Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____компт'ютерних наук_____
(повна назва)
Кафедра _____системотехніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

____Розробка та аналіз інформаційної системи продажу програмного забезпечення для промислових підприємств____
(тема)

Виконав:
студент 2 курсу, групи ІТПМ-22-2_____
_____Шелманов Д. О._____
(прізвище, ініціали)

Спеціальність 122 – Комп'ютерні_____
_____науки_____
(код і повна назва спеціальності)

Тип програми ___освітньо-наукова___

Освітня програма Інформаційні_____
_____технології проектування_____
(повна назва освітньої програми)

Керівник проф. Саваневич В. Є._____
(посада, прізвище, ініціали)

Допускається до захисту

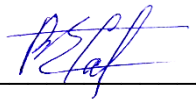
Зав. кафедри _____ ___Гребеннік І. В.____
(підпис) (прізвище, ініціали)

2024 р.

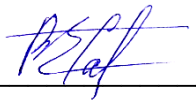
Я як студент(ка) ХНУРЕ розумію і підтримую політику закладу із академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

16.01.2024  *Шелманов Д. О.*

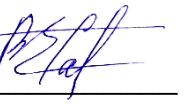
Кваліфікаційна робота не містить відомостей, заборонених до відкритого опублікування.

Керівник кваліфікаційної роботи 

Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні.

Керівник кваліфікаційної роботи 

Попередній захист проведено 12.01.2024

Керівник кваліфікаційної роботи 

Харківський національний університет радіоелектроніки

Факультет ___КН_____

Кафедра_____системотехніки_____

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність_____122 – Комп'ютерні науки_____

(код і повна назва)

Тип програми _____освітньо-наукова_____

Освітня програма _____Інформаційні технології проектування_____

(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові____*Шелманову Денису Олексійовичу*_____

1. Тема роботи: Розробка та аналіз інформаційної системи продажу програмного_____
_____забезпечення для промислових підприємств_____
_____

затверджена наказом університету від 20 листопада 2022 р. № _____1373 Ст_____

2. Термін подання студентом роботи до екзаменаційної комісії 16 січня 2024 р.

3. Вихідні дані до роботи: *В системі передбачити додавання та редагування інформації про нові обладнання, їх опис та інші дані, створення та перегляд інформації, ведення та фіксування використання.*

4. Перелік питань, що потрібно опрацювати у роботі *4.1 Вступ 4.2 Опис стану розвитку систем для промислових підприємств 4.3 Аналіз застосування рішень в предметній області 4.4 Аналіз проблем та способів їх вирішення 4.5 Опис об'єкта задачі 4.6 Вхідні дані 4.7 Вихідні дані 4.8 Опис очікуваного результату дослідження та розробки 4.9 Методи організації системи 4.10 Система пошуку програмного забезпечення 4.11 Система рекомендацій 4.12 Монетизація та оплата 4.13 Загальні принципи побудови системи 4.12 UML проєктування ПЗ 4.13 База даних 4.14 Серверна частина 4.15 Клієнтська частина 4.16 Висновки 4.17 Перелік джерел посилання*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *5.1 Тенденції маркету з продажу промислового програмного забезпечення 5.2 Відношення промислового програмного забезпечення до вбудованих в*

обладнання технологій 5.3 Зайнятість розробників за сферами 5.4 Діаграма компонентів 5.5 Діаграма класів 5.6 Функціонал системи 5.7 Схема модульної архітектури 5.8 Діаграма прецедентів 5.9 Діаграма послідовностей системи 5.10 Діаграма діяльності 5.11 Схема бази даних 5.12 Життєвий цикл обробки запиту 5.13 Загальна схема роботи додатків на Next.js 5.14 Форма авторизації додатка 5.15 Приклад форми оплати 5.16 Головна сторінка 5.17 Профіль користувача 5.18 Сторінка програмного забезпечення. 5.19 Сторінка з підписками користувача

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів атестаційної роботи	Термін виконання етапів роботи	Примітка
1.	*Отримання завдання кваліфікаційної роботи*	*10.11.2023*	*Виконано*
2.	*Аналіз предметної області, проблематики та аналогів існуючих систем з теми атестаційної роботи*	*10.11.2023-20.11.2023*	*Виконано*
3.	*Вибір мови програмування та системи управління базами даних для розробки функціонала додатку*	*21.11.2023*	*Виконано*
4.	*Розробка вимог до інформаційної системи маркетплейсу*	*21.11.2023-23.11.2023*	*Виконано*
5.	*Розробка програмного коду застосунку*	*23.11.2023-17.12.2023*	*Виконано*
6.	*Тестування програми*	*18.12.2023*	*Виконано*
7.	*Розробка «Посібника користувача»*	*20.12.2023*	*Виконано*
8.	*Оформлення пояснювальної записки та документація програмного коду*	*20.12.2023-8.01.2024*	*Виконано*
9.	*Оформлення графічної частини презентаційних матеріалів, комп'ютерних матеріалів для захисту атестаційної роботи*	*за 4 днів*	*Виконано*
10.	*Представлення на рецензування*	*за 3 дні*	*Виконано*
	Представлення атестаційної роботи в ДЕК	*за 2 дні*	*Виконано*

Дата видачі завдання « 10 » листопада 2023 р.

Студент _____ Шелманов Д. О.
(підпис)

Керівник роботи _____ __проф. Саваневич В. Є.___
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра містить, 147 стор., 28 рис., 2 табл., 25 джерел, 3 додатки.

ВИКОНАВЕЦЬ, ПІДПРИЄМЕЦЬ, ПОКАЗНИКИ, ПРОГРАМНА СИСТЕМА, САЙТ, СЕРВЕР, NEST JS, NEXT JS, POSTGRESQL, REST API, SQL, TYPESCRIPT.

Об’єкт дослідження – процес розробки та аналізу інформаційної системи з продажу програмного забезпечення для промислових підприємств.

Мета роботи – спроектувати та реалізувати програмну систему для процесів продажу програмного забезпечення для промислових підприємств, розробити відповідне програмне забезпечення з зазначеними необхідними функціями.

Методи дослідження – концептуальне моделювання, логічне моделювання предметної області, ER, проектування реляційних баз даних, об’єктно-орієнований підхід до розробки програмного додатку, REST API, модульна архітектура.

Розроблено програмну систему, що дозволяє зберігати, переглядати та додавати дані про програмне забезпечення для промислових підприємств та його замовлень. Також за допомогою системи можна керувати ліцензіями придбаного програмного забезпечення.

Для розробки використовувалися PostgreSQL, React JS, Next Js, середовище розробки Visual Studio Code, мови HTML та TypeScript.

ABSTRACT

The explanatory note contains 147 pages, 28 figures, 2 tables, 25 sources, 3 annexes.

EXECUTIVE, ENTREPRENEUR, INDICATORS, SOFTWARE, SITE, SERVER, NEST JS, NEXT JS, POSTGRESQL, REST API, SQL, TYPESCRIPT.

The object of research is the process of development and analysis of the information system for the sale of software for industrial enterprises.

The purpose of the work is to design and implement a software system for software sales processes for industrial enterprises, to develop appropriate software with the specified necessary functions.

Research methods – conceptual, ER, REST API, logical domain modeling, relational database design, object-oriented approach to software application development, modular architecture.

A software system has been developed that allows you to store, view and add data about software for industrial enterprises and its orders. You can also use the system to manage the licenses of purchased software.

PostgreSQL, React JS, Next Js, Visual Studio Code development environment, HTML and TypeScript languages were used for development

ЗМІСТ

ВСТУП

У сучасному світі існує дуже багато різноманітних видів промисловості з їхніми особливостями та нюансами. Крім того, велике значення має спеціальне обладнання, яке використовується для створення тих або інших продуктів або товарів. Тому, дуже актуальним є сервіси, що зможуть запропонувати методи знаходження та замовлення певного необхідного програмного забезпечення для відповідних підприємств. З розвитком інформаційних технологій існує декілька способів для вирішення подібної задачі.

На промислових підприємствах використання програмного забезпечення визначається потребами оптимізації та управління різними аспектами виробничої діяльності. Це широкий спектр рішень, які охоплюють різні етапи виробництва та дозволяють підприємствам досягати високої ефективності та конкурентоспроможності.

Програмне забезпечення впроваджується для автоматизації ключових виробничих процесів, включаючи контроль та управління обладнанням, моніторинг виробничих ліній, а також збір та аналіз даних для прийняття обґрунтованих управлінських рішень.

Особливу увагу приділяється системам управління виробничою ланкою, які забезпечують координацію виробничих операцій, починаючи від планування до виконання, що сприяє оптимізації ресурсів та зниженню часу виробництва.

Системи управління якістю впроваджуються для забезпечення контролю та підтримки високих стандартів якості продукції. Це включає в себе контроль якості на різних етапах виробництва та виявлення можливих дефектів.

Додатково, на промислових підприємствах використовують програми для управління ресурсами, фінансами, логістикою, а також ентерпрайз-ресурсні планувальники для інтеграції всіх аспектів діяльності підприємства.

Загалом, програмне забезпечення на промислових підприємствах є необхідним інструментом для досягнення оптимальної ефективності, забезпечення якості та здатності адаптуватися до змін виробничого середовища.

Метою даної роботи є дослідження ключових проблемних елементів в сфері використання програмного забезпечення на промислових підприємствах. Також необхідно реалізувати інформаційну систему для допомоги пошуку програмного забезпечення для промислових підприємств. Крім того, можна буде автоматизувати та оптимізувати роботу. Таким чином, на підприємствах буде підвищено ефективність та продуктивність працівників.

В даній системі можна буде зберігати інформацію про певні параметри експлуатації. З допомогою створеного додатку, можна буде переглянути ці дані, зробити певну статистику та виявити певні можливі відхилення під час користування обладнанням.

Результатом даної роботи будуть визначена дослідження проблематика та інформаційна система, що може бути використана для впровадження на підприємствах для покращення контролю експлуатації обладнання

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис стану розвитку систем для промислових підприємств

Станом на сьогодні все більше і більше сфер робочої діяльності людей включає застосування різного програмного забезпечення, яке допомагає покращити рівень продуктивності та ефективності праці. Різні програми застосовуються для автоматизації процесів у сферах навчання, медицини, фінансів та промисловості. Тому зростає попити на якісне та конкурентоспроможне програмне забезпечення.

Так як, галузь інформаційних технології відносно стрімко розвивається, існує декілька підходів вирішення таких задач. Тому не дивно, що уже створено багато програм, які за допомогою різних способів, можуть вирішити потреби людей, які безпосередньо беруть участь в промислових процесах на підприємствах.

Оскільки варіативність подібних робочих моментів дуже велика, то і зростає попит на таке програмне забезпечення, яке могло б повністю покрити необхідний функціонал.

Окреме місце займають платформи та сервіси, які надають змогу власникам підприємств придбати те чи інше програмне забезпечення, яке вони зможуть інтегрувати для власних вимог. Особливу увагу варто приділяти здатності такого програмного забезпечення правильно функціонувати при вирішенні різних задач і різноманіттям вхідних даних. Якщо ж такого рішення не було знайдено, то виникає питання в розробці нового програмного забезпечення, яке буде задовольняти умови користувачів.

Також необхідною умовою є функціонал створення та релізу програмного забезпечення розробниками. Таким чином, на платформі буде варіативність програм, які зможуть вирішувати подібні потреби, і, в результаті, можна буде уникнути монополії певного виду інформаційних систем.

З розвитком технологій, все більше і більше промислових підприємств залучають у виробництво різноманітне програмне забезпечення для покращення умов та ефективності праці. На рисунку 1.1 наведено діаграму зростання маркету продажу програмного забезпечення [1].

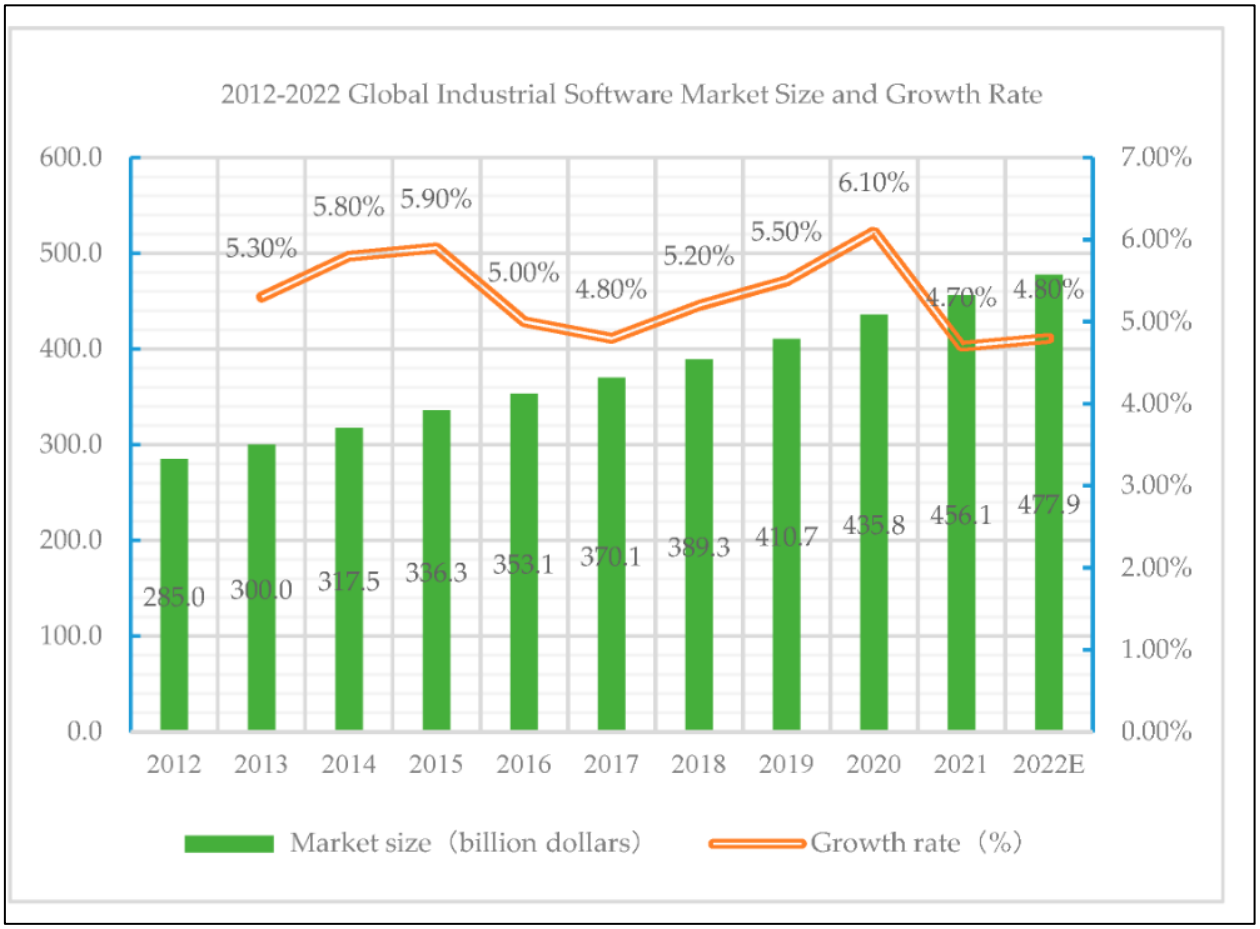


Рисунок 1.1 – Тенденції маркету з продажу промислового програмного забезпечення

Важливо також відзначити процес взаємодії клієнтів, які будуть шукати певне необхідне програмне забезпечення та авторів, які зможуть надавати його.

Стан розвитку програмного забезпечення для промислових підприємств відзначається рядом ключових тенденцій:

- інтернет речей та індустрія 4.0 - інтеграція промислових об'єктів з Інтернетом речей набуває все більшого значення [2]. Це дозволяє отримувати

реальні дані з обладнання, що підвищує ефективність виробництва та дозволяє впровадження концепцій Індустрії 4.0;

- штучний інтелект (AI) та Аналітика даних - промислове програмне забезпечення активно використовує штучний інтелект для прогнозування збоїв у виробництві, оптимізації процесів та використання великих обсягів даних для прийняття рішень;
- масштабованість та оптимізація виробництва - програмне забезпечення активно працює над розширенням можливостей масштабування, забезпечуючи пристосування до зростаючого обсягу даних та вимог користувачів;
- безпека - у зв'язку зі збільшенням кількості підключених пристроїв, кібербезпека для промислового програмного забезпечення стає ще більш актуальною. Зростає увага до заходів забезпечення відповідності та захисту від кіберзагроз;
- інтеграція та отримання даних у реальному часі - програмне забезпечення надає можливість отримувати дані у реальному часі, що полегшує миттєве реагування на події та вдосконалення різних виробничих процесів;
- мобільні застосунки для моніторингу - застосунки для моніторингу виробництва та управління процесами стають більш доступними та дозволяють віддалений доступ та управління;
- інтеграція з існуючим обладнанням - програмне забезпечення виробників стає все більше сумісним та інтегрованим з різними видами обладнання, що вже використовується в промислових підприємствах.

Ці напрямки вказують на те, що програмне забезпечення для промислових підприємств надається не тільки для автоматизації виробництва, а й для підвищення ефективності, забезпечення безпеки та інтеграції з новітніми технологіями.

Розвиток програмного забезпечення для промислових підприємств в порівнянні з іншими галузями високою мірою залежить від конкретної галузі та її потреб.

Крім того, важливо відзначити, що зростає попит на спеціальне програмне забезпечення, яке буде націлене на конкретні промислові підприємства з їх потребами, а не лише для підтримки промислового обладнання. На рисунку 1.2 наведено тенденцію відношення промислового програмного забезпечення до вбудованих в обладнання технологій [3]. За останні декілька років ситуація сильно змінилася на користь програмного забезпечення для промислових підприємств.

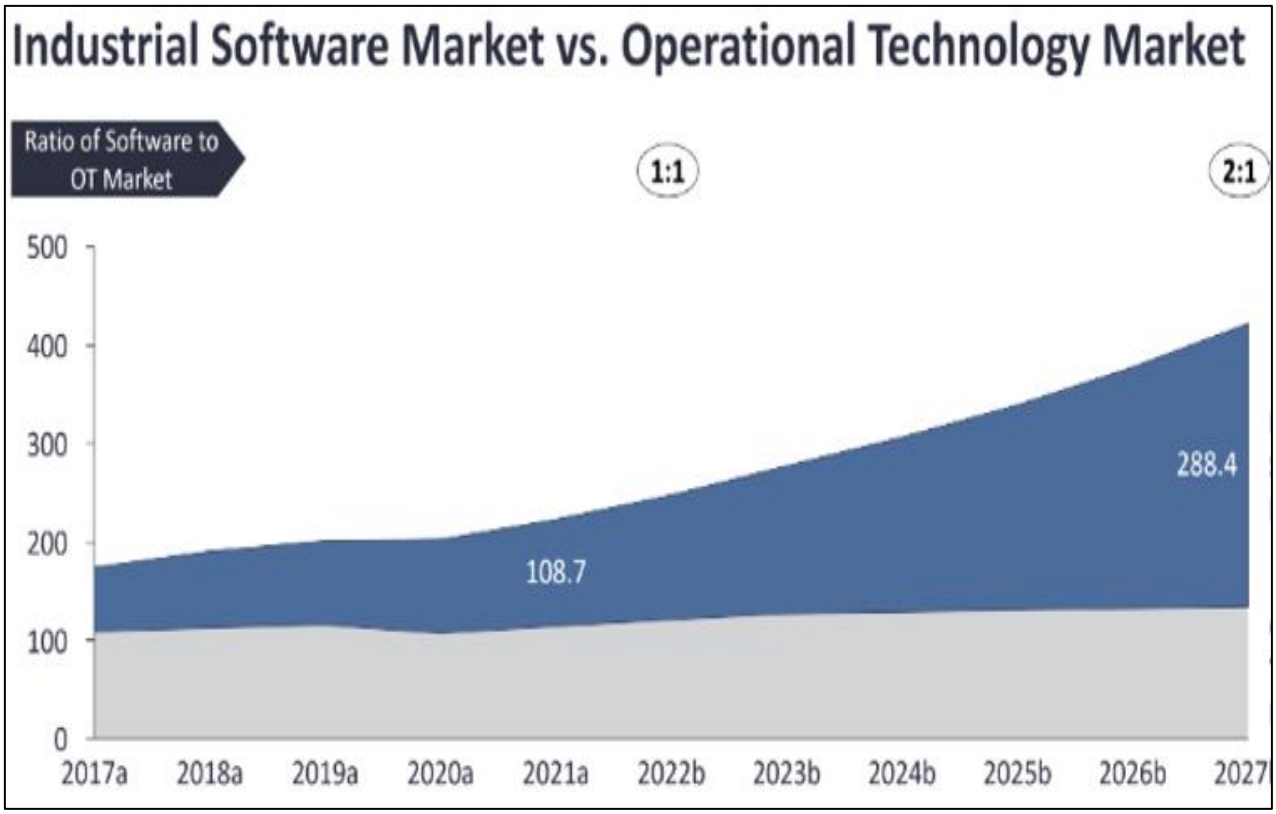


Рисунок 1.2 – Відношення промислового програмного забезпечення до вбудованих в обладнання технологій

Для промислових підприємств створюється програмне забезпечення, що враховує особливості та специфіку промислових процесів. Це може включати в себе системи автоматизації, управління виробництвом та моніторингу

обладнання. У інших галузях, таких як банківська сфера чи послуги, програмне забезпечення може бути спрямоване на обробку фінансових транзакцій, керування клієнтськими даними, аналітику та інші завдання, що відображають специфіку конкретного сектору.

Зад даними досліджень, процес інтеграції програмного забезпечення займає провідне місце в рейтингу зайнятості робітників в сфері інформаційних технологій [4]. На рисунку 1.3 наведено детальний графік.

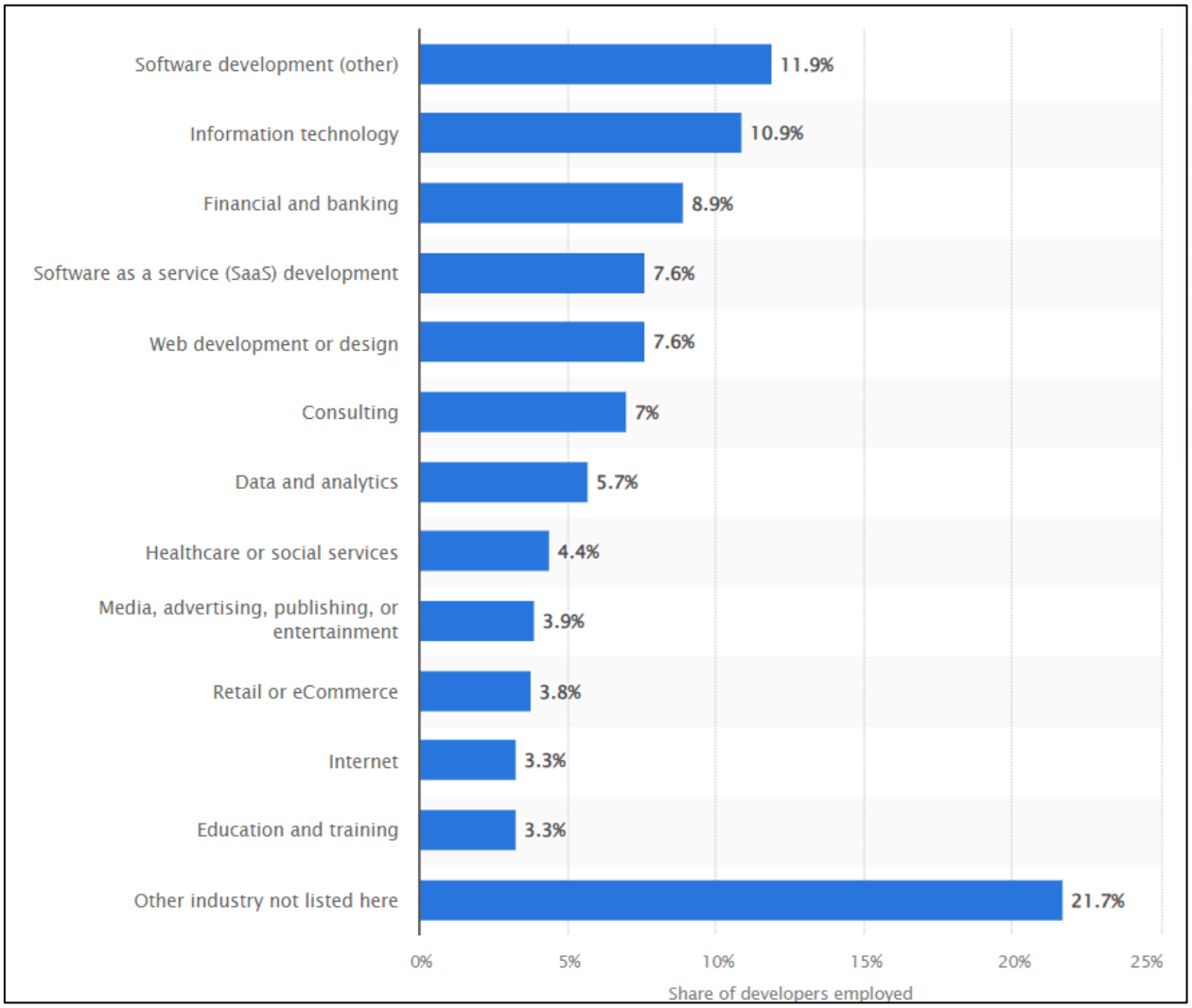


Рисунок 1.3 – Зайнятість розробників за сферами

У промисловості вимагається висока надійність та безпека, оскільки виробничі збої чи проблеми можуть призвести до серйозних наслідків. Промислове ПЗ часто може бути піддане суворим стандартам та

випробуванням на відповідність. У сферах, де не стільки важлива фізична надійність, а більше важлива конфіденційність даних (наприклад, в IT-сфері або галузях фінансів), акцент може бути зроблений на кібербезпеці та захисті від несанкціонованого доступу.

У промисловості важливо мати можливість інтегрувати програмне забезпечення з різними видами обладнання та систем, а також забезпечити його масштабованість для росту виробництва. В інших сферах може бути менше необхідності в інтеграції з фізичним обладнанням, і більше акцент може бути зроблений на інтеграції з іншими програмами, послугами та платформами.

Деякі промислові системи вимагають обробки та аналізу даних у реальному часі, особливо там, де потрібно миттєве реагування на події. У інших галузях може бути більше акценту на обробці партіями, а не в реальному часі.

Хоча загальні принципи розробки програмного забезпечення є спільними для багатьох галузей, специфіка промислового ПЗ полягає в унікальних вимогах та викликах, які виникають в умовах виробництва та використання фізичних об'єктів.

1.2 Аналіз застосування рішень в предметній області

Обираючи сервіс для пошуку програмного забезпечення, важливо врахувати кілька ключових аспектів, які допоможуть зробити інформований вибір. Ось кілька кроків та критеріїв, які можна врахувати:

- тип ПЗ, яке ви шукаєте - різні сервіси можуть спеціалізуватися на різних типах програмного забезпечення. Деякі фокусуються на бізнес-програмах, інші - на іграх або відкритому коді. Необхідно визначити, який тип ПЗ вам потрібен, і обирайте сервіс відповідно;

- відгуки та рейтинги – варто переглядати відгуки користувачів та рейтинги на обраних сервісах. Це дозволить отримати уявлення про досвід інших користувачів та якість сервісу;
- кількість та різноманітність продуктів - більше сервісів може означати більший вибір програмного забезпечення. Необхідно переконатися, що обраний сервіс пропонує широкий асортимент продуктів, щоб ви могли відзначити ті, які найкраще відповідають вашим потребам;
- спеціалізовані характеристики – необхідно вивчити, чи має сервіс спеціальні характеристики, які вам можуть знадобитися. Наприклад, деякі сервіси надають інструменти для порівняння альтернативного програмного забезпечення;
- легкість використання та інтерфейс – під час пошуку програмного забезпечення, частіше схиляються до платформ, у яких інтерфейс сервісу простий та зрозумілий. Легкість використання важлива для ефективного пошуку та порівняння;
- актуальність і оновлення - варто сервіс, який регулярно оновлює свою базу даних програмного забезпечення та відслідковує нові версії продуктів;
- доступність додаткових ресурсів – необхідно звернути увагу, чи певний сервіс надає додаткові ресурси, такі як статті, блоги чи поради щодо вибору програмного забезпечення.

Після ретельного розгляду цих аспектів можна зробити вибір сервісу для пошуку програмного забезпечення, який найкраще відповідає потребам підприємців.

Виділяючи головні моменти, було виділено основні поняття та процеси для розробки інформаційної системи програмного забезпечення для промислових підприємств. Для більш чіткішого розуміння стану речей, проведемо аналіз популярних систем, що в тій чи іншій мірі можуть допомогти з вирішенням завдань предметної області.

Для простого способу продавати цифрові продукти онлайн, можна використовувати систему «Podia». Ця комерційна платформа «все в одному» дозволяє створити спеціальний веб-сайт, де можна продавати онлайн-курси або будь-що, від членства до вебінарів. Однією з видатних особливостей Podia є можливість завантажувати необмежену кількість файлів і відео.

Крім того, можна налаштувати різні плани оплати та навіть пропонувати своїм клієнтам тести. Більше того, є можливість попередньо продавати свої онлайн-курси, щоб оцінити інтерес і надавати вміст поетапно, щоб усі навчалися з однаковою швидкістю. Без комісії за транзакції та миттєвих виплат Podia — це простий вибір для тих, хто хоче продавати цифрові продукти онлайн.

Продавати цифрові завантаження може бути дещо складно, але тут на допомогу приходить Payhip. Ця неймовірна платформа допомагає користувачам з легкістю продавати електронні книги, навчальні онлайн-ресурси та програмне забезпечення. Крім того, також можна продавати підписки через Payhip.

Тут можна продавати свої цифрові продукти абсолютно будь-де, від свого блогу до сторінок у соціальних мережах. Бути на зв’язку зі своїми клієнтами можна за допомогою простих оновлень електронною поштою або партнерською системою, щоб стимулювати їх продавати цифрові продукти.

SendOwl - проста у використанні платформа дозволяє швидко створити та налаштувати онлайн-магазин, який можна легко додати до свого блогу чи облікових записів у соціальних мережах. А завдяки можливості обробляти багатомовні та багатовалютні транзакції можна легко охопити глобальну аудиторію.

Крім того, SendOwl пропонує гнучкі варіанти оплати та спрощений процес оформлення замовлення, що змусить ваших клієнтів повертатися за новими. Незалежно від того, продаєте ви членство, ліцензію чи підписку, ця платформа допоможе вам. А завдяки таким функціям, як афілійований

маркетинг і маркетинг електронною поштою, можна вивести свої продажі на новий рівень.

Squarespace - конструктор веб-сайтів, який забезпечує зручну систему перетягування, яка дозволяє будь-кому легко створити веб-сайт професійної якості для свого бізнесу. Завдяки безлічі шаблонів на вибір, навіть ті, хто не має досвіду кодування, можуть створити добре зроблений сайт, який безперебійно працюватиме на комп'ютерах і мобільних пристроях.

І це стосується не лише цифрових продуктів. Squarespace також пропонує шаблони, розроблені спеціально для онлайн-магазинів, і навіть має інструменти, які дозволяють клієнтам призначати зустрічі та керувати підписками.

Було виявлено, що вже існують рішення, які в тій чи іншій мірі можуть допомогти вирішити завдання. Серед них не було виявлено такі, які повністю можуть охопити зазначений функціонал. Тому можна розробити систему, яка зможе це виконати. З її допомогою можна буде покращити процес інтеграції програмного забезпечення в промисловості.

Далі виконаємо певне порівняння програмних продуктів.

Для подібних систем важливо гарно спроектувати легкість впровадження та надійність. Крім того, варто враховувати інші фактори, що включають зручність, масштабованість та швидкість. Можна виокремити, необхідні для покращення роботи підприємств, характеристики:

- зручність використання;
- гнучкість пошуку необхідного програмного забезпечення;
- підтримка взаємодії з видавцями;
- зручна система оплати;
- подальша підтримка придбаного програмного забезпечення;

Після визначення, можна виконати порівняння даних програмних систем. В таблиці 1.1 наведено порівняння.

Таблиця 1.1 Порівняння систем

	Дана система	Podia	Payhip	SendOwl	Squarespace
Гнучкість пошуку програмного забезпечення	+	-	-	+	-
Підтримка взаємодії з видавцями	+	+	+	-	-
Зручна система оплати	+	+	-	+	+
Подальша підтримка програного забезпечення	+	-	+	-	+
Відповідність вимогам підприємців	+	+	-	+	_

Таким чином було виконано порівняння інформаційних систем, які в тій чи іншій мірі виконують завдання з поширення програмного забезпечення. За визначеними умовами та обмеженнями, було визначено, що, інформаційна система, що буде розроблюватися під час виконання кваліфікаційної роботи, матиме необхідні характеристики.

1.3 Аналіз проблем та способів їх вирішення

Програмне забезпечення для промислових підприємств має певний перелік вимог та проблем, які відрізняють від інших. Для вдалої реалізації інформаційних систем для промислових підприємств необхідно приділити увагу вирішенню цих проблем в першу чергу. Це допоможе успішно інтегрувати систему.

Сфера промислового програмного забезпечення стикається з рядом певних проблем і викликів, і їх вирішення вимагає ретельного та комплексного підходу та інноваційних рішень. Загалом, в даній сфері, можна виділити такі проблеми:

- кібербезпека та кіберзагрози;
- інтеграція та сумісність;
- великі обсяги даних та аналітика;
- нестабільність та збої виробництва;
- відсутність стандартів та єдиних підходів;
- специфіка галузі та вимоги;
- швидка зміна технологій;
- вимоги до ефективності та продуктивності;
- ресурсозатратність на вдосконалення;
- кадровий дефіцит та недостатність навичок.

Розглянемо детальніше ці проблеми та наявні можливі способи для їх вирішення.

Кібербезпека та кіберзагрози стали серйозною проблемою в інформаційному суспільстві [5]. Кіберзагрози включають в себе досить широкий спектр атак та вторгнень, спрямованих на комп'ютерні системи, мережі та дані. Це може включати хакерські атаки, віруси, фішинг, атаки з використанням шкідливого програмного забезпечення та інші форми кіберзлочинності.

Проблема полягає в тому, що зростання використання технологій в усіх сферах життя суспільства супроводжується ще більшим збільшенням кількості можливих точок вразливості, які в майбутньому можуть бути використані кіберзлочинцями. Такі загрози можуть призвести до неправомірного доступу до конфіденційної інформації, втрати даних, порушення приватності та завдати значних економічних та репутаційних збитків. Тому дуже важливо приділяти увагу способам захисту в сфері інформаційних технологій.

Для вирішення такої комплексної задачі необхідно розробляти вдосконалення систем безпеки, використовувати кіберзахист на різних рівнях систем (мережі, додатки, дані), виконувати регулярне оновлення програмного забезпечення, навчання персоналу, який взаємодіє з зовнішніми сервісами, з питань кібербезпеки.

Інтеграція та сумісність програмного забезпечення визначають, наскільки легко різні програмні продукти та інформаційні системи можуть взаємодіяти та працювати разом в єдиному середовищі. Це є важливою проблемою, оскільки в бізнес-та технологічних екосистемах часто використовуються різні програмні рішення від різних виробників. Проблеми інтеграції та сумісності можуть виникати на різних рівнях, таких як рівень апаратного забезпечення, операційної системи та додаткового програмного забезпечення.

Для вирішення цієї проблеми необхідно приділяти увагу на стандарти для забезпечення сумісності, використання відкритих протоколів, розробка інтерфейсів, які сприяють легкій інтеграції. Більш того, команди розробників та підтримки програмного забезпечення повинні завчасно закласти такі архітектуру системи, яке буде відкритою для подальших змін, залежно від бізнес вимог клієнтів.

З покращенням цифрової індустрії, зростає попит на системи зберігання, контролю та використання даних. На промислових підприємствах варто впровадити використання потужних систем аналізу даних, впровадження

технологій великих даних, використання деяких інструментів штучного інтелекту та машинного навчання для ефективного аналізу та використання даних [6].

Нестабільність та збої виробництва є серйозними проблемами, які можуть суттєво вплинути на ефективність, якість продукції та фінансовий стан промислового підприємства. Ці проблеми можуть мати різні причини та наслідки, і вирішення їх вимагає ретельного комплексного підходу. Для рішення цього необхідно залучити використання надійних систем моніторингу та діагностики в реальному часі, автоматичне реагування на проблеми за допомогою алгоритмів, регулярна технічна підтримка та оновлення.

Відсутність стандартів та єдиних підходів у різних галузях та сферах також може призводити до численних проблем, які впливають на ефективність, сумісність та розвиток промислових підприємств. Активна участь в стандартизації визначеної галузі, створення та використання єдиних стандартів для розробки та взаємодії систем, допоможе вирішити цю проблему.

Кожна галузь має свої унікальні характеристики, вимоги та специфічні особливості, які можуть стати причиною численних проблем. Тому дуже важлива розробка спеціалізованих рішень, які враховують конкретні потреби промислових секторів, індивідуалізація програмного забезпечення з урахуванням вимог конкретної галузі.

Для вирішення проблеми швидкої зміни використаних технологій, необхідно звертати увагу на такі речі як: гнучка архітектура наявного програмного забезпечення, використання більш сучасних технологій та рішень, активна діяльність у дослідженні та впровадженні новітніх технологій в інформаційних системах.

Також дуже важливе питання про ефективність та швидкість роботи програмного забезпечення. Для цього виконується оптимізація алгоритмів та

програмного коду, використання оптимізованих технологій, постійне вдосконалення та аналіз ефективності систем.

Висока потреба ресурсів - це проблема, коли деякі аспекти діяльності, проекти або технології вимагають значних ресурсів (фінансових, людських, матеріальних) для свого розвитку, утримання чи ефективної експлуатації [7]. Ця проблема може виникати з різних причин. Серед необхідних рішень можна виділити розробку бізнес-стратегії, яка враховує поетапні вдосконалення та підтримку, використання інструментів автоматизації та управління необхідними процесами [8].

Також важливо пам’ятати про кадровий дефіцит та недостатність навичок. Тому підприємцям слід робити інвестиції в навчання та розвиток персоналу, співпрацю з освітніми установами, вивчення та застосування нових методологій та підходів.

Вирішення цих проблем може забезпечити більш стійку та ефективну роботу промислового програмного забезпечення, сприяючи підвищенню продуктивності та конкурентоспроможності промислових підприємств.

Серед проблематики предметної області даної інформаційної системи можна виділити:

- зручний спосіб видавцям представити своє програмне забезпечення у застосунку;
- зручний спосіб для підприємців знайти необхідне програмне забезпечення;
- гнучка система рекомендацій, яка буде надавати рекомендоване програмне забезпечення для підприємців;
- структурна організація замовлень;
- зручний та безпечний спосіб оплати.

Для видавців дуже важливими є способи представлення їхнього продукту в інформаційній системі. Адже це напряму буде впливати на розповсюдження розробленого програмного забезпечення. Тому необхідно розробити таку структуру даних, яка буде дозволяти найбільш влучно описати

та представити в інформаційній системі певне програмне забезпечення для користувачів.

Важливою складовою є здійснення пошуку підприємцями. Вони можуть навіть не здогадуватися про певне існуюче програмне забезпечення, яке цілком буде підходити для їх підприємств. Тому необхідно розробити одночасну просту та ефективну систему пошуку, яка буде включати не лише назва, а й опис, теги, оцінки та ключові слова щодо представленого програмного забезпечення.

Також додатковим елементом, що допоможе знаходити відповідне програмне забезпечення є гнучка система рекомендацій. Вона буде заснована на раніше створених замовленнях, та представляти подібні рішення. Для цього необхідно буде обрати певний алгоритм, що буде найбільше підходити для даної інформаційної системи, та реалізувати його.

Для зручного користування системою, необхідно розробити зрозумілий інтерфейс для відображення замовлень. Підприємцям важливо знати про терміни їх ліцензій та підписок та їх активний стан. Тому необхідно передбачити відповідну архітектуру.

Для способу оплати можливо варто звернути увагу на інтеграцію з існуючими системами. Це зможе надати змогу інтуїтивно зрозумілої оплати та безпечного використання платіжної системи.

2. ПОСТАНОВКА ЗАДАЧІ

2.1 Опис об'єкта задачі

Ціль даної системи передбачає створення платформи для допомоги пошуку програмного забезпечення для використання разом з промисловим обладнанням. За допомогою певних фільтрів та пошуку можна буду знайти вже готові рішення, або ж створити запит на реалізацію нової системи, яке буде задовольняти потреби клієнтів.

В подальшому передбачено інтеграцію системи в інші сфери, в яких буде виникати попит на різноманітне програмне забезпечення. З точки зору розробки, подальшими кроками буде реалізації мобільного додатку та використання хмарних технологій. Крім того, можна буду розширяти функціонал за потребами користувачів.

У проблематиці даної галузі можна виділити наступні певні складові частини:

- пошук програмного забезпечення за параметрами;
- здійснення замовлення на програмне забезпечення;
- продовження замовлення;
- оформлення підписки;
- взаємодія з власниками програмного забезпечення;
- ведення звітів та історії;
- можливість завантаження власного програмного забезпечення;
- система відгуків;
- аналіз статистичних даних.

Серед переваг подібної платформи буде забезпечення працівників промислових підприємств необхідним функціоналом. Додатково зросте можливість різним командам розробників проявити себе та заявити про якість власної праці. Таким чином, система матиме великий потенціал та, що не менш важливо, актуальність у сфері.

Для успіху платформа повинна мати такий спектр необхідних властивостей:

- автоматизація пошуку;
- надійність;
- ведення історії пошуку та використання;
- система рекомендацій;
- можливість безпечного завантаження програмного забезпечення;
- складання статистики;
- надійність оформлення підписок та створення ліцензій;
- структуроване надання інформації;
- надійний процес авторизації;
- безпечний процес оплати.

Після проведеного аналізу проблематики предметної області, можна виділити певні окремі сутності, які будуть головними в досліджуваній інформаційній системі, а саме - «підприємець», «видавець», «програмне забезпечення», «ліцензія», «підписка». Розробку програмного забезпечення даної системи та її складових компонентів буде в першу чергу скеровано саме на них.

2.2 Вхідні дані

Промислові підприємства в наш час переживають стрімке впровадження цифрових технологій для покращення ефективності, оптимізації бізнес-процесів та підвищення конкурентоспроможності. Одним із важливих кроків в цьому напрямку є розробка та впровадження спеціалізованого програмного забезпечення. Для успішної розробки необхідно детально розуміти вхідні дані, що визначають функціональні та нефункціональні вимоги до системи.

Серед вхідних даних можна виділити:

- вимоги підприємців, що шукають програмне забезпечення;
- бізнес-процеси на підприємствах;

- архітектурні вимоги;
- безпека та вимоги до захисту даних;
- інтеграційні вимоги;
- вимоги до продуктивності та навантаження;
- стандарти та вимоги до документації;
- технічні обмеження та інфраструктура;

Перший крок у розробці програмного забезпечення - це ретельне вивчення вимог замовника [9]. Це включає в себе детальний опис функціональних та нефункціональних вимог, які враховують особливості та потреби конкретного промислового підприємства. Технічні специфікації, вимоги до продуктивності та безпеки - усе це стає основою для подальшої розробки.

Важливим етапом є аналіз поточних бізнес-процесів на підприємстві. Розробникам потрібно розуміти, яким чином організовано виробничі процеси, де виникають можливості для автоматизації та як можна оптимізувати діяльність.

Для створення ефективного програмного забезпечення важливо визначити його архітектуру [10]. Це включає в себе розробку структури системи, інтеграційні точки та вимоги до масштабованості. Архітектурні діаграми та технічні специфікації забезпечення стають ключовими елементами такого етапу [11].

У сучасному світі, де дані стають все ціннішим ресурсом, важливо визначити вимоги щодо захисту конфіденційності та цілісності інформації. Вимоги до шифрування, контролю доступу та політики безпеки стають невід'ємною частиною розробки.

Промислові підприємства часто використовують різноманітне обладнання та системи. Розробники повинні врахувати цю різноманітність та визначити інтеграційні зв'язки з іншими системами чи обладнанням.

Для забезпечення ефективності роботи програмного забезпечення необхідно визначити вимоги до продуктивності та навантаження системи [12].

Технічні параметри та тести навантаження допомагають забезпечити стабільну роботу системи під високим навантаженням.

Розробка програмного забезпечення для промислових підприємств повинна відповідати встановленим стандартам. Визначення стандартів та вимог до документації допомагає створити надійне та документоване рішення.

Розробники повинні враховувати технічні обмеження та визначити інфраструктуру, яка буде використовуватися для впровадження програмного забезпечення.

Для даної інформаційної системи вхідними даними, перш за все, буде інформація для пошуку промислового програмного забезпечення підприємцями. Серед такої інформації можна виділити:

- назва програмного забезпечення;
- опис сфери для програмного забезпечення;
- ключові слова;
- теги для програмного забезпечення;
- сфера застосування.

Крім того, для того, щоб придбати необхідно програмне забезпечення для промислового підприємства, будуть надані зручний пошук та система рекомендацій. Так чином можна буде зробити систему більше націленою на потреби користувачів.

Для видавців, вхідними даними, в цілому, можна назвати інформацію про відповідне програмне забезпечення. Саме вони будуть вносити дані про опис, назву, теги, властивості, ціну та терміни для промислового програмного забезпечення.

2.3 Вихідні дані

Вихідними даними в цілому можна назвати наданий функціонал системи, що буде забезпечувати взаємодію між видавцями програмного забезпечення та промисловими підприємцям. Далі розглянемо дані про

шукане програмне забезпечення, які будуть доступні в системі, та як їх можна використовувати для поліпшення бізнес-процесів на промислових підприємствах.

Найочевиднішим вихідним результатом є готовий продукт - програмне забезпечення, яке відповідає вимогам та очікуванням замовника. Це може бути велика монолітна система або комплекс різних модулів, адаптованих під конкретні потреби підприємства [13].

Важливою складовою вихідних даних є документація, яка описує архітектуру, функціональні та нефункціональні вимоги, інструкції з експлуатації та технічні характеристики [14]. Це забезпечує якісне управління програмним продуктом та сприяє його ефективній експлуатації.

Вихідні дані включають плани та звіти з тестування, які містять інформацію про проведені тести, виявлені помилки та виправлені дефекти. Це дозволяє забезпечити високу якість та надійність програмного продукту.

Якщо програмне забезпечення інтегрується з іншими системами чи обладнанням, вихідні дані включають рішення та конфігурації, необхідні для успішної інтеграції.

Вихідні дані також містять інструкції користувача, які полегшують впровадження та ефективне використання програмного продукту. Плани підтримки та служби підтримки надають користувачам необхідну допомогу.

Вихідні дані також включають плани та результати процесу запуску програмного продукту. Це може включати календарний графік, звітність про успішність впровадження та засоби для виявлення можливих проблем.

Програмне забезпечення на промисловому підприємстві є динамічним і розвивається разом з компанією. Вихідні дані можуть включати плани та засоби для подальших оновлень, розширень та адаптації під вимоги, що змінюються.

Важливо отримувати відгуки від користувачів та фахівців, які використовують програмний продукт. Ці відгуки можуть бути використані для подальшого вдосконалення продукту та покращення задоволеності клієнтів.

Остаточним етапом є аналіз результатів впровадження програмного продукту на підприємстві. Вихідні дані включають звіти про ефективність, виявлені позитивні зміни та обґрунтування користі від використання програмного забезпечення.

Для даної інформаційної системи, перш за все, вихідними даними буде інформація на сторінках, які будуть відображатися на створеному вебсайті, а саме:

- профіль користувача;
- власні підписки;
- власні ліцензії;
- рекомендації, щодо придбаного програмного забезпечення;
- інформація, щодо конкретного програмного забезпечення;
- інформація для видавців про власне додане програмне забезпечення;
- інформація зі статистикою для програмного забезпечення.

Одним із найголовніших результатів для підприємців, буде можливість придбати певне програмне забезпечення. Для цього буде реалізовано зручний пошук за фільтрами та ключовими слова та сторінка з рекомендаціями, що будуть надаватися на основі раніше придбаного програмного забезпечення. На сторінках з інформацією про підписки та ліцензії, можна буде переглянути їх термін, ціну при оформленні та інші необхідні деталі.

Для видавців буде створено сторінку з переліком доданого програмного забезпечення, сторінку з переглядом конкретного програмного забезпечення разом зі статистикою.

2.4 Опис очікуваного результату дослідження та розробки

Для успішного проектування та розробки даного програмного забезпечення, перш за все, необхідно визначити, для кого буде функціонувати дана система [15]. По-перше, це будуть підприємці, які мають певне

промислове обладнання та бажають отримати певне програмне забезпечення для нього. По-друге, свої послуги та програмне забезпечення будуть пропонувати видавці. По-третє, серед ролей системи можна виділити адміністраторів платформи, яку будуть керувати нею.

Для того щоб придбати програмне забезпечення, підприємці зможуть оформити ліцензії та підписки. З допомогою ліцензій можна буде користуватися певною версією програм. Після оформлення підписки, підприємці зможуть користуватися деякий період останніми версіями програмного забезпечення.

В предметній області можна виділити певні елементи, в залежності від функціоналу для користувачів. Для кращого розуміння вимог та розробки системи наведено опис для них.

Підприємцям необхідно мати можливість оформлювати ліцензії та підписки на певне програмне забезпечення. На платформі буде відображено деталі кожної з них.

Видавці додають своє програмне забезпечення, опис, деталі, ціну та цільове призначення. Крім того, вони зможуть переглядати інформацію про всі покупки свої програм.

Серед основного функціоналу інформаційної системи можна виділити процес придбання програмного забезпечення підприємцями. Тому особливу увагу варто приділити системі пошуку та рекомендацій за певними параметрами. З точки зору видавців, необхідно надати можливість найбільш влучно описати випущені програми, їх переваги та можливості після придбання.

Ряд вимог, яким має відповідати система:

- пошук підприємцями програмного забезпечення за певними фільтрами;
- наявність структурованого надання необхідної інформації, що зберігається;
- зручність та зрозумілість системи для звичайного повсякденного

користувача;

- за відсутності знайденого програмного забезпечення, можливість створити запити на створення нового;
- забезпечення надійності, точності та контролю доступу до даних ліцензій та підписок;
- здатність впровадження нового програмного забезпечення, яке буде використовуватися з існуючим в системі;
- ведення аналізу та відображення статистичних даних використання програмного забезпечення видавцям;
- легкість впровадження даної системи для різних промислових підприємств.

Серед ризиків системи можна виокремити:

- нездатність легкого та швидкого пошуку необхідного програмного забезпечення;
- неспроможність системи мати унікальний підхід до завантаження та реалізації різного програмного забезпечення, яке буде використовуватися;
- несумісність платформи з пропонованим програмним забезпеченням;
- невідповідність системи рекомендації та актуальних пропозицій для підприємців;
- недовіра потенціальних користувачів до надійності та безпеки даного продукту.

Початковий концепт системи можна представити у вигляді системи, що буде надавати можливість придбання програмного забезпечення для промислових підприємств.

Нефункціональні вимоги визначають якості, властивості та обмеження системи. Для даної інформаційної системи можна навести такий ряд нефункціональних вимог:

- продуктивність;
- час відгуку системи;

- пропускна здатність;
- надійність;
- безпека;
- горизонтальна масштабованість;
- вертикальна масштабованість;
- сумісність з браузерами;
- зручність інтерфейсу;
- швидкість відгуку;
- локалізовані інтерфейси.

Відповідно до предметної області, потрібно організувати систему, таким чином, щоб охопити інформацію про видавців, підприємців, програмне забезпечення, ліцензії та підписки. Для цього було створено відповідну діаграму компонентів (Див рис. 2.1).

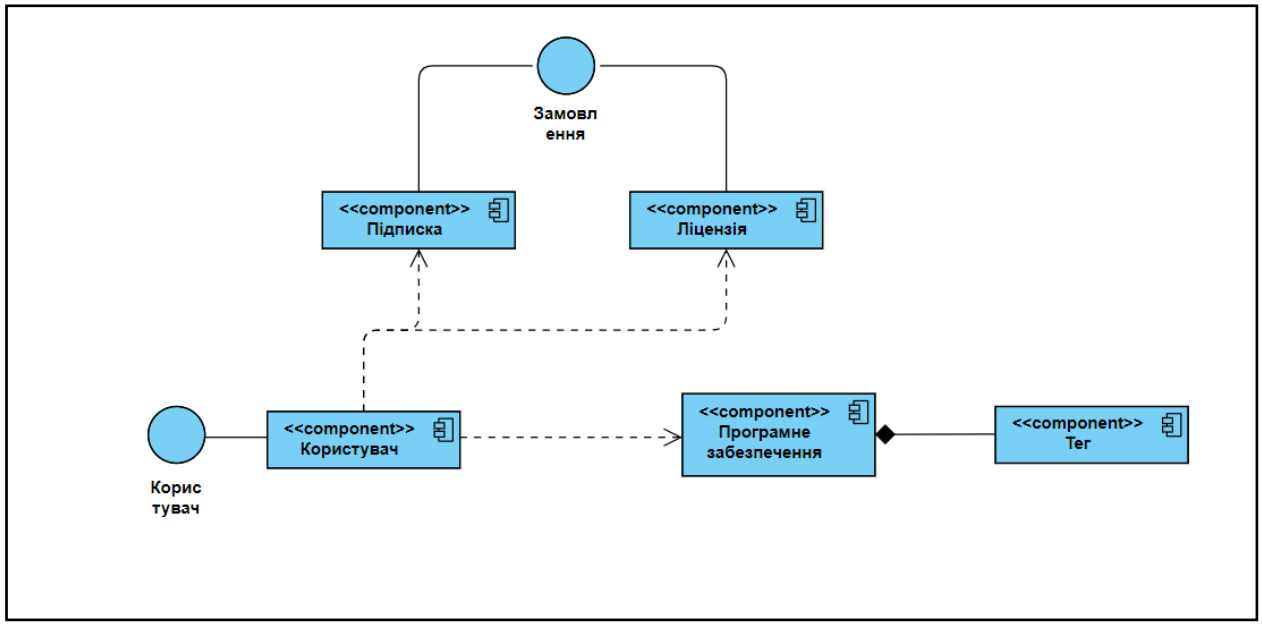


Рисунок 2.1 – Діаграма компонентів

В інформаційній системі було визначено декілька сутностей, які будуть взаємодіяти між собою. Користувачі зможуть додавати дані про програмне забезпечення та оформлювати замовлення. Для розуміння функціональної реалізації було створено діаграму класів (Див. рис. 2.2).

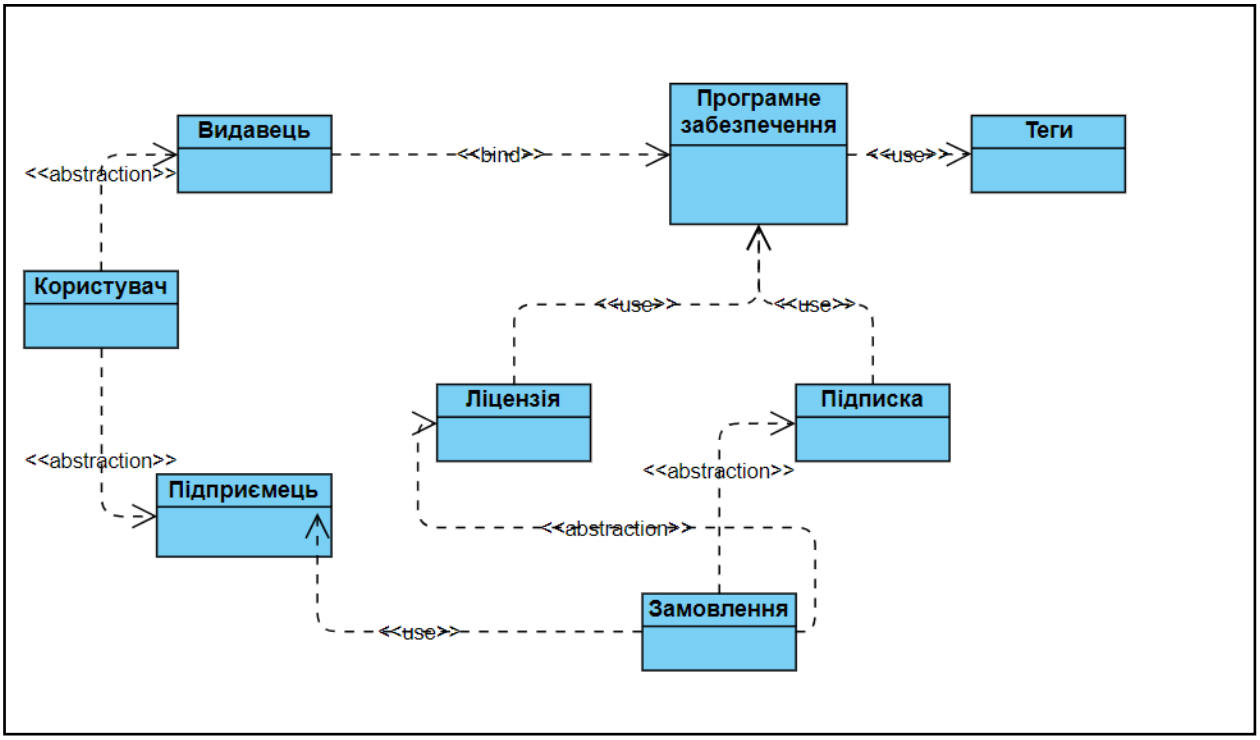


Рисунок 2.2 – Діаграма класів

Функціональні вимоги визначають, як система повинна взаємодіяти з користувачем, іншими системами або з самою собою [16]. Опишемо головний функціонал системи.

Підприємець матиме змогу пошуку програмного забезпечення за фільтрами та створення запитів за відсутності прийнятних результатів пошуку. Крім того, буде наявна можливість перегляду створених замовлень, підписок та ліцензій.

Серед функціоналу видавців можна виокремити завантаження та публікацію програмного забезпечення, перегляд активного стану ліцензій та підписок на їх програми. Більше того, буде розроблено функцію перегляду популярних запитів.

До системи можна сформувати ряд вимог, які будуть необхідні для використання:

- реєстрація та авторизація;
- пошук програмного забезпечення;

- створення підписок та ліцензій;
- можливість додати нове забезпечення;
- перегляд активного стану створених покупок;
- обробка даних;
- відображення статистики.

На підприємствах можуть бути використані різні підходи для ведення контролю. Тому варто приділити увагу для надання можливості користувачам використовувати систему так, щоб можна було отримати структуровану інформацію про пропоноване програмне забезпечення. Крім того, необхідно продумати можливості внесення інформації. Буде наявність створення запитів на створення нового програмного забезпечення або внесення оновлень до існуючого.

Таким чином, результатом дослідження та розробки буде отримано таку систему, яка буде допомагати здійснювати пошук та придбання необхідного програмного забезпечення для різних промислових підприємств та установ.

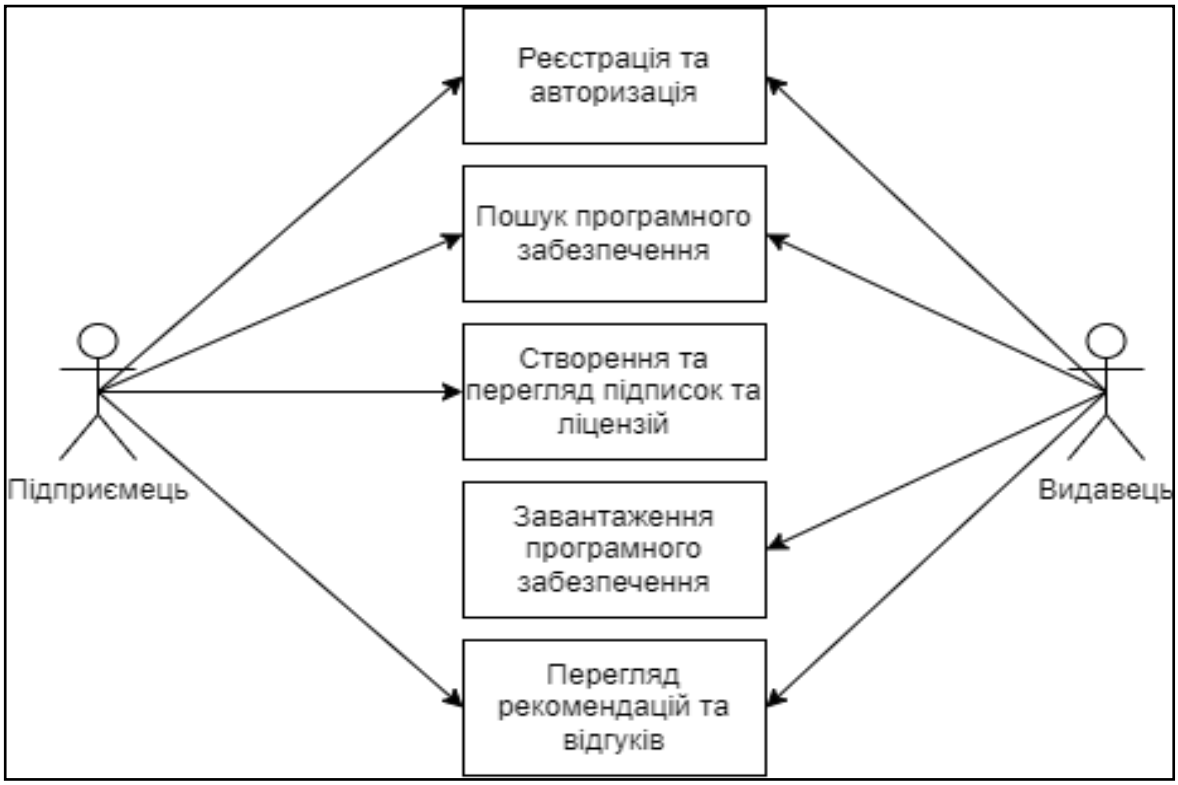


Рисунок 2.3 – Функціонал системи

В подальших релізах даної інформаційної системи планується впровадження до системи можливості покращення функціонування на різних великих промислових підприємствах, покращення інтернаціоналізації та локалізації.

Варто також зауважити, що користуватися системою будуть підприємці з різних сфер, тому необхідно розробити такий інтерфейс та дизайн, які будуть сприяти покращенню використання. Для видавців необхідно реалізувати змогу зручного завантаження та опису власних розроблених програм та застосунків.

Таким чином було визначено вимоги до програмної системи, яку буде реалізовано.

3. ДОСЛІДЖЕННЯ МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

3.1 Методи організації системи

В сфері інформаційних технологій існує декілька видів організації структури програмного забезпечення. Тому важливо підібрати такий, який буде найбільш доцільним та ефективним при виконання завдання. Організація програмного забезпечення включає в себе розподіл, керування та структурування компонентів програми з метою ефективного використання ресурсів та підтримки легкості розробки та обслуговування. Існують різні підходи та методи для організації програмного забезпечення [17]. Ось кілька загальних стратегій:

- монолітна архітектура;
- компонентна (модульна) архітектура;
- сервісно-орієнтована архітектура;
- мікросервісна архітектура;
- контейнеризація;
- розподілені системи.

Опис різних організацій систем з їх перевагами та недоліками наведено в таблиці 3.1.

Таблиця 3.1 Порівняння архітектури

Назва	Опис	Переваги	Недоліки
Монолітна архітектура:	Усі компоненти програми об'єднані в один великий блок. Зазвичай використовується в однокомпонентних застосунках	Простота розробки та підтримки для невеликих проектів	Складно в масштабуванні, важко в підтримці для великих проектів

Продовження таблиці 3.1

Компонентна (модульна) архітектура	Програма поділена на окремі компоненти або модулі, які можуть бути розроблені та тестовані окремо, а потім об'єднані	Легше управління великими проектами, забезпечення перевикористання коду.	Потребує детального планування, може виникнути складність управління великою кількістю компонентів
Сервісно-орієнтована архітектура	Розробка програмного забезпечення як набору взаємозалежних служб (сервісів), які можуть взаємодіяти між собою	Гнучкість, легкість масштабування, здатність до перевикористання	Складність при перехідному процесі, вимагає великої кількості інфраструктури.
Мікросервісна архітектура	Розбиття програми на невеликі, автономні, функціонально повністю незалежні мікросервіси, які взаємодіють між собою	Гнучкість, масштабованість, висока відмовостійкість	Складність управління великою кількістю сервісів, високі витрати на розгортання та обслуговування
Контейнеризація	Використання контейнерів для ізоляції та розгортання програми та її залежностей.	Консистентність навколишнього середовища, швидке розгортання	Збільшення складності управління контейнерами для великих проектів.
Розподілені системи	Використання різних серверів та компонентів для обробки різних завдань та надання сервісів	Масштабованість, висока доступність	Складність управління розподіленими системами

Розробка програмного забезпечення - це складний та відповідальний процес, який визначається численними аспектами, включаючи обрану

архітектуру. Вибір правильної архітектури є критичним етапом, який впливає на ефективність, масштабованість та підтримку програмного продукту протягом його життєвого циклу. Після проведення аналізу ключових принципів та особливостей, які варто враховувати при виборі архітектури для програмного забезпечення, було вирішено, що для даної інформаційної системи найкраще підходить компонента архітектура. На рисунку 3.1 наведено загальну схему модульної архітектури, що буде застосовано в додатку [18].

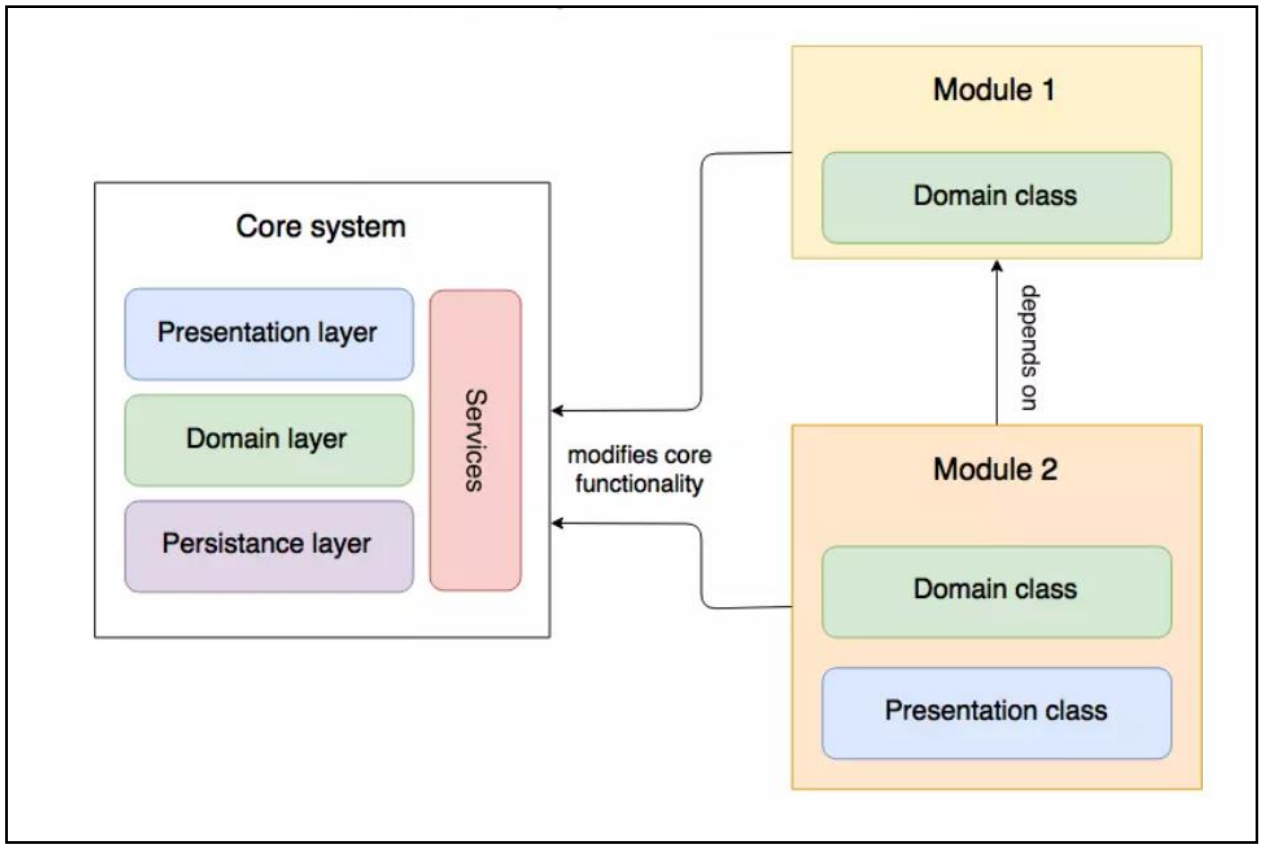


Рисунок 3.1 – Схема модульної архітектури

Логічні елементи інформаційної системи буде поділено на структурно подібні модулі, які в свою чергу будуть взаємодіяти між собою. Таким чином можна буде досягти кращого розширення системи, адже для внесення змін необхідно буде оновити лише відповідний модуль. Це робить процес розробки швидшим та простішим.

3.2 Система пошуку програмного забезпечення

Організація системи пошуку на власному веб-сайті є важливою для полегшення навігації користувачів та забезпечення швидкого доступу до потрібної інформації. Нижче подано кілька елементів, які можна виконати для ефективної організації системи пошуку:

- вибір пошукового рішення – необхідно визначити, чи потрібно використовувати стандартні засоби пошуку від визначених стандартів або інших веб-фреймворків, чи розглядати сторонні сервіси пошуку, які можна інтегрувати в системі;
- індексація контенту – необхідно переконатися, що весь контент вашого сайту індексується для пошуку. Важливо включити у пошук текст описів програмного забезпечення, назви, метатеги та інші релевантні елементи;
- налаштування поля пошуку – варто визначити, які поля будуть враховуватися у пошуку. Наприклад, можна дозволити пошук тільки за окремим заголовками, або враховувати всі текстові елементи, які відповідають параметрам;
- фільтрація та сортування результатів – можна додати можливість фільтрації та сортування результатів, що було знайдено. Користувачам може бути зручно використовувати фільтри за категоріями, типами контенту чи датою;
- автозаповнення та підказки – варто також необхідно буде надати функцію автозаповнення та підказки для полів пошуку які допоможуть користувачам швидко знаходити потрібну інформацію при введенні запиту;
- співставлення запитів – важливо також налаштувати систему пошуку на врахування співставлення запитів. Можна використовувати певні теги, синоніми, індексацію словоформ, можливість використовувати регулярні вирази тощо;

- візуальне відображення результатів – в системі необхідно розробити зручний та зрозумілий інтерфейс для відображення результатів пошуку. Варто також включити елементи, такі як зображення, іконки, короткі описи, щоб полегшити розуміння, сприйняття та вибір користувачем програмного забезпечення;
- аналітика та оптимізація – слід звернути увагу на аналітичні засоби для відстеження використання системи пошуку. Варто аналізувати популярні запити, пропозиції та виправляти можливі проблеми для оптимізації результатів;
- мобільна підтримка – варто забезпечити адаптивність та оптимізацію для мобільних пристроїв, оскільки користувачі все частіше використовують пошук на своїх смартфонах та планшетах. Більше адаптивності системи буде сприяти гарно враженню для користувачів і видавців.

Запуск системи пошуку для подібної інформаційної системи вимагає ретельного планування та тестування. Це є одним із критеріїв системи, на які будуть часто звертати увагу при використанні. Варто враховувати потреби аудиторії та забезпечити зручний та ефективний інструмент для пошуку інформації на веб-ресурсі.

3.3 Система рекомендацій

Організація системи рекомендацій - це важливий аспект для підвищення залучення користувачів та полегшення їхнього вибору. Існує кілька підходів до створення ефективних систем рекомендацій [19]. Розглянемо їх детальніше.

Фільтрація за вмістом - цей метод рекомендацій базується на властивостях та характеристиках самого контенту. Наприклад, для фільмів це може бути жанр, режисер, актори. Система порівнює вміст об'єкта з іншими об'єктами та рекомендує ті, які мають схожі характеристики. Ефективно для

об'єктів з чітко визначеними характеристиками. Не вимагає великої кількості даних про користувачів. Може вести до "фільтрування баннера", коли система рекомендацій видає схожий контент без розширення інтересів користувача.

Сумісна фільтрація користувачів використовує поведінкові дані користувачів для знаходження спільності між користувачами. Є два типи сумісної фільтрації:

- за пам'яттю - використовуються безпосередньо дані про оцінки користувачів;
- за моделлю - будується математична модель для прогнозування рейтингів;

Не вимагає апріорної інформації про контент об'єктів. Здатний рекомендувати неочевидний контент. Проблеми з холодним стартом (новий користувач або об'єкт), можливість "фільтрування шуму" (рекомендація відображає популярність, а не індивідуальний смак).

Гібридні методи комбінують різні методи рекомендацій, такі як фільтрація за вмістом і сумісна фільтрація даних користувачів або об'єктів. Саме це дозволяє вирішити проблеми кожного окремого методу та значно покращити точність рекомендацій. Серед переваг є здатність вирішувати проблеми, які можуть виникнути в чистих методах. Вони є більш гнучкі та ефективні. Серед недоліків є складність розробки та підтримки гібридних систем.

Поведінкова рекомендація - використовує дані про попередні дії користувачів (кліки, перегляди, покупки) для рекомендацій. Може використовувати алгоритми машинного навчання для визначення подібних елементів в поведінці користувачів. Метод заснований на конкретних діях користувачів, тому може бути дуже точним. Проте, він вимагає великої кількості даних та може бути вразливий до неочікуваних змін в поведінці користувачів.

Активне навчання рекомендацій використовує принципи такого активного навчання, де система навчається на основі власного досвіду та

результатів. Користувачі отримують рекомендації, і їхні реакції використовуються для поліпшення алгоритму. Серед переваг можна виділити здатність навчатися в реальному часі та адаптуватися до змін у виборах користувачів. Серед недоліків є високі вимоги до обчислювальних ресурсів та складність налаштування.

Вибір конкретного підходу залежить від конкретних потреб платформи та користувачів, а також від наявних даних та ресурсів. Зазвичай, комбінування різних методів в гібридній системі дозволяє отримати більш точні та різноманітні рекомендації.

Для даної інформаційної системи було обрано алгоритм, що використовує вагові коефіцієнти. Алгоритм рекомендацій за методом вагових коефіцієнтів використовує вагові значення для різних характеристик об'єктів та враховує їхню вагомість при формуванні рекомендацій. Загальний алгоритм, який може бути адаптований та використаний до даної предметної області:

- збір даних про користувачів та об'єкти (придбане та переглянуте програмне забезпечення, рейтинг, дата останнього оновлення), а також їх характеристики. Ці дані можуть включати рейтинги користувачів, теги, відгуки, інші взаємодії;
- визначення вагових коефіцієнтів для кожної характеристики об'єктів або взаємодій користувачів. Вагові коефіцієнти можуть визначатися експертно, на підставі аналізу даних або за допомогою алгоритмів машинного навчання;
- нормалізація даних для забезпечення однакового масштабу між різними характеристиками. Це може включати стандартизацію рейтингів або інші методи нормалізації;
- обчислення вагового рейтингу для кожного об'єкта для кожного користувача. Необхідно використовувати вагові коефіцієнти та нормалізовані значення для кожної характеристики. Наприклад, ваговий рейтинг може бути

сумою добутків вагових коефіцієнтів на нормалізовані значення характеристик;

- сортування та вибір рекомендацій за ваговим рейтингом у зворотному порядку. Вибір та представлення топ-N об'єктів для рекомендації користувачеві.
- надання рекомендацій користувачеві, наприклад, через інтерфейс веб-сайту, або мобільного додатка.

Цей підхід було обрано через відносно просту та швидку реалізацію ефективної системи рекомендацій. Таким чином, за допомогою даних користувачів про придбане програмне забезпечення можна отримати ваговий рейтинг WR_{ui} для кожної пари користувач-об'єкт та обчислити за допомогою формули:

$$WR_{ui} = \sum_{f \in F} R_{ui} \times W_f, \quad (3.1)$$

де W - множина вагових коефіцієнтів для кожної характеристики;

R - матриця рейтингів користувачів для об'єктів (де R_{ui} - рейтинг, який користувач *u* присвоїв об'єкту *i*);

U - множина користувачів;

I - множина об'єктів (програмне забезпечення);

F - множина характеристик об'єктів;

R_{ui} - рейтинг, який користувач u присвоїв об'єкту *i*;

Wf - ваговий коефіцієнт для характеристики *f*.

Отже, ваговий рейтинг є зваженою сумою рейтингів об'єкта і за всіма характеристиками з відповідними ваговими коефіцієнтами.

3.4 Монетизація та оплата

Монетизацію даної платформи в основному буде складено з покупки ліцензій та оформлення підписок. В сучасному світі це є серед найбільш оптимальних та гнучких способів. Крім того, вагому частку буде сформовано з можливості видавців висвітлювати своє програмне забезпечення на

платформі. Оскільки процес оплати в інтернеті вже досить розповсюджений, існує велика кількість способів його організації. Розглянемо вже готові рішення, які можна буде інтегрувати в дану інформаційну систему.

Stripe - це платіжна система та інфраструктура для онлайн-платежів [20]. Вона дозволяє компаніям та індивідуумам приймати платежі через Інтернет і забезпечує простий та безпечний механізм обробки транзакцій. Основні можливості Stripe включають роботу з кредитними картками, а також інші фінансові операції.

Основні характеристики та послуги Stripe:

- онлайн-платежі: Stripe дозволяє підприємствам приймати платежі через Інтернет без необхідності встановлювати складні платіжні шлюзи;
- кредитні та дебетові картки: Stripe підтримує роботу з кредитними та дебетовими картками, що дозволяє клієнтам сплачувати за товари та послуги;
- підписки та планування платежів: для підприємств, які пропонують платіжні плани або підписки, Stripe дозволяє легко впроваджувати та керувати цими послугами;
- мобільні платежі: Stripe надає інструменти для обробки платежів у мобільних додатках, що полегшує роботу розробників мобільних програм;
- безпека та захист від шахрайства: Stripe надає інструменти для захисту від шахрайства та забезпечує безпеку фінансових транзакцій;
- платежі у кількох валютах: Stripe дозволяє приймати платежі у різних валютах, що зручно для бізнесів з міжнародним спрямуванням;
- додаткові послуги: Stripe надає функціонал для ведення обліку та керування фінансами підприємств;
- розширені інструменти розробника: Stripe пропонує розширені API та бібліотеки для розробників, що дозволяє легко інтегрувати платіжні функції у власні додатки та веб-сайти;
- партнерські програми: Stripe працює з різними партнерами, такими як платіжні платформи, сервіси електронної комерції та інші.

Stripe став популярним рішенням серед розробників та бізнесів завдяки своїй легкості використання, гнучкості та високому стандарту безпеки.

PayPal - це електронна платіжна система, яка надає онлайн-платіжні послуги та діє як електронний гаманець для особистих та комерційних транзакцій. PayPal стала однією з найбільших та найпопулярніших платіжних систем у світі. Основна мета PayPal - надавати зручні та безпечні способи здійснення грошових переказів та платежів через Інтернет.

Основні характеристики та послуги PayPal включають:

- електронний гаманець: користувачі можуть зберігати гроші на своєму PayPal-рахунку, який діє як електронний гаманець;
- платіжні транзакції: можливість здійснювати платежі та перекази коштів як фізичним особам, так і компаніям через інтернет;
- карткові платежі: підтримка прийому платежів через кредитні та дебетові картки;
- міжнародні платежі: здійснення міжнародних платежів та переказів грошей;
- онлайн-магазини та платіжні кнопки: надання інструментів для інтеграції платіжних можливостей у веб-сайти та онлайн-магазини;
- партнерські програми: партнерські взаємодії та інтеграції з різними сервісами та платіжними системами;
- масштабні послуги для бізнесу: PayPal пропонує рішення для масових платежів, підтримку для ринків та масштабні комерційні послуги;
- безпека та захист від шахрайства: високий стандарт безпеки та захист від шахрайства для всіх транзакцій;
- мобільні додатки: можливість здійснювати платежі та переглядати баланс через мобільні додатки.

PayPal є популярним вибором для онлайн-торгівлі, переказів грошей та здійснення платежів через Інтернет завдяки своїй простоті використання та високому рівню безпеки.

Square - це платіжна система та технологічна компанія, яка надає різноманітні фінансові послуги та рішення для бізнесу. Заснована в 2009 році, Square стала популярною серед підприємців, власників малого та середнього бізнесу для прийому платежів, управління фінансами та інших фінансових операцій. Основна мета Square - забезпечувати доступні та ефективні інструменти для зростання бізнесу та обробки платежів.

Основні характеристики та послуги Square включають:

- точки продажу: Square пропонує апаратне обладнання та програмне забезпечення для точок продажу, що дозволяє підприємцям приймати платежі у фізичних точках продажу;
- мобільні платежі: здійснення платежів через мобільні пристрої за допомогою спеціальних рішень, таких як Square Reader та мобільні додатки.
- онлайн-платежі: інтеграція з веб-сайтами та онлайн-магазинами для прийому платежів через інтернет;
- платіжні термінали: постачання платіжних терміналів для різних видів бізнесів;
- платіжні карти: видання пластикових платіжних карт Square для зручності використання коштів;
- управління інвентарем та аналітика: послуги для ведення обліку товарів, вивчення аналітики та управління бізнес-процесами;
- послуги кредитування: Square пропонує позики та кредитування для підприємців;
- інтеграція з додатками: можливість інтегрувати платіжні рішення Square з іншими додатками та сервісами;
- контактна оплати: підтримка безконтактних платежів та оплати через мобільні пристрої.

Square визначається своєю простотою використання та доступністю для різних типів бізнесів, особливо для малих та середніх підприємств.

Після проведення аналізу наявних характеристик цих різних платіжних систем, було прийнято рішення використати Stripe для даної інформаційної системи.

3.5 Загальні принципи побудови системи

Важливо відзначити, що на підприємствах використовують різне обладнання. Через це, воно може мати свої особливості та певні специфікації. Більш того, параметри для пошуку та рекомендацій інформаційної системи можуть бути різними та характерними для певного програмного забезпечення. Тому необхідно розробити таку систему, яка матиме свій універсальний підхід до організації пошуку та виведення рекомендацій для різного програмного забезпечення.

Також варто передбачити, що буде необхідно мати доступ до структурованої інформації, щоб користувачі знали про стан ліцензій та підписок, які використовуються. Це буде необхідним для системи, що буде використовуватися на підприємствах.

Для подібних систем важливо гарно спроектувати легкість впровадження та надійність. Крім того, варто враховувати інші фактори, що включають зручність, масштабованість та швидкість.

Для покращення даної системи необхідно належним чином продумати легкість користування та надійність, які можуть бути важливими факторами при виборі програмних продуктів.

Технології для створення програмної частини буде обрано в залежності від їх переваг.

В наш час ми можемо зробити процеси пошуку програмного забезпечення зручнішими. Для цього можна буде розробити відповідне програмне забезпечення. Також важливо відзначити, що серед переваг можна винести:

- зрозумілий інтерфейс;

- простий принцип роботи;
- швидке функціонування;
- ведення історії користування;
- забезпечення надійності;
- продуманий підхід до здійснення пошуку.

У такий спосіб, після реалізації подібної інформаційної системи, можна не тільки покращити та систематизувати умови праці робітникам, а й допомогти власникам підприємств, так як вони не будуть витрачати власний час.

Базу даних необхідно розробити так, забезпечити ефективне використання інформації та можна було формувати результати пошуку та рекомендацій. Крім того, для функціонування системи, варто зберігати інформацію про підприємців, видавців, програмне забезпечення з описом, підписки та ліцензії. Також необхідно продумати масштабування програмної системи.

Для серверної частини необхідно реалізувати взаємодію інформації, що буде отримуватися з бази даних, та клієнтської частини. Більш того, варто розробити механізми обробки інформації. Також необхідно реалізувати функціонал, відповідно до бізнес логіки.

Крім того, необхідно розробити таку архітектуру, яка буде оптимально підтримувати вимоги користувачів. Тому на ранніх етапах розробки варто приділити увагу UML проєктуванню системи. Це дасть змогу не тільки описати функціонал, архітектуру та залежності, а й допоможе в розумінні конкретних бізнес задач. Це, в свою чергу, сприятиме покращенню розробки та розумінню правильного підходу до організації проекту.

Також для системи варто передбачити масштабування. Зі змінами потреб клієнтів та покращенням існуючих технологій, може змінитися задача системи в цілому. Для розробки нових елементів системи буде необхідно реалізувати окрему логіку, не впливаючи на готові частини рішення.

В подальшому, можливе використання серверної частини даного програмного забезпечення для інших сервісів. Тому не варто орієнтуватися на клієнтську частину лише для промислового обладнання. Логіку сервера необхідно зробити ізольованою та інкапсулювати. Це зробить систему більш гнучкою та стійкою до зовнішніх впливів. Краще реалізувати універсальну серверну частину, яка матиме такі можливості, які стануть в нагоді для використання іншими програмами. Це буде сприяти зручному масштабуванню системи.

Підсумовуючи, можна сказати, що краще розробити серверну частину для цього додатку, яка не буде залежати від інших елементів системи.

Для клієнтської частини реалізувати взаємодію користувачів з серверною частиною. Для цього буде створено окремі сторінки, що будуть відповідати різним елементам системи. Серед таких можна виділити:

- реєстрація;
- авторизація;
- відображення головної сторінки;
- сторінка з підписками;
- сторінка з ліцензіями;
- сторінка з завантаженим програмним забезпеченням;
- сторінка пошуку;
- сторінка профілю;
- сторінка окремої підписки;
- сторінка окремого програмного забезпечення;
- сторінка з редагуванням даних користувача;
- сторінка оплати;
- сторінка завантаження програмного забезпечення.

Саме такі елементи буде розроблено для клієнтської частини. Крім того, буде реалізовано відповідні сторінки з доданням та редагуванням даних, які будуть в системі.

Важливо відзначити, що, ще на етапі проєктування необхідно продумати інтерфейс системи. Це необхідно не лише для покращення взаємодії з користувачем, а й для правильного вибору архітектури клієнтської частини. Відповідно до цього будуть розроблені структурні елементи, які будуть відповідати конкретними частинам на сторінках додатку. Він повинен бути простим та зрозумілим для звичайного користувача. Це дасть змогу для кращого впровадження системи.

Після розробки необхідно провести тестування. На початкових етапах варто приділяти увагу інтеграційним тестам та юніт-тестам. Вони дадуть змогу перевірити розроблений функціонал та взаємодію, відповідно до вимог системи.

Для розгортання системи необхідно обрати таку платформу, яка буде одночасно надійною і простою у використанні. На початкових етапах варто простежити за процесами інтеграції, та обрати найбільш вдалий сервіс. Це допоможе легко розгорнути програмне рішення для даної інформаційної системи.

Також варто приділити окрему увагу структурі даної інформаційної системи. Вона повинна бути такою, щоб в подальшому можна було легко впровадити певні окремі елементи, які необхідно буде додати до даної системи.

4. АРХІТЕТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 UML проєктування ПЗ

Для виконання завдання було продумано архітектуру, що дозволяє розробити програмне забезпечення з можливістю додання нового функціоналу.

Для більш наглядного розуміння, наведено діаграму прецедентів (Див. рис. 4.1). Реалізацію було виконано відповідно до специфікації REST. REST - це архітектурний стиль для розробки веб-сервісів, який базується на кількох ключових принципах. У цій архітектурі, всі дані та функціональність розглядаються як ресурси, і кожен ресурс ідентифікується унікальним URI.

Узагальнено, REST спрощує взаємодію між клієнтами та серверами, забезпечуючи прозорий та ефективний обмін даними в мережі. Було створено відповідні сутності, якими буде маніпулювати система, та контролери, які відповідають за правильність виконання запитів [21]. В подальшому, їх будуть використовувати клієнтські частини.

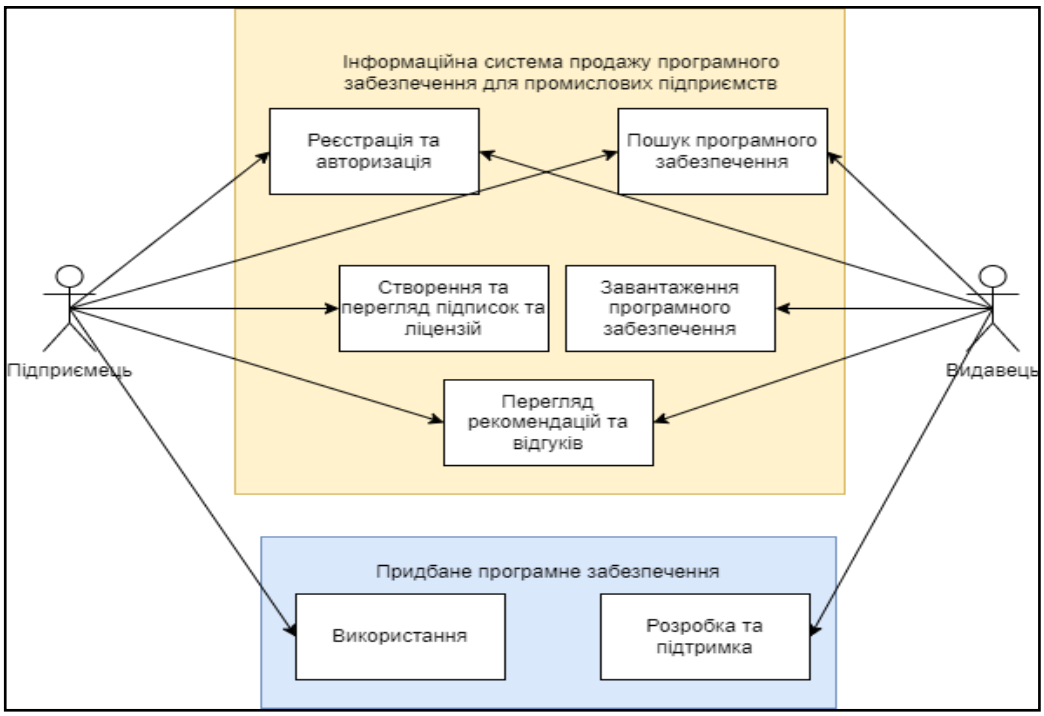


Рисунок 4.1 – Діаграма прецедентів

Серед необхідних функцій системи було виділено головні та ключові. Для користувачів:

- авторизація;
- реєстрація;
- пошук програмного забезпечення;
- завантаження програмного забезпечення;
- перегляд ліцензій та підписок;
- створення запиту на необхідне програмне забезпечення.

Для системи:

- обробка та зберігання даних;
- перевірка використання;
- структурування статистичних даних.

В даній інформаційній системі може виокремити три головні сутності, а саме: програмне забезпечення, тег та замовлення. Взаємодію між ними розглянуто на рис. 4.2.

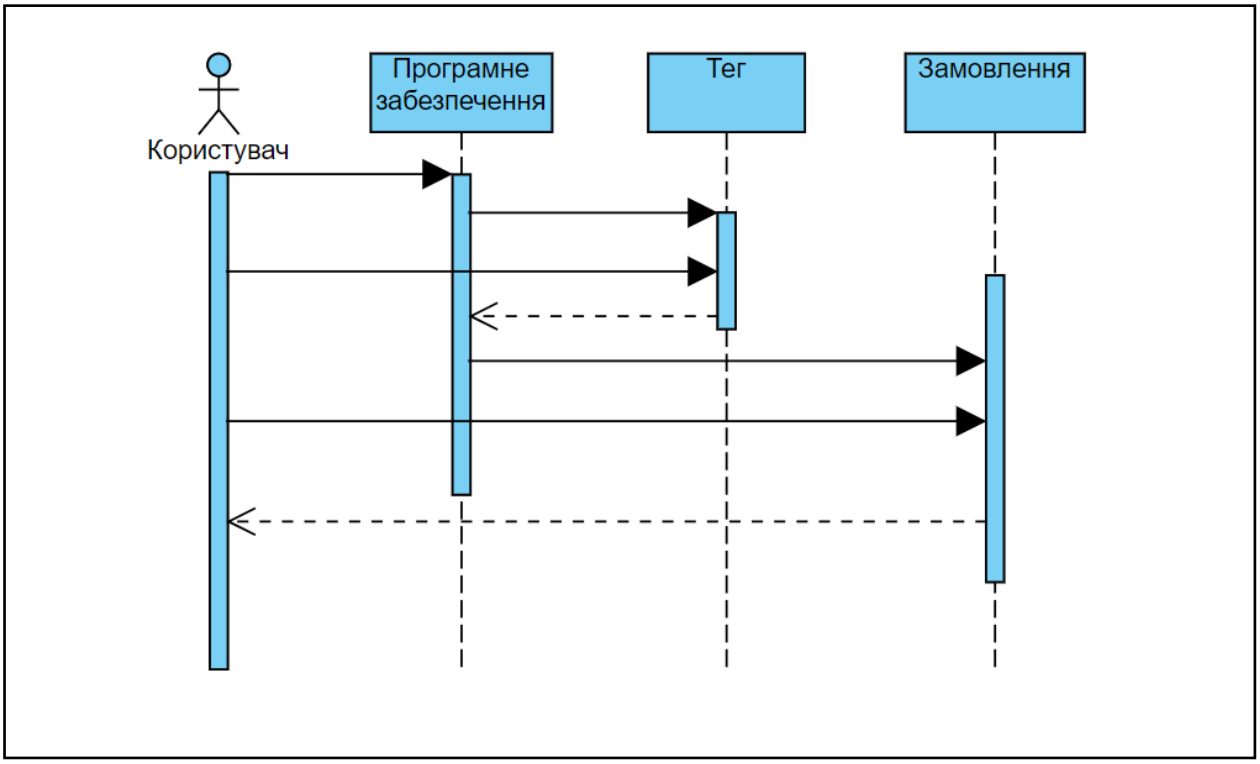


Рисунок 4.2 – Діаграма послідовностей системи

Зазначені функцію необхідно буде реалізувати у застосунку. Важливо розуміти ключові етапи та життєві цикли компонентів для інформаційної системи. Більш детально функціонал розглянуто на діаграмі діяльності (Див. рис. 4.3).

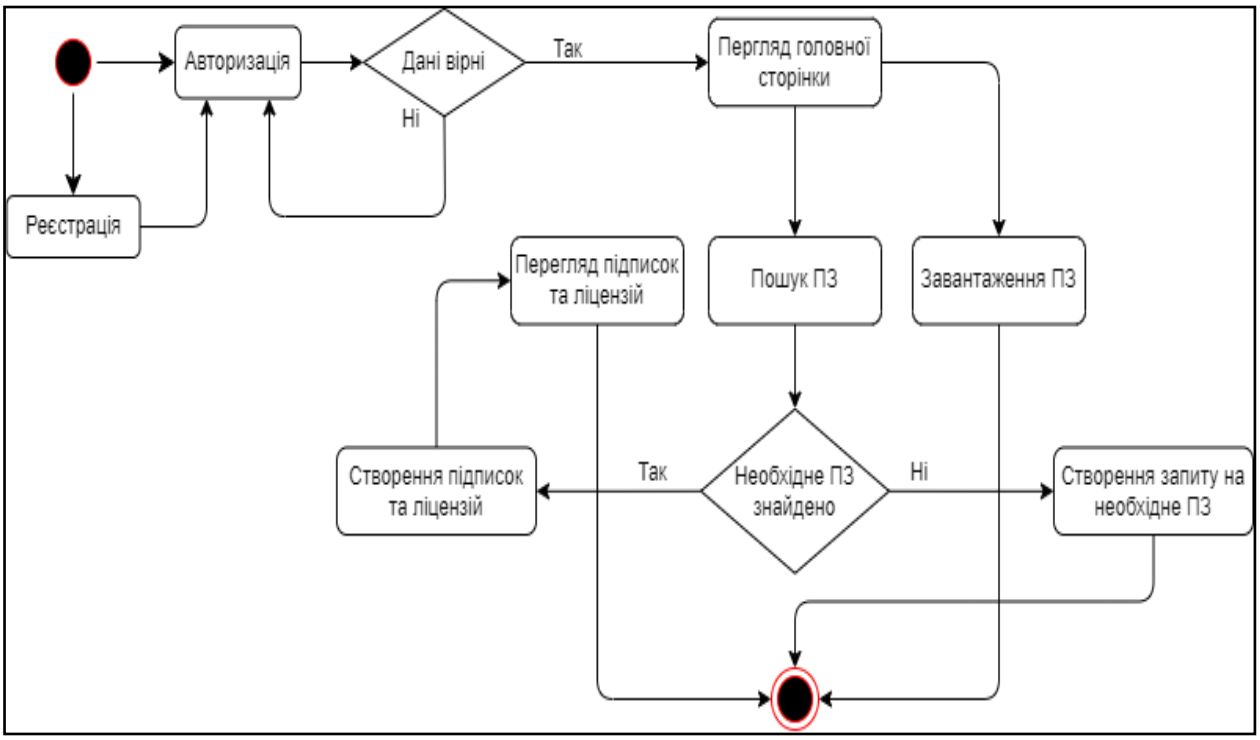


Рисунок 4.3 – Діаграма діяльності

З допомогою проведеного аналізу було визначено вимоги та функції, що будуть необхідні для даної програмної системи.

4.2 База даних

Проектування бази даних - це складний процес, що передбачає створення структури та організації бази даних для ефективного зберігання та обробки інформації. Цей процес включає в себе кілька ключових етапів.

На початку відбувається аналіз вимог, де визначаються потреби користувачів та бізнес-вимоги до зберігання даних. На цьому етапі створюється концептуальна модель даних, де визначаються сутності, їх атрибути та взаємозв'язки.

Далі проводиться нормалізація даних, що передбачає розбиття таблиць на менші для уникнення аномалій та забезпечення простого та ефективного управління даними. Після цього створюється логічна модель, де концептуальна модель перетворюється в структуру таблиць, визначаються типи даних та ключі.

Для даної інформаційної системи буде використано тертю нормальну форму[22]. Третя нормальна форма передбачає, що таблиці повинні бути в другій нормальній формі, і жоден неключовий атрибут не повинен залежати від іншого неключового атрибута. Крім того, не повинно існувати транзитивної залежності між атрибутами.

Після вибору системи керування базами даних (СКБД) розробляється фізична модель, що включає оптимізацію структури бази даних та визначення індексів для підвищення продуктивності. Важливим етапом є розробка запитів та процедур для взаємодії з даними. Це допомагає швидко та ефективно знаходити необхідну інформацію, без використання додаткових запитів або залежностей.

Важливо також враховувати певні аспекти безпеки та доступу, визначаючи права користувачів, ролі та залежності. Після цього проводиться тестування та оптимізація бази даних для забезпечення її ефективності та коректності.

Завершальними етапами є впровадження бази даних в робоче середовище та забезпечення підтримки для подальшого розвитку та вдосконалення системи. Проектування бази даних є ключовим етапом в розробці інформаційних систем та дозволяє забезпечити ефективне та безпечне управління даними в організації.

Для забезпечення реалізації функціоналу було спроектовано та створено базу даних, що складається з 4 таблиць. Для всіх таблиць є спільні поля, такі як ідентифікатор, дата створення та дата оновлення. Вони необхідні для кращої організації даних.

Розглянемо таблиці детальніше.

Користувач:

- ім’я;
- прізвище;
- електронна адреса;
- хеш значення паролю;
- роль.

Програмне забезпечення:

- посилання;
- назва;
- опис;
- перегляди;
- рейтинг;
- ідентифікатор користувача.

Тег:

- назва;
- опис;
- ідентифікатор програмного забезпечення;

Замовлення:

- ціна;
- кінцева дата;
- вид;
- статус;
- ідентифікатор програмного забезпечення;
- ідентифікатор користувача.

Для більш детального розуміння наведено схему бази даних, що буде використовуватися (Див. рис. 4.4).

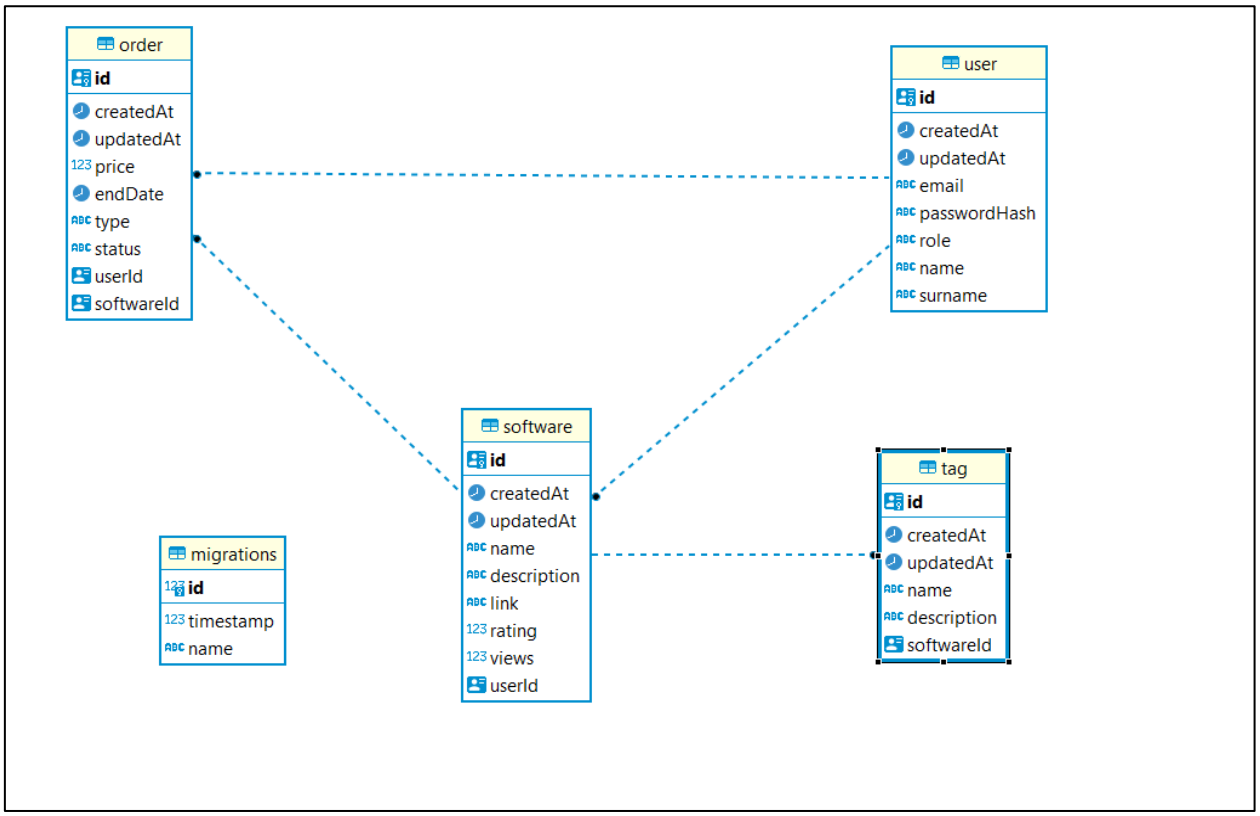


Рисунок 4.4 – Схема бази даних

Для системи керування було обрано PostgreSQL. PostgreSQL - це потужна та високоефективна система управління базами даних (СКБД), яка завоювала популярність завдяки своїм розширеним можливостям та відкритому вихідному коду [23]. Однією з ключових переваг PostgreSQL є його висока продуктивність та здатність ефективно опрацьовувати великі обсяги даних.

Ця СКБД повністю сумісна з мовою SQL, що робить її легко інтегрованою та зрозумілою для розробників. Також вона гарантує дотримання принципів ACID (Atomicity, Consistency, Isolation, Durability), що забезпечує надійність та цілісність даних.

PostgreSQL підтримує розширеність та гнучкість, надаючи можливість створювати власні функції та типи даних для адаптації до конкретних вимог проекту. Ця система також підтримує взаємодію з різними мовами програмування, що полегшує інтеграцію з різними технологіями.

PostgreSQL володіє вбудованою підтримкою роботи з JSON та іншими

форматами даних, що робить його ефективним для зберігання та обробки різноманітних структурованих і неструктурованих даних.

Однією з сильних сторін PostgreSQL є активна спільнота розробників, яка забезпечує швидке виправлення помилок, регулярні оновлення та розвиток системи. Також PostgreSQL дозволяє розширювати свою функціональність за допомогою різноманітних розширень.

Узагальнюючи, PostgreSQL є потужною та гнучкою СКБД, яка відповідає високим стандартам якості та надійності.

4.3 Серверна частина

Для реалізації серверної частини було обрано фреймворк NestJS. NestJS - це фреймворк для розробки серверних застосунків на мові програмування TypeScript, побудований на основі Node.js. Він використовує модульну архітектуру та принципи програмування, що орієнтовані на об'єкти (ООП), щоб забезпечити структурований та масштабований код.

Основні особливості NestJS включають в себе використання TypeScript для розвитку, вбудовану підтримку модульної організації застосунків, систему внесення залежностей (DI), обробку HTTP-запитів через контролери, використання middleware для обробки запитів, а також підтримку різних протоколів, таких як WebSocket.

NestJS дозволяє розробникам ефективно будувати серверні додатки для вебу та мобільних застосунків, забезпечуючи зручний інструментарій та структурований підхід до розробки.

NestJS використовує модульний підхід для структурування та організації коду в застосунках на основі Node.js і TypeScript. Модульний підхід дозволяє логічно розділити функціональність застосунка на невеликі, незалежні модулі, які можна буде потім легко використовувати в інших частинах системи.

У NestJS фундаментальними будівельними блоками є модулі. Модуль представляє собою область об'єкта, яка згруповує компоненти, контролери, провайдери та інші об'єкти застосунка. Використовуючи модулі, розробник може ефективно організувати та структурувати свій код. Крім того це сприяє кращому масштабуванні системи.

Компоненти у NestJS є основними будівельними блоками для роботи зі структурою модулів. Кожен компонент є класом, який може містити декоратори та метадані, і використовується для виконання конкретних функцій.

Контролери у NestJS відповідають за обробку HTTP-запитів та взаємодію з клієнтом. Вони розміщуються в межах модулів та використовуються для визначення маршрутів та обробки запитів. Життєвий цикл запиту представлений на рис. 4.5.

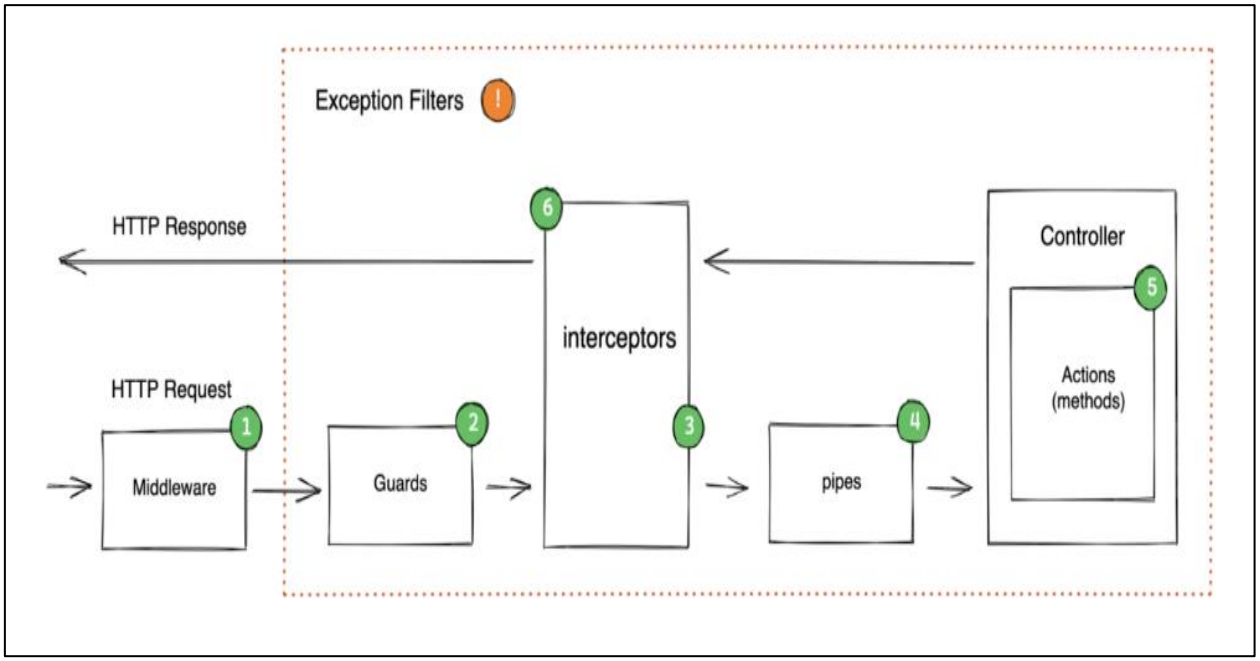


Рисунок 4.5 – Життєвий цикл обробки запиту

Провайдери використовуються для управління залежностями та внесенням служб в інші класи. Вони можуть бути використані для створення і

обслуговування об'єктів, які будуть використовуватися в різних частинах застосунка.

Сервіси представляють собою класи, які виконують конкретні функції та можуть бути використані в різних частинах застосунка. Вони часто використовуються для бізнес-логіки та взаємодії з базою даних.

Декоратори у NestJS використовуються для надання метаданих та додавання додаткової функціональності до класів.

NestJS використовує принцип застосування залежностей (Dependency Injection), що дозволяє легко управляти залежностями між класами та компонентами.

Модульний підхід дозволяє створювати застосунки, які є легко розширюваними та підтримуваними, сприяючи чіткій організації та структурі коду.

Для створення REST-сервісу, буде створено моделі, відповідно до бази даних.

Для взаємодії з базою даних буде використано вбудовану бібліотеку TypeORM. Вона дозволяє вам взаємодіяти з базами даних з використанням об'єктно-реляційного відображення. Це дозволяє взаємодіяти з підключеною базою даних за допомогою створених об'єктів та класів, замість прямих SQL-запитів. Крім того, є можливість створення міграцій для БД. Це є досить зручним інструментом для розробки. Спочатку необхідно розробити в коді класи, що будуть відповідати сутностям. Приклад класу користувача:

```
import { Entity, PrimaryGeneratedColumn, Column } from "typeorm";
@Entity()
export class User {
  @PrimaryGeneratedColumn()
  id: number;
  @Column()
  name: string;
  @Column()
```

```
  age: number;
}
```

На серверній частині буде розроблено елементи, що будуть реалізовувати обробку та взаємодію даних про:

- користувачів;
- програмного забезпечення;
- підписок;
- ліцензій.

Процес розробки серверної частини даного програмного забезпечення для інформаційної системи відповідно до існуючої бази даних було перш за все розпочато з аналізу вимог та використання структури бази даних [24]. Цей етап передбачає ретельне вивчення функціональних та нефункціональних вимог до системи, а також особливостей взаємодії з базою даних.

На етапі проектування бази даних було визначено її структуру, що включає в себе таблиці, поля, ключі та зв'язки між ними. Важливо враховувати оптимальність схеми бази даних для забезпечення ефективного доступу та обробки інформації.

Після визначення структури бази даних було розпочато написання серверної частини. Цей процес включає розробку логіки обробки запитів, яка враховує взаємодію з базою даних. Необхідно було реалізувати функції для збереження, оновлення, видалення та вибору даних з бази.

Додатково, важливо розглядати питання безпеки та оптимізації викликів до бази даних. Захист від SQL-ін'єкцій, аутентифікація та авторизація користувачів - це важливі аспекти, які слід враховувати під час розробки серверної частини.

У процесі розробки слід також було передбачено механізми обробки різних помилок, ведення журналів та моніторингу, щоб забезпечити стабільність та ефективність роботи серверної частини у виробничому середовищі.

NestJS використовує TypeScript, тому необхідно було описати моделі відповідно до схеми бази даних з використанням певного синтаксису TypeScript.

Після підготовчих етапів процес розробки перейшов до створення контролерів. Контролери визначають обробники запитів, які обробляють HTTP-запити та взаємодіють з сервісами для обробки бізнес-логіки та інших завдань.

Сервіси відповідають за виконання бізнес-логіки додатка. Було створено методи сервісів для взаємодії з базою даних, виконання операцій над даними та інші завдання.

NestJS використовує концепцію middleware для обробки запитів перед тим, як вони потрапляють до контролера. Вони є в нагоді для логування, автентифікації, перевірки прав доступу та інших завдань.

Для окремої частини бізнес логіки, необхідно було реалізувати механізм рекомендацій для користувачів. Код, що виконує логіку генерації рекомендацій за методом вагомих коефіцієнтів:

```
interface RatingItem {
  criterion: string;
  weight: number;
  score: number;
}

class RatingCalculator {
  private items: RatingItem[];
  constructor(items: RatingItem[]) {
    this.items = items;
  }
  calculateWeightedRating(): number {
    const totalWeight = this.calculateTotalWeight();
    if (totalWeight === 0) {
      throw new Error("Total weight cannot be zero.");
    }
```

```
    let weightedSum = 0;
    this.items.forEach((item) => {
      weightedSum += (item.score * item.weight);
    });
    return weightedSum / totalWeight;
  }

  private calculateTotalWeight(): number {
    return this.items.reduce((total, item) => total + item.weight, 0);
  }
}
```

Загалом, процес розробки серверної частини вимагав тісної взаємодії з існуючою базою даних та врахування різноманітних аспектів від аналізу вимог до реалізації безпеки та оптимізації.

4.4 Клієнтська частина

Розробка клієнтської частини для веб-сайтів включає в себе кілька ключових етапів, які охоплюють від аналізу вимог до реалізації та тестування [25]. Ось загальний процес розробки клієнтської частини:

- аналіз вимог - визначення функціональних та дизайнерських вимог для клієнтської частини. Розуміння цільової аудиторії, функціональності та дизайну допомагає визначити, яким має бути клієнтський додаток;
- проектування інтерфейсу- створення макетів та дизайну користувацького інтерфейсу (UI). Розробка структури сторінок, розміщення елементів, кольорової схеми та інших дизайнерських елементів;
- вибір технологій - визначення технологій, які будуть використовуватися для розробки клієнтської частини. Це може включати в себе вибір фреймворків, бібліотек, мов програмування та інших інструментів;

- розробка - написання коду клієнтської частини відповідно до визначених дизайну та функціональних вимог. Використання HTML, CSS та JavaScript (або його фреймворків, таких як React, Angular, Vue.js) для створення інтерактивної та зручної для використання сторінки;
- тестування - виконання тестів для перевірки коректності та працездатності клієнтської частини. Включає тестування взаємодії, адаптивності, валідації даних та інших аспектів;
- оптимізація - покращення продуктивності та оптимізація коду для швидкого завантаження та відображення сторінок. Виправлення будь-яких помилок та відлагодження коду для забезпечення стабільної роботи.

Загальний процес розробки клієнтської частини підпорядкований конкретним вимогам та обставинам проекту, але ці етапи можуть слугувати загальною основою для веб-розробки.

Для реалізації клієнтської частини було обрано Next.js. Цей фреймворк має кілька переваг, що роблять його популярним вибором для розробників веб-додатків:

- дозволяє виконувати завантаження сторінок на сервері, що сприяє швидкому завантаженню та покращенню SEO. Також підтримує SSG, що дозволяє генерувати окремі необхідні статичні файли під час розгортання системи;
- забезпечує просту маршрутизацію на основі структури файлів та каталогів, що полегшує організацію всього проекту та робить код більш зрозумілим;
- дозволяє змінювати код компонентів без повторного завантаження всього додатку, що прискорює процес розробки та відображення сторінок;
- мінімізує розмір модулів та залежностей, оптимізує завантаження необхідних сторінок та їх ресурсів для максимальної швидкості завантаження додатку;

- підтримка різних рішень для стилю додатка, а також можливість використання звичайного CSS або препроцесорів стилів для зручності стилізації;
- легко інтегрується з API, надаючи можливість використовувати вбудовані функції для здійснення запитів, обробки даних та взаємодії з зовнішніми сервісами;
- підтримка різних архітектур і можливість вибору різних стратегій завантаження сторінки залежно від потреб проекту.

Процес розробки на Next.js включає в себе кілька ключових етапів, які спрямовані на створення високопродуктивних веб-додатків. На рис. 4.6 наведено загальну схему роботи.

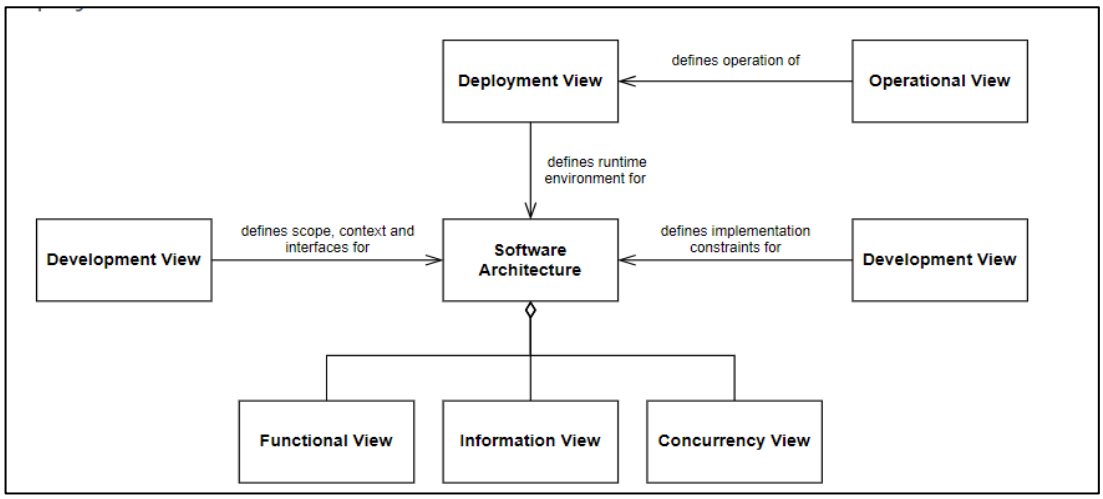


Рисунок 4.6 – Загальна схема роботи додатків на Next.js

На початку створюється новий проект за допомогою інструментів встановлення пакетів. Далі визначаються основні параметри проекту, такі як розташування статичних та динамічних файлів, визначення файлу з змінними середовища.

Основою розробки є створення різних сторінок, які визначають структуру додатку. Next.js використовує систему файлів та каталогів для виконання автоматичного маршрутизації. Це дозволяє значно спростити процес розробки.

Однією з важливих можливостей Next.js є підтримка SSR (Server-Side Rendering) та SSG (Static Site Generation), що сприяє високій продуктивності та забезпечує швидке завантаження сторінок. Сторінки, що в основному містять статичні дані, буде згенеровано на сервері та надіслано результат на клієнтську частину

Розробники можуть використовувати компоненти та різноманітні бібліотеки для покращення функціональності та вигляду додатку. Стилізація може проводитися за допомогою CSS, CSS-in-JS рішень чи інших доступних методів.

Крім того, необхідно передбачити такі сторінки, зазначені у вимогах, а саме:

- реєстрація;
- авторизація;
- головна сторінка;
- підписки;
- ліцензії;
- завантажене програмне забезпечення;
- пошук;
- профіль;
- оплата;

Всі сторінки виконано в ідентичному стилі без навантаження інформації, з логічним розділенням компонентів. На рисунку 4.6 наведено форму авторизації додатка.

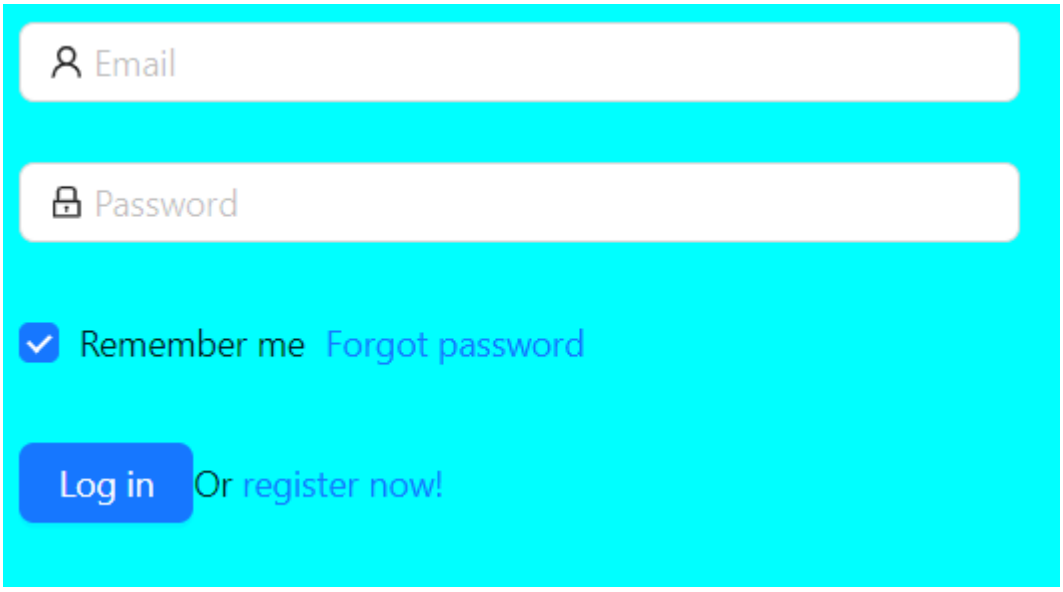


Рисунок 4.6 – Форма авторизації додатка

На сторінках можна переглядати всю необхідну інформацію. На рис 4.7 наведено сторінку перегляду програмного забезпечення.

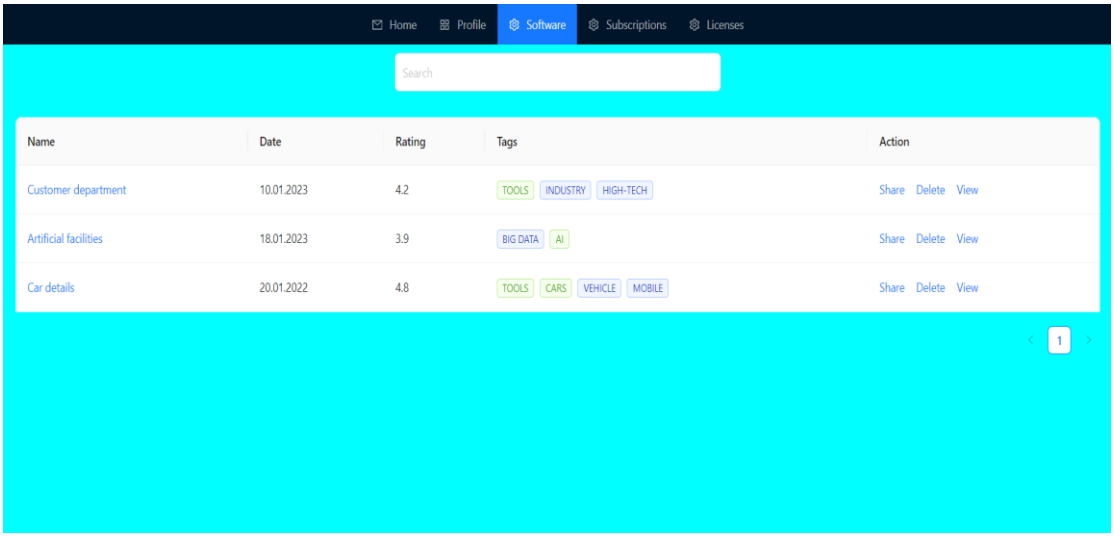


Рисунок 4.7 – Сторінка з програмним забезпеченням

Окремою частиною було створення форми для оплати. Для цього було виконано інтеграцію зі Stripe. Після встановлення відповідних модулей, було додано та налаштовано Stripe елементи у додатку. Для цього необхідно було використовувати компоненти Stripe, такі як CardElement, для створення безпечних форм оплати прямо на сторінках вашого додатку.

У React-коді було реалізовано логіку обробки платежів, використовуючи ключі API Stripe та передаючи необхідну інформацію, таку як сума платежу та валюта.

Під час процесу оформлення платежу, реалізовано взаємодію з Stripe API для створення запиту на ініціалізацію платежу, перевірки картки та обробки оплати.

Забезпечуючи правильність введених даних, необхідно відправити платіжні дані до серверної частини, де вони використовуються для взаємодії з Stripe API для обробки платежу.

Остаточно, результат операції (успіх чи помилка) передається назад у ваш React-додаток, де можна відобразити відповідний результат користувачеві.

Таким чином, інтеграція Stripe з React дозволяє легко впроваджувати платіжні можливості у веб-додатку, забезпечуючи безпеку та зручність для користувачів. Приклад форми оплати наведено нижче.

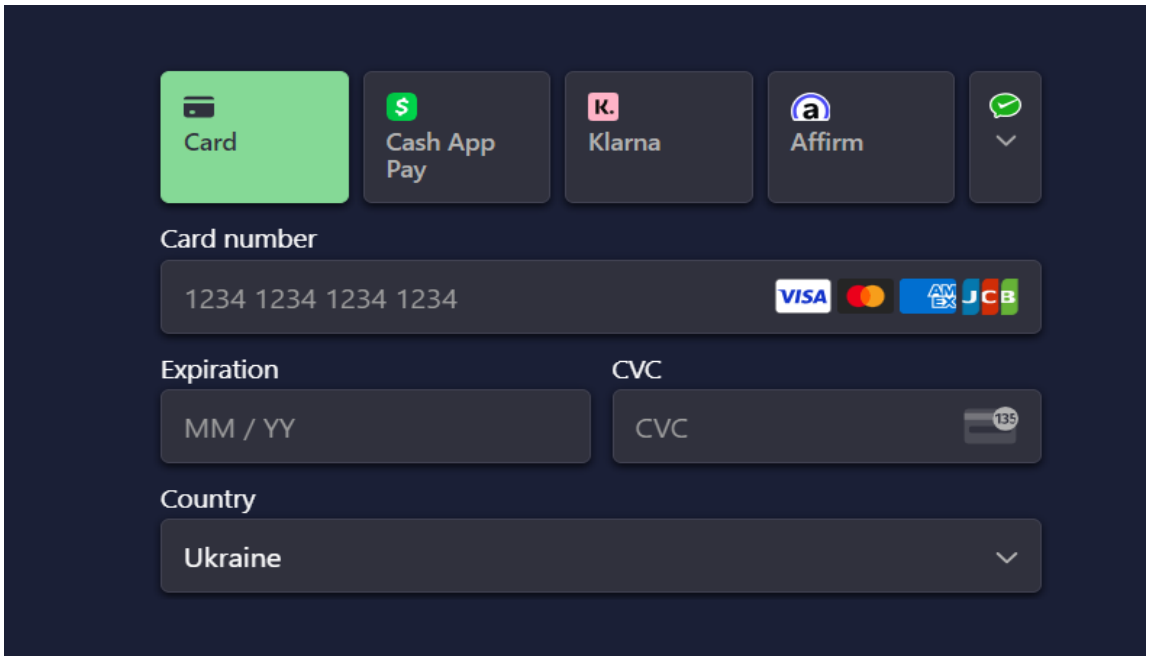


Рисунок 4.8 – Приклад форми оплати

Таким чином було реалізовано клієнтську частину для інформаційної системи для продажу програмного забезпечення.

ВИСНОВКИ

В ході виконання завдання кваліфікаційної роботи магістра було досліджено сучасний стан предметної області з продажу та інтеграції програмного забезпечення, виокремлено ключові елементи та спроектовано програмну систему для продажу програмного забезпечення для промислових підприємств. Для реалізації проекту було створено відповідні серверну та клієнтську частини. В процесі реалізації було вивчено та закріплено теоретичні знання з області розробки програмного забезпечення та технологій, що для цього використовуються. Крім того, були отримані практичні навички з реалізації та проектування.

Дана програмна система надає опції та можливості, що відповідають початковому необхідному функціоналу. Для їх виявлення було проведено аналіз предметної області, на основі результатів якого було визначено вимоги до програмного продукту, що пізніше були реалізовані в додатку. У логічній моделі бази даних було відображено об’єкти предметної області та відношення між ними.

В результаті завдання було спроектовано інформаційну системі , яка повністю відповідає заданим вимогам.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Дослідження зростання маркету промислового програмного забезпечення / URL: https://www.mdpi.com/2227-7390/11/24/4944 (дата звернення: 15.11.2023)

2. Adrian McEwen , Hakim Cassimally Designing the Internet of Things: 2013. – 194 с.

3. Дані, щодо використання IoT в програмному забезпеченні для промислових підприємств / URL: https://iot-analytics.com/industrial-software-companies/ (дата звернення: 20.11.2023)

4. Статистика з розробки програмного забезпечення для промислових підприємств / URL : https://www.statista.com/statistics/793624/worldwide-developer-survey-industry-employment/ (дата звернення: 29.11.2023)

5. Богуш В.М., Богуш В.В., Бровко В.Д., Настрадін В.П. Основи кіберпростору, кібербезпеки та кіберзахисту: Навч. пос. 2021. – 215 с.

6. Peter Harrington Machine Learning in Action: 2012. – 491 с.

7. Mark Richards, Neal Ford. Fundamentals of Software Architecture An Engineering Approach: 2020. – 378 с.

8. Jez Humble, David Farley Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation: 2010. – 328 с.

9. Юрій Грицюк Аналіз вимог до програмного забезпечення: 2018. – 213 с.

10. Rober Martin Clean architecture: 2019. – 368 с.

11. Scott Berinato Good Charts: 2016. – 453 с.

12. David Thomas, Andrew Hunt The pragmatic programmer: 1999. – 522 с.

13. Christian Ciceri, Dave Farley, Neal Ford, Andrew Harmel-Law, Michael Keeling, Carola Lilienthal, João Rosa, Alexander von

Zitzewitz, Rene Weiss, Eoin Woods Software Architecture Metrics: 2022. – 329 с.

14. Jared Bhatti , Sarah Corleissen , Jen Lambourne, David Nunez, Heidi Waterhouse Docs for Developers: An Engineer's Field Guide to Technical Writing: 2021. – 324 с.

15. Jeff Patton, Martin Fowler, Peter Economy, Alan Cooper, Marty Cagan User Story Mapping: Discover the Whole Story, Build the Right Product: 2014. – 374 с.

16. Karl Wiegers, Joy Beatty Software Requirements (Developer Best Practices): 2013. – 297 с.

17. Len Bass, Paul Clements, Rick Kazman Software Architecture in Practice: 2012. – 471 с.

18. David Wallance The Future of Modular Architecture: 2021. – 268 с.

19. Kim Falk Practical Recommender Systems: 2019. – 432 с.

20. Визначення Stripe / URL: https://stripe.com/payments/features (дата звернення: 20.12.2023).

21. Визначення специфікації REST / URL: https://en.wikipedia.org/wiki/Representational_state_transfer (дата звернення: 24.12.2023).

22. Stephen Morris Database Principles: Fundamentals of Design, Implementation, and Management: 2012. – 736 с.

23. Документація PostgreSQL / URL : https://www.postgresql.org/docs/ (дата звернення: 12.11.2022).

24. Krzysztof Cwalina. Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries (Hardcover). 2009. – 436 с.

25. Jeff Johnson Designing with the Mind in Mind: 2014. – 382 с.