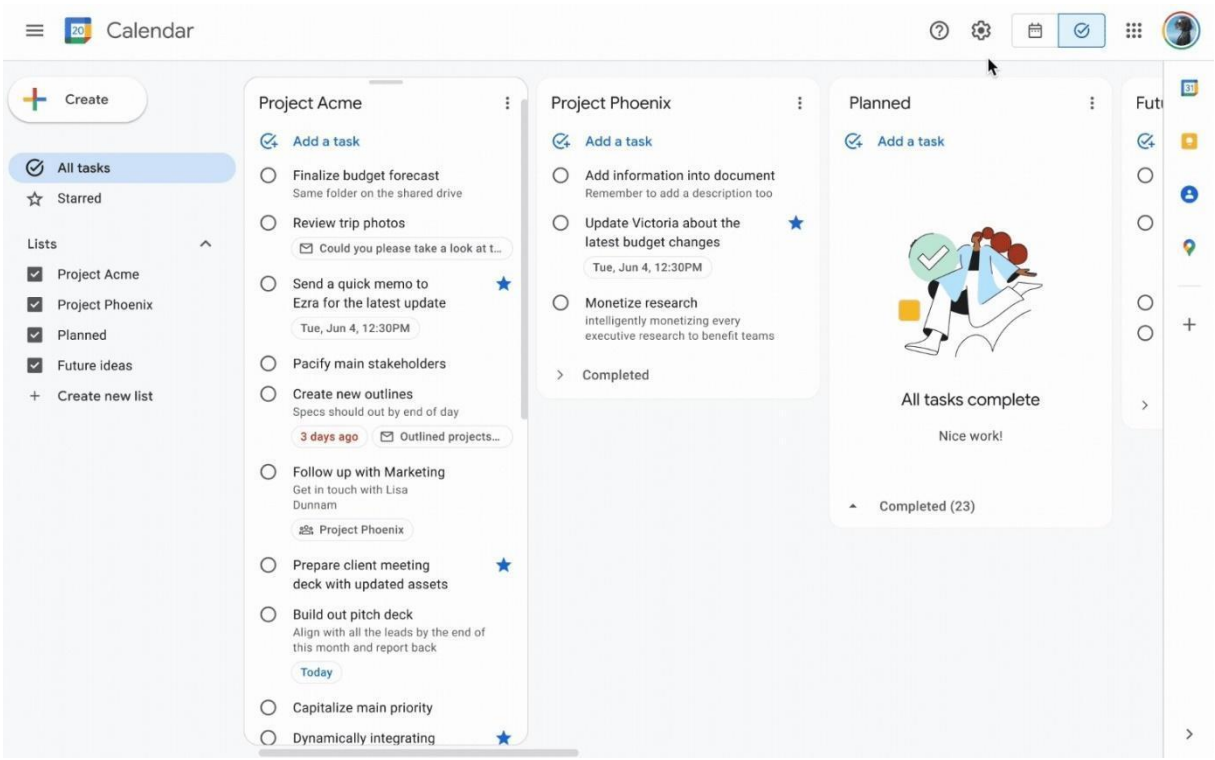


Рис 1.6. Інтерфейс додатку Google Tasks

Google Tasks найкраще підходить для:

- Користувачів Екосистеми Google: Для тих, хто вже користується різними продуктами Google, такими як Gmail, Google Календар та Google Диск, Google Tasks забезпечує зручну інтеграцію та синхронізацію завдань.

1.4. Узагальнені дані

Дослідивши предметну область обраної тематики кваліфікаційної роботи, а також актуальності теми, можна зробити такі висновки:

1) Зростання кількості фізичних осіб-підприємців. У багатьох країнах світу спостерігається тенденція до зростання кількості фізичних осіб-підприємців, які ведуть свій бізнес. Це створює попит на ефективні інструменти для управління роботою та завданнями.

2) Потреба в ефективному управлінні. ФОП відчувають потребу в ефективному управлінні своєю діяльністю, адже вони зазвичай працюють самі на себе та повинні вирішувати багато завдань одночасно.

3) Роль технологій. Сучасні інформаційні та комунікаційні технології надають можливості для автоматизації та оптимізації робочих процесів ФОП.

4) Конкуренція. У бізнес-середовищі конкуренція постійно зростає. ФОП, які володіють ефективними інструментами для управління роботами, можуть бути більш конкурентоспроможними та здатними задовольняти вимоги своїх клієнтів.

5) Необхідність оптимізації витрат. ФОП прагнуть зменшити витрати та підвищити ефективність своєї діяльності. Системи організації управління можуть сприяти оптимізації витрат та збільшенню прибутку.

6) Важливість якості та контролю. Забезпечення якості виконання робіт та завдань є критичним для успіху ФОП. Системи організації управління можуть допомагати в цьому аспекті.

7) Регуляторні рамки і законодавство. Регуляторні вимоги і законодавство можуть вимагати від ФОП дотримуватися певних стандартів управління та звітності.

Після аналізу існуючих систем стало зрозуміло, що важливо використати усі найкращі практики та виключити негативні аспекти аналогів, створивши таким чином власну систему, яка буде конкурентоздатною та здатною вразити користувачів, що бажають бачити зручний та універсальний інструмент для управління роботами. Уважно переглянувши усі особливості, переваги і недоліки можна узагальнити:

Todoist:

- Переваги: Інтуїтивний інтерфейс, доступність на різних платформах, гнучка організація завдань, ефективні нагадування, спільна робота.

- Недоліки: Обмеженість безкоштовної версії, відсутність гант-діаграм та календарного перегляду, обмеженість у роботі з прикріплювальними файлами.

Trello:

- Переваги: Візуальна орієнтація, гнучкість та кастомізація, комунікація та коментарі, зручність для проектів з багатьма етапами.
- Недоліки: Обмеженість управління завданнями, відсутність календарного перегляду, залежність від візуальної орієнтації.

Asana:

- Переваги: Гнучкість та комплексність, метаробочі простори, шаблони проектів, завдання з багаторівневою деталізацією, співпраця та комунікація.
- Недоліки: Складність для новачків, відсутність безкоштовної версії, обмежена взаємодія з деякими іншими інструментами.

Google Tasks:

- Переваги: Простота та мінімалізм, безкоштовність, інтеграція із завданнями в Gmail.
- Недоліки: Брак розширених функцій, обмежені функції перегляду та сортування, відсутність автономності.

При виборі системи управління завданнями важливо враховувати особисті або командні потреби. Todoist ідеально підходить для простих та ефективних завдань, Trello високо цінується за візуальний підхід, Asana найкраще підходить для складних проектів, а Google Tasks пропонує мінімалістичний підхід для швидкого виконання завдань. При виборі важливо враховувати функціональність, зручність використання та потреби конкретного користувача чи команди.

Висновок

У цьому розділі була розглянута актуальність дослідження з приводу кваліфікаційної роботи та проведений аналіз предметної області майбутньої системи для організації управління робітами ФОП. Аспекти робіт ФОП ретельно розглянуті з різних поглядів, а також були докладно проаналізовані існуючі системи організації управління роботами, визначені їх переваги та недоліки, які були узагальнені у вигляді списку. Це дозволяє зосередитися на ключових аспектах проекту, сприяючи покращенню ефективності, контролю та якості робіт ФОП. Для розробки системи буде використовуватися інтегроване середовище розробки для мови програмування Java Script та платформи Node.js, таке як Microsoft Visual Studio Code.

РОЗДІЛ 2
ОПИС ВИМОГ ДО СИСТЕМИ

Вимоги до програмного забезпечення – набір вимог щодо властивостей, якості та функцій програмного забезпечення, що буде розроблено, або знаходиться у розробці. Вимоги визначаються в процесі аналізу вимог та фіксуються в специфікації вимог, діаграмах прецедентів та інших артефактах процесу аналізу та розробки вимог. Розробка вимог до програмної системи може бути розділена на декілька етапів: [5]

- Знаходження вимог (збір, визначення потреб заінтересованих осіб та систем).
- Аналіз вимог (перевірка цілісності та закінченості).
- Специфікація (документування вимог).

Аналіз вимог - це процес систематичного вивчення, розуміння, формалізації і документування вимог до системи чи продукту. Цей процес передбачає збір, уточнення та документування інформації про те, як система повинна працювати або які функції вона повинна виконувати, а також які вимоги повинні бути враховані в процесі розробки.

Аналіз вимог включає в себе взаємодію з зацікавленими сторонами, виявлення ключових функціональних і нефункціональних вимог, уточнення деталей та визначення обмежень, які повинні бути враховані при розробці системи чи продукту. Отримана інформація використовується для створення документації вимог, такої як "Специфікація вимог", яка служить основою для подальшого проектування та розробки. Узгодження і правильне розуміння вимог є критичним етапом у життєвому циклі розробки програмного забезпечення або створення продуктів.

2.1 Вимоги до системи

Специфікація вимог до програмного забезпечення є документом, який детально визначає очікувані функції та роботу програмного продукту. Цей документ також визначає функціональні можливості, які необхідні для задоволення потреб усіх зацікавлених сторін, таких як бізнес та користувачі. У подальшому ми розподілятимемо наші загальні вимоги за категоріями, щоб вичерпно описати всі характеристики нашого продукту:

Бізнес-вимоги

Бізнес-вимоги — це конкретні характеристики, функціональність, або умови, які визначає бізнес, і які новий продукт чи система має задовольняти для досягнення бізнес-цілей та вирішення бізнес-проблем. Ці вимоги визначаються на рівні організації та відображають потреби та очікування бізнес-спільноти, що вони визначають, що повинно бути враховано або створено в новому продукті чи сервісі. [2]

Бізнес-вимоги мають визначити, які завдання має вирішувати продукт, який функціонал має бути включений, а також якісні та кількісні показники успіху для бізнесу. Вони можуть включати в себе вимоги до функціональності, ефективності, надійності, безпеки, масштабованості, інтеграції та інші аспекти.

Узагальнюючи, це такі вимоги, що визначаються з точки зору бізнесу та допомагають створити продукт чи рішення, яке найкращим чином відповідає потребам та стратегії бізнесу. Бізнес-вимоги до системи організації управління робіт ФОП можуть включати:

1. Автоматизація процесів робіт: Забезпечення можливості автоматизації ключових етапів робочих процесів, що допоможе вдосконалити та прискорити виконання завдань.

2. Моніторинг та аналіз продуктивності: Реалізація інструментів для відстеження та аналізу ефективності виконання завдань, що дозволить удосконалити робочі процеси та приймати обґрунтовані управлінські рішення.

3. Система звітності: Впровадження засобів для створення звітів та аналітики, які нададуть докладну інформацію щодо робочих результатів, ресурсів та інших ключових метрик.

4. Інтеграція з іншими системами: Забезпечення можливості інтеграції з іншими корисними інструментами та платформами для оптимізації обміну даними та покращення спільної роботи.

5. Забезпечення безпеки даних: Захист конфіденційної інформації та даних про клієнтів, а також забезпечення відповідності з законодавством щодо захисту персональних даних.

6. Масштабованість: Створення системи, яка може ефективно масштабуватися, адаптуючись до зростання обсягів роботи та потреб бізнесу.

7. Зручний інтерфейс користувача: Розробка інтуїтивного та зручного інтерфейсу для користувачів, що сприятиме легкості використання та швидкому освоєнню системи.

Вимоги користувача

Вимоги користувача — це конкретні характеристики, функціональність та обмеження, які користувачі очікують від продукту чи системи. Ці вимоги визначають, як продукт повинен взаємодіяти з користувачем та які функціональні можливості має надавати, щоб відповідати потребам користувачів. Для системи організації управління робіт ФОП можливі вимоги користувача включають:

1. Інтуїтивний інтерфейс: Користувачі можуть мати вимоги до зручного та легкого використання інтерфейсу системи.

2. Можливість ефективного планування робіт: Користувачі можуть очікувати функціоналу для швидкого створення, призначення та відстеження завдань.

3. Спільна робота та обмін інформацією: Якщо система призначена для командної роботи, користувачі можуть очікувати зручних інструментів для спільної роботи та обміну інформацією.

4. Наявність зручних засобів аналізу та звітності: Користувачі можуть потребувати можливостей аналізу робочого процесу та отримання звітів для ефективного управління роботами.

5. Безпека та конфіденційність: Якщо в системі обробляються чутливі дані, користувачі можуть мати вимоги до високого рівня безпеки та захисту інформації.

6. Зручна інтеграція з іншими інструментами: Користувачі можуть вимагати можливостей інтеграції системи з іншими інструментами, такими як електронна пошта, календарі, або інші програми, які вони вже використовують.

7. Мобільний доступ: Користувачі можуть очікувати можливості використання системи на мобільних пристроях для зручного доступу в будь-який час та в будь-якому місці.

Ці вимоги визначаються на основі потреб та очікувань користувачів і використовуються для створення системи, яка найкращим чином задовольняє їхні очікування та сприяє їхній продуктивності.

Функціональні вимоги

Функціональні вимоги визначають конкретні функції та операції, які повинна виконувати система. Це опис того, що система має здатність робити. Для системи організації управління робіт ФОП функціональні вимоги можуть включати:

1) Створення та Планування Завдань: Система повинна дозволяти ФОПам створювати нові завдання, призначати їх, та планувати терміни виконання.

2) Підтримка Проектів: Якщо ФОП працює над проектами, система повинна забезпечувати можливість організації завдань в рамках проектів.

3) Спільна Робота та Обмін Інформацією: Функціонал для спільної роботи над завданнями, коментування та обміну файлами між ФОПами або членами команди.

4) Моніторинг та Аналіз Продуктивності: Можливість відстеження стану завдань, прогресу та аналізу продуктивності ФОП.

5) Нагадування та Сповіщення: Система повинна надавати можливість налаштовувати нагадування та отримувати сповіщення про терміни виконання завдань.

6) Інтеграція з Електронною Поштою та Календарем: Зручна інтеграція з електронною поштою та календарем для організації робочого часу та комунікації.

7) Захист Даних та Приватності: Функції, які забезпечують безпеку і конфіденційність даних, особливо якщо вони включають чутливу інформацію.

Візьмемо за приклад ФОП, який веде власну майстерню з ремонту електроніки, може використовувати систему, щоб створювати та відстежувати замовлення від клієнтів. Він може визначати терміни ремонту, вести графік своєї робочої діяльності та ділитися інформацією про стан робіт зі своїм технічним персоналом або клієнтами через цю систему. Така функціональність спрощує процес управління та спільної роботи, поліпшуючи продуктивність та задоволення клієнтів.

Нефункціональні вимоги

Нефункціональні вимоги визначають атрибути та обмеження системи, які не стосуються конкретних функцій, але впливають на її ефективність, надійність та інші характеристики. До нефункціональних вимог для системи організації управління робітами ФОП можуть входити:

- Безпека та Конфіденційність: Система повинна забезпечувати високий рівень захисту даних клієнтів та внутрішніх інформаційних ресурсів.

- Продуктивність: Система повинна працювати ефективно та забезпечувати швидкий доступ до інформації, навіть при великій кількості користувачів або об'ємі даних.

- Надійність та Доступність: Система повинна бути стійкою до відмов та готовою до використання у будь-який момент робочого часу ФОП.

- Сумісність та Інтеграція: Можливість інтеграції системи з іншими інструментами, такими як облікові програми чи електронні платіжні системи.

- Використання Ресурсів: Система повинна ефективно використовувати ресурси, такі як пам'ять та потужність обчислень, для оптимальної продуктивності.

Нефункціональна вимога до системи організації управління робітами ФОП може бути пов'язана із продуктивністю. Наприклад, вимога, що система повинна завантажувати сторінки або відповідати на користувацькі запити протягом певного часу, щоб забезпечити швидкий та ефективний робочий процес.

2.2 Вимоги до безпеки

Вимоги безпеки у контексті системи організації управління робіт ФОП є визначальними умовами, що забезпечують захищеність від можливих ризиків та загроз. Їх основна мета - забезпечити конфіденційність, цілісність та доступність даних та ресурсів, зменшити можливість несанкціонованого доступу та втрати інформації.

Ці вимоги є необхідним елементом для забезпечення захищеності системи та мінімізації ризиків. Вони визначаються через аналіз потенційних загроз, врахування стандартів безпеки та врахування вимог стейкхолдерів, таких як користувачі та власники бізнесу. З метою забезпечення довіри до системи та зменшення можливості виникнення проблем важливо дотримуватися встановлених стандартів та практик безпеки. Це особливо

актуально у випадку, коли в системі обробляються чутливі дані, що потребують особливого захисту.

Аналіз ризиків є ключовим етапом у визначенні вимог безпеки, де ідентифікуються потенційні загрози та оцінюється їх вплив на систему. Забезпечення безпеки системи включає в себе використання відповідних технологій та засобів безпеки, дотримання політик та стандартів, а також постійне оновлення заходів безпеки для врахування нових загроз та вразливостей. Цей ітеративний процес визначення вимог безпеки має велике значення протягом усього життєвого циклу системи та допомагає забезпечити високий рівень захищеності та надійності управління робіт ФОП.

Вимоги до безпеки в системі організації управління робітами ФОП є критичним елементом, оскільки вони спрямовані на захист конфіденційності, цілісності та доступності даних. Такі вимоги стосуються як технічних, так і організаційних аспектів. Нижче розглянуті ключові аспекти вимог до безпеки та технології, які можуть бути використані для їх реалізації:

Ключові аспекти вимог до безпеки в системі:

- Контроль доступу: Забезпечення ефективного контролю доступу до системи та конфіденційних даних. Впровадження ролевої моделі доступу, двофакторної аутентифікації та обмеження прав доступу на рівні користувачів та ролей.
- Шифрування даних: Захист даних від несанкціонованого доступу та забезпечення конфіденційності. Використання шифрування на рівні бази даних та протоколів HTTPS для захищеної передачі даних.
- Захист від атак: Мінімізація ризиків від різних видів атак, таких як SQL-ін'єкції, кросс-сайтові сценарії та інші. Використання параметризованих запитів у базі даних та активне виявлення та блокування аномальної активності.

- Резервне копіювання та відновлення: Забезпечення можливості швидкого відновлення системи в разі втрати даних або виникнення інших кризових ситуацій. Використання автоматизованих систем резервного копіювання в хмарних службах та регулярні тестування процедур відновлення.
- Моніторинг та аудит безпеки: Забезпечення постійного моніторингу та аудиту безпеки для виявлення та реагування на потенційні загрози. Використання систем безпеки подій та інформаційних подій (SIEM) та автоматизоване виявлення аномалій у поведінці користувачів.

Технології якими можна убезпечити систему:

- Firewalls та IDS/IPS: Використання брандмауерів для контролю мережевого трафіку та систем виявлення та запобігання вторгнень.
- Антивірусні програми та антимальварні рішення: Захист від шкідливих програм та вірусів.
- Віртуалізація та контейнеризація: Використання ізольованих середовищ для підвищення безпеки додатків та даних.
- Шифрування та безпека на рівні додатків: Застосування методів шифрування та захисту на рівні програм.
- Ідентифікація та управління ідентичністю: Застосування технологій управління доступом та ідентифікації користувачів.

Врахування цих вимог і використання відповідних технологій забезпечить високий рівень безпеки системи організації управління робітами ФОП.

2.2.1 Firewalls та IDS/IPS

Firewall (брандмауер) є ключовим елементом в системах безпеки комп'ютерних мереж, спроектованим для контролю та моніторингу мережевого трафіку між комп'ютерами. Основна його мета — забезпечити безпеку мережі, фільтруючи та блокуючи небезпечний чи недозволений трафік. Цей захисний

шар між внутрішньою мережею та зовнішніми мережами дозволяє адміністраторам задавати правила доступу, контролювати вхідний та вихідний трафік, а також застосовувати стратегії безпеки для запобігання несанкціонованому доступу.

Система виявлення та запобігання вторгнень (IDS/IPS) — це комплекс технологій та програм, спрямованих на виявлення та усунення загроз безпеки в мережі. DS виявляє потенційно небезпечний трафік чи вторгнення, подаючи сигнали адміністраторам або активуючи механізми автоматичного реагування. У свою чергу, IPS включає в себе не лише виявлення, але й активні заходи для запобігання або припинення атак. [26]

Firewalls та системи виявлення та запобігання вторгнень (IDS/IPS) грають ключову роль у захисті вашої системи від кіберзагроз та небезпек. Вони блокують небажані та шкідливі з'єднання, дозволяючи лише безпечному трафіку пройти через систему. IDS виявляє аномалії у мережевому трафіку, а IPS може автоматично реагувати на виявлені загрози, блокуючи IP-адреси чи призупиняючи підключення. Також, вони сприяють використанню шифрування для захисту конфіденційної інформації та допомагають в регулярному моніторингу та аудиті безпеки, ідентифікуючи потенційні проблеми в системі. Використання цих технологій створює ефективний захисний щит, забезпечуючи цілісність, конфіденційність та доступність системи.

2.2.2 Віртуалізація та контейнеризація

Віртуалізація - це процес створення віртуальних екземплярів операційних систем або ресурсів на одному фізичному сервері. Гіпервізор відповідає за створення та управління цими віртуальними машинами, що ізолює їх одну від одної. Це дозволяє запускати різні операційні системи та додатки на одному сервері, використовуючи об'єднані ресурси.

Контейнеризація - це легковаговий метод віртуалізації, де окремі додатки та їх залежності запускаються в окремих контейнерах. Контейнери ізолюють

додатки один від одного, але використовують спільний ядро операційної системи, що робить їх більш ефективними та швидкими.

Docker є однією з ключових технологій контейнеризації. Він дозволяє упаковувати додатки та їх залежності в контейнери, які можуть бути перенесені між різними середовищами. Docker забезпечує ізоляцію, що дозволяє уникнути конфліктів між додатками та спільно використовувати ресурси.

Kubernetes - це система для автоматизації розгортання, масштабування та управління контейнеризованими додатками. Вона дозволяє автоматизувати багато аспектів роботи з контейнерами, такі як масштабування, керування ресурсами та аварійне відновлення.

Ці технології допомагають системі, забезпечуючи гнучкість та ізоляцію середовищ, полегшуючи розгортання та масштабування додатків. Вони дозволяють швидше реагувати на зміни та ефективно використовувати ресурси серверів, підвищуючи загальний рівень безпеки системи.

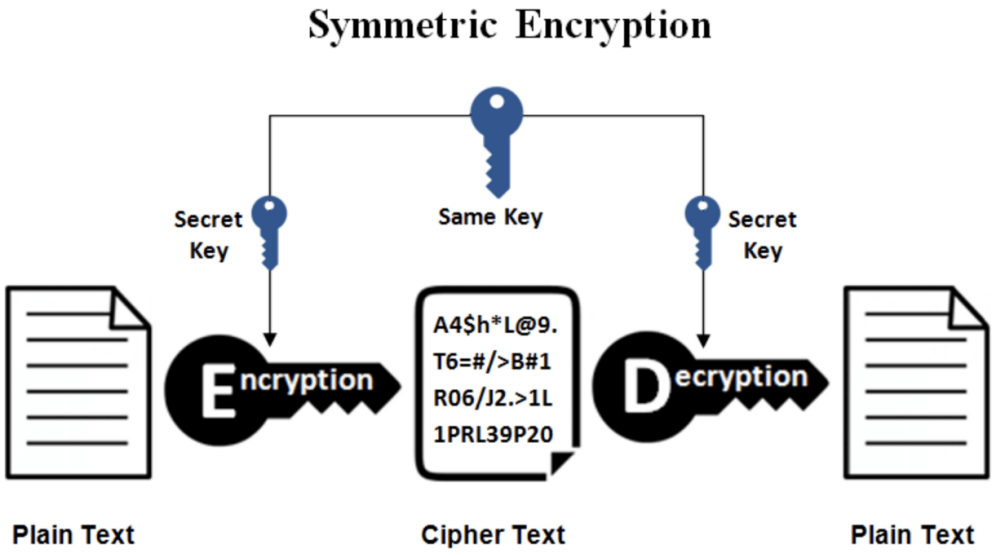
2.2.3 Шифрування та безпека на рівні додатків

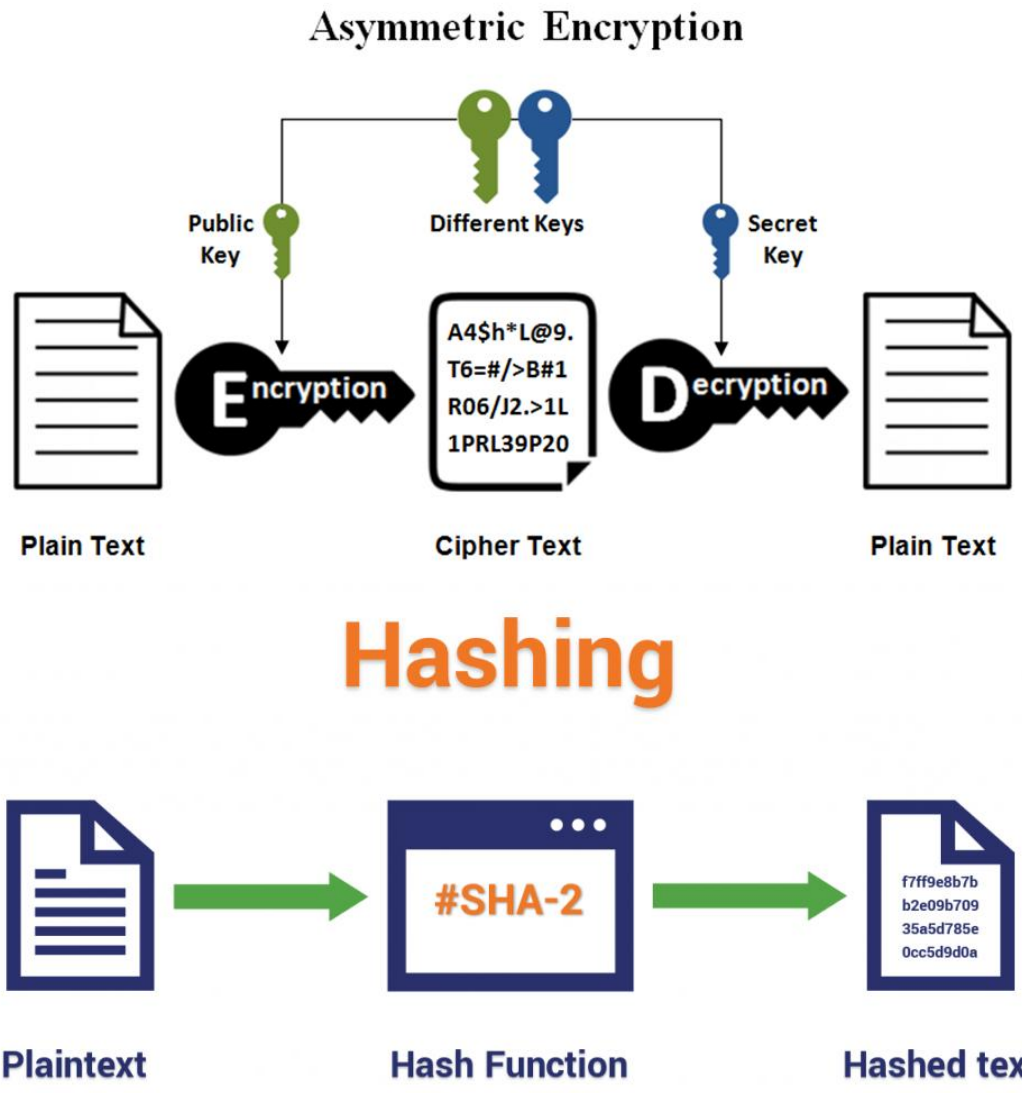
Будь-який веб-сайт Шифрування та безпека на рівні веб-додатків є невід'ємною частиною забезпечення високого рівня захисту в інтернет-середовищі. Ці аспекти важливі для забезпечення конфіденційності, цілісності та доступності веб-додатків.

Шифрування веб-додатків — це важлива ланка в системі забезпечення інформаційної безпеки, а є кілька видів шифрування, які використовуються для захисту даних в онлайн середовищі.

- Симетричне шифрування: Цей вид шифрування використовує один ключ для як шифрування, так і розшифрування інформації. Його принцип роботи полягає в тому, що обидва боки спілкуються за допомогою спільного ключа, який залишається конфіденційним. Однак в умовах веб-додатків це може бути менш ефективним, оскільки обидва боки повинні зберігати та обмінюватися одним ключем.

- Асиметричне шифрування: Тут використовуються два ключі — публічний та приватний. Публічний ключ використовується для шифрування, а приватний — для розшифрування. Це забезпечує більшу безпеку, адже приватний ключ не обмінюється іншими користувачами. Однак, в такому випадку, процес шифрування може бути більш обчислювально витратним, що важливо враховувати у веб-додатках з великим обсягом даних. [1]
- Хешування: Це не повноцінне шифрування, але важливий елемент безпеки веб-додатків. Хеш-функції перетворюють введені дані в фіксований рядок, і вони використовуються для зберігання паролів у вигляді хешів. Коли користувач вводить свій пароль, веб-додаток порівнює хеш введеного паролю з збереженим. Це ускладнює виявлення пароля навіть у випадку проникнення в систему. [9]





Рисунки 2.1-2.3 Схема того як працює симетричне та асиметричне шифрування та хешування

Один із найбільш поширених протоколів шифрування – HTTPS. Це розширення стандартного протоколу HTTP, призначене для забезпечення безпеки передачі даних між користувачем і веб-сайтом. Процес шифрування в HTTPS використовує протоколи SSL (Secure Sockets Layer) або їхню більш сучасну версію — TLS (Transport Layer Security). SSL/TLS забезпечує конфіденційність даних, пересланих між користувачем і веб-сайтом, шляхом їхнього шифрування.

SSL/TLS також використовує сертифікати, які відіграють роль електронних паспортів для веб-сайтів. Ці сертифікати видаються довіреними центрами сертифікації і підтверджують ідентичність веб-сайту. При переході на

веб-сайт за HTTPS, браузер перевіряє валідність цього сертифіката, що додає елемент аутентифікації та впевненості в безпеці з'єднання. Застосування SSL/TLS у HTTPS дозволяє ефективно захищати дані від несанкціонованого доступу та забезпечує безпечний обмін інформацією в онлайн-середовищі.

Безпека на рівні веб-додатків охоплює велику кількість практик і технологій. Зокрема, фільтрація та валідація введених даних допомагають уникнути атак, таких як SQL-ін'єкції та кросс-сайтові сценарії. Використання протоколів безпеки, наприклад, OAuth або OpenID Connect, дозволяє надійно автентифікувати користувачів та керувати доступом до ресурсів.

Важливою частиною безпеки є також регулярні аудити та моніторинг веб-додатків для виявлення та вирішення потенційних загроз. Це може включати в себе використання систем безпеки подій та інформаційних подій (SIEM) та інші засоби для виявлення аномалій та вторгнень.

Всі ці заходи спрямовані на забезпечення надійності та захисту веб-додатків, зменшення ризиків інформаційної безпеки та забезпечення високого рівня довіри користувачів.

2.3 Вимоги до інтерфейсу

Вимоги до інтерфейсу відображають очікування щодо зовнішнього вигляду та функціональності користувацького інтерфейсу програмного продукту. Ці вимоги спрямовані на те, щоб забезпечити ефективну та задовільну взаємодію між користувачем і системою. Вони можуть включати різноманітні аспекти, такі як дизайн, навігацію, доступність, швидкодію, зручність використання та інші.

Вимоги до інтерфейсу описують, як користувач буде взаємодіяти з програмним продуктом, визначають структуру елементів інтерфейсу, його кольорову палітру, шрифти, розміщення кнопок та інших елементів. Вони також можуть враховувати принципи взаємодії, такі як консистентність та інтуїтивність, щоб сприяти комфортному користуванню.

Наприклад, вимоги до інтерфейсу можуть включати запитання щодо розміру кнопок на екрані, розташування меню, наявність візуальних ефектів, які підкреслюють важливі дії, і багато іншого. Усі ці аспекти призначені для створення ефективного та зручного інтерфейсу, який відповідає потребам та очікуванням користувача. Дизайн інтерфейсів складається з двох основних частин:

2.3.1 UI Design

Дизайн інтерфейсу користувача (UI) є вирішальним аспектом створення успішного продукту, веб-сайту або програми. Це візуальне представлення продукту, з яким взаємодіє користувач. Дизайн інтерфейсу користувача відіграє вирішальну роль у забезпеченні того, щоб продукт був не лише зручним для використання, але й візуально привабливим і відповідав ідентичності бренду. [21]

Дизайн інтерфейсу користувача (UI) — це процес розробки інтерфейсу, з яким користувачі взаємодіють під час використання продукту. Це поєднання візуального дизайну, дизайну взаємодії та інформаційної архітектури. Метою дизайну інтерфейсу користувача є створення візуально привабливого та зручного для користувача інтерфейсу, який дозволяє користувачам легко та ефективно виконувати завдання. Дизайн інтерфейсу користувача зосереджується на передбаченні того, що користувачам може знадобитися зробити, і забезпеченні того, щоб інтерфейс мав елементи, які легко отримати, зрозуміти та використовувати для полегшення цих дій. Інтерфейс користувача об’єднує концепції дизайну взаємодії, візуального дизайну та інформаційної архітектури.

Процес проектування інтерфейсу користувача

Процес розробки інтерфейсу користувача складається з кількох етапів, починаючи з дослідження та аналізу та закінчуючи створенням остаточного дизайну. Ось основні етапи розробки інтерфейсу користувача:

1) Дослідження та аналіз: цей етап передбачає розуміння потреб, уподобань і поведінки користувача, а також бізнес-цілей і завдань. Це робиться шляхом дослідження користувачів, аналізу конкурентів та опитування зацікавлених сторін.
2) Ідея: цей етап передбачає генерування ідей для дизайну, таких як макет, колірні схеми, типографіка та зображення. Це робиться за допомогою мозкового штурму та створення ескізів.
3) Каркас: на цьому етапі створюється низькоточне представлення дизайну, наприклад, каркас, щоб показати основну структуру та макет інтерфейсу.
4) Створення прототипу: цей етап передбачає створення високоточного представлення дизайну, наприклад прототипу, який можна натиснути, щоб показати інтерактивні елементи та потік користувачів.
5) Тестування користувачами: цей етап включає тестування дизайну з користувачами для виявлення будь-яких проблем із зручністю використання та отримання відгуків щодо дизайну.
6) Доопрацювання: цей етап передбачає внесення змін до дизайну на основі відгуків користувачів і результатів тестування.
7) Виробництво: цей етап передбачає доопрацювання дизайну та підготовку його до розробки.

Ключові принципи та практики дизайну інтерфейсу користувача

Ключові принципи дизайну інтерфейсу користувача визначають основні принципи, які спрямовані на створення ефективних та зручних для користувачів інтерфейсів. Принципи, якими слід керуватися при розробці інтерфейсу користувача:

- Простота: зберігайте дизайн простим і легким у використанні. Уникайте безладу та непотрібних елементів.

- Узгодженість: використовуйте узгоджені візуальні елементи, такі як колір, типографіка та макет, у всьому інтерфейсі, щоб створити цілісну та єдину взаємодію.
- Чіткість: переконайтеся, що дизайн чітко та ефективно передає свою мету. Використовуйте описові мітки та надсилайте відгук користувачам, коли вони взаємодіють з інтерфейсом.
- Гнучкість: створюйте інтерфейс для адаптації до різних розмірів екрана, роздільної здатності та типів пристроїв. Переконайтеся, що дизайн адаптований до різних контекстів і потреб користувачів.
- Доступність: переконайтеся, що дизайн доступний для користувачів з обмеженими можливостями, наприклад, людей із вадами зору або руховими вадами. Використовуйте відповідний колірний контраст, надайте альтернативний текст для зображень і переконайтеся, що інтерфейсом можна переміщатися за допомогою комбінацій клавіш.

Найкращі практики для дизайну інтерфейсу користувача:

- Використання системи сітки, щоб забезпечити вирівнювання та баланс у макеті.
- Використання обмеженої палітри кольорів, щоб створити цілісну та послідовну візуальну ідентичність.
- Використання типографії, щоб створити ієрархію та акцент.
- Використання зображень та значків, щоб допомогти передати інформацію та зробити інтерфейс більш візуально привабливим.
- Використання пробілів, щоб створити відчуття відкритості та підкреслити важливі елементи

Дизайн інтерфейсу користувача є критично важливим аспектом створення успішного продукту, веб-сайту чи програми. Добре розроблений інтерфейс користувача є важливим для забезпечення позитивного досвіду користувача (UX). Важливо розуміти основи дизайну інтерфейсу користувача, передові практики та інструменти щоб створити ефективний дизайн інтерфейсу

користувача, який відповідає потребам ваших користувачів. Головне пам’ятати про те, щоб він був простим, послідовним і доступним, надаючи зворотний зв’язок і створюючи візуальну ієрархію.

2.3.2 UX Design

Дизайн користувальницького досвіду, або UX-дизайн, є важливим аспектом створення програм та веб-сайтів, спрямованим на створення задоволення та зручності для користувачів. Основною метою UX-дизайну є розуміння потреб користувачів і забезпечення такого досвіду, який би відповідав їхнім очікуванням та сприяв ефективному взаємодії з продуктом. Цей підхід включає в себе аналіз поведінки користувачів, створення прототипів, тестування та постійне вдосконалення інтерфейсу. Засоби UX-дизайну допомагають визначити логіку взаємодії, структуру інформації та зовнішній вигляд продукту.

Успішний UX-дизайн дозволяє підняти рівень задоволення користувачів, зменшити кількість помилок та покращити віддзеркалення бренду через взаємодію з програмними продуктами. Необхідність враховувати потреби користувачів у кожному етапі розробки гарантує, що створюваний продукт буде високоякісним та конкурентоспроможним на ринку. [11]

Ключові принципи та практики дизайну користувальницького досвіду

Дизайн користувальницького досвіду (UX) базується на ключових принципах та практиках, спрямованих на створення ефективних та задовільних взаємодій між користувачем і продуктом. Деякі з них наведені нижче:

- Розуміння користувачів: Суть UX-дизайну полягає в ретельному вивченні та розумінні потреб та очікувань цільової аудиторії. Аналіз поведінки користувачів, їхніх цілей та контексту використання є ключовим етапом.

- Прототипування: Створення прототипів дозволяє випробувати концепції та ідеї до остаточного релізу продукту. Це допомагає виявити можливі проблеми та змінити дизайн на ранніх етапах розробки.
- Консистентність: Один з ключових принципів — це забезпечення консистентності у всій системі. Користувачі очікують однакового досвіду на різних екранах та сторінках, тому важливо дотримуватися єдиної стилістики та взаємодії.
- Простота та зрозумілість: Дизайн повинен бути якнайбільше інтуїтивним. Простота взаємодії та зрозумілість інтерфейсу сприяють полегшенню використання продукту для різних категорій користувачів.
- Доступність: Забезпечення доступності для всіх користувачів, включаючи тих, хто має обмеження, є невід'ємною частиною дизайну UX. Врахування різноманітності користувачів сприяє створенню більш широкого та інклюзивного продукту.
- Задоволення користувачів: Кінцева мета — це забезпечення задоволення та позитивного враження від взаємодії з продуктом. Важливо враховувати емоції користувачів та створювати приємний досвід.
- Застосування дизайну згідно з контекстом: Урахування контексту використання — це важливий аспект, оскільки користувачі можуть використовувати продукт в різних умовах та ситуаціях. [10]

З дотриманням усіх вище вказаних принципів можуть допомогти такі практики:

- Дослідження користувачів: Проведення глибокого аналізу та дослідження потреб та очікувань цільової аудиторії. Врахування поведінки користувачів, їхніх проблем та цілей дозволяє створити дизайн, який дійсно відповідає їхнім потребам.
- Прототипування: Використання прототипів для передбачення та тестування дизайну перед фінальною реалізацією. Це дозволяє виявити

можливі проблеми та внести зміни на ранніх етапах розробки, що економить час та ресурси.

- Тестування з користувачами: Залучення реальних користувачів для тестування продукту в реальних умовах. Зворотний зв'язок від користувачів допомагає виявити слабкі місця та покращити дизайн.
- Консистентність у дизайні: Забезпечення єдиної стилістики та взаємодії на всій платформі або в рамках продукту. Користувачі повинні легко розпізнавати та взаємодіяти з різними частинами продукту.
- Врахування візуального та функціонального балансу: Створення гармонії між естетикою та функціональністю. Забезпечення того, щоб елементи дизайну були привабливими та одночасно виконували потрібні завдання для користувачів.

Загальна ідея полягає в тому, щоб створити такий дизайн, який не лише відповідає функціональним вимогам, але і приносить задоволення користувачам у процесі взаємодії з продуктом.

Висновок

Другий розділ проекту визначав ключові вимоги до розроблюваної системи, ретельно розглядаючи бізнес вимоги, вимоги користувача, функціональні та нефункціональні вимоги, а також вимоги до інфраструктури, на якій має функціонувати продукт. Зокрема, враховувалась необхідність надійності системи, що призвело до ретельного розгляду вимог до безпеки, таких як використання брандмауерів, систем виявлення та запобігання вторгнень, і методів шифрування даних. В особливості розглянуті принципи UI та UX дизайну, які визначають ефективність та зручність взаємодії користувача з продуктом. Важливість збалансованості між функціональністю та естетикою була освітлена, підкреслюючи необхідність створення гармонійного дизайну. В цілому, чітко визначені вимоги становлять основу для подальшого проектування та розробки системи, забезпечуючи зручний та ефективний інструмент для користувачів при збереженні високого рівня безпеки.

РОЗДІЛ 3
СТРУКТУРА СИСТЕМИ

Архітектура програмного забезпечення визначає спосіб організації внутрішніх компонентів продукту, і правильний підбір архітектурних рішень впливає на його підтримуваність та масштабованість. З ростом розміру кодової бази стає важливим забезпечити чітку структуру, яка дозволяє легко знаходити, змінювати та доповнювати функціонал, не порушуючи роботу існуючих компонентів. Архітектурні шаблони і патерни грають ключову роль у полегшенні розробки та утримання ПЗ. Вони пропонують стандартні рішення для типових завдань, сприяючи створенню модульного та розширюваного коду. При цьому важливо враховувати якнайбільше аспектів, таких як ефективність, безпека, масштабованість та тестування.

3.1 Архітектура

Архітектура служить планом системи, що забезпечує абстракцію для управління складністю системи та встановлення механізму зв’язку та координації між компонентами. Вона визначає структуроване рішення, яке відповідає всім технічним і експлуатаційним вимогам. Одночасно оптимізує загальні атрибути якості, такі як продуктивність і безпека. [15]

Крім того, це включає ряд важливих рішень щодо організації, пов’язаних із розробкою програмного забезпечення, і кожне з цих рішень може мати значний вплив на якість, зручність обслуговування, продуктивність і загальний успіх кінцевого продукту. Ці рішення складаються з:

- Вибір структурних елементів та їх інтерфейсів, з яких складається система.
- Поведінка, визначена у співпраці між цими елементами.

- Композиція цих структурних і поведінкових елементів у велику підсистему.
- Архітектурні рішення відповідають цілям бізнесу.
- Архітектурні стилі керують організацією.

Дизайн

Розробка програмного забезпечення забезпечує план проектування, який описує елементи системи, як вони підходять і працюють разом, щоб виконати вимоги системи. Цілі плану дизайну такі:

- Обговорювати системні вимоги та встановлювати очікування з клієнтами, маркетингом і управлінським персоналом.
- Дійте як план під час процесу розробки.
- Керуйте завданнями впровадження, включаючи детальне проектування, кодування, інтеграцію та тестування.
- Він з’являється перед детальним проектуванням, кодуванням, інтеграцією та тестуванням і після аналізу домену, аналізу вимог і аналізу ризиків.

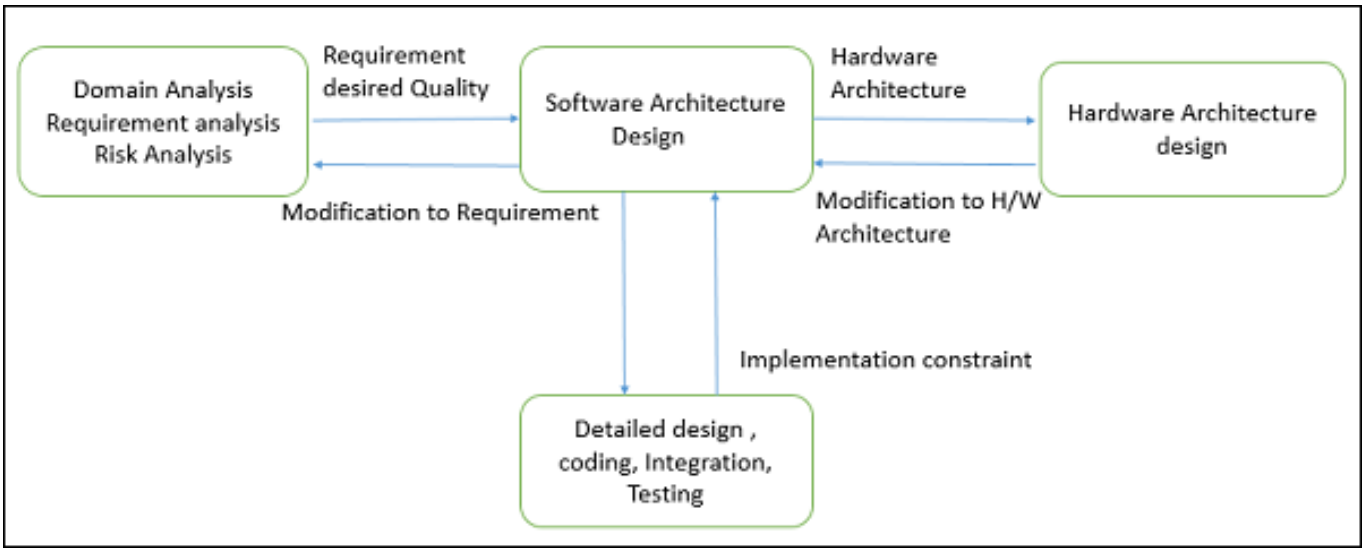


Рис. 3.1. Модель процесу дизайну архітектури майбутньої системи

Цілі архітектури

Основною метою архітектури є визначення вимог, які впливають на структуру програми. Добре продумана архітектура зменшує бізнес-ризики, пов’язані зі створенням технічного рішення, і створює міст між бізнесом і технічними вимогами. Деякі з інших цілей такі:

- Відкрити структуру системи, але приховати деталі її реалізації.
- Реалізуйте всі варіанти використання та сценарії.
- Спробуйте задовольнити вимоги різних зацікавлених сторін.
- Задовольняти як функціональні, так і якісні вимоги.
- Знизити мету власності та покращити ринкову позицію організації.
- Покращте якість і функціональність системи.
- Підвищення зовнішньої довіри до організації чи системи.

Роль розробки архітектури та чому це так важливо?

Розробка архітектури в системі є невід'ємною частиною її життєвого циклу та відіграє ключову роль у процесі створення програмного забезпечення. Цей етап визначає загальну структуру та організацію системи, визначає ключові компоненти, їх взаємодію та способи розв'язання проблем. Розробка архітектури включає в себе вибір архітектурних патернів, стилів та технологій, що забезпечують оптимальні характеристики системи.

Архітектура системи визначає логічну та фізичну структуру, взаємодію компонентів, розподіл ресурсів та механізми забезпечення безпеки та ефективності. Цей етап вирішує ключові питання щодо організації коду, його модульності, забезпечення розширюваності та підтримуваності системи в подальшому.

Крім того, розробка архітектури визначає технічні аспекти, такі як вибір технологій, патернів взаємодії та підходів до розв'язання конкретних завдань. Цей етап забезпечує зручність тестування, впровадження нового функціоналу та підтримує найкращі практики в розробці програмного забезпечення. Усі ці

аспекти сприяють створенню ефективної, стійкої та зручної для розвитку системи. Організована архітектура системи допомагає підтримувати внутрішню якість, що ще більше покращує програмне забезпечення.

У Netflix керувати доступністю дає архітектура мікросервісу, тоді як у Salesforce або Google керована доменом конструкція допомагає їм керувати складною логікою домену. Давайте зрозуміємо це на наступних прикладах. Припустімо, що два подібні продукти були запущені протягом місяця. Через три місяці вони потребують додавання нових функцій.

Зараз є два сценарії:

1. Запуск — 31 травня: код безладний і заплутаний. Користувачі не мають до цього жодного відношення, але відстежити масштаб змін і впровадити їх стало складно.

2. Запуск — 30 червня: код ідеально організований. Користувачі не мають до цього відношення, але команда розробників програмного забезпечення може легко впоратися зі змінами та впровадити їх.

Що б за таких обставин вибрала компанія з розробки програмного забезпечення? Зазвичай, незважаючи на заплутані кодові блоки, команда вибирала більш ранній запуск, оскільки це мало значення на даний момент — чим швидший запуск, тим більше можливостей для монополізації ринку. Однак у другому сценарії зміни займуть час, оскільки якість продуктивності та якість коду мають однакове значення. Це негативно вплине на час виходу на ринок. Але чітко визначена архітектура системи у вигляді мікросервісів допоможе полегшити обслуговування. Ваша компанія не тільки заощадить час, але й задовольнить користувачів швидкими та регулярними оновленнями.

Рис. 3.2. Модель процесу дизайну архітектури майбутньої системи

У висновку маємо ситуацію коли безладний і заплутаний код міг вивести продукт на ринок раніше, але вимагає більше, ніж потрібно, щоб включити нові зміни. Навпаки, організований код, можливо, постраждав від відкладеного запуску, але зрештою забезпечить точні та своєчасні оновлення.

3.2 Триярусна архітектура

Діаграма архітектури веб-додатку представляє макет із усіма програмними компонентами (базами даних, програмами та проміжним програмним забезпеченням) і тим, як вони взаємодіють. Він визначає, як дані доставляються через HTTP, і гарантує, що клієнтська сторона та внутрішні сервери можуть їх зрозуміти. Більше того, це також гарантує наявність дійсних даних у всіх запитах користувачів. Він створює та керує записами, забезпечуючи доступ і автентифікацію на основі дозволів. Вибір правильного дизайну визначає зростання вашої компанії, надійність і сумісність, а також майбутні IT-потреби. Таким чином, важливо розуміти компоненти, що складають архітектуру веб-програм. [20]

Як правило, діаграма архітектури веб-додатку складається з трьох основних компонентів:

1. Веб-браузер: браузер, компонент на стороні клієнта або зовнішній компонент є ключовим компонентом, який взаємодіє з користувачем, отримує вхідні дані та керує логікою презентації, одночасно контролюючи взаємодію користувача з програмою. За потреби введені користувачем дані також перевіряються.

2. Веб-сервер: веб-сервер, також відомий як серверний компонент або компонент на стороні сервера, керує бізнес-логікою та обробляє запити користувачів, направляючи запити до потрібного компонента та керуючи всіма операціями програми. Він може запускати та контролювати запити від широкого кола клієнтів.

3. Сервер бази даних: сервер бази даних надає необхідні дані для програми. Він виконує завдання, пов’язані з даними. У багаторівневій архітектурі за допомогою збережених процедур сервери баз даних можуть керувати бізнес-логікою.

У традиційній 2-рівневій архітектурі є два компоненти: клієнтська система або інтерфейс користувача та серверна система, якою зазвичай є сервер бази даних. Бізнес-логіка включена в інтерфейс користувача або сервер бази даних. Недоліком 2-рівневої архітектури є зниження продуктивності зі збільшенням кількості користувачів. Крім того, безпосередня взаємодія бази даних і пристрою користувача викликає проблеми безпеки. Системи залізничного бронювання та системи керування вмістом — це кілька додатків, які зазвичай будуються з використанням цієї архітектури.

Існує три рівні 3-рівневої архітектури:

- Рівень презентації / рівень клієнта
- Прикладний рівень / бізнес-рівень
- Рівень даних

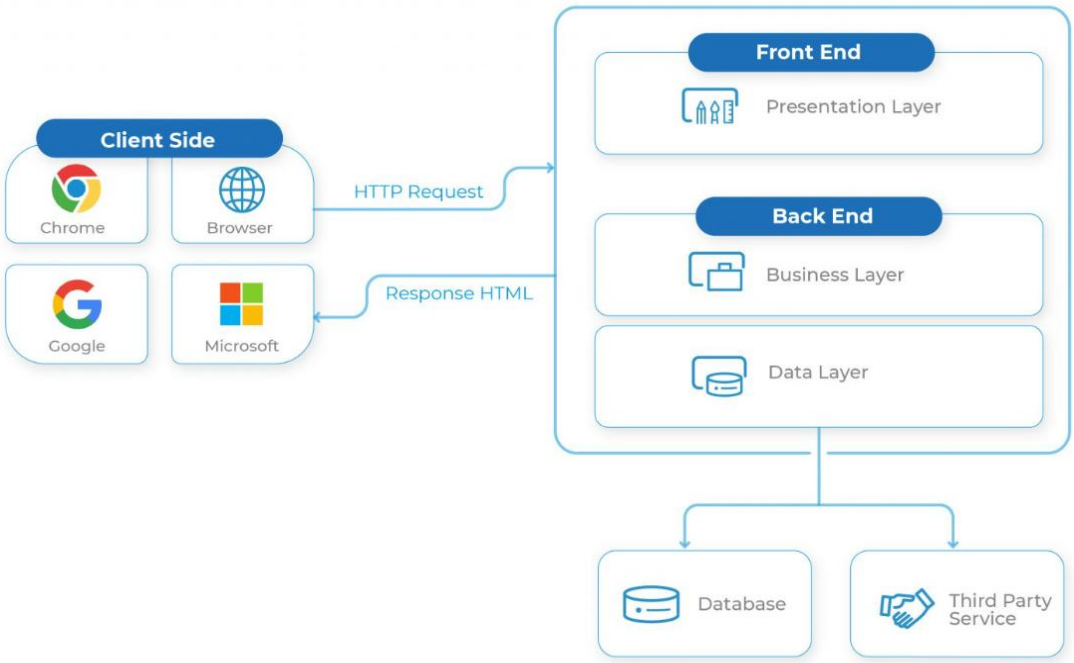


Рис. 3.3. Схема триярусної архітектури

У цій моделі проміжні сервери отримують запити клієнтів і обробляють їх, координуючи роботу з підлеглими серверами, застосовуючи бізнес-логіку. Зв'язок між клієнтом і базою даних управляється проміжним прикладним рівнем, що дозволяє клієнтам отримувати доступ до даних з різних рішень СУБД.

Трирівнева схема архітектури веб-додатків є більш безпечною, оскільки клієнт не має прямого доступу до даних. Розгортання серверів додатків на кількох машинах забезпечує кращу масштабованість, кращу продуктивність і краще повторне використання. Ви можете масштабувати його горизонтально, масштабуючи кожен елемент окремо. Ви можете абстрагувати основний бізнес від сервера бази даних для ефективного балансування навантаження. Цілісність даних покращується, оскільки всі дані проходять через сервер додатків, який вирішує, як і хто має отримати доступ до даних. З цієї причини зміна керівництва є легкою та економічно ефективною. Клієнтський рівень може бути тонким клієнтом, що означає зниження витрат на обладнання. Ця

модульна модель дозволяє змінювати один рівень, не впливаючи на інші компоненти.

Побудова багаторівневої діаграми архітектури сучасної веб-програми допомагає визначити роль кожного компонента та легко вносити зміни до відповідного рівня, не впливаючи на загальну програму. Це дозволяє легко писати, налагоджувати код, керувати ним і повторно використовувати його.

Три рівні діаграми архітектури веб-додатку:

- Рівень презентації / рівень клієнта
- Рівень програми / Рівень бізнес-логіки
- Рівень даних

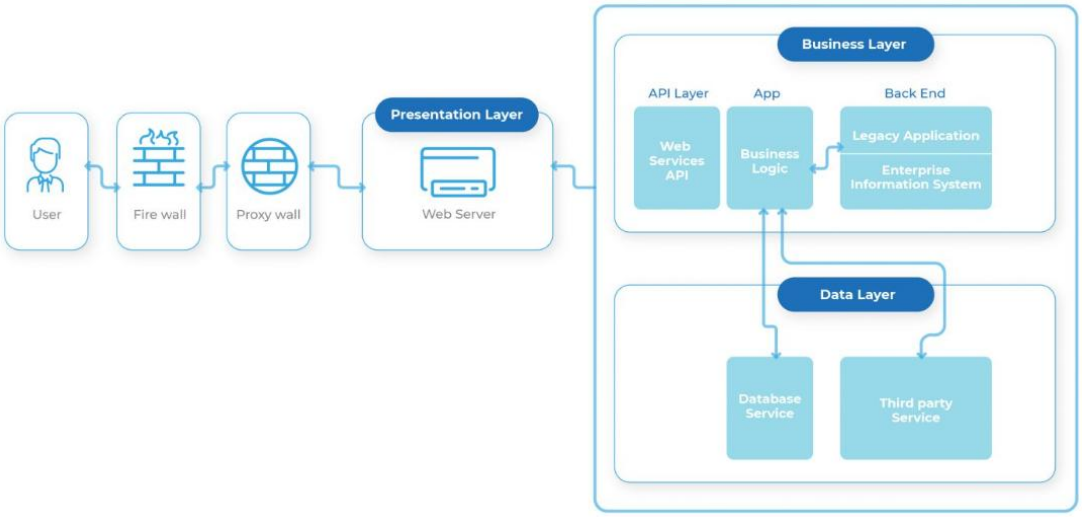


Рис. 3.4. Діаграма рівнів архітектури веб-додатків

3.2.1 Рівень програми: веб-сервер

Що таке веб-сервер? Простіше кажучи, веб-сервер запускає один або кілька веб-сайтів або веб-додатків. Веб-сервер використовує протокол передачі гіпертексту (HTTP) та інші протоколи для перегляду запитів користувачів через браузер. Він обробляє їх, застосовуючи бізнес-логіку та доставляючи

запитуваний вміст кінцевому користувачеві. Веб-сервер може бути апаратним пристроєм або програмним забезпеченням:

- Апаратний веб-сервер: комп'ютерний пристрій, підключений до Інтернету та містить програмне забезпечення веб-сервера та компоненти веб-додатків, такі як зображення, документи HTML, файли JS і таблиці стилів CSS.
- Програмний веб-сервер: це програмне забезпечення, яке розуміє URL-адреси та протоколи HTTP. Користувачі можуть отримати доступ до нього через доменні імена, щоб отримати запитаний вміст.

У той час як статичний веб-сервер доставляє вміст у браузер таким, яким він є, динамічний веб-сервер оновлює дані перед їх доставкою.

Apache — популярний веб-сервер із відкритим кодом від Apache Software Foundation. Він був розроблений Робертом Маккулом у C та XML у 1995 році. Apache базується на моделі, керованій процесом, у якій кожен запит призводить до створення нового потоку. Модульний дизайн Apache дозволяє легко масштабувати окремі ресурси. З мінімальною конфігурацією ви можете керувати навіть великим трафіком. Він працює в середовищах MacOS, Windows і Linux. Однак Linux є найбільш бажаним середовищем для Apache.

Хоча він використовує файлову систему для обробки статичного вмісту, динамічний вміст обробляється на сервері. Використовуючи файли .htaccess, ви можете виконувати додаткові налаштування параметрів сервера. Безпека хороша. Він пропонує підтримку через IRC, Stack Overflow і списки розсилки.

NGINX — ще один популярний веб-сервер під назвою «Engine X». NGINX, розроблений Ігорем Сисоєвим у 2004 році, швидко став популярним. Він працює за моделлю, керованою подіями, в якій тисячі запитів обробляються в одному потоці, надаючи більше з мінімальними ресурсами. Він використовує PHP для обслуговування статичних ресурсів і обслуговує статичний вміст у 2,5 рази швидше, ніж Apache. Динамічний вміст подається через зовнішні процеси. Що стосується інтерпретації запитів, Apache передає розташування файлової

системи, тоді як NGINX передає URI. Ця функція розширює можливості NGINX як балансувальника навантаження, кешу HTTP та проксі-сервера.

Хоча він підтримує ОС Unix, сумісність з Windows обмежена. Ви не можете зробити додаткові налаштування. Менша кодова база забезпечує кращу безпеку. Динамічні модулі не підтримуються. Разом зі списками розсилки та IRC також доступний форум. NGINX має перевагу над Apache, оскільки виконує функції веб-сервера та проксі-сервера. Підхід, керований подіями, який обробляє тисячі запитів в одному потоці, забезпечує більшу продуктивність, швидкість і економічну ефективність.

3.2.2 Рівень презентації: клієнтський компонент (Front-End)

Рівень презентації, відомий як клієнтський компонент або фронт-енд, представляє собою ключовий елемент багатьох програмних систем. Його головна мета - створення інтерфейсу, який взаємодіє з користувачем, тобто тим, хто використовує програму. Цей рівень відповідає за візуальний аспект системи, включаючи вигляд, поведінку та взаємодію з користувачем. Розробники клієнтського компонента прагнуть створити інтуїтивний та ефективний інтерфейс, спроектований для максимально зручного використання.

Основна функція клієнтського компонента - відображення інформації користувачу в зручному та естетичному вигляді. Різноманіття елементів управління, від кнопок та полів вводу до складніших графічних компонентів, створюють можливість взаємодії з програмою. Навички дизайну, ергономіки та психології користувача стають важливими на цьому етапі, оскільки вони визначають, наскільки користувачеві легко орієнтуватися та використовувати систему.

Покращення продуктивності та зручності використання вимагають постійного вдосконалення клієнтського компонента. Застосування новітніх технологій, оптимізація завантаження сторінок та удосконалення анімаційних ефектів - усе це спрямовано на підвищення задоволення користувачів від використання програми. Взаємодія із клієнтським компонентом стає першочерговим інтерфейсом для кінцевого користувача, і тому його розробка вимагає не лише технічної експертизи, але й ретельного вивчення потреб та побажань аудиторії.

На рівні презентації, клієнтський компонент також відповідає за обробку введених даних користувача та передачу їх на серверний рівень для подальшої обробки. У такий спосіб, цей рівень виступає важливим посередником між користувачем і іншими компонентами системи. Сучасні технології, такі як бібліотеки та фреймворки, значно полегшують розробку клієнтського компонента, дозволяючи швидко впроваджувати інноваційні рішення та вдосконалення. Зміни в цьому рівні можуть суттєво впливати на загальний вигляд і функціональність продукту, тому важливо враховувати сучасні тенденції та вимоги користувачів при його розробці.

У сучасному розробництві клієнтського компонента, або фронт-енду, використовують різноманітні мови програмування та технології для досягнення оптимальної продуктивності та високоякісного взаємодії з користувачем. JavaScript залишається основною мовою для створення динамічних та інтерактивних веб-сторінок, а його розширення, такі як TypeScript, забезпечують підтримку строгих типів та полегшують розробку масштабних проектів. Для верстки і стилізації використовують мови, такі як HTML та CSS, а також їх сучасні розширення, наприклад, HTML5 та CSS3, щоб забезпечити більше можливостей та анімацій. Фреймворки, такі як React, Angular та Vue.js, дозволяють ефективно організовувати код та створювати складні інтерфейси. Для роботи з візуальними компонентами і графікою використовують бібліотеки, наприклад, D3.js або Three.js. Це лише кілька прикладів, і вибір

конкретних технологій може залежати від вимог проекту та вподобань розробників.

3.2.3 Рівень програми: серверний компонент (Back-End)

Рівень програми, або серверний компонент (Back-End), відіграє важливу роль у забезпеченні функціональності та обробці даних на веб-сайті чи додатку. Однією з ключових задач цього рівня є забезпечення ефективної комунікації між клієнтом та сервером. Для цього використовуються різні протоколи, такі як HTTP або WebSocket, які дозволяють обмінюватися інформацією між фронт-ендом та бек-ендом.

У високоякісних архітектурах серверний компонент побудований з урахуванням принципів масштабованості та високої доступності. Велику увагу приділяють обробці запитів, забезпечуючи їх ефективність та оптимізацію ресурсів. Кешування інформації, оптимізація баз даних та використання кластеризації є популярними стратегіями для досягнення високої продуктивності.

Важливою частиною серверного компонента є безпека. Заходи безпеки, такі як автентифікація, авторизація та захист від атак, використовуються для забезпечення конфіденційності та цілісності даних. Розробники також можуть імплементувати протоколи шифрування для захисту інформації під час передачі через мережу.

Нарешті, розширюваність є важливим аспектом серверного компонента. Системи повинні легко адаптуватися до зростання обсягу даних та користувацького трафіку. Використання мікросервісної архітектури чи контейнеризації може полегшити масштабування та управління ресурсами.

У рівні програми, який представляє серверний компонент (Back-End), інтеграція з базами даних відіграє важливу роль у забезпеченні ефективного зберігання та управління даними. Використання оптимальних моделей баз даних, таких як реляційні або NoSQL бази даних, залежить від конкретних потреб проекту. Розробники також можуть використовувати ORM (Object-Relational Mapping) для полегшення взаємодії з базою даних і використання мови програмування для операцій з даними.

Однією з важливих характеристик є побудова серверного компонента так, щоб він був готовий до інтеграції з іншими сервісами чи зовнішніми API. Це дозволяє розширювати функціональність системи за допомогою взаємодії з різноманітними сервісами та інструментами.

У контексті безпеки, серверний компонент відповідає за захист конфіденційності даних та управління правами доступу. Використання хешування паролів, механізмів автентифікації та імплементація принципів найменших привілеїв є ключовими елементами безпеки серверного рівня.

Нарешті, серверний компонент може використовувати різні стратегії кешування для поліпшення продуктивності та зменшення навантаження на базу даних. Розробники мають можливість вибрати оптимальний механізм кешування відповідно до особливостей додатку та його завдань.

3.2.4. Рівень програми: інтерфейс прикладного програмування (API)

Інтерфейс прикладного програмування (API) — це не технологія, це концепція, яка дозволяє розробникам отримувати доступ до певних даних і функцій програмного забезпечення. Простіше кажучи, це посередник, який дозволяє програмам спілкуватися один з одним. Він містить протоколи, інструменти та визначення підпрограм, необхідні для створення програм.

Наприклад, коли ви входите в програму, програма викликає API, щоб отримати дані вашого облікового запису та облікові дані. Програма зв’яжеться з відповідними серверами, щоб отримати цю інформацію та повернути ці дані програмі користувача. Веб-API – це API, доступний в Інтернеті через протокол HTTP. Його можна створити за допомогою таких технологій, як .NET і Java.

Завдяки API розробникам не потрібно створювати все з нуля, а використовувати наявні функції, представлені як API, щоб підвищити продуктивність і швидше вийти на ринок. Зменшуючи витрати на розробку, API значно знижують витрати на розробку. Це також покращує співпрацю та підключення в екосистемі, покращуючи взаємодію з клієнтами.

Існують різні типи API:

- RESTful API: Representational State Transfer API у легкому форматі JSON. Він дуже масштабований, надійний і забезпечує високу продуктивність, що робить його найпопулярнішим API.
- SOAP: простий протокол доступу до об’єктів використовує XML для передачі даних. Це вимагає більшої пропускної здатності та розширеної безпеки
- XML-RPC: розширювана мова розмітки – виклики віддалених процедур використовують спеціальний формат XML для передачі даних
- JSON-RPC: для передачі даних використовується формат JSON

3.2.5 Рівень даних: база даних

База даних — це ключовий компонент веб-програми, який зберігає та керує інформацією для веб-програми. Використовуючи функцію, ви можете шукати, фільтрувати та сортувати інформацію на основі запиту користувача та надавати необхідну інформацію кінцевому користувачеві. Вони дозволяють доступ на основі ролей для підтримки цілісності даних.

Вибираючи базу даних для архітектури веб-програми, розмір, швидкість, масштабованість і структура – це чотири аспекти, які вимагають вашої уваги. Для структурованих даних хорошим вибором є бази даних на основі SQL. він підходить для фінансових програм, де цілісність даних є ключовою вимогою.

Для обробки неструктурованих даних NoSQL є хорошим варіантом. Він підходить для програм, у яких характер вхідних даних непередбачуваний. Бази даних Key Value пов’язують кожне значення з ключем і підходять для баз даних, які зберігають профілі користувачів, відгуки, коментарі в блогах тощо. Для аналітики бази даних Wide Column є хорошим вибором.

3.3 Рекомендації щодо архітектури веб-додатків

Розробка архітектури веб-додатків є критичним етапом у створенні ефективного та надійного продукту. [22] Нижче наведено рекомендації, які можуть виявитися корисними при цьому процесі:

Масштабований веб-сервер

Масштабований веб-сервер має вирішальне значення для забезпечення сталої продуктивності додатка незалежно від кількості одночасних користувачів, місця та часу. Існує три типи параметрів масштабування: горизонтальне, вертикальне та діагональне. У той час як вертикальне масштабування стосується оновлення/зменшення конфігурації пристрою, горизонтальне масштабування стосується збільшення/зменшення кількості пристроїв. Діагональне масштабування — це поєднання обох моделей. Рекомендовано вибрати модель горизонтального масштабування, оскільки ви не будете обмежені конфігурацією чи кількістю серверів. Крім того, ви можете комбінувати вертикальне масштабування, де це можливо.

Адаптуйте хмару за допомогою еластичної інфраструктури оскільки гібридні та багатохмарні середовища стають все більш популярними, адаптація

до хмари та проактивне надання ресурсів є ключовими для створення високопродуктивних веб-програм. Еластична інфраструктура включає попередньо налаштовані мережеві системи, сервери віртуальних машин, сховище та обчислювальні ресурси, що дозволяє легко керувати середовищем за допомогою порталів самообслуговування. Це забезпечує гнучкість для швидкої адаптації до мінливих потреб ринку та очікувань клієнтів.

Незмінна інфраструктура

Простіше кажучи, незмінну інфраструктуру неможливо змінити після розгортання. Це дозволяє адміністраторам автоматично надавати ресурси за допомогою коду. Коли сервери потрібно оновити або змінити, вони автоматично замінюються новішими.

Конфігураційний дрейф є великою проблемою для змінної інфраструктури. Проблеми з масштабуванням і налагодженням під час тиражування виробничого середовища ускладнюють цю проблему. Незмінна інфраструктура використовує перевірений і контрольований за версіями образ для надання нових серверів для кожного розгортання. Отже, попередній стан сервера не викликає занепокоєння. Ви можете протестувати сервери перед їх розгортанням. Він усуває конфігураційні дрейфи та дозволяє горизонтальне масштабування, пропонуючи простий відкат і відновлення з узгодженими середовищами проміжки.

Мікросервіс і безсерверний підхід

Мікросервіси та безсерверні обчислення мають вирішальне значення для розробки веб-додатків. Однак різниця полягає в тому, що архітектура мікросервісів пропонує довгострокове рішення з високою масштабованістю, тоді як безсерверні обчислення пропонують ефективність коду. Безсерверні функції запускаються лише тоді, коли вони запускаються.

Поєднуючи обидві моделі, ви можете отримати найкраще з обох світів. Ви можете використовувати AWS Step Functions, щоб призначити тригери, щоб

об’єднати кілька функцій у службу та призначити їм тригери. За допомогою мікросервісів, ініційованих подіями, ви можете побудувати комбіновану систему для підвищення ефективності коду, довгострокової стабільності, економічної ефективності та масштабованості.

Висновок

В третьому розділі були було розглянуто та описано внутрішню будову нашого продукту. За основу була обрана триярусна архітектура, а також проведений аналіз цієї архітектури. Вибір триярусної архітектури для веб-додатку є обґрунтованим та забезпечить високу ефективність, розширюваність та надійність системи. Даний підхід передбачає розділення додатку на три основні рівні: клієнтський (front-end), серверний (back-end) та рівень даних. Клієнтський рівень відповідає за взаємодію з користувачем і відображення інформації. Використання технологій, таких як HTML, CSS, та JavaScript, дозволяє створювати дружелюбний та інтуїтивно зрозумілий інтерфейс, що підвищує задоволення користувачів. Серверний рівень відповідає за обробку бізнес-логіки та взаємодію з базою даних. Застосування відповідних мов програмування та фреймворків дозволяє оптимізувати роботу сервера та забезпечує швидке виконання запитів. Рівень даних є фундаментальною складовою системи, забезпечуючи зберігання та обробку інформації. Використання певних типів баз даних та їх оптимальне конфігурування гарантує ефективне управління даними. В цілому, триярусна архітектура дозволяє створити збалансовану та добре структуровану систему, яка відповідає вимогам бізнесу та забезпечує високий рівень користувацького досвіду. Однак важливо продовжувати вдосконалювати архітектуру, враховуючи зростання обсягів даних та нові вимоги, що можуть виникнути з плином часу. Сформувавши вимоги, розробивши архітектуру та проаналізувавши рекомендації можна впевнено переходити до реалізації системи.

РОЗДІЛ 4

РЕАЛІЗАЦІЯ СИСТЕМИ

4.1 Інструментарій для розробки системи

В цьому розділі будуть описані інструменти та технології які будуть використані для розробки нашої системи.

4.1.1 Java Script

JavaScript – це високорівнева, інтерпретована мова програмування, що використовується для створення динамічних та інтерактивних веб-сайтів, додавання функціоналу до HTML-сторінок та забезпечення користувачам ефективного інтерфейсу. Синтаксис мови є зрозумілим і схожим на інші мови програмування, що полегшує її вивчення. Вона підтримується всіма сучасними веб-браузерами, що робить її однією з ключових технологій для фронтенд-розробників. [14]

JavaScript дозволяє розробникам створювати асинхронні застосунки, обробляти події та взаємодіяти з DOM, що робить її ідеальною для створення сучасних та відзначених інтерфейсів користувача. Однією з ключових особливостей JavaScript є можливість використання на обох рівнях: клієнтському (front-end) та серверному (back-end). На клієнтському рівні JavaScript відповідає за створення динамічних ефектів, анімації та взаємодії користувача, тоді як на серверному рівні використовується у платформі Node.js для обробки запитів та взаємодії з базою даних.

Багатофункціональність JavaScript доповнюється різноманіттям бібліотек та фреймворків, таких як React, Angular, Vue.js, які сприяють розробці високопродуктивних та модульних веб-додатків. Ці інструменти дозволяють розробникам швидко та ефективно створювати високопродуктивні та модульні застосунки з сучасним інтерфейсом користувача.

JavaScript взаємодіє з різними API та технологіями, що дозволяє створювати різноманітні функції, такі як геолокація, робота з веб-камерою, масштабування графіки тощо. Це робить мову важливою для сучасних веб-застосунків, що стикаються з різноманітними вимогами користувачів. JavaScript вигідно відрізняється асинхронністю, яка дозволяє виконувати неблокуючі операції та ефективно обробляти події. Це робить мову ідеальною для веб-розробки, де відсутність затримок дуже важлива.

Однією з переваг JavaScript є те, що вона постійно розвивається. Стандарт ECMAScript регулярно оновлюється, додаючи нові можливості та покращення. Розробники можуть користуватися останніми досягненнями технологій та інновацій в галузі програмування. За всю свою історію мова змінилася від простої скриптової мови до потужного інструменту для створення різноманітних застосунків, і її роль у веб-розробці залишається невід'ємною.

Ця мова знаходить застосування не тільки у веб-розробці, але і у розробці мобільних додатків та інтернету речей (IoT). Вона є ключовим елементом технологічного прогресу та постійно адаптується до зростаючих потреб сучасного програмування. JavaScript використовується у сучасних трендах, таких як прогресивні веб-застосунки (PWA) та single-page applications (SPA), що підкреслює його актуальність та роль у сфері веб-розробки.

Для розробки системи організації управління робіт ФОП за допомогою JavaScript можна використовувати різноманітні технології та інструменти. Нижче наведені аспекти системи, та які технології допоможуть нам у розробці:

1. Front-end розробка:

- React або Vue.js: Використання бібліотек React або Vue.js дозволяє створювати ефективний та взаємодіючий користувацький інтерфейс для системи. Ці бібліотеки полегшують структуру і розгортання компонентів і дозволяють ефективно керувати станом додатку.

2. Back-end розробка:

- Node.js з Express або Nest.js: Використання Node.js для створення серверної частини системи дозволяє використовувати JavaScript на обох рівнях

– клієнтському та серверному. Express або Nest.js – це потужні фреймворки для розробки серверної логіки.

3. Бази даних:

- MongoDB або MySQL: В залежності від потреб системи, ви можете вибрати між NoSQL базами даних, такими як MongoDB, або реляційними базами, наприклад MySQL. Вони забезпечують можливість зберігання та управління даними вашого додатку.

4. Аутентифікація та авторизація:

- JSON Web Tokens (JWT): Для забезпечення безпеки системи та реалізації механізму аутентифікації та авторизації, використання JWT може бути зручним вибором. Вони дозволяють зберігати інформацію про користувача у вигляді токенів.

5. Комунікація з сервером:

- AJAX або Fetch API: Для взаємодії з сервером та обміну даними можна використовувати AJAX або Fetch API. Вони дозволяють асинхронно завантажувати дані із сервера, не перезавантажуючи сторінку.

6. Real-time взаємодія:

- Socket.io: Якщо система вимагає реального часу для спілкування з користувачами або оновлення даних, використання бібліотеки Socket.io дозволяє зручно реалізувати двосторонню комунікацію між клієнтом та сервером.

7. Тестування:

- Jest або Mocha: Для забезпечення якості коду та виявлення помилок можна використовувати фреймворки для тестування, такі як Jest або Mocha.

8. Документація:

- Swagger або YARD: Для зручного документування API системи можна використовувати інструменти, такі як Swagger або YARD.

9. Інструменти розробки:

- Visual Studio Code: Як потужний та зручний редактор коду, Visual Studio Code є популярним вибором серед розробників JavaScript-програм.

10. Управління залежностями:

- npm або Yarn: Для ефективного управління залежностями та пакетами JavaScript проекту використовують npm або Yarn.

4.1.2 HTML та CSS

HTML розшифровується як HyperText Markup Language. Гіпертекст — це фактично текст у тексті, а мова розмітки — це мова, зрозуміла комп'ютерам, призначена для опису веб-сторінок і робить текст, який ви використовуєте, більш інтерактивним. [23]

Його було винайдено ще в 1989 році як мову публікації в Інтернеті, і, простіше кажучи, це перший будівельний блок для створення веб-сторінки. Три основні з них - це елементи, теги та атрибути. HTML5 — це остання «версія» HTML, і знання в цій галузі є зростаючою тенденцією на ринку праці. Загалом, він має таку саму функціональність, як стандартний HTML, але набагато динамічніший і використовує набагато менше коду для створення чогось комплексного.

По суті, HTML використовується для створення основного вмісту веб-сторінки, надання їй структури. Ви починаєте з написання слів, потім до цих слів додаєте теги або елементи. Потім веб-браузер читає це й може зрозуміти заголовок сторінки, будь-які абзаци, а також те, де сторінка починається й закінчується, таким чином наповнюючи вашу веб-сторінку вмістом.

HTML підтримується кожним окремим браузером і встановлено майже на кожній існуючій веб-сторінці. Для використання не потрібні жодні ліцензії, не потрібно платити за це, і його можна досить легко вивчити та кодувати. Якщо ми можемо порівняти веб-сторінку з тілом людини, то HTML – це кістки тіла.

CSS — це абревіатура від Cascade Styling Sheets. Коротше кажучи, це мова стилю аркуша, яка є типом мови, який можна використовувати для опису подання мови розмітки – у цьому випадку для опису рухів HTML. Він

ефективно визначає, як будівельні блоки, закладені HTML, оформлені та представлені користувачеві.

CSS був вперше створений приблизно в 1996 році, щоб зрозуміти HTML і зробити веб-сторінку виглядати та відчувати себе чудово. Як і в будь-якій мові, ви повинні писати CSS, і знання того, як ми пишемо CSS, є навиком, яким повинен володіти будь-який розробник веб-сторінок. Простий в обслуговуванні, CSS є другою частиною двокомпонентного набору інструментів для створення веб-сторінок.

Веб-доступність дійсно важлива для багатьох роботодавців і компаній сьогодні, і існує жорстка конкуренція у створенні найдоступнішої та добре розробленої веб-сторінки. Вивчення деяких основ розробки програмного забезпечення за допомогою CSS може допомогти вам навчитися створювати доступні веб-сторінки.

Якщо HTML — це кістки тіла, то CSS — це шкіра, яка його покриває. Він використовує для кольору фону, стилю, макету, рамок, затінення – усіх важливих елементів дизайну, за допомогою яких веб-сторінка виглядає вишуканою та вишуканою. CSS дозволяє відрізнити презентацію від вмісту, змінюючи дизайн і відображаючи елементи HTML.

Презентація та простота використання — це кілька основних речей, які CSS додав у веб-дизайні, транслюючи те, як виглядає вміст на веб-сторінці та що ще на ньому доповнює цей вміст. Хоча часто використовується в поєднанні з HTML, він фактично не відрізняється від нього і може використовуватися з будь-якою мовою розмітки на основі XML.

Технології HTML, CSS і JavaScript можуть бути використані в системі організації управління робіт ФОП для таких завдань: HTML визначає структуру інтерфейсу, CSS стилізує та оформлює його, а JavaScript забезпечує динамічні можливості та взаємодію з користувачем. JavaScript дозволяє реалізувати валідацію форм, асинхронне завантаження даних та інші динамічні функції, що поліпшують ефективність та зручність використання системи. Такий комплекс веб-технологій створює повноцінний інтерфейс, допомагаючи забезпечити

оптимальне управління роботами ФОП та задовольняючи раніше визначені потреби користувачів.

4.1.3 React

React - це бібліотека для створення інтерфейсів користувача, яка дозволяє розробникам побудувати ефективні та динамічні веб-додатки. Основними характеристиками React є компонентний підхід, віртуальний DOM та одностороння потік даних. [17]

React дозволяє створювати перевикористовувані компоненти, які визначають вигляд та поведінку частини користувацького інтерфейсу. Це полегшує підтримку та розширення коду, що є важливим для розробки складних систем.

Віртуальний DOM використовується для ефективного оновлення інтерфейсу, зменшуючи кількість фактичних маніпуляцій з реальним DOM. Це призводить до покращеної продуктивності додатка та відчутного користувацького досвіду.

Односторонній потік даних дозволяє контролювати потік даних у додатку, забезпечуючи структурований та передбачуваний спосіб управління станом.

У контексті розробки системи організації управління робіт ФОП, React допомагає створити сучасний та ефективний користувацький інтерфейс. Завдяки компонентному підходу можна легко взаємодіяти з різними елементами системи, такими як списки завдань, фільтри чи статуси. Віртуальний DOM сприяє швидкій відповіді інтерфейсу на зміни, що може бути важливим для динамічного управління завданнями та роботами. Односторонній потік даних робить управління станом додатку більш передбачуваним, що полегшує роботу з різними даними та їх оновленням у реальному часі.

У розробці подібної системи за допомогою React, можна використати низку корисних класів та структур:

1. Компоненти: Основним класом у React є компонент. Ви можете створювати власні компоненти для представлення різних частин інтерфейсу, таких як завдання, фільтри, меню тощо.

```
class TaskList extends React.Component {
  render() {
    return (
      <div>
        {/* Ваш вміст компонента */}
      </div>
    );
  }
}
```

2. Стейт і Пропси: Використовуйте стейт та пропси для управління даними та їх передачі між компонентами. Стейт дозволяє зберігати та оновлювати дані в компоненті, а пропси передаються від батьківських компонентів.

```
class Task extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      status: 'inProgress',
    };
  }
  render() {
    return (
      <div>
        {this.props.title} - {this.state.status}
```

```
      </div>
    );
  }
}
```

3. Життєвий цикл компонента: Використовуйте методи життєвого циклу (наприклад, `componentDidMount`, `componentDidUpdate`) для виконання дій при монтуванні та оновленні компонента.

```
class TaskList extends React.Component {
  componentDidMount() {
    // Викликається після монтування компонента
    console.log('Компонент монтується');
  }
  render() {
    return (
      <div>
        {/* Ваш вміст компонента */}
      </div>
    );
  }
}
```

4. Хуки: Використовуйте хуки, якщо ви працюєте з функціональними компонентами. Наприклад, `useState` для управління станом.

```
import React, { useState } from 'react';
function TaskForm() {
  const [taskTitle, setTaskTitle] = useState('');
  const handleInputChange = (event) => {
    setTaskTitle(event.target.value);
```

```
  };

  return (
    <form>
      <input type="text" value={taskTitle} onChange={handleInputChange} />
      <button type="submit">Додати завдання</button>
    </form>
  );
}
```

5. Контекст: Використовуйте контекст для передачі даних глибоко вниз по дереву компонентів без явної передачі пропсів.

```
const ThemeContext = React.createContext('light');

function ThemedComponent() {
  return (
    <ThemeContext.Consumer>
      {(theme) => <div>Поточна тема: {theme}</div>}
    </ThemeContext.Consumer>
  );
}
```

Ці класи та структури є основними будівельними блоками для розробки інтерфейсу системи організації управління робіт ФОП за допомогою React. Розуміння їхнього використання дозволить ефективно створювати функціональні та динамічні компоненти.

4.1.4 Node.js

Node.js — це кросплатформне середовище виконання JavaScript із відкритим кодом, яке може працювати в Windows, Linux, Unix, macOS тощо. Node.js працює на механізмі JavaScript V8 і виконує код JavaScript поза веб-браузером. [16]

Node.js дозволяє розробникам використовувати JavaScript для написання інструментів командного рядка та сценаріїв на стороні сервера. [3] (“Встановлення Node.js”) Можливість запуску коду JavaScript на сервері часто використовується для створення динамічного вмісту веб-сторінки перед тим, як сторінка надсилається у веб-браузер користувача. Отже, Node.js представляє парадигму «JavaScript скрізь», об’єднуючи розробку веб-додатків навколо однієї мови програмування, на відміну від використання різних мов для програмування на стороні сервера чи клієнта.

Node.js дозволяє створювати веб-сервери та мережеві інструменти за допомогою JavaScript і колекції «модулів», які обробляють різні основні функції. Надаються модулі для введення/виведення файлової системи, мереж (DNS, HTTP, TCP, TLS/SSL або UDP), двійкових даних (буферів), функцій криптографії, потоків даних та інших основних функцій. Модулі Node.js використовують API, призначений для зменшення складності написання серверних програм.[4]

JavaScript — єдина мова, яку Node.js підтримує нативно, але доступно багато мов для компіляції в JS. У результаті програми Node.js можна писати на CoffeeScript, Dart, TypeScript, ClojureScript та інших.

Node.js в основному використовується для створення мережевих програм, таких як веб-сервери. Найсуттєвіша відмінність між Node.js і PHP полягає в тому, що більшість функцій у PHP блокуються до завершення (команди виконуються лише після завершення попередніх команд), тоді як функції Node.js не блокуються (команди виконуються одночасно або навіть паралельно,

і використовувати зворотні виклики, щоб сигналізувати про завершення або невдачу).

Node.js дозволяє розробляти швидкі веб-сервери в JavaScript за допомогою програмування, керованого подіями. Розробники можуть створювати масштабовані сервери без використання потоків за допомогою спрощеної моделі, яка використовує зворотні виклики для сигналізації про завершення завдання. Node.js поєднує простоту мови сценаріїв (JavaScript) із потужністю мережевого програмування Unix.

Node.js було побудовано на основі механізму Google V8 JavaScript, оскільки він був відкритий за ліцензією BSD, і він містить повну підтримку фундаментальних протоколів, таких як HTTP, DNS і TCP. Існуюча популярність JavaScript зробила Node.js доступним для спільноти веб-розробників.

В контексті розробки системи організації управління робіт ФОП, Node.js має кілька переваг та можливостей:

1. Ефективність та швидкодія: Node.js використовує асинхронний та подієвий підхід, що дозволяє обробляти багато операцій одночасно. Це особливо важливо для обробки багатокористувацьких запитів у системах управління робітами.

2. Широкі можливості для реалізації серверної логіки: Node.js надає можливість розробляти серверні додатки, включаючи API та обробники запитів, що дозволяє нам створювати потужні та масштабовані серверні частини системи.

3. Використання однієї мови (JavaScript) на обох сторонах: Node.js дозволяє використовувати JavaScript як на стороні клієнта, так і на стороні сервера. Це спрощує розробку та забезпечує більш однорідне середовище для розробників.

4. Модульність та велика кількість пакетів: Node.js має велику кількість доступних пакетів (за допомогою npm), які дозволяють швидко впроваджувати

різноманітні функції, такі як робота з базами даних, обробка запитів, автентифікація тощо.

5. Підтримка WebSocket для реального часу: Управління роботами може вимагати функцій реального часу, таких як сповіщення чи оновлення стану завдань. Node.js підтримує технологію WebSocket, що полегшує реалізацію таких функцій.

Використання Node.js у сполученні з іншими технологіями та фреймворками може допомогти створити потужну та ефективну серверну частину системи управління робітами ФОП.

4.1.5 SQL та NoSQL Server

При виборі сучасної бази даних одним із найважливіших рішень є вибір реляційної (SQL) або нереляційної (NoSQL) структури даних. Обидві системи пропонують унікальні переваги та задовольняють різні потреби, що робить вибір між ними вирішальним для оптимального керування даними.

SQL, або мова структурованих запитів, — це мова програмування з традиційним підходом, який дозволяє реляційним базам даних, які моделюють попередньо визначені схеми, керувати структурованими даними, такими як рядки та таблиці. З іншого боку, NoSQL, що означає «Не тільки SQL», пропонує більш гнучкий, нереляційний підхід, ідеальний для обробки неструктурованих або динамічних даних. Оскільки бізнес розвивається, а дані стають дедалі різноманітнішими, важливо розуміти основні відмінності між SQL і NoSQL. [24]

SQL — це доменно-спеціальна мова, яка використовується для запитів і керування даними. Він працює, дозволяючи користувачам запитувати, вставляти, видаляти та оновлювати записи в реляційних базах даних. SQL також дозволяє застосовувати складну логіку за допомогою транзакцій і вбудованих процедур, таких як збережені функції або представлення.

NoSQL означає не тільки SQL. Це тип бази даних, яка використовує нереляційні структури даних, такі як документи, бази даних графів і сховища ключ-значення для зберігання та отримання даних. Системи NoSQL розроблені так, щоб бути більш гнучкими, ніж традиційні реляційні бази даних, і можуть легко масштабуватися відповідно до змін у використанні або навантаженні. Це робить їх ідеальними для використання в додатках

Рішення про те, який тип бази даних використовувати – SQL чи NoSQL – залежатиме від конкретних потреб і вимог проекту. Наприклад, якщо вам потрібна швидка, масштабована та надійна база даних для веб-додатків, система NoSQL може бути кращою. З іншого боку, якщо ваша програма вимагає складних запитів даних і підтримки транзакцій, система SQL може бути кращим вибором. Зрештою, не існує універсального рішення – все зводиться до того, що вам потрібно від вашої бази даних і який тип системи може забезпечити це найефективнішим способом. Перш ніж прийняти рішення, краще ретельно вивчити обидва варіанти.

У багатьох випадках NoSQL віддають перевагу над SQL, оскільки він пропонує більшу гнучкість і масштабованість. Основна перевага використання системи NoSQL полягає в тому, що вона надає розробникам можливість швидко й легко зберігати та отримувати доступ до даних без накладних витрат на традиційну реляційну базу даних. У результаті команди розробників можуть зосередитися на швидшому забезпеченні функцій і основної бізнес-логіки, не турбуючись про реалізацію основного сховища даних.

NoSQL пропонує низку функцій. Нижче наведені кілька прикладів цих функцій які допоможуть у розробці системи організації управління робіт ФОП:

1. Гнучка схема даних: NoSQL бази даних дозволяють зберігати дані без строгої схеми, що полегшує розширення та модифікацію структури даних в майбутньому. Це особливо важливо в умовах вимог та потреб бізнесу, що постійно змінюються.

2. Горизонтальне масштабування: NoSQL бази даних призначені для роботи з великими обсягами даних та можуть легко масштабуватися

горизонтально, додаючи нові сервери або вузли, щоб забезпечити високу доступність та продуктивність.

3. Ключ-значення (Key-Value) модель: Використання ключ-значення дозволяє швидко знаходити та отримувати доступ до даних за допомогою унікальних ключів. Це корисно для зберігання та отримання інформації про робіт ФОП, наприклад, за номером ідентифікації клієнта чи проєкту.

4. Документальна модель: Схожа на ключ-значення, але здатна зберігати більш складні дані у вигляді документів (наприклад, у форматі JSON або BSON). Це особливо ефективно для зберігання та організації даних у вигляді документів, які можуть містити різні поля та вкладені структури.

5. Графова модель: Ідеальна для вирішення задач, пов'язаних із складними взаємозв'язками між об'єктами, які можна відобразити у вигляді графа. Наприклад, для відстеження взаємодій між різними суб'єктами та подіями у системі управління роботами ФОП.

Ці функції NoSQL баз даних можуть допомогти побудувати ефективну та легко масштабовану систему організації управління робіт ФОП, забезпечуючи оптимальну продуктивність та гнучкість в роботі з даними.

4.1.6 MongoDB

MongoDB є NoSQL базою даних, що використовує документальну модель для зберігання даних у форматі BSON (Binary JSON). Вона використовує JSON-подібні документи для організації та зберігання даних, дозволяючи вкладати один документ в інший. Гнучкість схеми дозволяє додавати нові поля до документів без модифікації існуючих записів. MongoDB підтримує горизонтальне масштабування за рахунок розподілення даних на кілька серверів. Забезпечує індексацію для швидкого доступу до даних та геопросторові можливості для обробки географічних даних. Також надає агрегаційні фреймворки для виконання складних операцій агрегації та обробки даних агрегаційними пайплайнами. MongoDB знаходить застосування в

розробці веб-додатків, систем управління ресурсами та інших областях, де важливо ефективно зберігати та обробляти дані. [27]

MongoDB володіє також механізмами для забезпечення високої доступності та надійності даних за допомогою реплікації, де дані розміщуються на кількох вузлах, і шардування, яке дозволяє розділити великі обсяги даних на менші частини для оптимізації роботи з ними. Крім того, MongoDB підтримує вбудовані можливості для текстового пошуку та індексації, що полегшує реалізацію функцій пов'язаних із пошуком в додатках. Із застосуванням MongoDB, розробники можуть ефективно впроваджувати функціональність бази даних у свої додатки, забезпечуючи швидкий та масштабований доступ до даних.

MongoDB відмінно підходить для системи організації управління робіт ФОП завдяки своїй гнучкості та здатності працювати з великими обсягами неструктурованих даних. Завдяки реплікації та шардуванню, можна забезпечити високу доступність та оптимальну продуктивність для системи. Крім того, можливість ефективного роботи з текстовим пошуком та індексацією полегшує реалізацію функцій, пов'язаних із пошуком та аналітикою, що є важливими для системи управління роботами. Таким чином, MongoDB допомагає створити потужну та масштабовану базу даних для оптимізації роботи з інформацією в системі ФОП.

4.2 Деталі розробки системи

Створення системи організації управління робіт ФОП потребує комплексного підходу та використання різноманітних інструментів. Ось описаний процес, враховуючи використання JavaScript, HTML, CSS, React, Node.js та MongoDB:

1. Аналіз та Проектування:

Почнемо з аналізу вимог до системи та створення детального проекту. Визначивши основні функціональні та нефункціональні вимоги, розробимо схему бази даних для MongoDB. Схема бази даних в системі організації управління робіт ФОП включає кілька колекцій для зберігання різних типів даних. Ось загальна структура схеми:

- Колекція "Users" (Користувачі):
  - Зберігання інформації про користувачів системи.
  - Поля:
    - `_id`: Унікальний ідентифікатор користувача.
    - `username`: Логін користувача.
    - `password`: Хеш пароля користувача.
    - `email`: Адреса електронної пошти користувача.
    - `role`: Роль користувача (наприклад, адміністратор, звичайний користувач).

- Колекція "Tasks" (Завдання):
  - Зберігання інформації про завдання, які пов'язані з роботою ФОП.
  - Поля:
    - `_id`: Унікальний ідентифікатор завдання.
    - `title`: Заголовок або опис завдання.
    - `description`: Деталізований опис завдання.
    - `status`: Статус завдання (наприклад, "В роботі", "Завершено").
    - `assignedTo`: Ідентифікатор користувача, якому призначено завдання.

- Колекція "Calendar" (Календар):
  - Зберігання інформації про події та зустрічі, пов'язані з управлінням роботами ФОП.
  - Поля:
    - `_id`: Унікальний ідентифікатор події.

- `title`: Заголовок або опис події.
- `startDateTime`: Дата та час початку події.
- `endDateTime`: Дата та час закінчення події.
- `location`: Місце проведення події.

Це лише частина бази даних, схема ширша та модифікована відповідно до конкретних вимог та функціональності системи. Наприклад, присутні додаткові поля для кожної колекції та визначені взаємозв'язки між ними для оптимізації запитів.

2. Front-End Розробка:

Створимо основний інтерфейс системи за допомогою HTML та CSS, використовуючи React для побудови компонентів. Розробимо різні розділи, такі як завдання, календар, звіти тощо. Використовуючи React, було успішно створено гнучкі та динамічні інтерфейси для різних розділів, таких як завдання, календар та звіти. Ці інтерфейси ефективно реагують на взаємодію користувача, забезпечуючи зручність використання. Також застосовані анімації для поліпшення візуального враження та забезпечення плавності переходів. Крім того, реалізовано функціонал фільтрації та сортування завдань, щоб користувачі могли легко керувати вмістом системи та швидко знаходити необхідну інформацію.

```
// Приклад React компонента для відображення завдань
class TaskList extends React.Component {
  render() {
    return (
      <div>
        <Task title="Підготовити звіт" />
        <Task title="Зустріч з клієнтом" />
        {/* Додайте інші компоненти */}
```

```
      </div>
    );
  }}
```

3. Back-End Розробка:

Для повного функціонування системи створимо серверний застосунок з використанням Node.js та Express. Важливо правильно визначити маршрути для обробки запитів, включаючи отримання, додавання, оновлення та видалення завдань у базі даних MongoDB.

```
// Приклад маршруту для отримання завдань
app.get('/api/tasks', (req, res) => {
  Task.find({}, (err, tasks) => {
    if (err) throw err;
    res.json(tasks);
  });
});
```

4. Взаємодія з Back-End:

Використовуючи Axios API для взаємодії з сервером Node.js пропишемо код для обробки запитів та відправки даних на сервер. Щодо взаємодії з Back-End, використовуючи Node.js та MongoDB, було успішно реалізовано механізм зв'язку між клієнтським та серверним компонентами системи. Реалізував відправку та отримання запитів до сервера за допомогою асинхронних запитів, що сприяє ефективній комунікації та обміну даними між фронт-ендом і бек-ендом.

Впроваджено архітектурні рішення, які дозволяють забезпечити безпеку та надійність обміну даними. Використовуючи HTTPS-протокол для забезпечення шифрування та захисту передачі конфіденційної інформації через

мережу. Також впроваджено механізми перевірки автентичності користувачів для забезпечення безпеки доступу до системи.

Загалом, взаємодія з бек-ендом була побудована таким чином, щоб забезпечити ефективну, безпечну та надійну обробку та обмін даними між компонентами системи організації управління робіт ФОП.

```
// Приклад використання Axios для отримання завдань з сервера
useEffect(() => {
  axios.get('/api/tasks')
    .then(response => {
      // Обробка отриманих завдань
    })
    .catch(error => {
      // Обробка помилок
    });
}, []);
```

5. Робота з Базою Даних:

- Використовується бібліотека Mongoose для моделювання та взаємодії з базою даних MongoDB та збереження даних про завдання, користувачів та іншу інформацію. Щодо обробки даних на бек-енді, використовуючи MongoDB, успішно реалізовано зберігання та управління інформацією. Створено оптимізовані запити до бази даних для швидкого отримання та оновлення даних. Застосовано концепцію NoSQL для гнучкості в роботі з динамічною структурою даних.

```
// Приклад моделі для завдань в Mongoose
const taskSchema = new mongoose.Schema({
  title: String,
  description: String,
```

```
  status: String,
});
const Task = mongoose.model('Task', taskSchema);
```

6. Аутентифікація та Авторизація:

- Додано механізми аутентифікації та авторизації користувачів для захисту конфіденційної інформації та визначення рівнів доступу.

```
// Приклад Passport.js для аутентифікації користувачів
passport.use(new LocalStrategy(
  function(username, password, done) {
    // Логіка перевірки логіну та пароля
  }
));
```

7. Тестування:

Для тестування було використано фреймворк Jest як Front-End, так і Back-End коду. У сфері тестування системи організації управління робіт ФОП було проведено ретельне тестування, охоплюючи різні аспекти функціональності та властивостей системи.

Тестування функціональності включало в себе перевірку всіх основних функцій, таких як додавання та редагування завдань, ведення календаря подій, генерація звітів та інші операції. Переконався, що кожна функція працює належним чином та відповідає вимогам замовника.

Також проведено тестування сумісності для переконання, що система працює на різних пристроях та в різних браузерах, забезпечуючи користувачам однаковий функціонал та зручний інтерфейс.

8. Розгортання:

Процес розгортання на Amazon Web Services (AWS) починається зі створення віртуальних серверів у хмарному середовищі AWS EC2. Для забезпечення безпеки та обмеження доступу налаштовується віртуальна приватна хмарна мережа (VPC), і були створені підмережі для різних компонентів системи.

Клієнтська частина застосунку, розроблена на React, була збирана та завантажена до сервісу зберігання об'єктів Amazon S3. Це дозволило забезпечити ефективне розгортання та доступ до статичних ресурсів. [28]

Для балансування та розподілу навантаження між екземплярами серверів використовується сервіс AWS Elastic Load Balancing (ELB), що забезпечує стабільну роботу системи та відсутність однієї точки відмови.

Для забезпечення моніторингу та виявлення проблем використовується сервіс моніторингу AWS CloudWatch, який дозволяє відслідковувати різноманітні метрики та налаштовувати сповіщення про можливі проблеми.

Щоб забезпечити безпеку передачі даних, обов’язково використовувати сертифікати SSL/TLS, і всі комунікації були шифровані з використанням протоколу HTTPS.

Також система налаштовується для автоматичного масштабування з використанням сервісу AWS Auto Scaling, що дозволяє адаптувати ресурси до змінного навантаження, забезпечуючи ефективне використання ресурсів та високу доступність системи.

9. Підтримка та Масштабування:

Надання можливості користувачам надсилати зворотній зв'язок, вирішення виникаючих проблем та створення ідей для розширення функціоналу системи відповідно до потреб та побажань користувачів.

Використання триярусної архітектури є також ефективним рішенням. В рамках цієї архітектури різні рівні системи (презентації, логіки та даних)

функціонують автономно, що дозволяє їм незалежно масштабуватись та розвиватись.

На рівні презентації, використання React дозволяє створити динамічний та інтуїтивний інтерфейс, піддається легкій модифікації та розширенню. Взаємодія клієнта з фронтендом є швидкою та зручною завдяки використанню JavaScript.

На рівні логіки бізнесу, Node.js використовується для створення ефективного серверного середовища. Він забезпечує швидке виконання серверного коду та можливість обробки багатьох одночасних запитів, що є важливим для високопродуктивних та масштабованих систем.

MongoDB, яка використовується на рівні даних, пропонує гнучку та масштабовану схему зберігання даних типу NoSQL. Це дозволяє легко масштабувати обсяги даних та пристосовувати структуру до зростання обсягів інформації.

У контексті AWS рішення, використання Elastic Load Balancing, Auto Scaling та CloudWatch сприяє автоматизованому масштабуванню та моніторингу, забезпечуючи оптимальну продуктивність системи.

Отже, використання триярусної архітектури, у поєднанні зі згаданими технологіями, створює добре збалансовану та гнучку систему, готову до зростання, забезпечуючи ефективну підтримку.

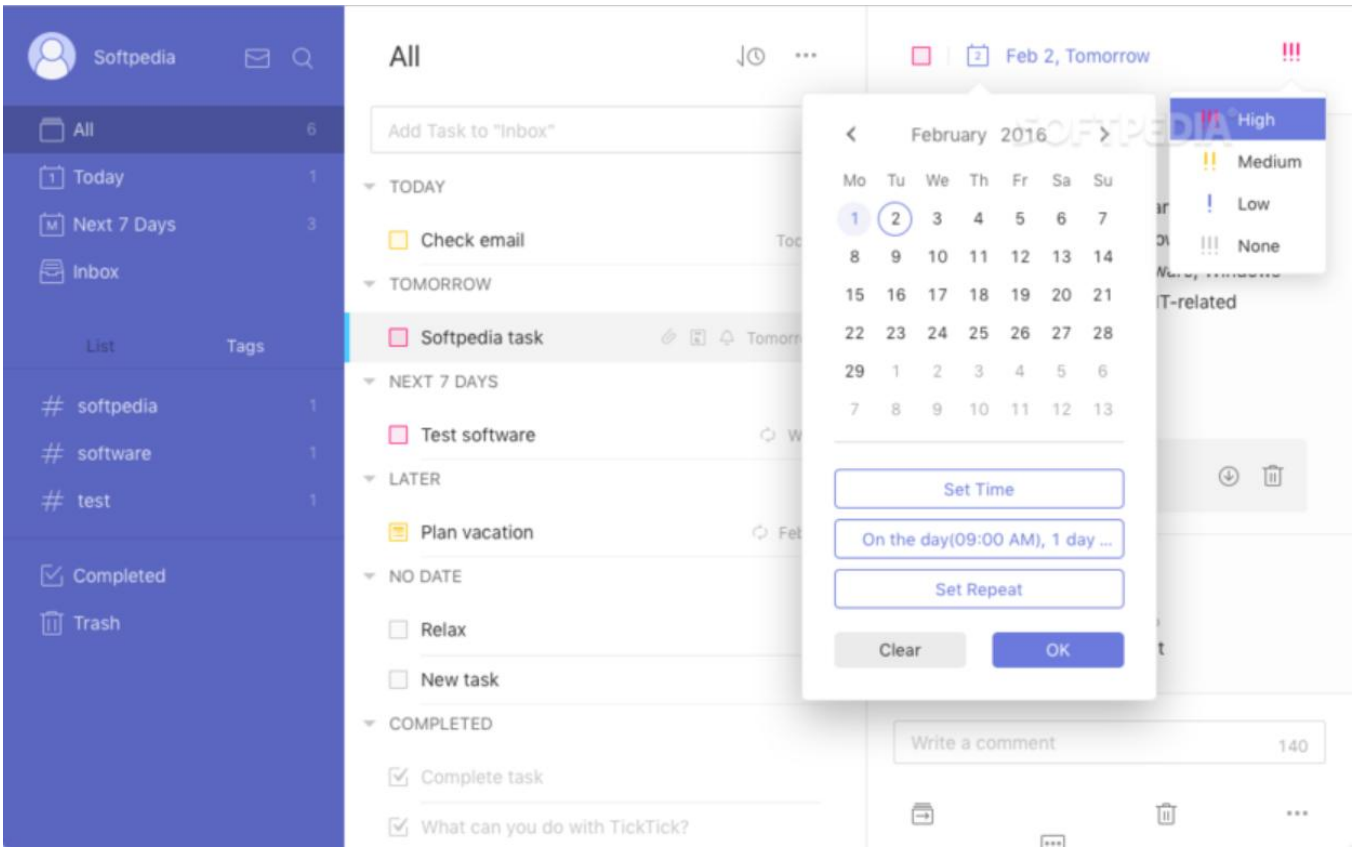


Рис. 4.1. Сторінка головного меню з представленим функціоналом

На рисунку 4.1. зображена головна сторінка що має наступний набір опцій:

- Profile: Розділ "Profile" надає ФОП інформацію про їх профіль та статистику виконаних завдань. Це допомагає в оцінці власної продуктивності та плануванні подальших дій.
- Today: Вкладка "Today" дозволяє ФОП переглядати та керувати завданнями, які повинні бути виконані протягом поточного дня. Вона допомагає організувати пріоритети та визначити, над чим потрібно працювати в першу чергу.
- Add Task: Кнопка "Add Task" дозволяє ФОП додавати нові завдання до свого списку. Це зручно для швидкого запису ідей або нових завдань, які потрібно виконати.
- Tags: Використання тегів допомагає класифікувати та групувати завдання за різними категоріями або проектами, спрощуючи навігацію та пошук.

- Completed: Вкладка "Completed" дозволяє переглядати завдання, які вже виконані. Це допомагає відстежувати досягнення та отримувати відчуття завершення.
- Trash: Вкладка "Trash" служить контейнером для видалених, непотрібних або застарілих завдань, підтримуючи чистоту та порядок у списку завдань.
- Calendar: Розділ "Calendar" надає можливість визначити терміни виконання завдань та планувати робочий графік. Це допомагає ефективно розподіляти час та виконувати завдання в строки.
- Habits: За допомогою функції "Habits" можна створювати та відстежувати різні аспекти особистого та професійного зростання.
- Pomodoro: Техніка Помодоро, вбудована в функцію "Pomodoro", допомагає підвищити продуктивність шляхом розділення робочого часу на періоди активності та коротких перерв. Це сприяє зосередженню та уникненню втоми.
- Explore: Кнопка "Explore" відкриває можливість ділитися та отримувати завдання від інших користувачів. Для ФОП це може бути корисним при співпраці з колегами або делегуванні завдань.

Кожна з цих функцій спрямована на полегшення управління завданнями, планування робочого часу та підвищення продуктивності, що є важливим для ФОП в організації робіт.

Висновок

В останньому розділі детально розкрито вибір інструментів та технологій для розробки проекту, присвяченого системі організації управління робіт ФОП. Було вирішено скористатися мовою програмування JavaScript для створення веб-додатка. Front-End частину було реалізовано з використанням HTML і CSS, а також залучено Node.js для Back-End частини. Використання бібліотеки React сприяло побудові динамічного веб-додатка з використанням компонентного підходу, віртуального DOM та одностороннього потоку даних. Зокрема, детально описано використання MongoDB в розробці. Ця NoSQL база даних дозволила системі зберігати дані у форматі JSON-подібних документів, а в новому форматі BSON (Binary JSON) вкладати один документ в інший, що сприяє ефективній організації та зберіганню інформації. Надалі, детально викладено кожен етап розробки, починаючи від написання коду та завершуючи тестуванням та розгортанням системи. Цей опис включає в себе не лише технічні аспекти, але й функціональні особливості створеної системи, надаючи повний огляд розробленого продукту.

ВИСНОВОК

Дана дипломна робота була направлена на створення системи організації управління робіт ФОП. У сучасному бізнес-середовищі такі системи відіграють важливу роль у забезпеченні ефективного та систематизованого підходу до робочих процесів. Зростання конкуренції, високий темп життя та вимоги до продуктивності вимагають від ФОП інноваційних інструментів управління. Ці системи дозволяють ефективно планувати та розподіляти робочий час, вести облік завдань і проектів, спрощують внутрішню комунікацію і співпрацю, що дає можливість краще контролювати роботу та реагувати на зміни. Вони також сприяють підвищенню продуктивності, униканню надмірного адміністративного планування та фокусуванню на важливих завданнях. Створення саме такої системи надасть можливості спростити життя підприємців у керуванні і організації їх бізнесу.

В пояснювальній записці описано усі процеси що передували створенню нашого застосунку, починаючи від аналізу предметної області, проходячи через визначення вимог та опис структури системи, і закінчуючи реалізацією системи. А отже, розроблена система відповідає вимогам та очікуванням, пов'язаним з управлінням робіт ФОП, та може стати корисним інструментом для підприємців у покращенні їхнього робочого процесу та досягненні більшої продуктивності.

Результатом кваліфікаційної роботи стало створення системи організації управління робіт ФОП з урахуванням усіх висновків з попередніх розділів проекту. Застосунок дозволяє зручно та швидко відстежувати і впроваджувати роботи пов’язані з діяльністю підприємця.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Асиметричне шифрування [Електронний ресурс] // Studfiles. – Режим доступу до ресурсу: https://studfile.net/preview/9094212/page:19/=
2. Бізнес-вимоги: розробка та приклади оформлення [Електронний ресурс] // What?. – Режим доступу до ресурсу: https://what.com.ua/biznes-vimogi-rozrobka-ta-pri/
3. Блог про програмування [Електронний ресурс] // Webjdev. – Режим доступу до ресурсу: https://www.webjdev.com/2022/08/nodejs.html
4. Введення в криптографію: симетричне шифрування [Електронний ресурс] // MarkupUa. – Режим доступу до ресурсу: https://markup-ua.com/vvedennya-v-kriptografiyu-simetrichne-shifruvannya/
5. Вимоги до програмного забезпечення [Електронний ресурс] // Studfiles. – Режим доступу до ресурсу: https://studfile.net/preview/5130988/page:2/
6. В Україні зареєстровано 2 млн ФОПів, 76% з них є платниками єдиного податку [Електронний ресурс] // LIGA ZAKON. – Режим доступу до ресурсу: https://biz.ligazakon.net/news/220491_v-ukran-zarestrovano-2-mln-fopv-76-z-nikh--platnikami-dinogo-podatku---dps
7. Методології управління проектами: виважена класика Waterfall та гнучкий Agile [Електронний ресурс] // IT Artel. – Режим доступу до ресурсу: https://it-artel.ua/blog/metodologiyi-upravlinnya-proektamy-vyvazhena-klasyka-waterfall-ta-gnuchkyj-agile/
8. Усе що потрібно знати про ФОП [Електронний ресурс] // Дія.Безбар'єрність.– Режим доступу до ресурсу: https://bf.diia.gov.ua/articles/use-shcho-potribno-znati-pro-fop
9. Хешування в криптографії і житті [Електронний ресурс] // Servicedesk. – Режим доступу до ресурсу: https://www.servicedesk.site/uk/2022/01/08/hashing/

10. 10 Timeless UX Design Best Practices [Електронний ресурс] // BairesDevBlog. – Режим доступу до ресурсу: https://www.bairesdev.com/blog/timeless-ux-design-best-practices/
11. 5 practical tips to make your UX design process more efficient right now [Електронний ресурс] // Wix. – Режим доступу до ресурсу: https://www.wix.com/studio/blog/ux-process-design?utm_source=google&utm_medium=cpc&utm_campaign=20415922026^153462531737&experiment_id=^^682623789497^&gclid=Cj0KCQiAgqGrBhDtARIsAM5s0_mlZt77YxA5j9WSgfE51pmTNr-I8o9k-L83sZLTZ3NDjkIalCPVAywaAll5EALw_wcB
12. Asana [Електронний ресурс] // Wikipedia. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Asana,_Inc.
13. Google Tasks [Електронний ресурс] // Zapier. – Режим доступу до ресурсу: https://zapier.com/blog/google-tasks-guide/
14. JavaScript [Електронний ресурс] // Wikipedia. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/JavaScript
15. Modern Web Application Architecture Explained: Components, Best Practices and More [Електронний ресурс] // Listlink. – Режим доступу до ресурсу: https://litslink.com/blog/web-application-architecture
16. Node.js [Електронний ресурс] // Wikipedia. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Node.js
17. React (software) [Електронний ресурс] // Wikipedia. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/React_(software)
18. Todoist [Електронний ресурс] // Wikipedia. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Todoist
19. Trello [Електронний ресурс] // Wikipedia. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Trello
20. Understanding Software Architecture: A Complete Guide [Електронний ресурс] // Medium. – Режим доступу до ресурсу:

https://sarrahpitaliya.medium.com/understanding-software-architecture-a-complete-guide-cb8f05900603

21. User Interface design: principles and best practices [Електронний ресурс] // Medium. – Режим доступу до ресурсу: https://bootcamp.uxdesign.cc/user-interface-design-principles-and-best-practices-72aa85203e46
22. Web Application Architecture: The Latest Guide 2024 [Електронний ресурс] // ClickIT. – Режим доступу до ресурсу: https://www.clickittech.com/devops/web-application-architecture/#h-web-application-architecture-best-practices
23. What are HTML and CSS used for? The basics of coding for the web [Електронний ресурс] // Future Learn. – Режим доступу до ресурсу: https://www.futurelearn.com/info/blog/what-are-html-css-basics-of-coding
24. What is a NoSQL database? [Електронний ресурс] // IBM. – Режим доступу до ресурсу: https://www.ibm.com/topics/nosql-databases#:~:text=NoSQL%2C%20also%20referred%20to%20as,structures%20found%20in%20relational%20databases.
25. What Is a Web Application? Definition & Examples [Електронний ресурс] // Founder Jar. – Режим доступу до ресурсу: https://www.founderjar.com/web-application/
26. What is IDS and IPS? [Електронний ресурс] // Juniper. – Режим доступу до ресурсу: https://www.juniper.net/us/en/research-topics/what-is-ids-ips.html
27. What is MongoDB [Електронний ресурс] // Anazon Web Services. – Режим доступу до ресурсу: https://aws.amazon.com/documentdb/what-is-mongodb/
28. What's the Difference Between Containers and Virtual Machines? [Електронний ресурс] // Amazon Web Services. – Режим доступу до ресурсу: https://aws.amazon.com/compare/the-difference-between-containers-and-virtual-machines/