МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему: **“ Система управління інформаційними ресурсами підприємства із застосуванням технології Entity Framework ”**

Виконав: __ст. гр. Іт-62_____

Спеціальності 126 – «Інформаційні системи та технології»_____
(шифр і назва)

_____Олещук Роман Ярославович_____
(Прізвище та ініціали)

Керівник: __к.т.н., доц. Луб П.М._____
(Прізвище та ініціали)

Рецензенти: _____
(Прізвище та ініціали)

_____
(Прізвище та ініціали)

ДУБЛЯНИ-2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Освітній ступінь «Магістр»
126 – «Інформаційні системи та технології»

"ЗАТВЕРДЖУЮ"
Завідувач кафедри_____
д.т.н., проф. А.М. Тригуба
"____"_____2022 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Олещук Роман Ярославович

1. Тема роботи: «Система управління інформаційними ресурсами підприємства із застосуванням технології Entity Framework»

Керівник роботи Луб Павло Миронович, к.т.н., доцент
Затверджені наказом по університету 28.04.2023 року № 133/к-с.
2. Строк подання студентом роботи 15.01.2024 р.

3. Початкові дані до роботи: 1. Система Entity Framework. 2. Мова програмування C#. 3. Інтегрована платформа розробки Visual Studio. 4. Хмарні сервіси Microsoft Azure. 5. Бази даних MySQL.

4. Зміст розрахунково-пояснювальної записки:

1. Аналіз IT-систем планування ресурсів виробництва

2. Вимоги до проектування системи управління інформаційними ресурсами підприємства

3. Методика побудови системи управління інформаційними ресурсами підприємства

4. Розробка і програмна реалізація фреймворку системи управління інформаційними ресурсами

5. Охорона праці та безпека в надзвичайних ситуаціях

Висновки та пропозиції.

Бібліографічний список.

Додатки.

5. Перелік графічного матеріалу: 1 та 2 – Тема, мета, завдання роботи; 3 – Головні поняття системи планування ресурсів підприємства ERP; 4 – Головні поняття системи планування ресурсів ERP; 5 – Проектування ERP-системи за модульним принципом; 6 – Загальні технічні та функціональні вимоги до розробки; 7 – Вибір технології розробки та доступу до баз даних – *Entity Framework core*; 8 – Розробка складових програмної системи; 9 – Розробка складових програмної системи; 10 – Програмна реалізація головних модулів фреймворку; 11 – Структура головного веб-сайту та API; 12, 13 – Висновки.

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 4, 6	*Луб П.М., доцент кафедри інформаційних технологій*		
5	*Городецький І.М., доцент кафедри фізики, інженерної механіки та безпеки виробництва*		

7. Дата видачі завдання – 12 травня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів роботи	Приміт-ка
1.	*Написання першого розділу та означення головних завдань роботи*	*28.04-20.06.23*	
2.	*Виконання другого розділу та опис інформаційних технологій для виконання завдань роботи*	*21.06-14.08.23*	
3.	*Виконання третього розділу, методика вирішення завдань та елементи наукових досліджень*	*15.08-31.10.23*	
4.	*Написання розділу: «Охорона праці та безпека в надзвичайних ситуаціях»*	*01.11-10.11.23*	
5.	*Головні результати отримані в роботі, оцінення розроблених пропозицій*	*20.11-30.11.23*	
6.	*Завершення оформлення розрахунково-пояснювальної записки та презентаційних матеріалів*	*30.11-01.12.23*	
7.	*Завершення роботи в цілому*	*01.12.23-10.01.24*	

Студент _____ Олещук Р.Я.
(підпис)

Керівник роботи _____ Луб П.М.
(підпис)

УДК: 658.51:631.3

Кваліфікаційна робота: 80 с. текст. част., 35 рис., 2 табл., 13 слайдів, 36 джерел.

Система управління інформаційними ресурсами підприємства із застосуванням технології Entity Framework. Олещук Р.Я. Кафедра ІТ. – Дубляни, Львівський НУП, 2024.

Проаналізовані сучасні ERP системи, і що вони із себе представляють. Визначено основні принципи їхньої роботи та архітектури. Проведений огляд систем-аналогів, детально розглянута концепція модульності, аргументовані причини використання даної концепції та її переваги.

Сформовано перелік важливих технічних та функціональних вимог до фреймворку. Вибрано технології, принципи і шаблони проектування та архітектури, які стали в нагоді в процесі розробки та значно полегшили його, а також процес тестування фреймворку.

За сучасними стандартами, шаблонами і принципами проектування і розробки створено фреймворк із перспективою його подальшого розвитку як комерційного проекту.

Розроблено фреймворк для створення веб-сервісів управління ресурсами підприємства, що відповідає технічним та функціональним вимогам.

Означено вимоги охорона праці та безпека в надзвичайних ситуаціях.

Ключові слова: система управління, інформаційні ресурси, ERP системи, архітектура, фреймворк, веб-сервісів.

ЗМІСТ

ВСТУП…… 7
РОЗДІЛ 1
АНАЛІЗ ІТ-СИСТЕМ ПЛАНУВАННЯ РЕСУРСІВ ВИРОБНИЦТВА…… 9
1.1. Головні поняття системи планування ресурсів підприємства ERP 9
1.2. Аналіз методів планування ресурсів виробництва та їх контролю 12
1.3. Аналіз відомих програмних рішень……...……... 16
РОЗДІЛ 2
ВИМОГИ ДО ПРОЕКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ ІНФОРМАЦІЙНИМИ РЕСУРСАМИ ПІДПРИЄМСТВА…….. 22
2.1. Проектування ERP-системи за модульним принципом…… 22
2.2. Вимоги до побудови локальної архітектури програмного забезпечення……..…….…… 25
2.3. Залучення хмарних сервісів та їх переваги ……. 28
2.4. Загальні технічні вимоги до систем управління інформаційними ресурсами підприємства…… 29
РОЗДІЛ 3
МЕТОДИКА ПОБУДОВИ СИСТЕМИ УПРАВЛІННЯ ІНФОРМАЦІЙНИМИ РЕСУРСАМИ ПІДПРИЄМСТВА……..…… 33
3.1. Вибір технології розробки та головних інструментів створення системи…… 33
3.2. Вибір постачальника хмарних даних…… 41
3.3. Розробка складових програмної системи…… 48
3.4. Інтеграція статичних і динамічних додатків та загальна структурна схема системи…….. 50
3.5. Моделі фільтрації даних та розбиття на сторінки…… 55
РОЗДІЛ 4.
РОЗРОБКА І ПРОГРАМНА РЕАЛІЗАЦІЯ ФРЕЙМВОРКУ СИСТЕМИ УПРАВЛІННЯ ІНФОРМАЦІЙНИМИ РЕСУРСАМИ ……..…….. 58
4.1. Програмна реалізація головних модулів системи фреймворку…….. 58
4.2. Реалізація ERP безпеки через механізм HMAC SHA256……..... 60
4.3. Загальна структура взаємодії хмарних сервісів…… 62

4.4. Розробка головного веб-сайту…… 63
4.5. Розробка API…… 69
РОЗДІЛ 5.
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ…….. 71
5.1. Розробка логіко-імітаційної моделі виникнення травм і аварій…… 71
5.2. Планування заходів із покращення умов праці…… 73
5.3. Безпека в надзвичайних ситуаціях……... 74
ВИСНОВКИ І РЕКОМЕНДАЦІЇ……..... 76
БІБЛІОГРАФІЧНИЙ СПИСОК……. 78

ВСТУП

Комп'ютерні системи що обслуговують інформаціно-комунікаціні процеси, а також зберігання даних, розвиваються трохи швидше, ніж інші, тому що їм доводиться йти в ногу з технологічними і соціальними змінами в світі. Розвиток систем управління процесами йде шляхом інтеграції в єдине ціле, і величезним кроком до цього є використання стандарту обробки даних за допомогою структурованої мови запитів SQL [5, 23, 27, 28].

Водночас, процес побудови бізнесу будь-якого рівня починається із встановлення системи управління та автоматизації бізнес-процесів, менеджменту її ресурсів із спільною базою даних тощо. Для вирішення цього завдання бізнесу доводиться звертатися до послуг IT- спеціалістів для розробки відповідних систем. Із часом у деяких IT-компаній виникла ідея створення загальної системи із усіма основними функціями, та постачати її в якості IT-послуги. Єдина проблема була в тому, що кінцевий, повністю готовий необхідний функціонал ніколи достеменно не відомий, оскільки у кожного замовника або компанії обов’язково виникнуть свої специфічні потреби.

Тому було вирішено створити свого роду «каркас» системи із базовим функціоналом, який, за потреби, може бути розширений встановленням додаткових сервісів та функціоналу. В результаті цього клієнт має вигоду в тому, що йому більше не потрібно звертатися до IT-спеціалістів за розробкою системи, що економить йому час та фінанси, а постачальник даної системи має прибуток від клієнта за її користування.

Мета роботи – підвищити ефективність управління ресурсами виробництва завдяки реалізації фреймворку для запуску веб-сервісів ERP систем зі здатністю до розширення шляхом інсталяції ERP модулів.

Завдання роботи: 1) проаналізувати IT-системи планування ресурсів виробництва; 2) навести вимоги до проектування системи управління інформаційними ресурсами підприємства; 3) описати методику побудови системи управління; 4) розробити і виконати програмну реалізацію фреймворку

системи управління інформаційними ресурсами.

Об’єкт – фреймворк для запуску веб-сервісів управління ресурсами підприємства.

Предмет роботи – показники швидкодії обробки даних завдяки використанню системи управління ресурсами підприємства.

Новизна одержаних результатів:

- структуровано функціональні та технічні вимоги до системи управління інформаційними ресурсами підприємства;

- розроблено фреймворк для запуску веб-сервісів ERP систем зі здатністю до розширення завдяки інсталяції ERP модулів;

- розроблено фреймворк, завдяки якому користувачі зможуть швидко запускати ERP системи із потрібними налаштуваннями в залежності від потреб їхнього бізнесу.

Практичне значення одержаних результатів полягає в створенні інформаційної системи взаємопов’язаних застосунків із використанням хмарних сервісів та технологій для управління фреймворком для створення систем управління ресурсами підприємства на основі необхідних користувачеві модулів. Завдяки даному фреймворку користувачі зможуть швидко запускати ERP системи із потрібними налаштуваннями в залежності від потреб їхнього бізнесу. Значну увагу приділено безпеці та надійності інформації, а також можливості інтеграції системи із іншими додатками.

Система управління інформаційними ресурсами підприємства призначена полегшити і спростити роботу виробничих підрозділів підприємства.

РОЗДІЛ 1
АНАЛІЗ ІТ-СИСТЕМ ПЛАНУВАННЯ РЕСУРСІВ ВИРОБНИЦТВА

1.1. Головні поняття системи планування ресурсів підприємства ERP

ERP – це скорочення від *Enterprise Resource Planning*, тобто планування ресурсів підприємства [1, 22, 24, 32]. Відомо, що в основі роботи будь-якого підприємства лежить низка взаємопов'язаних бізнес-процесів:

- управління фінансами, людськими ресурсами та закупівлями;
- планування ланцюга постачання;
- організація виробництва продукції чи надання певних сервісів;
- продажі та інші господарчі процеси, які залежать від специфіки діяльності самої компанії.

Головна функція ERP – об'єднання всіх бізнес-процесів (і ресурсів компанії) в одну систему, а також спрощення, прискорення й оптимізація. Цього вдається досягти завдяки автоматизації рутинних процесів та уникненню багаторазового введення тієї ж інформації виконавцями з різних відділів.

Проте ERP – це не просто бухгалтерія, кадри та закупівлі в одній програмі. Сучасні системи ERP оснащені такими новітніми технологіями, як штучний інтелект, інтернет речей, машинне навчання тощо, завдяки яким управління бізнесом стає ще прозорішим та ефективнішим.

Досить часто кожний відділ компанії користується різним програмним забезпеченням, яке не завжди сумісне між собою, тому доводиться вручну заносити в систему ті самі дані по декілька разів. Це забирає дорогоцінний час, а помилки та неточності під час такого введення інформації є просто неминучими. Тому досить часто спеціалісти з планування виробництва не знають точної кількості товарних запасів на складах, а бухгалтери не мають доступу до інформації, коли певний співробітник іде у відпустку.

Застосування ERP-системи передбачає використання єдиної чи інтегрованої бази даних, яка усуває зазначені труднощі. Це виключає

розбіжності, а також надає кожному відділу доступ до всієї необхідної інформації будь-коли й будь-де.

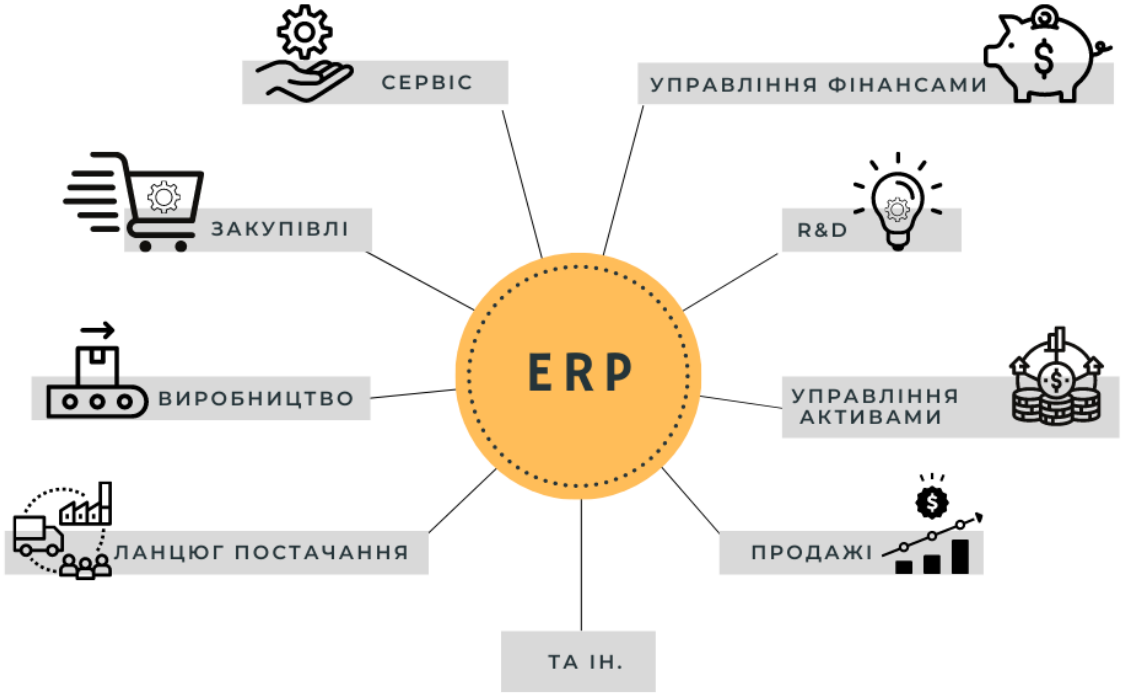


Рис. 1.1 – Структура інтегрованої ERP-системи [24]

Тому, коректне використання ERP-систем призводить до того що життя підприємця стає простішим та комфортнішим:

- ERP-системи дають змогу організувати робочий процес у торговій компанії так, щоб у супермаркеті не було ані порожніх полиць, ані непроданих прострочених продуктів. Система аналізує продажі й повідомляє, скільки та які товари потрібно закупити, щоб вони завжди були в наявності, але й не залежувалися на складах.
- швидка доставка з інтернет-магазинів – теж заслуга ERP-системи, яка оперативно обробляє замовлення та забезпечує прискорену логістику.
- промислові товари, які використовують щодня, теж акумульовані за допомогою ERP та автоматизації бізнес-процесів на виробництві. Ручна координація роботи виробничого підприємства потребувала б значних зусиль і часу, що в результаті призвело б до втрат доходу та прибутку.

Для прикладу, дати відповідь на запитання скільки та які ґудзики повинна

закупити фабрика одягу, щоб вчасно пошити нову колекцію, враховуючи збільшений попит перед різдвяними святами?! ERP-система зробить це автоматично та ще й покаже результати своїх розрахунків у наочній та зручній графічній формі.

Отже, саме ERP-системи допомагають уникнути простоїв і перевантаження виробничих потужностей, логістичних помилок, серйозних проблем із забезпеченням якості продукції та багатьох інших неприємностей для підприємця. ERP використовують не лише торгові, логістичні чи виробничі компанії, а й банки, медіакомпанії, авіалінії, мобільні оператори та інші підприємства, управління якими вимагає регулярної та швидкої обробки великої кількості даних і чіткої координації всіх бізнес-процесів. Тому, наявність такої системи – одна з необхідних умов для успішного функціонування та розвитку бізнесу будь-якого масштабу та галузі.

ERP-система – це комплекс програм для управління підприємством, які працюють на єдиній технологічній платформі, опираються на єдину базу даних (що може також бути інтегрованою з баз даних, отриманих із різних джерел чи систем) і синхронізуються між собою в реальному часі. Цей комплекс не є монолітним: підприємство може впроваджувати та використовувати тільки ті програмні модулі, які йому потрібні.

Частина модулів необхідна практично кожному підприємству, оскільки існують базові бізнес-процеси, такі як управління фінансами, продажами, персоналом, закупівлями тощо. Проте є і спеціалізовані рішення для різних галузей виробництва та сфери послуг. Наприклад, банки можуть користуватися особливими модулями для управління платіжними картками, а фабрики – модулями, які контролюють у цеху робочі процеси відповідно до виду виробництва (дискретне чи безперервне), або технічне обслуговування та ремонти обладнання.

ERP-системи зазвичай використовуються на великих підприємствах, проте існують і гнучкі ERP-рішення для середніх та малих підприємств. Звісно, найменші фірми можуть деякий час функціонувати й без ERP (якщо на

початковому етапі їм вистачає програми для бухгалтерського обліку та Excel). Але чим більше підприємство, тим більша ймовірність того, що воно потребує сучасної системи ERP.

1.2. Аналіз методів планування ресурсів виробництва та їх контролю

MRP II (Manufacturing resource planning – планування ресурсів виробництва) – метод ефективного планування ресурсів виробничого підприємства [7]. Дозволяє здійснювати виробниче планування та логістичне планування в натуральних одиницях вимірювання, фінансове планування – у вартісних одиницях вимірювання і надає можливість здійснювати моделювання з метою відповіді на питання типу «що буде, якщо…». Він складається з множини функцій, пов'язаних одна з одною:

- бізнес-планування;
- планування продажів і операцій (англ. *sales and operations planning*);
- планування виробництва (англ. *production planning*);
- формування Головного календарного плану виробництва (англ. *master production scheduling*);
- планування потреби в матеріалах;
- планування потреби в потужностях;
- система підтримки виконання планів для виробничих потужностей і матеріалів.

Система планування виробничих ресурсів (MRP II) — це інтегрована інформаційна система, яка використовується підприємствами. Планування виробничих ресурсів (MRP II) розвинулося на основі попередніх систем планування потреб у матеріалах (MRP) із додатковою можливістю інтеграції додаткових даних, таких як вимоги до робочої сили та фінансові потреби [1].

Система розроблена для централізації, інтеграції та обробки інформації для прийняття ефективних рішень у проєктуванні, плануванні, управлінні

запасами та контролю витрат у виробництві.

Рис. 1.2. – Модель планування в методології MRP II [7]

І MRP, і MRP II вважаються попередниками системи планування ресурсів підприємства (ERP), процесу, за допомогою якого компанія (зазвичай виробник) керує та інтегрує важливі частини свого бізнесу.

Інформаційна система управління ERP об’єднує планування, закупівлі, інвентаризацію, продажі, маркетинг, фінанси, людські ресурси та інші сфери. ERP найчастіше використовується в програмному середовищі, оскільки було розроблено багато великих програм для допомоги компаніям в процесі впровадження ERP.

MRP II — це комп’ютеризована система, яка використовує дані в реальному часі для створення детальних виробничих планів, які координують надходження комплектуючих матеріалів із наявністю обладнання та робочої сили. MRP II широко використовується сам по собі, але також використовується як модуль у більш просунутих системах планування ресурсів підприємства (ERP).

MRP II є розширенням оригінальної системи планування матеріальних

потреб (MRP I). Планування потреб у матеріалах (MRP) було однією з перших інтегрованих інформаційних систем на основі програмного забезпечення, призначених для підвищення продуктивності бізнесу.

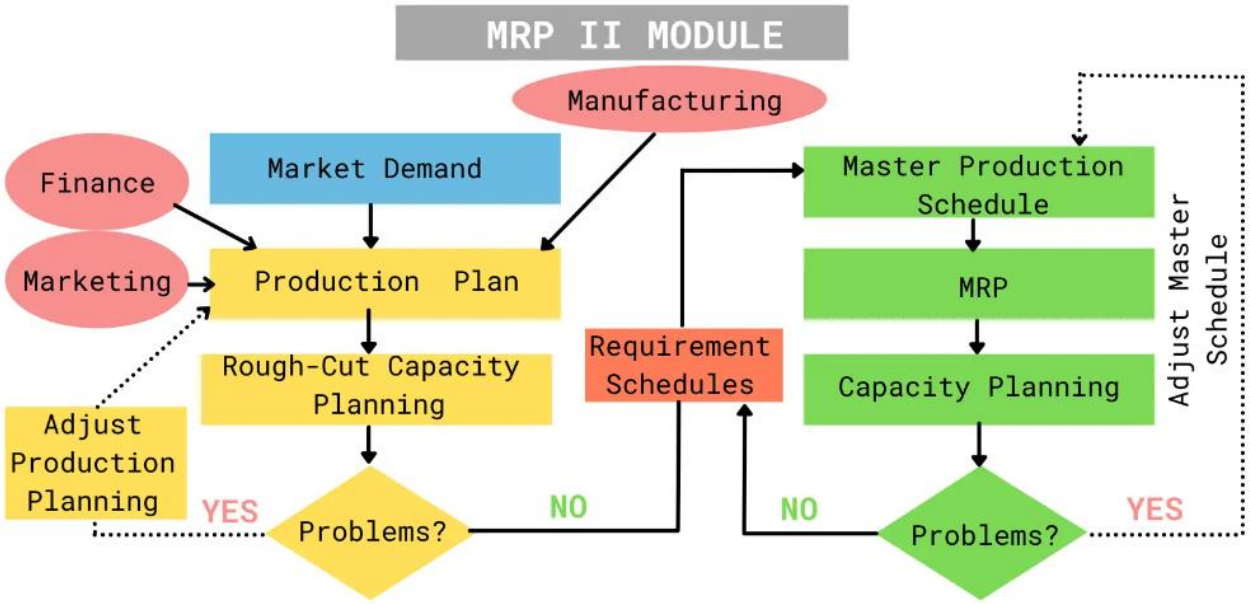


Рис. 1.3. – Структура модуля MRP II [10]

Інформаційна система планування потреб у матеріалах — це система, заснована на прогнозі продажів, яка використовується для планування постачання сировини та її кількості, враховуючи припущення щодо техніки і робочих одиниць, необхідних для прогнозування продажів.

Нижче наведено невелику вибірку деяких популярних постачальників програмного забезпечення MRP II станом на початок 2020 року:

– IQMS;
– Fishbowl;
– FactoryEdge;
– Prodsmart;
– Oracle Netsuite Manufacturing Edition;
– Epicor;
– S2K Enterprise.

Для всіх намірів і цілей MRP II фактично замінив програмне забезпечення MRP I. Більшість систем MRP II забезпечують усі функції системи MRP. Але

окрім планування основного виробництва, опису матеріалів (BOM) і відстеження запасів, MRP II також додатково забезпечує функціональність у сфері логістики, маркетингу та загальних фінансів.

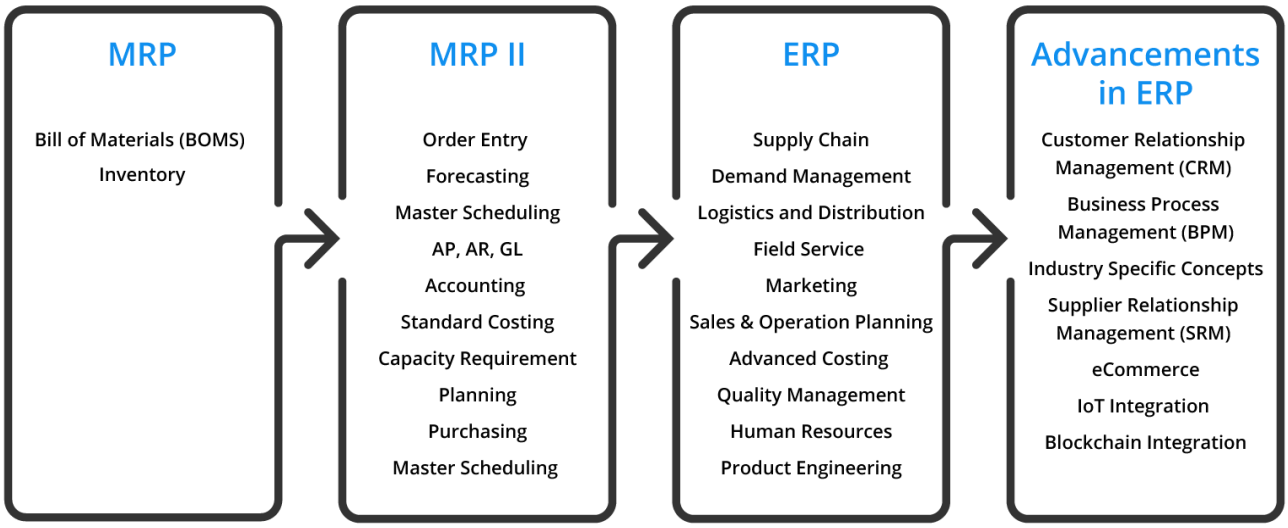


Рис. 1.4. – Інтеграція систем планування ресурсів виробництва MRP та ERP

Наприклад, MRP II може враховувати змінні, яких MRP не враховує, такі як можливості обладнання та персоналу, що дає більш реалістичне та цілісне уявлення про операційні можливості компанії. Багато рішень MRP II також пропонують можливості моделювання, які дозволяють оператору вводити змінні та спостерігати за ефектами. MRP II іноді називають системою замкнутого циклу, оскільки вона може надавати зворотний зв’язок щодо певних операцій. MRP I містив такі три основні функції:

– головне планування виробництва;
– опис матеріалів;
– відстеження запасів.

MRP II включає ці три, а також деякі додаткові:

– планування потужності техніки;
– прогнозування попиту;
– гарантія якості;
– загальний облік.

Система MRP II і сьогодні широко використовується виробничими компаніями як окреме рішення або як частина системи планування ресурсів підприємства (ERP). Програмна система Enterprise Resource Planning (ERP) вважається наступником програмного забезпечення MRP II.

Пакети ERP містять програми, що виходять далеко за межі виробництва. Це включає в себе все: від управління людськими ресурсами та відносинами з клієнтами до управління корпоративними активами.

1.3. Аналіз відомих програмних рішень

На основі проведеного попереднього аналізу були відібрані найбільш популярні аналоги даного фреймворку.

Microsoft D365. Одна з основних причин, через яку D365 є найпопулярнішим аналогом, полягає в тому, що він пропонує два різні рішення. Є Business Central, створений для малого бізнесу, та Finance and Operations, призначений для більших і складніших організацій [21].

Користувачі використовують дві абсолютно різні системи, що відповідають різним потребам різних типів організацій, але мають гнучкість і інтерфейс користувача Microsoft. Багатьом компаніям подобається цей інтерфейс, і вони цінують гнучкість, яку пропонує D365. Особливо в порівнянні з Oracle NetSuite або SAP S/4HANA, наприклад, Microsoft D365 є набагато простішим у формуванні.

Недоліком є те, що тільки те, що користувачі можуть змінити систему D365 не означає, що вони повинні це робити. Багато організацій настільки намагаються настроїти свої системи, що здаються під час впровадження. Ще одним привабливим аспектом Microsoft Dynamics є те, що її дуже легко інтегрувати з іншими системами.

На рисунку 1.5 можна побачити інтерфейс застосунку компанії Microsoft Business Central.

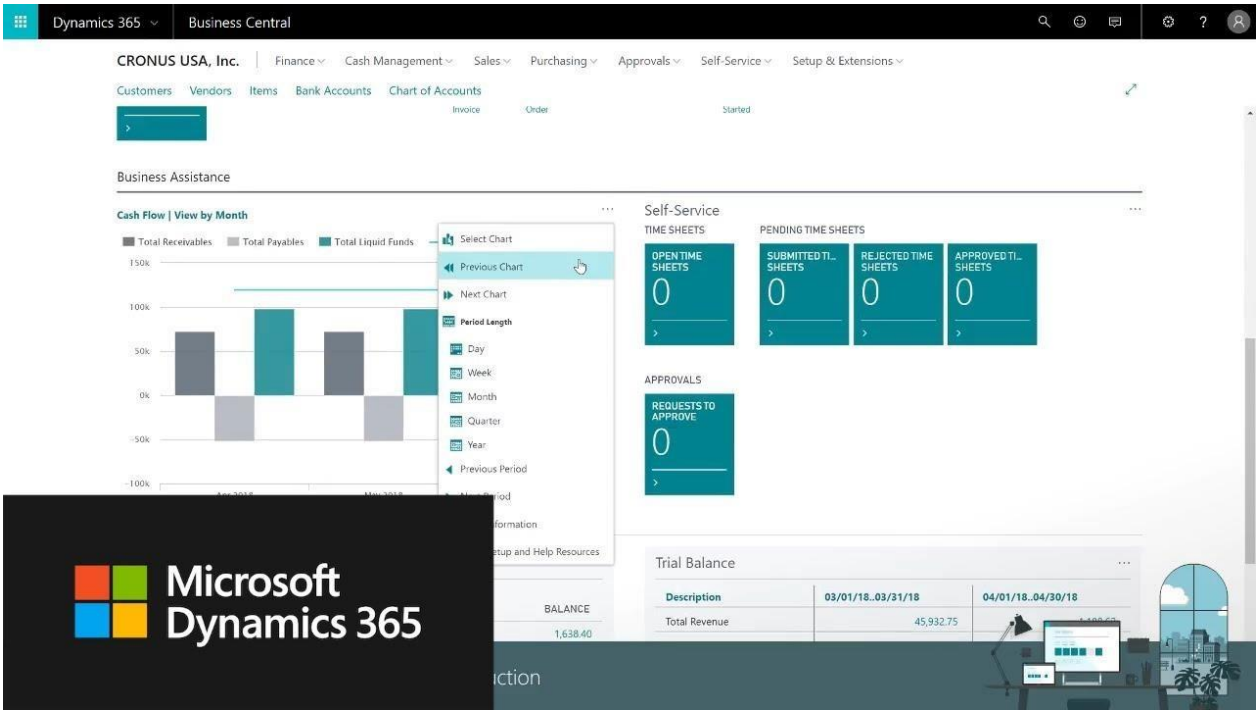


Рисунок 1.5 – Інтерфейс додатку MS Business Central

Oracle NetSuite. Позитивні аспекти Oracle NetSuite в тому, що це один із перших типів рішень «програмне забезпечення як послуга». Він знаходиться у хмарі вже 20 років, задовго до того, як його наздогнали решта постачальників. Таким чином, у них є дуже зріле рішення, яке все життя перебуває у хмарі. Він був створений для хмари та має хмарну архітектуру. NetSuite також орієнтований на малий бізнес, і в цьому випадку оновлення базової системи обліку до NetSuite може стати наступним логічним кроком у еволюції через цифрову трансформацію [25].

Недоліком Oracle NetSuite є досить висока ціна, особливо для малих і середніх організацій. Модель підписки, що повторюється, з великою кількістю прихованих витрат може з часом накопичуватися, тому в довгостроковій перспективі вона може бути дуже дорогою. Що дійсно стримує NetSuite, так це відсутність гнучкості в порівнянні з іншими системами на ринку.

На рисунку 1.6 зображено інтерфейс додатку Oracle NetSuite.

Oracle ERP Cloud. Oracle ERP Cloud є одним із стандартів для великих організацій зі списку Fortune 1000. Oracle ERP Cloud – це дуже широкий і надійний продукт, який може задовольнити потреби багатьох галузей, особливо

якщо для диверсифікованих, великих та складних організацій.

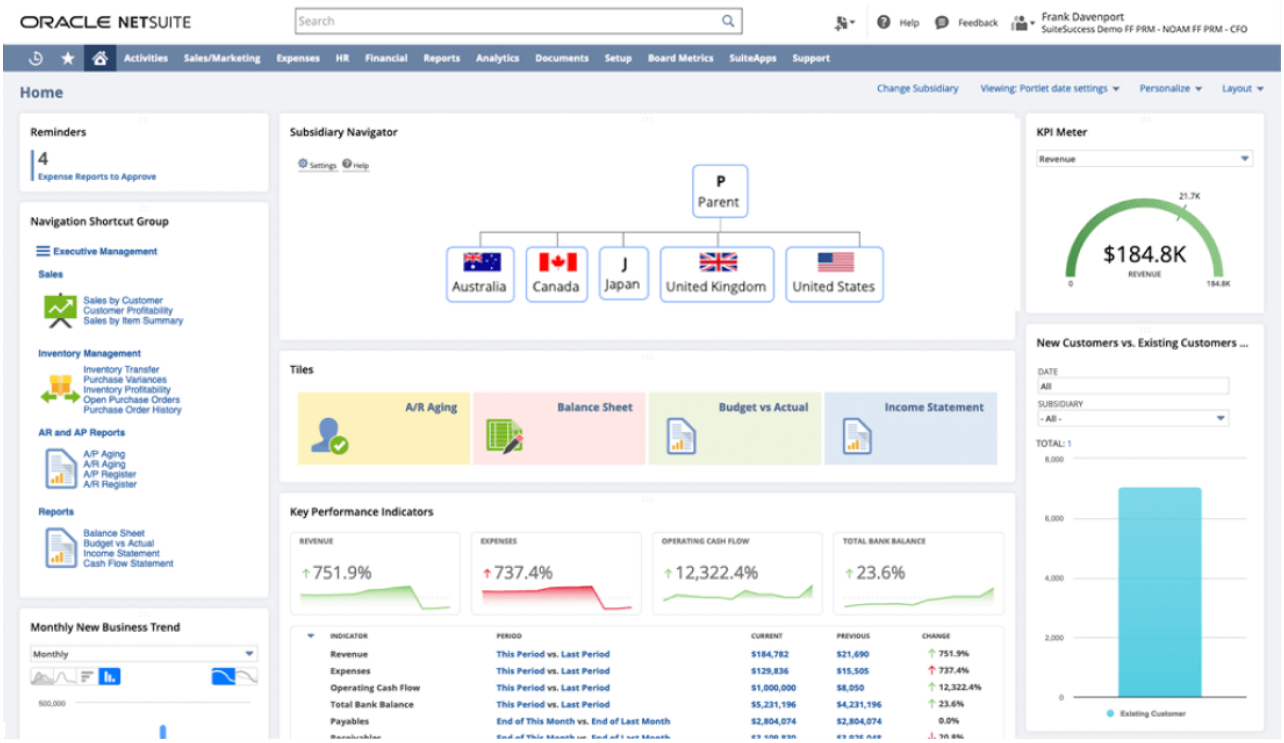


Рисунок 1.6 – Інтерфейс додатку Oracle NetSuite

Oracle Cloud ERP – це повний, сучасний хмарний пакет ERP, який надає командам розширені можливості, такі як штучний інтелект для автоматизації ручних процесів, які сповільнюють їх роботу, аналітику для реагування на зміни ринку в режимі реального часу та автоматичні оновлення, щоб бути в курсі і отримати конкурентну перевагу [24].

На рисунку 1.7 зображено інтерфейс додатку Oracle ERP Cloud.

Epicor Kinetic. Epicor Kinetic (раніше Epicor ERP) призначений для підтримки різних виробничих процесів, включаючи дискретні, виготовлення на замовлення (MTO), проєктування на замовлення (ETO), конфігурування на замовлення (CTO) та змішаний режим та середовище виготовлення на складі [17]. Epicor продовжує надавати найкращі в галузі рішення новим, швидкозростаючим виробникам, а також компаніям середнього розміру та дочірнім компаніям великих транснаціональних корпорацій. Epicor ERP – це модульна, багатофункціональна, масштабована система, яка підтримує зростання компанії за рахунок швидкого впровадження та легкого розширення,

незалежно від розміру чи складності виробничого процесу.

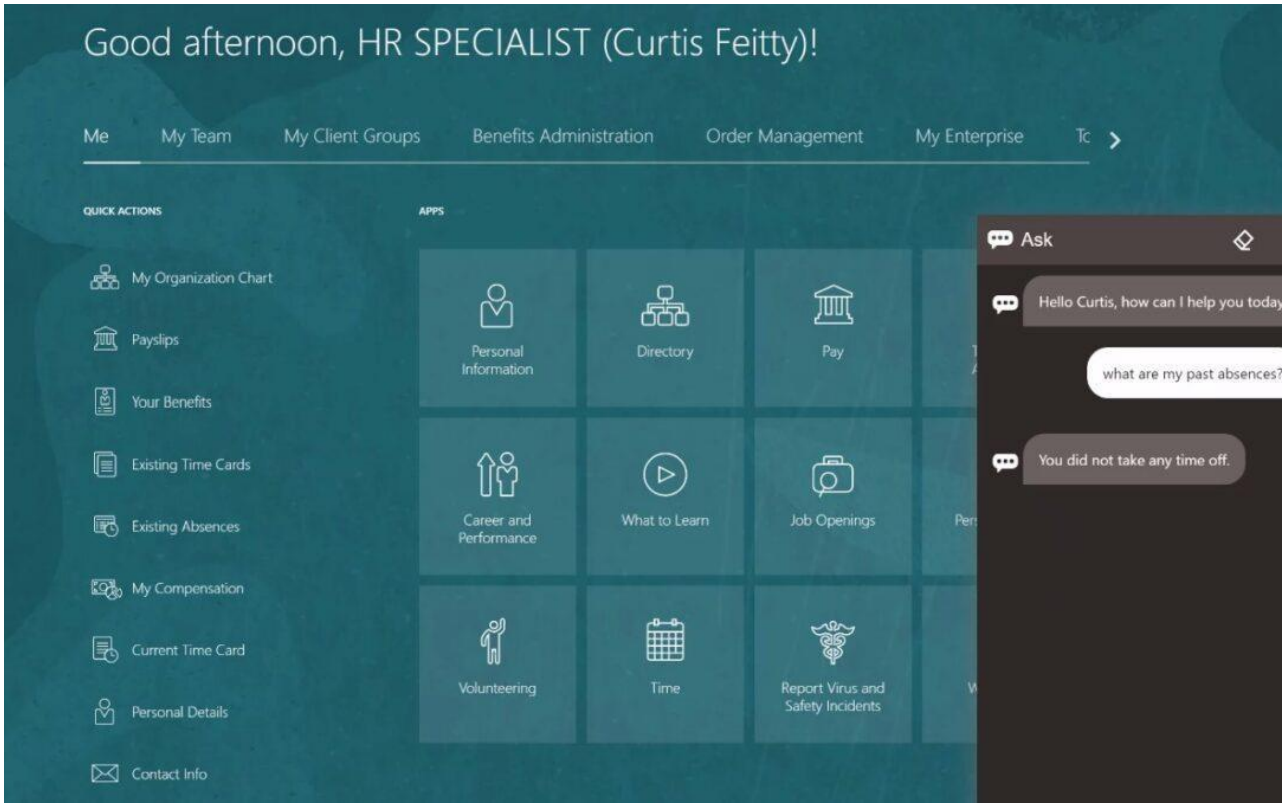


Рисунок 1.7 – Інтерфейс додатку Oracle ERP Cloud

Це рішення пропонує повну гнучкість у розгортанні. Користувачі можуть вибрати розгортання рішення у хмарі або локально. Будучи багатовимірним рішенням, Epicor Kinetic for Manufacturing має унікальну здатність керувати вимогами кількох галузей за допомогою єдиного рішення, включаючи промислове обладнання, автомобілебудування, аерокосмічну та оборонну промисловість, медичні пристрої, електроніку та високі технології, готові метали, послуги з металообробки, меблі та гумові чи пластмасові пристрої, майстерні та тощо [17]. Epicor має досвід роботи у багатьох виробничих галузях та розробив рішення, що відповідають унікальним вимогам забезпечення надійної функціональності у цих галузях.

У міру того, як бізнес росте і змінюється, власникам необхідне рішення, здатне впоратися з цим зростанням та змінами. Epicor Kinetic пропонує велику гнучкість, оскільки його можна розгорнути локально або у хмарі. Наприклад, якщо у компанії обмежені IT-ресурси, користувачі можуть спочатку розгорнути

рішення у хмарі. У міру зміни бізнесу користувачі можуть пізніше ухвалити рішення про повторне розгортання Epicor Kinetic On-Premises.

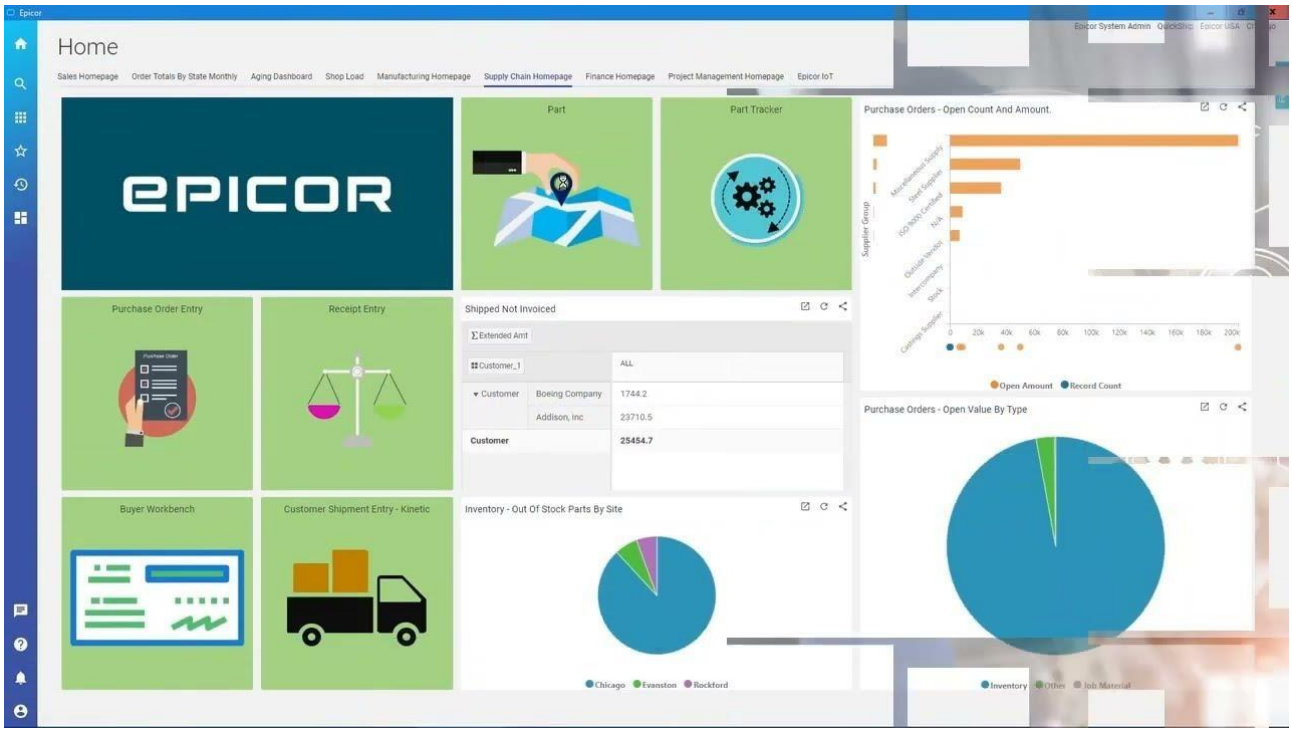


Рисунок 1.8 – Інтерфейс додатку Epicor Kinetic

SAP S/4HANA. S/4HANA відмінно підходить для фінансів, управління запасами, основних функцій ERP та багато іншого.

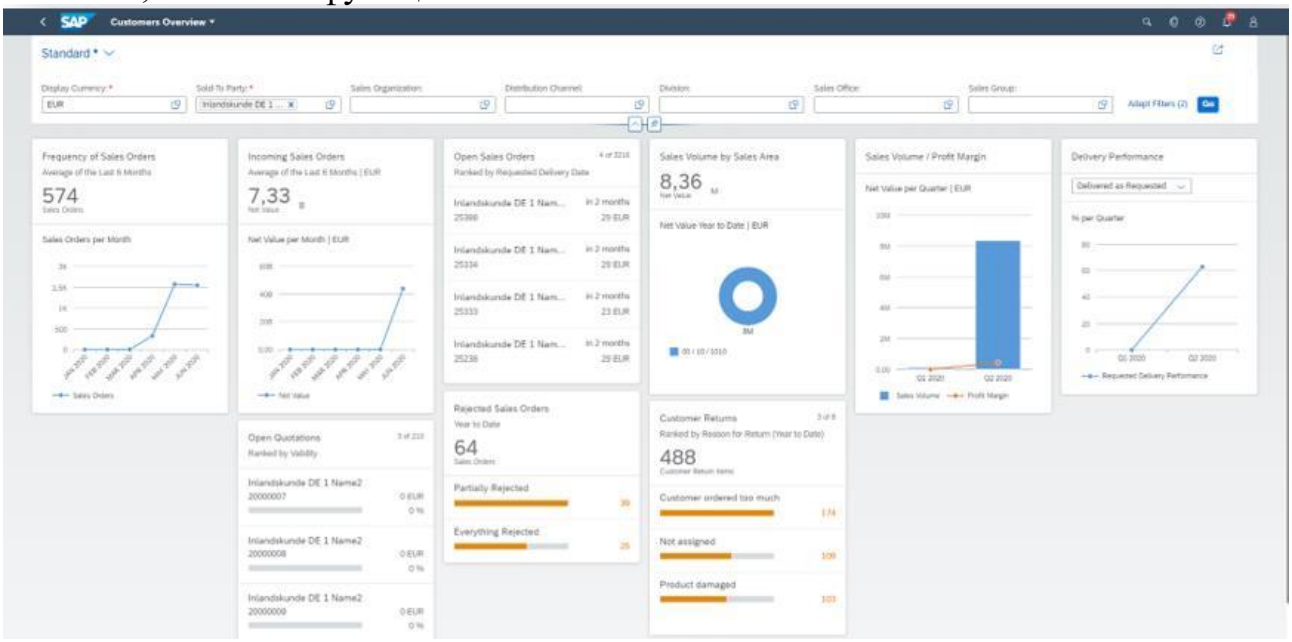


Рисунок 1.9 – Інтерфейс додатку SAP S/4HANA

Це просто один з кращих, коли йдеться про фінансову гнучкість,

можливості GL, цінність продукту та багато іншого. З іншого боку, S/4HANA відсутні розширені функції, такі як виробництво, планування та управління життєвим циклом продукту, і навіть відсутні деякі функції CRM.

Що SAP зробила для часткового вирішення цієї проблеми, то це придбала інші компанії. Вони придбали такі продукти, як Ariba для закупівель, Success Factors для управління людським капіталом та Concur для економії часу та витрат. Вони стали певною мірою кращими у своєму класі постачальниками, але разом з цим виникає проблема інтеграції кількох систем [30]. Дорожня карта SAP все ще трохи незграбна, і важко зрозуміти, які продукти підходять для SAP.

РОЗДІЛ 2
ВИМОГИ ДО ПРОЕКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ ІНФОРМАЦІЙНИМИ РЕСУРСАМИ ПІДПРИЄМСТВА

2.1. Проектування ERP-системи за модульним принципом

Ключовим аспектом будь-якої ERP-системи є модульність. Основою цієї концепції є поділ загальних сутностей або функціональних можливостей на окремі блоки. У той же час модульність — це тип властивості системи, яка вимірює ступінь, наскільки сильно пов’язані компоненти в системі можуть бути розділені на окремі сутності або кластери, які взаємодіють один з одним більше, ніж інші. У сильно взаємопов’язаних системах із низьким рівнем модульності несправності в одному компоненті можуть каскадно поширюватися на інший, збільшуючи ризик відмови всієї системи. І навпаки, у системі з високим ступенем модульності збурення одного компонента краще стримуються, і менш імовірно, що вони поширюватимуться на інші компоненти [6, 12]. Ідея модульності широко практикується в управлінні різними системами.

Система вважається «модульною», якщо, наприклад, її можна розділити на кілька компонентів, які можна комбінувати та поєднувати в різних конфігураціях. Компоненти можуть певним чином з’єднуватися, взаємодіяти або спільно використовувати ресурси (наприклад, дані), дотримуючись стандартизованих інтерфейсів. Модульний продукт — це система «слабко пов’язаних» компонентів, на відміну від тісно інтегрованого продукту, де кожен компонент спеціально розроблений для роботи з певними іншими компонентами в тісно пов’язаній системі.

У рамках ERP-системи модулі вважаються програмними пакетами зі своїми моделями, послугами та функціями [32]. Приклад змісту модуля зображено на рисунку 2.1.

Кожен модуль може бути реалізований як за вимогами клієнта, так і

компанією-власником цього фреймворку. Клієнтам надається можливість вибирати модулі, необхідні їм для ведення свого бізнесу, тому власники фреймворку повинні розробити якнайбільше різних модулів, щоб залучити якнайбільше користувачів, що буде в їхніх інтересах.

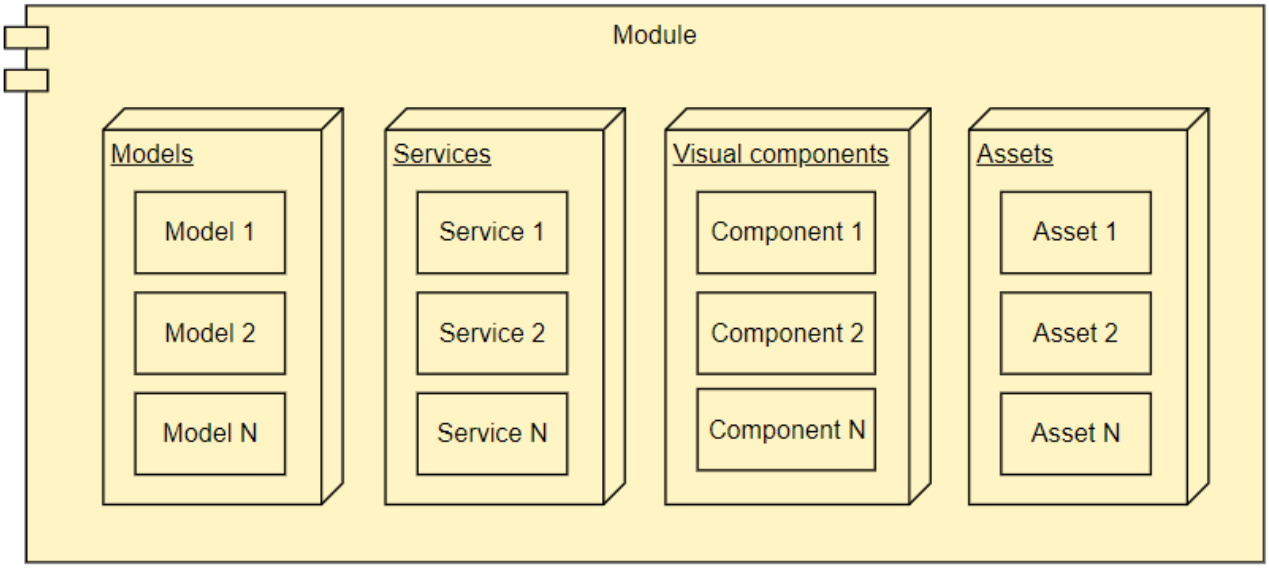


Рис. 2.1 – Схема змісту програмного пакета модуля

Як уже згадувалося в главі 1, ERP-системи призначені для централізації, консолідації та обробки інформації для ефективного прийняття рішень у галузі планування, проєктування, управління запасами та контролю витрат у виробничій інформаційній системі. Звідси можна виділити основні програмні елементи, які необхідні коректної роботи всієї системи:

— компонент роботи із базою даних (для зберігання корпоративної інформації компанії);

— компонент зв’язку і комунікації (для обробки зовнішніх запитів);

— головне програмне ядро (для управління взаємодією інших компонентів).

Окрім функціоналу, впровадженого шляхом встановлення того або іншого модуля, у ERP системі має бути встановлений певний набір функціональних особливостей.

Безпека ERP – це практика вжиття ефективних заходів безпеки для запобігання вторгненню у систему ERP. Захищена система ERP охоплює

безпечну конфігурацію сервера та дозволяє вести журнал безпеки, забезпечувати безпеку зв'язку системи та безпеку даних. Користувачі та авторизації однаково важливі.

Безпека ERP часто є сліпою плямою для багатьох компаній. У міру того, як компанії витрачають час і ресурси на модернізацію своїх ERP-систем та елементів управління, вони часто не беруть до уваги або недостатньо інвестують у належну кібербезпеку. Атаки на ERP-системи можуть мати руйнівні наслідки діяльності компанії, оскільки вони беруть участь у багатьох повсякденних операціях [32]. І зі зростанням загроз кібербезпеці потенціал фінансових втрат і репутаційного збитку стає реальним. Організації повинні захищати свої системи від внутрішніх та зовнішніх кіберзагроз, щоб підтримувати конфіденційність, доступність та цілісність.

Належні засоби контролю за допомогою впровадження рішень безпеки ERP та їх інтеграції з іншими безпековими операціями мають вирішальне значення для підтримки захисту та безпеки вашої системи ERP.

Інтеграція ERP – це метод підключення системного програмного забезпечення ERP до інших систем. Інтеграція забезпечує потік інформації між різними системами. Це спосіб автоматизувати бізнес-процеси та підвищити продуктивність всього підприємства [7]. Це дозволяє підприємствам швидше обробляти замовлення та збільшувати обробку замовлень на веб-сайті того ж дня, заощаджуючи багато грошей та часу. Без інтеграції з ERP адміністраторам довелося б вручну оновлювати рівні запасів у системі ERP і додатку електронної комерції. У міру збільшення обсягів продажу та зростання запасів неефективні та трудомісткі операційні процеси стають важко масштабованими. Інтеграція з ERP допоможе користувачам отримати огляд компанії та всіх запущених програм. Він допомагає відстежувати ключові показники ефективності в режимі реального часу, забезпечувати цілісність даних та багато іншого.

Роль систем ERP в системній інтеграції є вирішальною, оскільки вони містять дані про клієнтів, продукти та замовлення на продаж. Існує безліч типів

ERP інтеграцій, які оптимізують бізнес-процеси, серед них основними є:

— HTTP(S) інтеграція;

— ESB інтеграція.

HTTP(S) інтеграція з’єднує ERP систему із одним конкретним додатком (зазвичай – API сервером). Реалізовано це у вигляді звичайного HTTP (або HTTPS) запиту на вказану адресу із надсиланням даних. Дана інтеграція може бути використана як у системі декількох окремих додатків, розміщених на одному або різних серверах, так і для обміну даними серед додатками різних компаній (наприклад, ERP система інтернет-магазину може надіслати запит до API логістичної компанії для відправлення замовлення). Сам запит може бути налаштований (мати конкретний тип і/або заголовки запиту).

2.2. Вимоги до побудови локальної архітектури програмного забезпечення

Іншим рішенням є Enterprise Service Bus або ESB. Це локальна архітектура програмного забезпечення, яка дозволяє різним програмам спілкуватися одна з одною та обмінюватися даними [18]. Дані надходять на шину в певному форматі, часто JSON, і існують адаптери, які знаходяться між шиною та ERP системою, яке перекладає їхні дані в/з JSON і в потрібний формат. Схема підключенням різних додатків до ESB шини ERP системи зображено на рисунку 2.2.

Управління ідентифікацією та доступом (IAM) – це управління поточним життєвим циклом облікових записів та прав доступу до всіх корпоративних інформаційних ресурсів, як у корпоративних центрах обробки даних, так і у хмарі [18]. Ця функція є центральною для безпеки системи, оскільки вона автентифікує користувачів та регулює доступ до систем та даних. Підприємства можуть використовувати стратегію нульової довіри для визначення користувачів. Хмарні служби управління ідентифікацією використовують

інтеграцію з відкритими стандартами зниження накладних витрат і витрат за обслуговування.

Цей процес включає перевірку облікових даних користувача та відповідних системних прав доступу. Рішення IAM надає інструменти для керування цифровою ідентифікацією користувачів системи та забезпечення належного доступу до ресурсів компанії. Ці рішення дозволяють адміністраторам відстежувати дії користувачів, створювати звіти про дії користувачів та застосовувати політики для забезпечення відповідності вимогам.

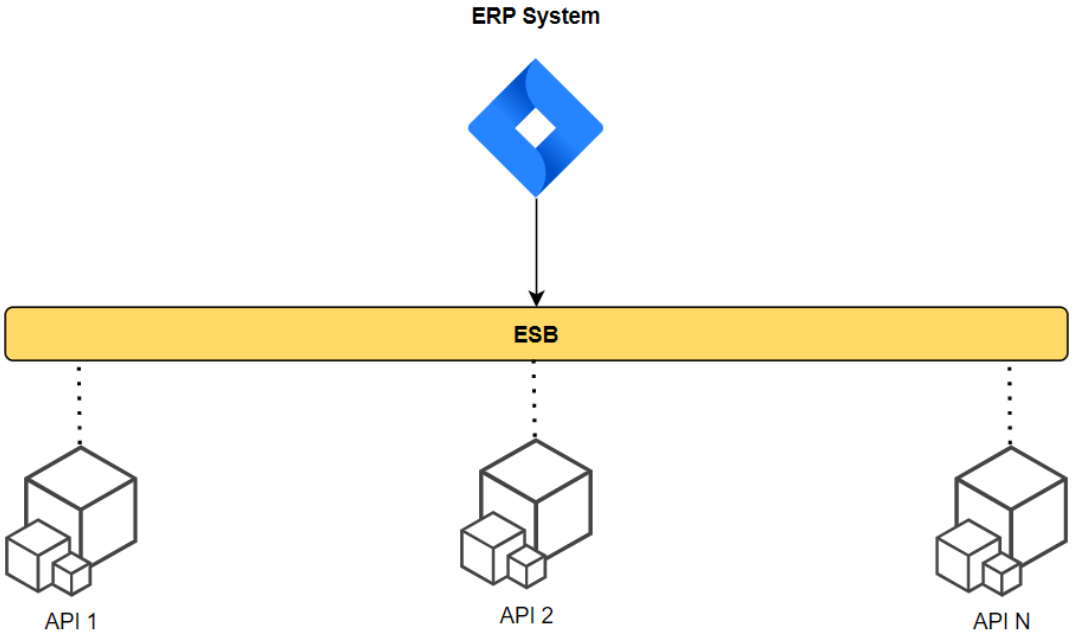


Рис. 2.2 – Приклад ESB інтеграції

IAM є найважливішим інструментом захисту корпоративних ресурсів від загроз кібербезпеці. Система IAM забезпечує узгодженість правил та політик доступу користувачів у всій компанії, а також правильне та точне застосування прав доступу до ресурсів у міру того, як користувачі змінюють роль компанії. Без автоматизованого моніторингу ресурсів та дій підприємства вразливі до витоку облікових записів та даних. Це часто скомпрометовано проблемою надмірних дозволів, які не керуються належним чином.

Рішення IAM забезпечують керування доступом на основі ролей, що дозволяє адміністраторам регулювати доступ окремих користувачів до систем або мереж. Ключові функції включають:

– гарантії життєвого циклу облікових записів: управління життєвим циклом облікових записів користувачів, включаючи права та ініціалізацію;

– керування доступом: єдиний вхід (SSO) та багатофакторна автентифікація (MFA) часто використовуються для управління уніфікованими політиками доступу.

– служба каталогів: централізоване та уніфіковане керування та синхронізація даних облікового запису.

– ініціалізація користувачів: автоматичне створення та призначення нових облікових записів користувачів.

– аналіз ідентифікаційних даних: застосування машинного навчання для виявлення та запобігання підозрілим діям з використанням ідентифікаційних даних.

– єдиний вхід або єдиний вхід (SSO): об'єднання пароля та облікових даних користувача в один обліковий запис та використання надійних паролів для спрощення доступу до служб.

– багатофакторна аутентифікація (MFA): складніша аутентифікація у вигляді додаткових факторів контролю аутентифікації. Це допомагає гарантувати справжність користувача та знижує ризик крадіжки облікових даних.

– аутентифікація на основі ризику: використання алгоритмів розрахунку ризику дій користувача. Блокуйте та повідомляйте про ризиковані дії.

– контроль над ідентифікаційними даними та їхнє керування ними: зменшення ризику надмірного доступу та привілеїв за допомогою елементів керування привілеями.

2.3. Залучення хмарних сервісів та їх переваги

На основі проведеного аналізу існуючих аналогів необхідно визначити детальні технічні та функціональні вимоги для проектної реалізації фреймворку, що розглядається в даній роботі.

Хмарні обчислення – це доставка IT-ресурсів на запит через Інтернет з оплатою за фактом використання [20]. Замість того, щоб купувати, володіти та обслуговувати фізичні центри обробки даних та сервери, користувачі можуть споживати обчислювальну потужність, сховище, бази даних та інші технологічні послуги на запит від хмарних провайдерів.

Організації всіх типів, розмірів та галузей використовують хмару для різних цілей, включаючи резервне копіювання даних, аварійне відновлення, електронну пошту, віртуальні робочі столи, розробку та тестування програмного забезпечення, аналіз великих даних та клієнтські веб-додатки. Наприклад, медичні компанії використовують хмарні технології розробки більш персоналізованих методів лікування своїх пацієнтів. Фінансові компанії використовують хмару для виявлення та запобігання шахрайству в режимі реального часу.

Перевагами хмарних обчислень є наступні пункти:

– швидкість: хмара надає користувачеві легкий доступ до широкого спектру технологій, для швидшого впровадження інновацій;

– еластичність: у разі використання хмарних обчислень користувачам не потрібно попередньо виділяти ресурси для відповідності майбутнім піковим рівням ділової активності. Натомість вони отримують ту кількість ресурсів, яка їм справді потрібна. Клієнти можуть масштабувати ці ресурси, щоб миттєво збільшувати чи зменшувати ємність у міру зміни потреб бізнесу;

– економія коштів: у хмарі користувачі можуть замінити фіксовані витрати (наприклад, центри обробки даних та фізичні сервери) змінними витратами та платити за IT лише за те, що використовують. Крім того, змінні витрати набагато нижчі, ніж користувачі платять самі собі через ефект

масштабу;

– вихід на глобальний рівень за лічені хвилини: хмара дозволяє користувачам освоювати нові географічні регіони та глобально розгортати програми та програми за лічені хвилини. Розміщення програм ближче до кінцевих користувачів скорочує час затримки та підвищує зручність роботи.

З урахуванням наведених вище аргументів, можна зробити висновок та прийняти за вимогу те, що усі додатки та функціонал сервісу має бути опублікований лише за допомогою хмарних технологій, що стане в нагоді в питанні економії часу і фінансів, а також полегшить процес управління архітектурою фреймворку в цілому.

2.4. Загальні технічні вимоги до систем управління інформаційними ресурсами підприємства

Фреймворк, що розробляється, повинен давати користувачу можливість швидко створювати веб-сервіси ERP систем у глобальному Інтернеті із глобальним Інтернет доменом. Створити та видалити ERP системи повинна бути змога як і через головний веб-сайт, так і через запит до API даного фреймворку. При процесі створення користувач повинен мати досить різноманітний вибір налаштувань, таких як назва системи, тип бази даних, логін і пароль для доступу до БД тощо. Сам процес запуску ERP системи має займати невеликий проміжок часу – не більше години.

Основною вимогою до системи є концепція модульності. Система повинна мати можливість легко доповнюватись новими модулями, так само швидко і легко модулі мають видалятися із системи за необхідності. Самі модулі мають коректно працювати як в якості незалежного компоненту, так і у взаємодії із іншими компонентами системи в цілому.

Фреймворк має забезпечувати швидкий і безпечний процес передачі запитів та їхньої обробки. Цей процес має бути реалізований із використанням

сучасних алгоритмів шифрування даних, таким як підпис запитів (англ. «request signing»).

У системі має бути ретельно реалізований процес ідентифікації користувача та його прав. Кожний запит від користувача повинен перевірятися із правами даного користувача та блокуватися у випадку недостатнього доступу.

Власник фреймворку повинен мати швидкий доступ до інформації даного сервісу у будь-який момент часу. Одним із варіантів вирішення даного питання є розробка окремого додатка конкретно для власника даного фреймворку.

Оскільки в перспективі даний сервіс може стати комерційним, і ERP системи клієнтів потребуватимуть фінансів для оплати хмарних ресурсів, важливо розробити механізм збереження інформації щодо використання користувачів тих або інших ресурсів, за які користувачеві надходитиме плата.

Функціональні вимоги до користувацьких можливостей головного веб-сайту. Головний веб-сайт є основним веб-застосунком, яким буде користуватися потенційний користувач фреймворку. До основних вимог щодо даного додатку належать:

– реєстрація за допомогою електронної пошти та паролю та верифікація шляхом надсилання електронного листа на вказану електронну пошту;

– авторизація за допомогою електронної пошти та паролю;

– можливість відновлення паролю через електронну пошту;

– можливість зміни паролю;

– перегляд створених ERP систем;

– створення ERP системи за бажаними налаштуванням та із бажаними доступними модулями;

– перегляд детальної інформації кожної ERP системи окремо;

– видалення ERP системи у будь-який момент після її повного запуску;

– перегляд рахунків та статистики використання ERP систем (кількість годин роботи тощо);

– ознайомлення із фреймворком в цілому та його конкретними

можливостями.

Вимоги до користувацьких можливостей бек-офісу. Бек-офіс (англ. «back office», «BO») є веб-застосунком, розробленим виключно для власника фреймворку. Він слугуватиме для отримання власником актуальної інформації щодо сервісу. До основних вимог щодо даного додатку належать:

– перегляд зареєстрованих користувачів фреймворку;

– перегляд створених ERP систем;

– перегляд існуючих модулів, додавання нових, а також оновлення та видалення існуючих;

– перегляд статусу працездатності інших застосунків або ресурсів фреймворку, а саме: головного веб-сайту, головної бази даних та API;

– перегляд статусу використання хмарних ресурсів, таких як обчислювальні машини, бази даних, інтернет доменів, хмарне сховище тощо.

Вимоги до користувацького процесу веб-інтерфейсу ERP системи. Веб-інтерфейс ERP системи також є не менш важливим, оскільки саме цей застосунок користувач побачить в якості кінцевого результату, і саме із цим додатком він буде працювати. До основних вимог щодо даного застосунку належать:

– перегляд встановлених модулів та їхнє видалення із ERP системи;

– доступ лише для власника ERP системи;

– унеможливлення неавторизованому користувачу переходу на всі сторінки, крім сторінки авторизації;

– перегляд даних кожних таблиць встановлених модулів;

– запис у системний журнал усієї активності та запитів від користувачів;

– перегляд системного журналу;

– створення, редагування та видалення IAM облікових записів;

– створення, редагування та видалення інтеграцій та їхній статус;

– перегляд налаштувань ERP системи;

– перегляд статусу мережі та статистики запитів;

– забезпечення швидкої і надійної обробки запитів від користувача та

повідомляти про помилки під час обробки;

– перегляд доступних невстановлених модулів та їхня інсталяція до системи.

Вимоги до користувацьких можливостей консолі взаємодії та управління. Консоль управління є веб-застосунком для працівників компанії. Саме через цей застосунок працівники підприємства будуть авторизуватись, використовуючи ключі доступу IAM облікових записів, і працювати із ERP системою. Основними вимогами до цього додатку є наступні пункти:

– унеможливити неавторизованим користувачам переходити на будь-які сторінки додатку, окрім сторінки авторизації;

– візуальне відображення та доступний функціонал має повністю залежати від прав доступу авторизованого користувача;

– стабільна система запитів до ERP системи користувача.

Вимоги до процесу роботи API. API даного фреймворку повинен надавати можливість отримувати інформацію для користувачів не лише через головний веб-сайт, а і через звичайні запити. Це дасть можливість іншим розробникам інтегрувати свої застосунки із даним фреймворком та отримувати необхідні дані. До основних вимог щодо даного застосунку належать:

– обробка запитів лише для зареєстрованих користувачів, що пройшли верифікацію;

– система запитів має бути захищена за існуючими стандартами ERP безпеки;

– запити мають оброблятися швидко, не залежачи від кількості трафіку.

Вимоги до розробки пакету інструментів розробки SDK. Останньою вимогою є створення пакету інструментів розробки (англ. «*Software Development Kit*»). SDK покращують стандартні процеси та надають доступ до необхідної інформації та створені для інформування, надання необхідних інструментів і шаблонів, завдяки чому розробники можуть сфокусуватися на розробці свого продукту [31, 34].

РОЗДІЛ 3
МЕТОДИКА ПОБУДОВИ СИСТЕМИ УПРАВЛІННЯ ІНФОРМАЦІЙНИМИ РЕСУРСАМИ ПІДПРИЄМСТВА

3.1. Вибір технології розробки та головних інструментів створення системи

Керуючись розробленими технічними та функціональними вимогами потрібно провести аналіз технологічного стеку, принципів побудови веб-застосунків та вибрати сучасні технології що повинні вирішене завдання кваліфікаційної роботи. Тому, головною технологією для розробки веб-застосунків нами вибрано платформу .NET [36] від компанії Microsoft та її головну мову програмування C#.

Основна перевага полягає в тому, що C# – це сучасна об'єктно-орієнтована мова програмування, також від Microsoft [9, 20, 35]. Ця мова є наступником мови C++, включає статичну типізацію даних та підтримує принципи ООП. Основні особливості цієї мови програмування:

– об'єктно-орієнтована, компонентно-орієнтована мова програмування;

– механізм складання сміття автоматично звільняє пам'ять, зайняту об'єктами,що не використовуються;

– обробка винятків забезпечує структурований та розширюваний підхід довиявлення та усунення помилок;

– синтаксис Language Integrated Query (LINQ) створює загальний шаблон длякерування даними з будь-якого джерела;

– мовна підтримка асинхронних операцій забезпечує синтаксис для побудовирозподілених систем;

– підтримка механізмів, що використовують залежність (Dependency Injection);

– C# підтримує керування версіями, тому програми та бібліотеки можутьзгодом розвиватися сумісним чином.

Blazor. Для створення веб-застосунків вибрано технологію реалізовану в платформі .NET, а також обрано сучасний механізм під назвою Blazor. Даний інструментарій на основі ASP.NET Core активно розвивається компанією Microsoft, є дуже популярним серед .NET розробників завдяки легкості роботи і стає все більш популярним [11].

Ця технологія є аналогом SPA від компанії Microsoft. SPA (Single Page Application) є сучасним типом веб-додатків, який є односторінковим додатком, яке завантажується на одну HTML сторінку, і далі, завдяки механізму мови JavaScript, динамічно оновлює контент сторінки, завантажуючи потрібні компоненти. Перевагою такого підходу є те, що в процесі роботи користувачеві може здатися, що він запустив не веб-сайт, а десктопний додаток, оскільки він миттєво реагує на всі його дії, без затримок та «підвисань», також такий додаток має вбудовану оптимізацію та розподіл ресурсів, що суттєво зменшить навантаження.

Основними фреймворками такого типу веб-застосунків саме на мові JS є:

– Angular;
– React;
– Vue JS.

Єдиною відмінністю між переліченими фреймворками та інструментом Blazor є те, що в Blazor при розробці застосунків використовується мова C#, а не JS, що полегшує роботу .NET розробникам, оскільки більше немає потреби вивчати JS в якості додаткової мови програмування, хоча навіть у випадку необхідності використання саме цієї мови (наприклад, для графічних задач), у Blazor є вбудований механізм виклику JS скриптів.

Blazor Server. На даний момент існує 2 типи веб-застосунків фреймворку Blazor:

– Blazor Server;
– Blazor WebAssembly.

Різниця між вказаними типами в тому, що у Blazor Server зміни у UI фіксуються та обробляються механізмом SignalR, і надсилаються на сервер, на

стороні якого генерується оновлений UI і повертається клієнтові (рис. 3.1), в той час як у Blazor WebAssembly маніпуляції із UI відбуваються за допомогою вищезгаданої мови JavaScript.

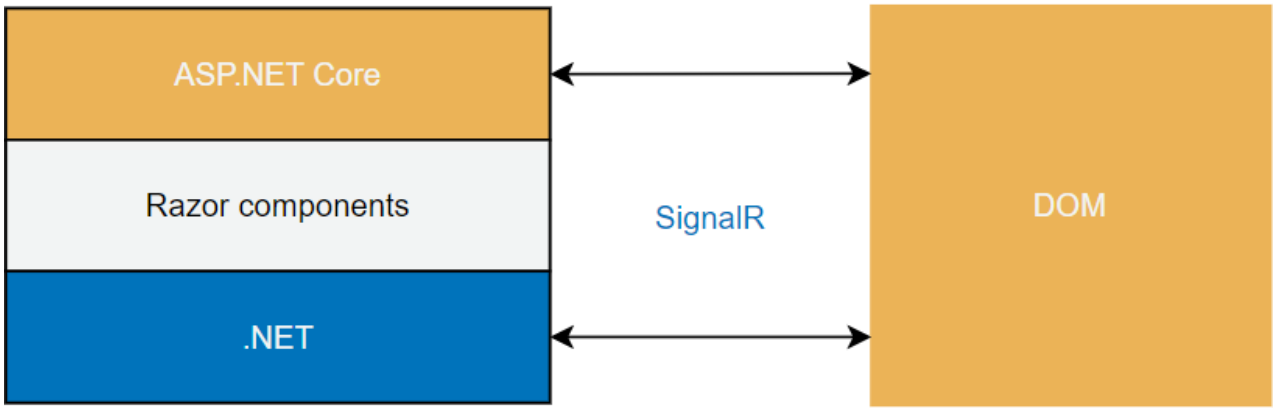


Рисунок 3.1 – Схема взаємодії UI та серверу у Blazor Server

WebAssembly за своєю суттю є PWA – прогресивна програма, або Progressive Web Application, яка взаємодіє із користувачем, як додаток. Вона може встановлюватися на головний екран смартфона, надсилати push-сповіщення та працювати в офлайн-режимі. Перевагами такого типу є кросплатформенність, висока швидкість роботи та можливість запуску і відображення даних в офлайн-режимі з моментальним завантаженням та висока швидкість встановлення.

Важливою перевагою Server над WebAssembly є те, що у випадку із WebAssembly бібліотеки із згенерованим програмним кодом будуть доступні клієнтові, оскільки обробка запитів відбувається саме на стороні клієнта. Це означає, що клієнт у будь-який момент може в налаштуваннях браузера, у секції Developer Tools, знайти ці бібліотеки та скачати вихідний код додатку, що є негативним аспектом із точки зору безпеки.

Саме через цю причину перевага буде надана Blazor Server.

SignalR. SignalR ASP.NET Core — це бібліотека з відкритим кодом, яка спрощує створення веб-функцій у режимі реального часу до програм [31]. Веб-функції в режимі реального часу дозволяють із серверу миттєво надсилати вміст клієнтам. SignalR надає API для створення викликів віддалених процедур між

серверами (RPC). RPC викликають функції на клієнтах із коду .NET Core на стороні сервера. Існує кілька платформ, що підтримуються, кожна з яких має відповідний клієнтський пакет SDK. Через це виклик RPC застосовує різні мови програмування.

До основних функцій SignalR належать:

– автоматично обробляє керування підключеннями;

– одночасно надсилає повідомлення всім підключеним клієнтам;

– надсилає повідомлення певним клієнтам чи групам клієнтів (рис. 3.2).

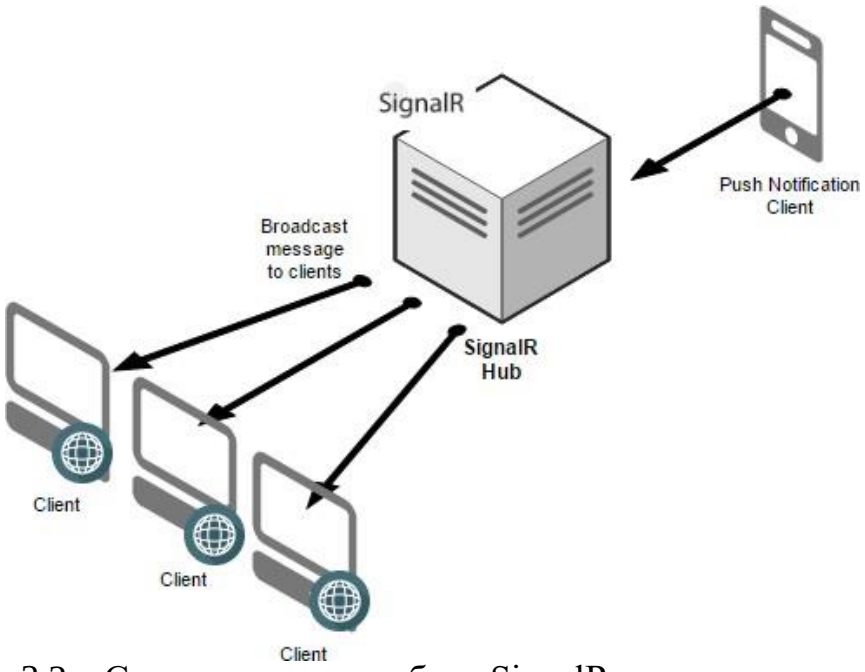


Рисунок 3.2 – Схема прикладу роботи SignalR

Entity Framework Core. Як і у будь-якому застосунку, даний фреймворк повинен мати механізм роботи із базою даних. В якості такого інструменту був вибраний Entity Framework Core.

Entity Framework (EF) Core – це полегшена, розширювана, крос-платформна версія популярної технології доступу до даних Entity Framework з відкритим вихідним кодом. EF Core також працює як об'єктно-реляційне зіставлення (ORM) [15, 16], який:

– дозволяє розробникам .NET працювати з базою даних за допомогою .NET об’єктів;

– усуває потребу в більшості коду доступу до даних, який зазвичай

потрібно писати.

EF Core використовує моделі для доступу до даних. Модель складається з класів сутностей та контексту, що представляє сеанс із базою даних. Саме через контекст відбуваються запити на читання та збереження даних (рис. 3.3).

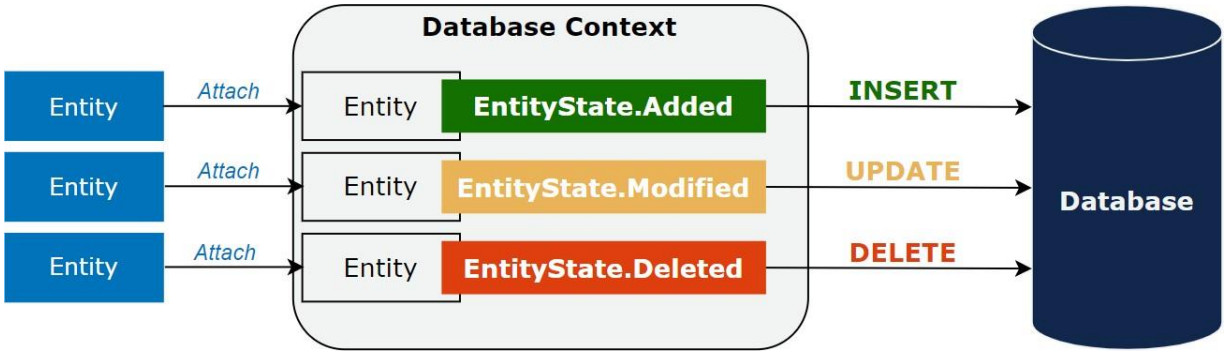


Рисунок 3.3 – Схема взаємодії із базою даних завдяки Entity Framework Core

У EF Core успішно абстрагується від більшої частини деталей програмування, але є деякі рекомендації, які застосовуються до будь-якої ORM, які допомагають уникнути поширених помилок у робочих додатках:

– базові знання сервера баз даних середнього чи просунутого рівня необхідні проектування, налаштування, профілювання та перенесення даних у високопродуктивних виробничих додатках. Наприклад, знання первинних та зовнішніх ключів, обмежень, індексів, нормалізації, операторів DML та DDL, типів даних, профілювання тощо;

– тест продуктивності. Звичайне використання деяких функцій погано масштабується. Наприклад, використання кількох колекцій, інтенсивне використання відкладеного завантаження (англ. «lazy loading»), умовні запити до неіндексованих стовпців, масові оновлення та вставки з використанням згенерованих значень, відсутність паралелізму, великі моделі та погана політика кешування тощо.

Шаблон проектування CQRS. CQRS інтерпретується як Розподіл відповідальності команд та запитів (англ. «*Command and Query Responsibility Segregation*») [13], шаблон, який поділяє операції читання та оновлення сховища

даних. Реалізація CQRS у додатку може максимізувати його продуктивність, масштабованість та безпеку. Гнучкість, створена переходом на CQRS, дозволяє системі краще розвиватися з часом і уникати виклику команд оновлення конфліктів злиття лише на рівні домену.

Контекст та суть шаблону. Традиційні архітектури використовували одну й ту саму модель даних для запитів та оновлень бази даних. Це підходить для основних операцій CRUD (створення, читання, оновлення, видалення). Однак у складніших додатках цей підхідможе стати громіздким. Наприклад, на стороні читання програма може робити безліч різних запитів, які повертають об'єкти передачі даних (DTO) різних форматів. Перегляд об'єктів може бути ускладненим. На боці запису моделі можуть реалізовувати складну перевірку та бізнес-логіку. В результаті можна отримати надмірно складні моделі.

Робочі навантаження читання та запису часто асиметричні, з дуже різними вимогами до продуктивності та масштабування.

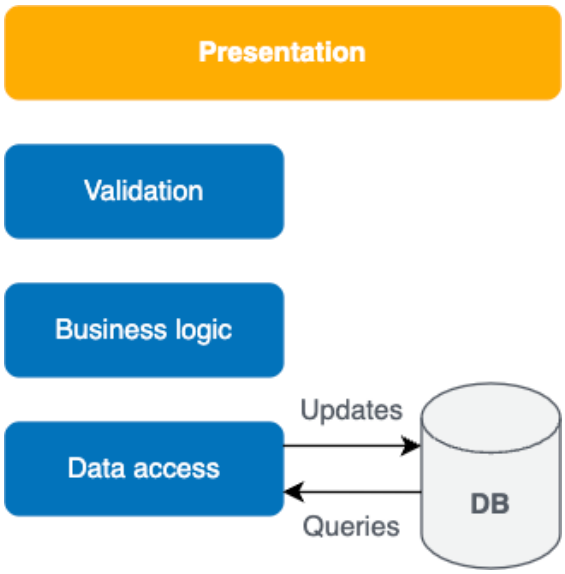


Рисунок 3.4 – Схема традиційної CRUD архітектури

Наявність окремих моделей запитів та оновлень спрощує проектування та реалізацію. Однак одним недоліком є те, що код CQRS не може бути автоматично згенерований із схеми бази даних із використанням механізмів інфраструктури, таких як інструменти O/RM.

MediatR. MediatR є бібліотекою для мови програмування C#, завдяки їй

можна інтегрувати шаблон CQRS у додаток [13]. На рівні коду даний шаблон реалізується двома взаємопов'язаними класами: один виступає в якості моделі запиту (реалізацією інтерфейсу IRequest<>, який вказує, який тип даних дає повертати даний запит) та обробника даного запиту (реалізації інтерфейсу IRequestHandler<>), який має реалізувати метод Handle, параметром якого є конкретний об'єкт запиту. Приклад використання MediatR для реалізації шаблону CQRS:

```
public class MediatrExampleRequest : IRequest<string>
{
    public int Property { get; set; }
}

public class MediatrExampleRequestHandler : IRequestHandler<MediatrExampleRequest, string>
{
    public Task<string> Handle(MediatrExampleRequest request, CancellationToken cancellationToken)
    {
        // Business logic
        throw new NotImplementedException();
    }
}
```

Після створення запиту та обробника, достатньо просто надіслати даний запит, використовуючи інтерфейс IMediator, попередньо додавши його до колекції сервісів механізму DI (Dependency Injection). Використання IMediator у механізмі DI:

```
public static IServiceCollection AddMediatR(this IServiceCollection services)
{
    services.AddMediatR(Assembly.GetExecutingAssembly());

    return services;
}
```

Приклад виклику запиту через механізм MediatR:

```
public class MediatrExampleService
{
    private readonly IMediator _mediator;
    public MediatrExampleService(IMediator mediator)
    {
        _mediator = mediator;
    }

    public async Task SomeMethod()
    {
        var request = new MediatrExampleRequest
        {
            Property = 1
        };

        var result = await _mediator.Send(request);
    }
}
```

Важливо вказати, що у кожного запиту має бути лише один обробник. Більш того, можна надсилати запити під час обробки цих самих запитів.

Приклад використання IMediator в обробниках запитів

```
public class MediatrExampleRequest1 : IRequest<int>
{
    public int MyProperty { get; set; }
}

public class MediatrExampleRequestHandler1 : IRequestHandler<MediatrExampleRequest1, int>
{
    public async Task<int> Handle(MediatrExampleRequest1 request, CancellationToken cancellationToken)
    {
        // Business logic
        return await Task.FromResult(1);
    }
}

public class MediatrExampleRequest2 : IRequest<int>
{
    public int MyProperty { get; set; }

}
public class MediatrExampleRequestHandler2 : IRequestHandler<MediatrExampleRequest2, int>
{
    private readonly IMediator _mediator;

    public MediatrExampleRequestHandler2(IMediator mediator)
    {
        _mediator = mediator;
    }

    public async Task<int> Handle(MediatrExampleRequest2 request, CancellationToken cancellationToken)
    {
        var request1 = new MediatrExampleRequest1
        {
            MyProperty = 1
        };

        var requestiResult = await _mediator.Send(request1, cancellationToken);
        return requestiResult + 1;
    }
}
```

PostgreSQL. Існує безліч систем управління реляційними базами даних (RDBMS) [28], з яких можна вибирати, якщо реляційна модель найкраще представляє дані. PostgreSQL – одна з найпопулярніших і найпродуктивніших у світі реляційних баз даних з відкритим кодом [29]. Одна з найбільш фундаментальних відмінностей між PostgreSQL та більшістю інших реляційних баз даних – це фундаментальна структура.

Більшість реляційних баз даних найкраще описати як РСУБД. СУБД – це програмне забезпечення, спеціально розроблене для роботи з реляційними базами даних, де дані зберігаються в табличних структурах з певними стовпцями та типами даних. Дані можна отримувати, змінювати та вилучати за допомогою методів, заснованих на реляційній алгебрі, зазвичай з

використанням мови структурованих запитів (SQL).

PostgreSQL є технічно системою управління об'єктно-реляційними базами даних (ORDBMS). Це означає, що вона має самі реляційні функції, як і РСУБД, але йдеякі об'єктно-орієнтовані функції.

Практично кажучи, це означає, що PostgreSQL дозволяє:

– визначати власні складні типи даних;

– перевантажувати функції для роботи з різними типами даних аргументів;

– визначати відносини успадкування між таблицями.

PostgreSQL відомий тим, що надає відмінні реалізації основних реляційних функцій, не обмежуючись традиційними середовищами СУБД. Жодна база даних не може задовольнити всі потреби, але PostgreSQL — відмінний варіант завдяки своїй універсальності для багатьох випадків використання.

3.2. Вибір постачальника хмарних даних

Як було зазначено у функціональних вимогах, даний сервіс має бути повністю опублікований із використанням хмарних технологій. Виходячи з цього, необхідно зробити аналіз найбільш популярних постачальників хмарних послуг, щоб вибрати найбільш надійного. У даному аналізі мають бути враховані такі показники як різноманіття сервісів, доступність пакету інструментів розробки (SDK) для вибраної мови програмування та ціна використання сервісів.

До переліку обов’язкових сервісів, які має надавати обраний провайдер, належать наступні:

– хмарне сховище об’єктів;

– сервіс баз даних;

– сервіс балансування навантажень;

– сервіс електронної пошти;
– безсерверні обчислення;
– сервіс реєстрації доменів.

Amazon Web Services (AWS) – це комерційна загальнодоступна хмара, що підтримується і розробляється Amazon з 2006 року. Абонентам надаються послуги як на основі інфраструктурної моделі (віртуальні сервери, ресурси зберігання), так і рівня платформи (хмарні бази даних, хмарне ПЗ, хмарні безсерверні обчислення, засоби розробки). AWS — найпопулярніша у світі хмарна платформа, що пропонує понад 200 повнофункціональних сервісів із центрів обробки даних по всьому світу. AWS використовують мільйони клієнтів, у тому числі найшвидші стартапи, найбільші підприємства та провідні державні установи [9].

Gartner Research позиціонує AWS у квадранті лідерів нового Магічного квадранту хмарної інфраструктури та платформних послуг (англ. CIPS) 2021 року. CIPS, у контексті цього магічного квадранта, визначаються як «стандартизовані високоавтоматизовані пропозиції, у яких ресурси інфраструктури (наприклад, обчислення, мережа та зберігання) доповнюються інтегрованими сервісами платформи» (рисунок 3.5).

Рисунок 3.5 – Квадрант хмарної інфраструктури та платформних послуг

Переваги AWS як постачальника хмарних послуг:

– максимальна функціональність. Від інфраструктурних технологій, таких як обчислення, зберігання та бази даних, до нових технологій, таких як машинне навчання та штучний інтелект, озера даних та аналітика, до Інтернету речей, AWS пропонує набагато більше послуг та можливостей, ніж будь-який інший постачальник хмарних послуг;

– найбільша спільнота клієнтів та партнерів AWS має найбільшу та динамічну спільноту з мільйонами активних клієнтів та десятками тисяч партнерів по всьому світу. Клієнти майже всіх галузей і розмірів, включаючи стартапи, підприємства та організації державного сектора, використовують AWS для різних сценаріїв використання. У партнерську мережу AWS входять тисячі системних інтеграторів, що спеціалізуються на сервісах AWS, та десятки тисяч незалежних постачальників програмного забезпечення, які адаптують свої технології до роботи на AWS;

— AWS розроблена як найгнучкіше і найбезпечніше середовище хмарних обчислень, доступне на сьогоднішній день. Наша основна інфраструктура призначена для задоволення вимог безпеки військових, міжнародних банків та інших важливих організацій. Це підтримується широким набором інструментів хмарної безпеки з більш ніж 300 службами та можливостями безпеки, відповідності та управління. AWS підтримує 98 стандартів безпеки та сертифікатів відповідності, і всі 117 сервісів AWS, де зберігаються дані клієнтів, пропонують можливість шифрування цих даних;

– найбільш оперативний досвід. AWS має неперевершений послужний список, надійність, безпеку та продуктивність, на які можна покластися при роботі з найбільш важливими програмами. Вже понад 16 років AWS надає хмарні послуги для різних варіантів використання мільйонам клієнтів по всьому світу. З усіх хмарних провайдерів AWS має найбільший досвід у масштабуванні;

– глобальна мережа регіонів. AWS має найбільш розгалужену глобальну хмарну інфраструктуру. Модель регіону та зони доступності AWS

визнана Gartner рекомендованим підходом для запуску корпоративних програм, що потребують високої доступності.

Окрім цього, AWS надає різноманітний набір пакетів інструментів розробки SDK, в тому числі і для обраної мови програмування C#. AWS SDK для .NET спрощує використання служб AWS, надаючи набір узгоджених і звичних для розробників .NET бібліотек. Усі AWS SDK забезпечують підтримку розгляду життєвого циклу API, такого як керування обліковими даними, повторні спроби, маршалінг даних і серіалізація. Зовнішній вигляд UI консолі управління AWS зображено на рисунку 3.6.

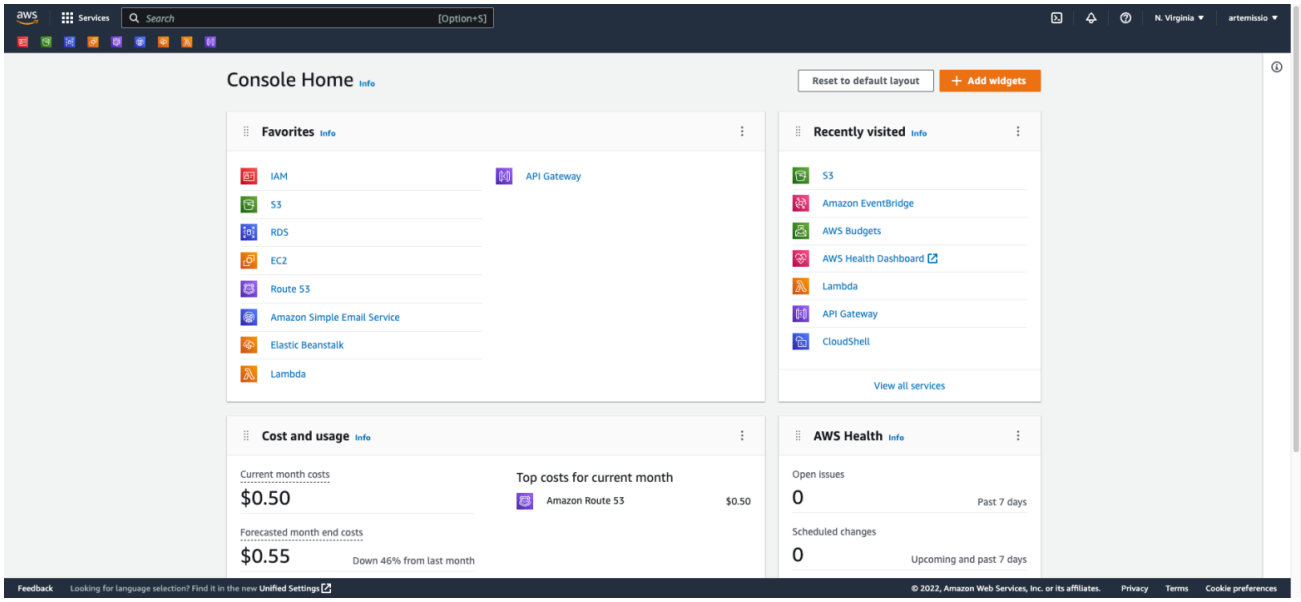


Рисунок 3.6 – UI консолі AWS Management Console

Ще однією особливістю AWS є наявність безкоштовного пробного періоду впродовж року з моменту реєстрації облікового запису. Доступні три різні типи безкоштовних пропозицій залежно від використовуваного продукту.

Усі необхідні для розробки фреймворку хмарні сервіси присутні у даного провайдера.

Google Cloud Platform. Google Cloud Platform (GCP) — це набір хмарних сервісів, які пропонують Google, які працюють на тій самій інфраструктурі, яку Google використовує для споживчих продуктів, таких як Google Search та YouTube. Крім інструментів управління, він також пропонує ряд модульних

хмарних сервісів, таких як хмарні обчислення, зберігання даних, аналіз даних та машинне навчання [19].

Google Cloud складається з набору фізичних активів, таких як комп'ютери та жорсткі диски, та віртуальних ресурсів, таких як віртуальні машини (ВМ), розташовані в центрах обробки даних Google по всьому світу. Кожен центр обробки даних знаходиться у межах регіону. Доступні регіони в Азії, Австралії, Європі, Північній Америці та Південній Америці. Кожен регіон є сукупністю зон, ізольованиходне від одного всередині регіону.

Такий розподіл ресурсів має кілька переваг, таких як відмово-стійкість та зменшення затримки за рахунок розміщення ресурсів ближче до клієнтів. Ця квота також запроваджує деякі правила спільного використання ресурсів.

Google Cloud надає клієнтські бібліотеки, які спрощують створення ресурсів та керування ними. Клієнтські бібліотеки Google Cloud надають API переважно для двохцілей:

– API-інтерфейси програм надають доступ до служб. API програми оптимізовано для мов, що підтримуються, таких як Node.js, Python і C#. Бібліотека заснована на метафорі служби, дозволяючи користувачам працювати зі службами природніше і писати менше стандартного коду. Бібліотека також надає помічників для автентифікації та авторизації;

– API керування надає функціональні можливості для керування ресурсами. Наприклад, якщо потрібно створити власні інструменти автоматизації, є можливість використовувати API управління.

Зовнішній вигляд UI консолі GCP Cloud Console зображено на рисунку 3.7.

Ще одна особливість GCP полягає в тому, що з реєстрації облікового запису можна скористатися 1-річним безкоштовним пробним періодом і кредитом у розмірі 300 доларів США для використання більш конкретних послуг.

Серед переліку необхідних хмарних сервісів у даного провайдера відсутні сервіс електронної пошти та сервіс реєстрації і обслуговування інтернет

доменів.

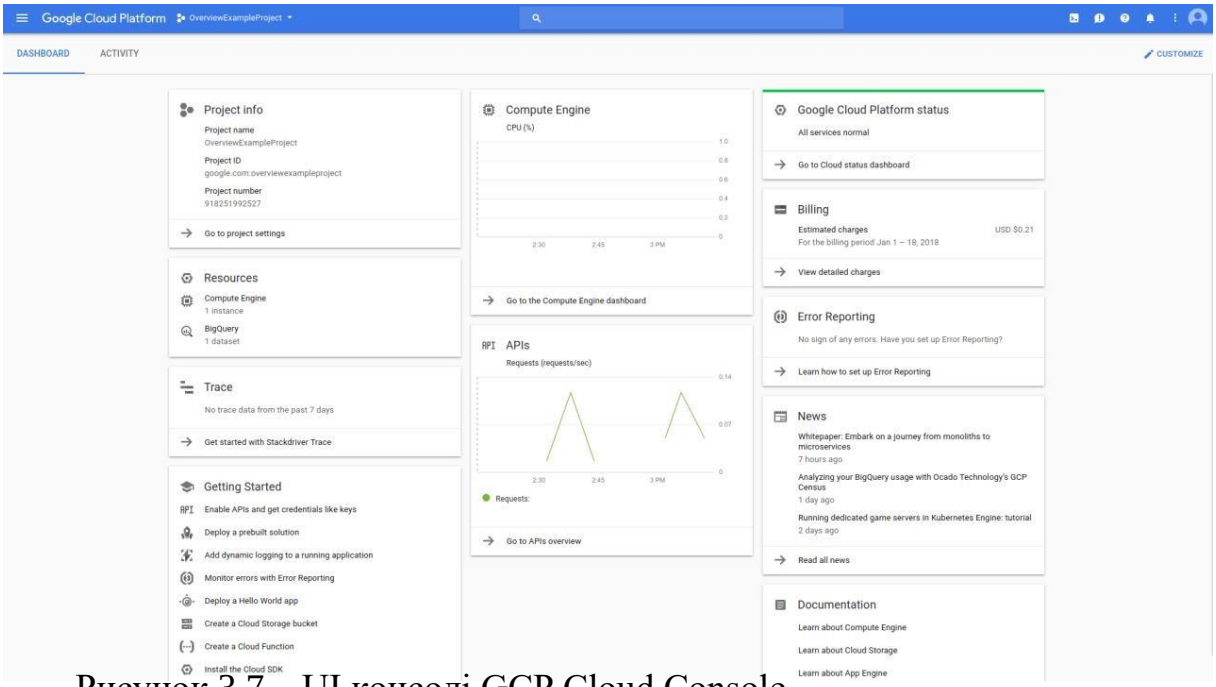


Рисунок 3.7 – UI консолі GCP Cloud Console

Microsoft Azure – це хмарна платформа Microsoft. Надає можливість розробляти, запускати програми та зберігати дані на серверах, розташованих у розподілених центрах обробки даних [20].

Microsoft Azure реалізує хмарні моделі «Платформа як послуга» (PaaS) та «Інфраструктура як послуга» (IaaS). У рамках моделі «Програмне забезпечення як послуга» (SaaS) можна використовувати як сторонні служби, так і служби Microsoft. Можливості платформи Microsoft Azure забезпечують глобальну мережу розподілених центрів обробки даних Microsoft.

На додаток до базових функцій операційної системи Microsoft Azure включає виділення ресурсів за запитом для масштабування, автоматичну синхронну реплікацію даних для більшої стійкості до збоїв і збої інфраструктури для постійної доступності. Є додаткові функції, такі як обробка.

Ще одна особливість Microsoft Azure полягає в тому, що з моменту реєстрації облікового запису користувач має право на один рік безкоштовного пробного періодута кредит у розмірі 200 доларів США для використання більш конкретних послуг.

Серед переліку необхідних хмарних сервісів у даного провайдера відсутні сервіс електронної пошти та сервіс реєстрації і обслуговування інтернет доменів.

Зовнішній вигляд UI консолі Microsoft Azure Portal зображено на рисунку 3.8.

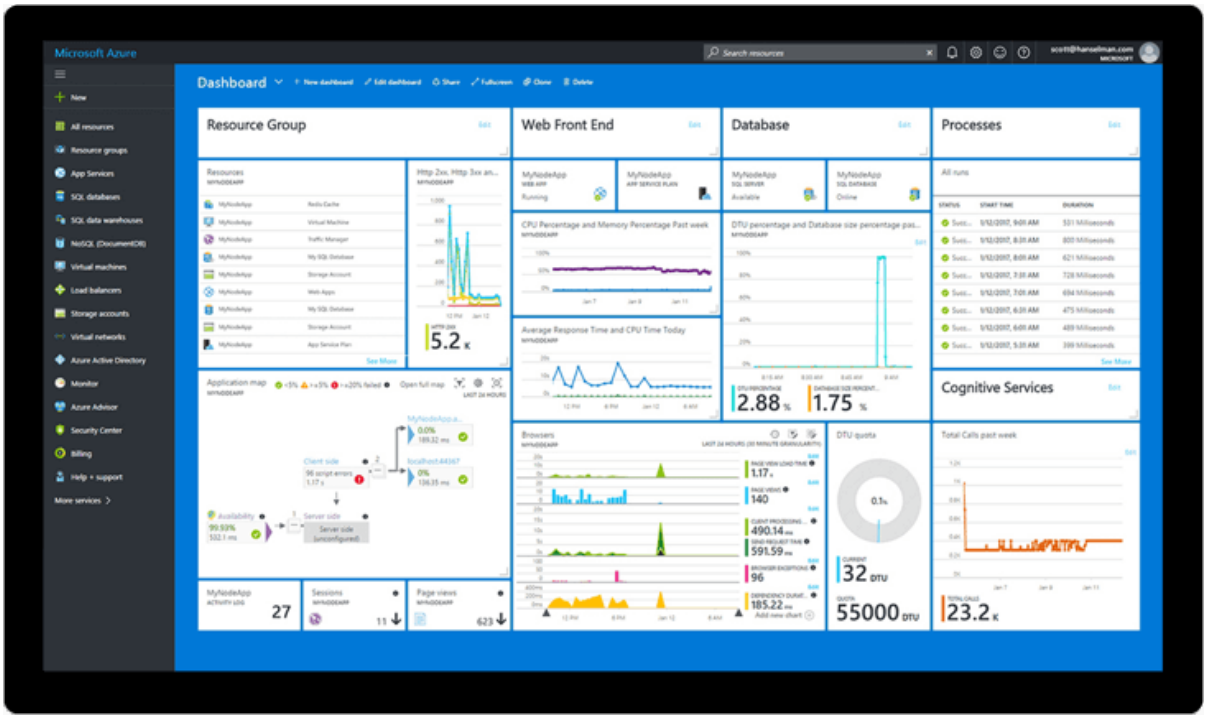


Рисунок 3.8 – UI консолі Microsoft Azure Portal

Порівняння цін на необхідні сервіси. Одним із ключових параметрів для аналізу хмарних постачальників є ціна за оплату використаних сервісів.

Таблиця 3.1 – Структура цін хмарних сервісів

Сервіс	AWS	GCP	Azure
Хмарне сховище об’єктів, $ за 1 Gb/місяць	0.025	0.02	0.01
Бази даних, $ за 1 год	0.02	0.05	0.057
Віртуальні сервери, $ за 1 год	0.012	0.03	0.012
Балансування навантажень, $ за 1 год	0.03	0.03	0.025
Електронна пошта, $ за 1000 листів	0.10	–	–
Безсерверні обчислення, $ за мільйон запитів	0.20	0.32	0.20
Домени, $ за 1 місяць	0.50	–	–

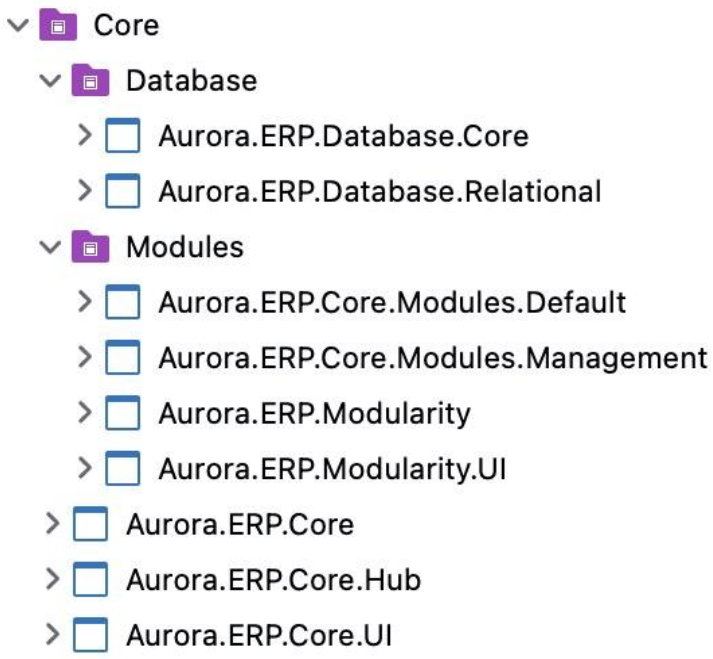
На таблиці 3.1 наведені ціни за користування необхідними сервісами того або іншого хмарного провайдера.

3.3. Розробка складових програмної системи

На основі визначених у розділі 2 функціональних та технічних вимог до системи, а також огляді технологій та шаблонів і принципів проектування у розділі 3, розроблено систему взаємопов'язаних додатків, кожний із яких відповідає за певну задачу.

Для зручності розробки рішення було поділене на кілька частин:

– ядро фреймворку. До даної частини належать проекти із головними моделями, сервісами, бізнеслогікою, елементами UI, програмна реалізація модулів, а також загальна логіка роботи із БД (на момент розробки була реалізована логіка для реляційних БД, в перспективі можлива реалізація також для NoSQL, графових БД тощо). Проект ядра фреймворку наступний:



На ньому можна побачити:

а) Aurora.ERP.Core – головний проект бібліотеки класів із головними моделямита сервісами фреймворку;

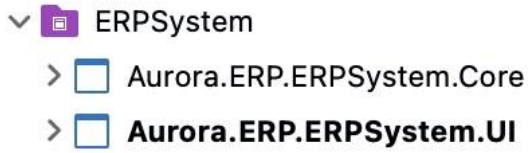
б) Aurora.ERP.Core.UI – бібліотека UI елементів;

в) Aurora.ERP.Core.Hub – «огортка» SignalR із власним додатковим функціоналом;

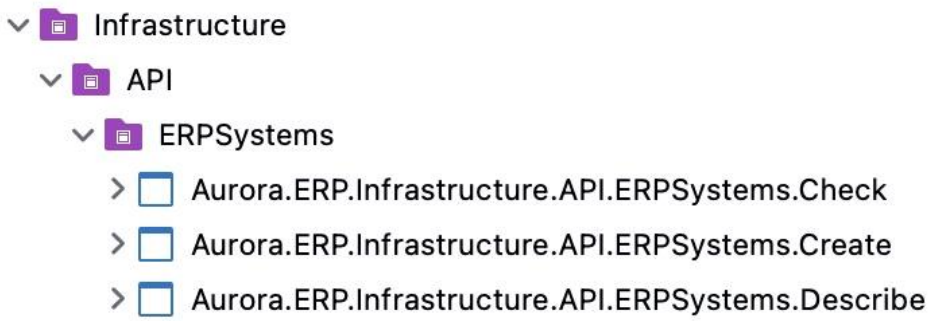
г) Aurora.ERP.Database.Core та Aurora.ERP.Database.Relational – загальна логіка взаємодії із БД в вигляді абстракцій та реалізація даної логіки для реляційних БД відповідно;

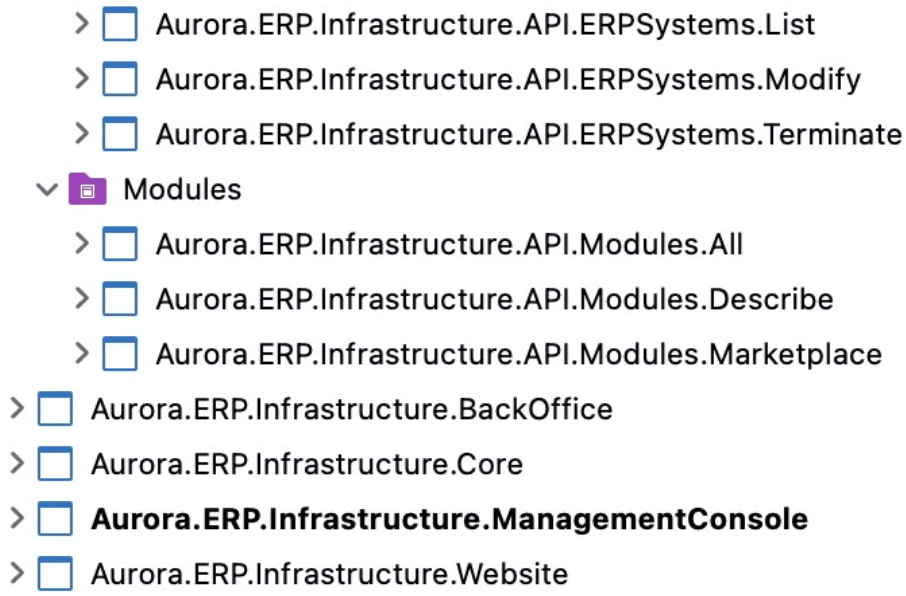
г) Aurora.ERP.Core.Modularity, Aurora.ERP.Core.Modules.Default та Aurora.ERP.Core.Modules.Management – логіка та основні моделі для впровадження модульності та 2 модулі: за замовчуванням (даний модуль буде встановлений у ERP системі за замовчуванням без необхідності користувачу вибирати його для встановлення) та модуль Управління (тестовий модуль) відповідно.

– ERP система. До цієї частини програмного рішення належать 2 проєкти: Aurora.ERP.ERPSystem.Core (головна бізнес логіка, моделі та сервіси, необхідні для коректної роботи ERP системи) та Aurora.ERP.ERPSystem.UI (власне веб-сервіс ERP системи, його UI та шлюз для обробки запитів). Дану частину можна побачити:

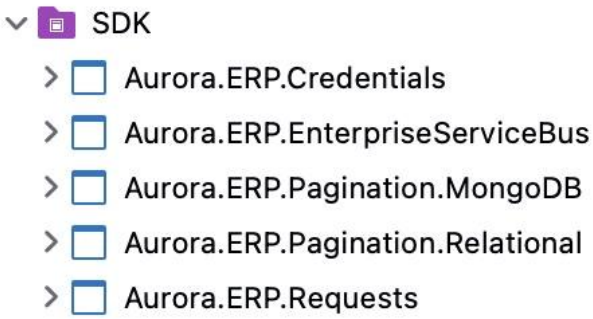


– інфраструктура фреймворку. До даної частини програмного рішення належать додатки для роботи із даним фреймворком для користувачів та адміністраторів фреймворку. В цій частині знаходяться проекти головного веб-сайту, консолі взаємодії, API фреймворку та бек-офісу. Список додатків та проектів цієї частини фреймворку:





– SDK фреймворку. Однією із умов розробки було створення пакету розробки для даного фреймворку. Зазвичай IT-компанії створюють такі набори для декількох мов програмування. На даному етапі розробки такий SDK створений лише для мови програмування C#. Список проектів даної частини SDK фреймворку:



3.4. Інтеграція статичних і динамічних додатків та загальна структурна схема системи

Фреймворк, що розробляється, являє собою систему взаємопов’язаних додатків, кожний із яких відповідає за поставлену задачу, водночас працює у спільному зв’язку із іншими додатками. В загальній структурній схемі фреймворку є 4-и статичні додатки (додатки, що будуть опубліковані власноруч адміністраторами та власниками фреймворку):

– головний веб-сайт (взаємодіє повністю із усією інфраструктурною частиноюфреймворку та хмарними ресурсами);

– API (взаємодіє повністю із усією інфраструктурною частиною фреймворку та хмарними ресурсами);

– бек-офіс (взаємодіє повністю із усією інфраструктурною частиною фреймворку та хмарними ресурсами, окрім можливості управління ERP системи);

– консоль управління (взаємодіє лише із створеними користувачами ERP системами).

До динамічних додатків належить лише власне веб-сервіс ERP системи – даний застосунок буде запускатися користувачами фреймворку під час запуску ERP систем. Сам застосунок (а точніше його збірка та виконавчий файл) зберігатиметься в закритому доступі у хмарі, завдяки чому жоден користувач у світі не зможе його отримати та дізнатися кодову базу додатку. Доступ до файлу матимуть лише адміністратори фреймворку.

Програмні моделі фреймворку. Для розробки даного фреймворку використовуються принципи ООП, а також для коректної роботи і зручної розробки був використаний набір моделей для абстракції реальних речей в реальному житті.

Моделі запитів та відповідей. Оскільки даний фреймворк, які і безліч інших систем, оснований на системі запитів-відповідей, для полегшення розробки були створені абстракції для запитів та відповідей:

```
/// <summary>
/// Base model of Aurora ERP request
/// </summary>
public abstract class AuroraERPRequest : Serializable
{
    /// <summary>
    /// Request identifier
    /// </summary>
    [JsonInclude]
    public Guid RequestID { get; set; }

    /// <summary>
    /// Request data object
    /// </summary>
    public object Data { get; set; }
```

```
        /// <summary>
        /// Initializes new instance of <see cref="AuroraERPRequest"/>
        /// </summary>
        [JsonConstructor]
        public AuroraERPRequest()
        {
            RequestID = Guid.NewGuid();
            Data = new();
        }
    }
```

Дана модель складається із наступних властивостей:

– `RequestID` – унікальний ідентифікатор запиту. Дана властивість призначена для можливості відслідковування запитів та надає можливість реалізації своєрідногожурналу запитів;

– `Data` – дана властивість надає можливість передавати будь-який об’єкт у тілозапиту.

Модель відповідей на запити наступна:

```
/// <summary>
/// Base model of Aurora ERP response
/// </summary>
public abstract class AuroraERPResponse : Serializable
{
    /// <summary>
    /// Response identifier
    /// </summary>
    [JsonInclude]
    public Guid ResponseID { get; set; }

    /// <summary>
    /// Request identifier
    /// </summary>
    public Guid RequestID { get; set; }

    /// <summary>
    /// Response status
    /// </summary>
    [ValueConverter(typeof(ResponseStatusValueConverter))]
    [JsonConverter(typeof(ResponseStatusJsonConverter))]
    public ResponseStatus Status { get; set; }

    /// <summary>
    /// Response message
    /// </summary>
    public string Message { get; set; }

    /// <summary>
    /// Response data
    /// </summary>
    public object? Data { get; set; }

    /// <summary>
    /// Initializes new instance of <see cref="AuroraERPResponse"/>
    /// </summary>
    public AuroraERPResponse()
    {
        ResponseID = Guid.NewGuid();
        RequestID = Guid.Empty;
        Message = string.Empty;
        Status = ResponseStatus.Error;
    }
}
```

Ця модель складається із таких властивостей:

– `ResponseID` – унікальний ідентифікатор відповіді. Дана властивість призначена для можливості реалізації своєрідного журналу відповідей;

– `RequestID` – ідентифікатор запиту, на який була зроблена дана відповідь;

– `Status` – статус відповіді. У відповіді може бути 3 можливі статуси:

1) `Success` – відповідь матиме даний статус у випадку, якщо операція при запиті була успішна;

2) `Error` – відповідь матиме даний статус у випадку, коли під час обробки запиту трапилася помилка;

3) `Cancelled` – даний статус буде повертатися при скасуванні запиту.

– `Message` – повідомлення про помилку під час обробки запиту;

– `Data` – об’єкт із даними відповіді.

Як побачимо з наведеного коду, вказані моделі є абстрактними класами, і одними із успадкувань цих класів є окремі моделі запиту та відповіді у ERP системі. Ці моделі описані класами `ApplicationRequest:`

```
/// <summary>
/// ERP system application request
/// </summary>
[JsonConverter(typeof(ApplicationRequestJsonConverter))]
public class ApplicationRequest : AuroraERPRequest
{
    /// <summary>
    /// Application request
    /// </summary>
    [ValueConverter(typeof(ApplicationOperationValueConverter))]
    [JsonConverter(typeof(ApplicationOperationJsonConverter))]
    [AuroraERPProperty(Required = true, ErrorMessage = "Operation is required")]
    public ApplicationOperation? Operation { get; set; }

    /// <summary>
    /// Initializes new instance of <see cref="ApplicationRequest"/>
    /// </summary>
    public ApplicationRequest()
    {
        Operation = null;
    }
}
```

та `ApplicationResponse` відповідно:

```
/// <summary>
/// ERP system application response
/// </summary>
[JsonConverter(typeof(ApplicationResponseJsonConverter))]
public class ApplicationResponse : AuroraERPResponse
{
    /// <summary>
    /// Application operation
    /// </summary>
    [ValueConverter(typeof(ApplicationOperationValueConverter))]
    [JsonConverter(typeof(ApplicationOperationJsonConverter))]
    public ApplicationOperation? Operation { get; set; }

    /// <summary>
    /// Initializes new instance of <see cref="ApplicationResponse"/>
    /// </summary>
    public ApplicationResponse()
    {
        Operation = null;
    }
}
```

Обидві моделі успадковуються від `AuroraERPRequest` та `AuroraERPResponse` відповідно та мають додаткову властивість – `Operation`, яка вказує, яка операція булазазначена у запиті. `ApplicationOperation` може набувати одного із наступних значень:

– `Database operation` – вказує на запит до БД. У випадку даної операції у полі Data вказується, над якою сутністю та якого модулю робиться операція;

– `Module operation` – вказує на конкретну операцію конкретного модуля. Кожний модуль може мати набір моделей операцій та логіку опрацювання тієї чи іншої операції;

– `Add module` – вказує на запит на інсталяцію нового модуля в ERP системі, а також на запит видалення модуля із ERP системи.

В залежності від значення `Operation` у полі `Data` запиту мають бути вказані конкретні дані, необхідні для обробки тієї або іншої операції.

3.5. Моделі фільтрації даних та розбиття на сторінки

Оскільки даний фреймворк розробляється для створення веб-сервісів ERP систем для комерційного бізнесу, бази даних такого сегменту можуть містити терабайти даних. Звісно, при вибірці даних із таблиць такого масштабу може повертатися велика кількість інформації. Спроба серіалізувати цю інформацію та передати мережею може викликати проблеми через розмір цих даних. Саме тому було прийнято за стандарт необхідність реалізації механізму фільтрації вибірки та розбиття її на сторінки (так званий пейджинг, англ: «pagination»). Саме завдяки такому механізму користувачу по всьому світі при відвідуванні, наприклад, інтернет магазину не бачать відразу всі товари (яких може бути тисячі і мільйони), а завантажують їх частково, наприклад, по 50. Це набагато зменшує пропускну здатність та навантаження на мережу і систему в цілому.

Саме в рамках цього стандарту аналогічний механізм був реалізований у даному фреймворку. Реалізований він у вигляді трьох моделей:

- `Expression`;
- `Filter`;
- `PagedResult`.

Наступною зображена модель логічного вираження для фільтрації вибірки даних за параметрами. Ця модель має дві властивості: `Text` (текст вибірки) та `Values` (значення параметрів, які вказуються при вибірці).

Формат тексту вираження аналогічний до звичайного SQL запиту для вибірки, в якому параметри беруться із колекції Values за індексом, вказаного після символу ‘@’. Наприклад, нехай існує таблиця замовлень і необхідного зробити наступну вибірку: дістати всі завершені замовлення (статус `Completed`) в Києві, або Львові станом на 1 грудня 2023 року. Тоді текст вираження буде: «((`Status` == @0) and (`City.Contains`(@1) or `City.Contains`(@2) and (`Date` == @3))», тоді як значення будуть: ‘`Completed`’, ‘`Kyiv`’, ‘`Lviv`’ та ‘2022.12.01’. Такий формат є зрозумілим як для розробників, так і для звичайних користувачів.

```
/// <summary>
/// Filtration expression model
/// </summary>
public struct Expression
{
    /// <summary>
    /// Expression text
    /// </summary>
    public string Text { get; set; }

    /// <summary>
    /// Expression values
    /// </summary>
    public IEnumerable<object> Values { get; set; }

    /// <summary>
    /// Initializes new instance of <see cref="Expression"/>
    /// </summary>
    public Expression()
    {
        Text = string.Empty;
        Values = Array.Empty<object>();
    }
}
```

Наступною покажемо модель фільтру. Саме ця модель буде вказана при запиті і на її основі буде виконана логіка фільтрації даних. Вона складається із наступних властивостей:

– `PageIndex` – вказує номер сторінки під час розбиття даних на сторінки.

Мінімальне значення – 1;

– `PageSize` – вказує максимальну кількість записів для витягнення на вказанійсторінці. За замовчуванням – 25;

– `OrderBy` – вказує назву колонки в таблиці, по якій треба робити сортування;

– `OrderDirection` – вказує тип сортування: Ascending (від меншого до більшого) або Descending (від більшого до меншого). За замовчуванням – Descending;

– `Expression` – модель логічного вираження;

– `FilterPageLimit` – вказує чи потрібно розбивати вибірку на сторінки (`Limited`)або ні (`Unlimited`). За замовчуванням – `Limited`.

```
/// <summary>
/// Model for filtering collections (using <c>Expression</c> property),sorting, ordering and paging
/// </summary>
public class Filter : Serializable
{
    /// <summary>
    /// Index of page to display
    /// </summary>
    public int PageIndex { get; set; }

    /// <summary>
    /// Amount of elements to display per one page
    /// </summary>
    public int PageSize { get; set; }

    /// <summary>
    /// Name of property to order by
    /// </summary>
    public string OrderBy { get; set; }

    /// <summary>
    /// Ordering direction - <c>Ascending</c> or <c>Descending</c>.
    /// Default value - <c>Descending</c>
    /// </summary>
    public string OrderDirection { get; set; }

    /// <summary>
    /// Filtration expression model
    /// </summary>
    [JsonConverter(typeof(ExpressionJsonConverter))]
    public Expression Expression { get; set; }

    /// <summary>
    /// Defines whether collection should filtered (<c>Limited</c>) or not (<c>Unlimited</c>).
    /// Default value - <c>Limited</c>
    /// </summary>
    public FilterPageLimit PageLimit { get; set; }

    [JsonConstructor]
    public Filter(FilterPageLimit pageLimit = FilterPageLimit.Limited)
    {
        PageLimit = pageLimit;

        PageIndex = GlobalConstants.DefaultPageIndex;
        PageSize = GlobalConstants.DefaultPageSize;
        OrderDirection = GlobalConstants.Descending;
        OrderBy = string.Empty;
        Expression = new();
    }
}
```

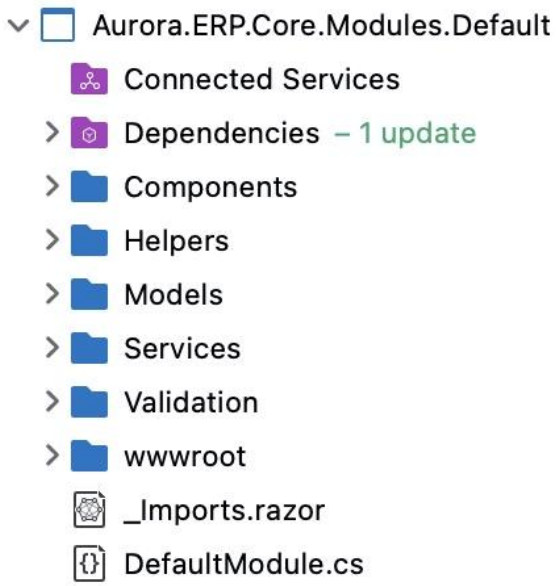
Останньою із трьох моделей для механізму фільтрації вибірки є власне результат даної фільтрації, який і буде повертатися у тіло відповіді на запит.

РОЗДІЛ 4.
РОЗРОБКА І ПРОГРАМНА РЕАЛІЗАЦІЯ ФРЕЙМВОРКУ СИСТЕМИ УПРАВЛІННЯ ІНФОРМАЦІЙНИМИ РЕСУРСАМИ

4.1. Програмна реалізація головних модулів системи фреймворку

Як вже зазначалося, концепція модульності дозволить розділити бізнес-логіку нашої системи на окремі компоненти. З точки зору розробки, дана концепція також стане в нагоді, оскільки в такому випадку розробка, тестування та впровадження кожного модуля відбуватиметься окремо. Завдяки цьому, кожен модуль потребуватиме залучення окремих команд розробників. Це дасть змогу виконувати розробку багатьох модулів одночасно і пришвидшить впровадження продукту на ринку та забезпечить різноманіття вибору необхідного функціоналу для користувачів фреймворку [29].

Проект модуля являтиме собою бібліотеку класів `Razor` – звичайна бібліотека класів, з єдиною відмінністю в тому, що надає можливість створювати Razor компоненти – основними `UI` елементами для `Blazor` додатків. Вигляд бібліотеки модуля такого проекту наступний.



Як можна побачити, структура проекту складається із наступних папок:

– `Components`: саме у цій папці зберігатимуться вищезгадані Razor

компоненти таблиці даних, форми для заповнення, ті або інші візуальні елементи, через які користувачі будуть взаємодіяти із ERP системою;

– `Models`: сутності даного модуля. Саме ці моделі будуть використовуватись для бізнес-логіки даного модуля;

– `Services`: абстракції та реалізації сервісів даного модуля;

– `Validation`: папка із логікою валідації тих або інших сутності при їхньому створенні;

– `wwwroot`: папка із статичним контентом, таким як файли стилів, зображення, веб-скрипти тощо.

Кожний модуль модель із трьома властивостями:

– `Name`: назва модуля, наприклад «`Project management`» або «`Logistic`», може бути локалізована для того чи іншого ринку;

– `Prefix`: префікс модуля, наприклад «`project-management`» або «`logistic`»;

– `Type`: тип модуля, назва його класу, це `DefaultModule`:

```
[PluralTitle("Modules")]
public class Module : SerializableEntity
{
    [Filter(nameof(GlobalConstants.Search))]
    [Display(Name = "Name")]
    public string Name { get; set; }

    [Filter(nameof(GlobalConstants.Search))]
    [Display(Name = "Prefix")]
    public string Prefix { get; set; }

    [Filter(nameof(GlobalConstants.Search))]
    [Display(Name = "Type")]
    public string Type { get; set; }

    public Module() : this(string.Empty, string.Empty, string.Empty) { }

    [JsonConstructor]
    protected Module(string name, string prefix, string type)
    {
        Name = name;
        Prefix = prefix;
        Type = type;
    }
}
```

4.2. Реалізація ERP безпеки через механізм HMAC SHA256

Однією із основних технічних вимог було створення механізму забезпечення конфіденційності та надійності обробки запитів. Для даного механізму був вибраний алгоритм HMAC SHA256.

HMAC (*Hash-based Message Authentication Code*) – це механізм перевірки цілісності інформації, що передається або зберігається в ненадійних середовищах [29]. Такі методи є невід'ємною та необхідною частиною світу відкритих обчислень та комунікацій. Механізми, які забезпечують такі перевірки цілісності на основі закритого ключа, зазвичай називають кодами автентифікації повідомлень (MAC). MAC зазвичай використовуються між двома сторонами, які спільно використовують секретний ключ для перевірки автентичності інформації, що надсилається між цими сторонами. Цей стандарт визначає MAC. Механізм, що використовує криптографічну хеш-функцію у поєднанні із закритим ключем, називається HMAC. Перевагами HMAC є:

– можливість використання хеш-функцій, які вже є в програмному продукті;

– відсутність необхідності внесення змін до реалізації існуючих хеш-функцій (внесення змін може призвести до погіршення продуктивності та погіршення криптостійкості);

– можливість заміни хеш-функції у разі появи більш безпечної або швидшої хеш-функції.

На рисунку 4.1 зображена схема алгоритму підпису запиту. Загалом він складається із чотирьох кроків, кожний із яких має більш детальну і складну логіку:

1) створити канонічний запит;

2) використати канонічний запит і додаткові метадані, щоб створити рядок для підпису;

3) отримайте ключ підпису зі свого секретного ключа доступу. Потім використати ключ підпису та рядок із кроку №3, щоб створити підпис;

4) додати отриманий підпис до HTTP-запиту в вигляді заголовку.

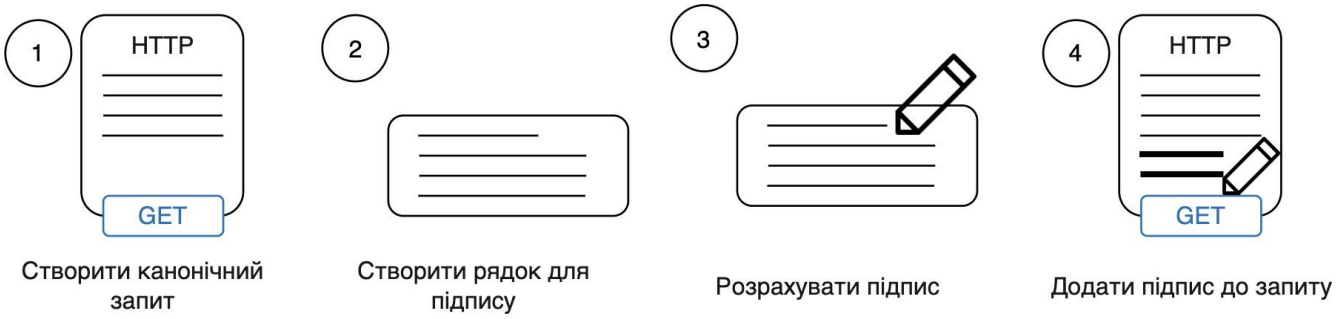


Рисунок 4.1 – Загальні етапи алгоритму верифікації підпису запиту

Наведемо приклад створення канонічного запиту. Канонічний запис запиту – запис, що формується за формулою 4.1 [27]:

CanonicalRequest = HTTPRequestMethod + '\n' + CanonicalURI + '\n' + CanonicalQueryString + '\n' + CanonicalHeaders + '\n' + SignedHeaders + '\n' + HexEncode(Hash(RequestPayload)) (4.1)

У цій формулі `Hash` представляє функцію, яка створює дайджест повідомлення, зазвичай SHA-256. `HexEncode` представляє функцію, яка зміст у hex форматі дайджесту символами нижнього регістру. Кожен вхідний байт повинен бути представлений рівно двома шістнадцятковими символами.

Щоб створити канонічний запит, потрібно об’єднати наступні компоненти з кожного кроку в єдиний рядок:

1) починається з методу запиту `HTTP` (`GET`, `PUT`, `POST` тощо) та символ нового рядку ‘`\n`’;

2) додати канонічний шлях запиту з подальшим символом ‘`\n`’. Канонічний шлях – це закодована версія компонента абсолютного шляху (від хоста HTTP до символу початку параметрів "?"). Якщо абсолютний шлях порожній, потрібно використати символ /;

3) додати канонічний рядок запиту з подальшим символом ‘`\n`’. Якщо запит не містить рядка запиту, використати порожній рядок. Для побудови канонічного рядка запиту, потрібно виконати наступні кроки:

а) відсортувати імена параметрів за у порядку зростання. Закодувати у форматі URI кожне ім'я та значення параметрів. Створити рядок канонічного

запиту, починаючи з першого імені параметра у списку відсортованих;

б) для кожного параметра додати URI-кодовану назву параметра з подальшим символом = та подальшим URI-кодованим значенням параметра. Використати порожній рядок для параметрів, які не мають значення;

в) додати символ & після кожного значення параметра, за винятком останнього значення у списку.

4) додати канонічні заголовки, за якими слідує '\n'. Канонічні заголовки складаються зі списку всіх заголовків HTTP, які будуть включені із підписаним запитом. Щоб створити список канонічних заголовків, потрібно конвертувати всі назви заголовків у нижній регістр.

4.3. Загальна структура взаємодії хмарних сервісів

У прийнятій нами загальній структурі взаємодії хмарних сервісів своєрідною вхідною точкою є сервіс Route53, оскільки саме в ньому створений глобальний інтернет домен, на який будуть надходити запити. Далі, в залежності від шляху, на який був відправлений запит, він перенаправляється на один із декількох інфраструктурних додатків або до ERP системи:

– erp-aurora.com – запит до головного веб-сайту. Цей додаток опублікований за допомогою сервісу Elastic Beanstalk. Він взаємодіє із S3 (для зчитування шаблонів електронних листів), RDS (сервісом БД) та SES (відправка електронних листів);

– backoffice.erp-aurora.com – запит до бек-офісу фреймворку. Цей додаток, аналогічно до головного веб-сайту, опублікований за допомогою сервісу Elastic Beanstalk. Він також взаємодіє із S3 (для завантажування файлів модулів та зчитування статистики), RDS (сервісом БД) та SES (для моніторингу статистики користування даним сервісом);

– api.erp-aurora.com – запит до API фреймворку. API опублікована за допомогою сервісу API Gateway, який перенаправляє запити до потрібної

Лямбда- функції, які в свою чергу також взаємодіють із RDS, S3, SES;

– console.erp-aurora.com – запит до консолі управління. Цей додаток також опублікований у сервісі Elastic Beanstalk, проте не взаємодіє із БД фреймворку та іншими інфраструктурними сервісами. Він створений в якості веб-інтерфейсу для взаємодії із створеними ERP системами.

Event Bridge регулюється самим AWS, незалежно від інших сервісів, та взаємодіє із однією єдиною лямбда-функцією, яка буде формувати фінансову звітність та статистику за минулий місяць, і зберігатиме її у S3.

Самі ERP системи з точки зору хмарних сервісів являють собою логічне об'єднання трьох сервісів – EC2 (віртуальний сервер (або декілька), на якому працює веб-сервіс), Elastic Load Balancing (для розподілення навантаження між серверами) та RDS (БД ERP системи).

4.4. Розробка головного веб-сайту

Згідно із технічними вимогами головний веб-сайт повинен слугувати відразу як ознайомчий сайт, на якому користувач може ознайомитися з усіма послугами даного фреймворку (модулями (їхніми описами та детальною інформацією), веб-сервісом ERP системи), так і функціональний веб-застосунок.

При переході на потрібну сторінку, користувач може знайти необхідну інформацію про модулі (рисунок 4.2), власну сторінку (рисунок 4.3).

На вкладці ERP Systems користувач знайде всі свої ERP системи та може створити нові, використовуючи надзвичайно прості налаштування – введення назви системи, вибрати необхідні модулі та вказати налаштування БД (рисунок 4.4-4.5).

В ході створення можна відмітити, чи хоче користувач отримати електронного листа про результат запуску (як успішний, так і у випадку помилки).

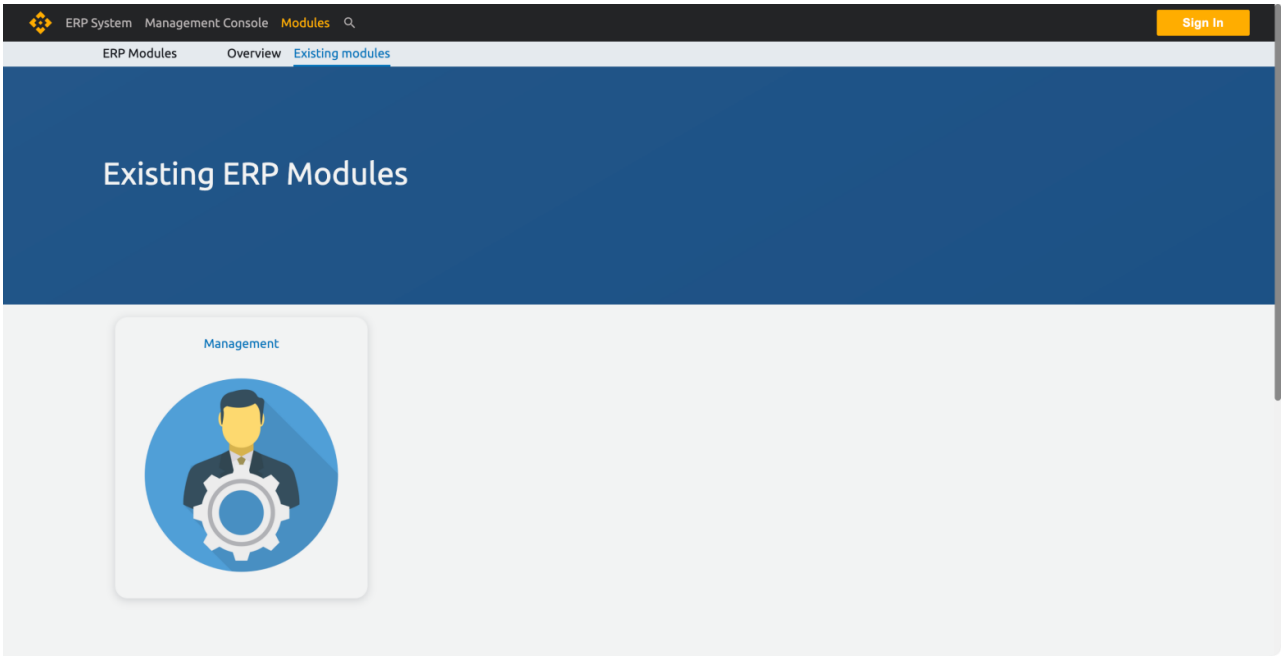


Рисунок 4.2 – Сторінка із модулями фреймворку

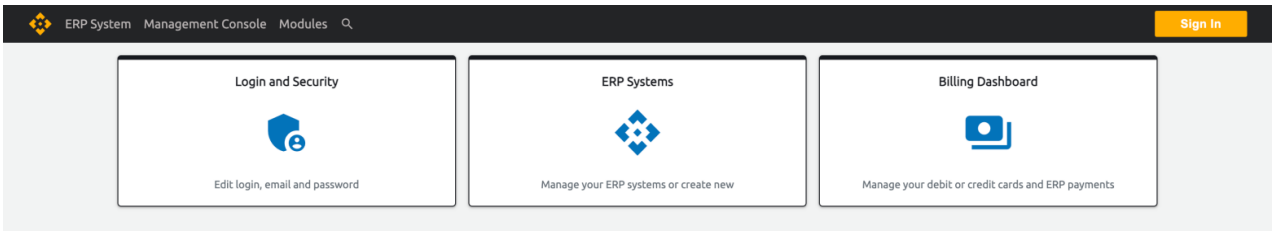


Рисунок 4.3 – Сторінка облікового запису користувача

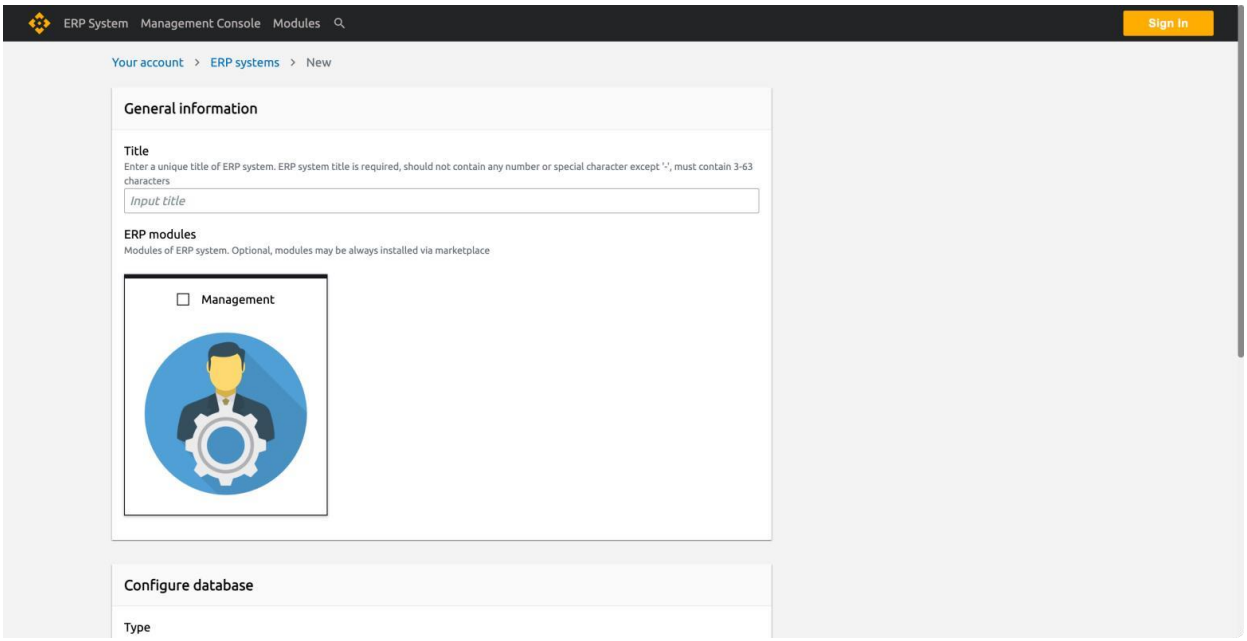


Рисунок 4.4 – Створення ERP системи (введення назви та вибір модулів)

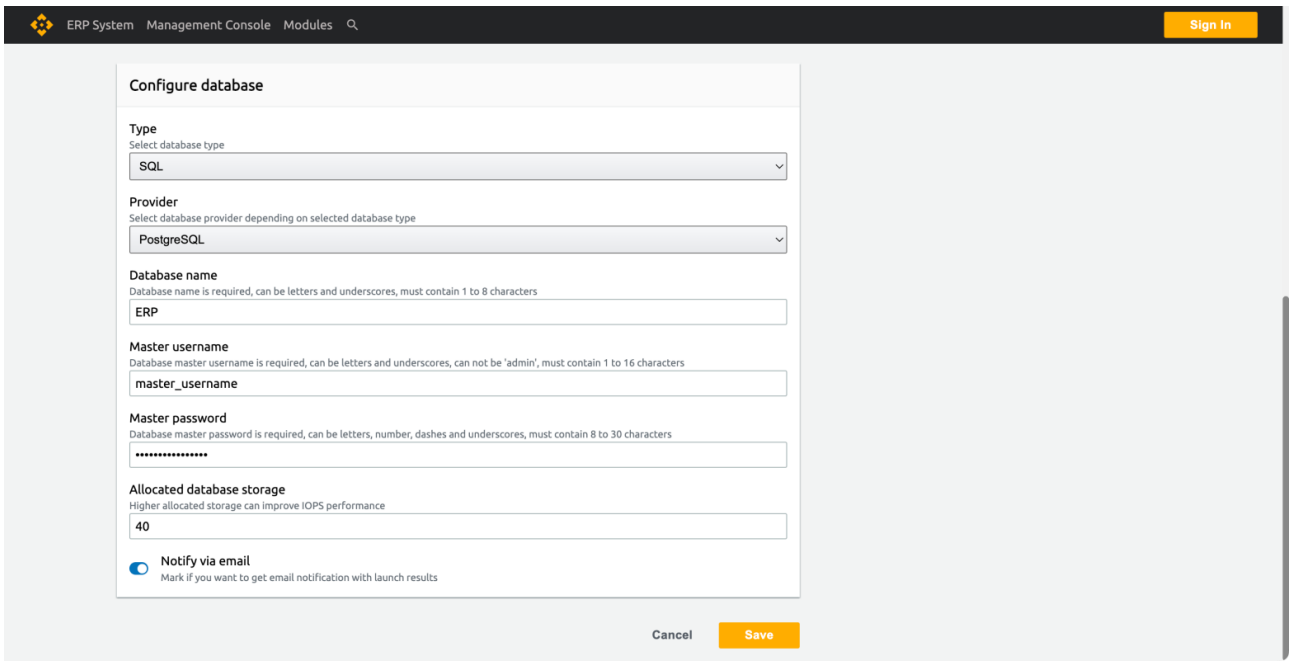


Рисунок 4.5 – Створення ERP системи (налаштування БД)

В результаті успішної операції на електронну адресу користувача прийде лист із результатами (рисунок 4.6).

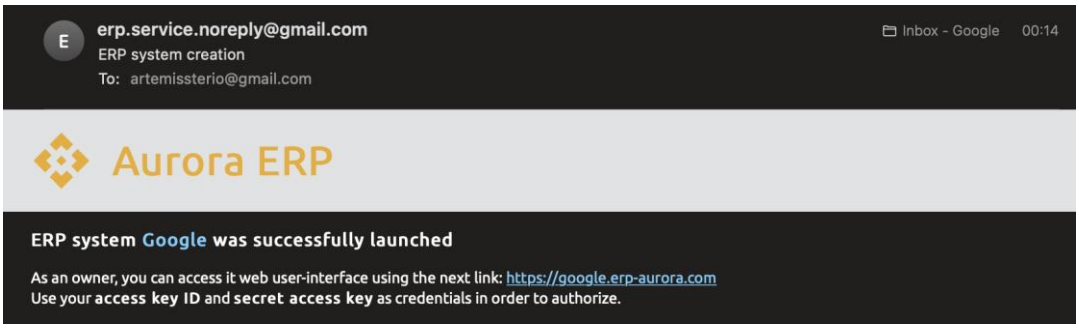


Рисунок 4.6 – Електронний лист про успішний запуск ERP системи

Процес запуску ERP системи складається із декількох наступних кроків:

– запис інформації про нову систему до БД. Після цього на вкладці ERP систем можна побачити, на якому етапі створення знаходиться вибрана система (рисунок 4.7). Всього існує 4 стани: Initializing (запис нової системи до БД), Creating database (створення екземпляру БД системи), Creating instance (створення екземпляру веб-серверу та балансувальника навантаження), Creating domain (створення глобального домену системи), Running (вказує, що система знаходиться у робочому стані (рисунок 4.11)), Shutting down (даний статус система набуватиме під час її видалення), Terminated (вказує, що система була

видалена та/або знаходиться у неробочому стані);

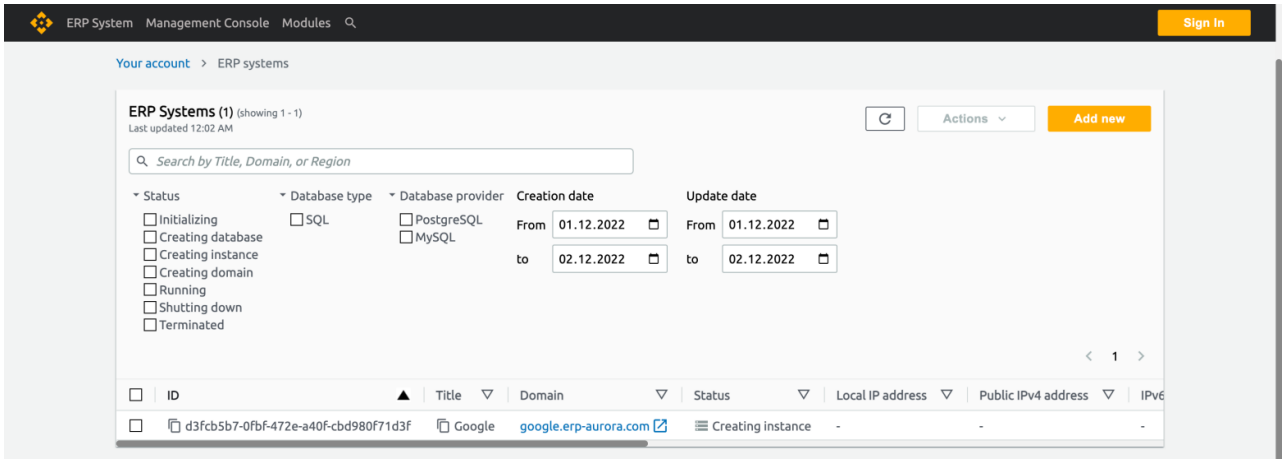


Рисунок 4.7 – Етапи створення ERP системи

– запуск БД системи в залежності від налаштувань. В першу чергу створюється БД система, адже саме до неї будуть перші запити від веб-серверу на створення таблиць в залежності від вибраних модулів. На рисунку 4.8 зображена панель налаштувань створеної БД у консолі AWS. Тільки після успішного запуску БД процес створення система перейде до наступного кроку;

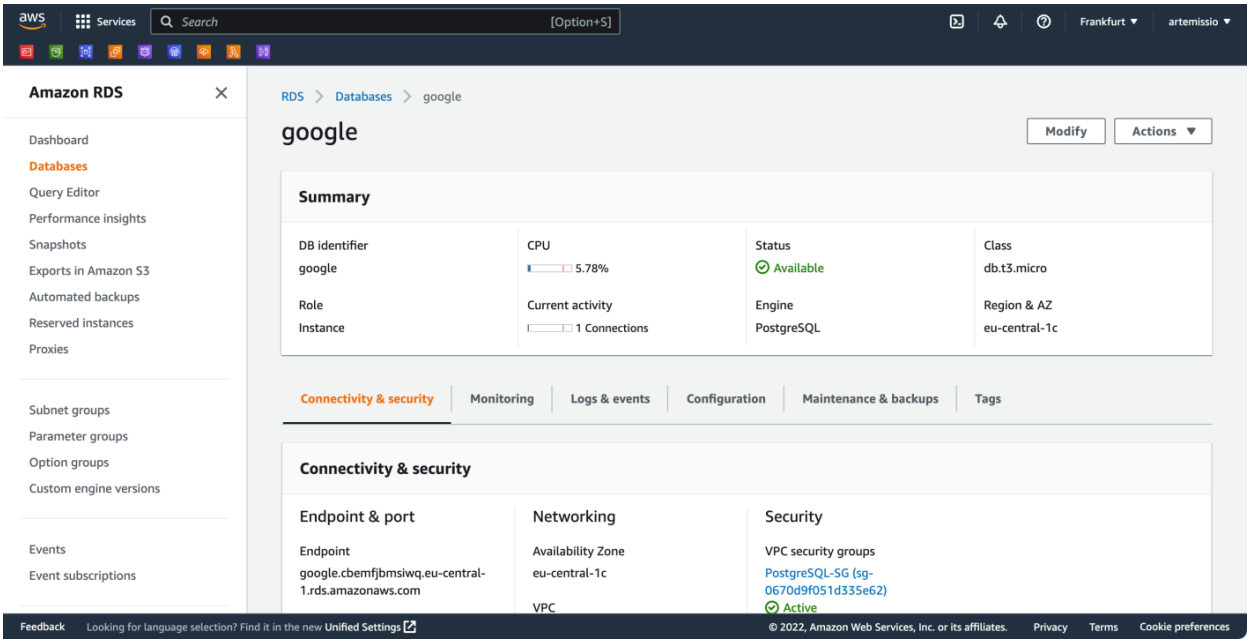


Рисунок 4.8 – Створена бази даних ERP системи

– запуск екземпляру веб-серверу. Після успішного запуску БД відбувається запуск віртуального серверу із скриптом на скачування виконуваного файлу для власне веб-сервісу і запускає його на 80 та 443 портах.

Згаданий файл зберігається у S3 бакеті у приватному доступі, але є доступним для екземплярів веб-серверів EC2. На рисунку 4.9 можна побачити деталі створеного серверу у консолі AWS;

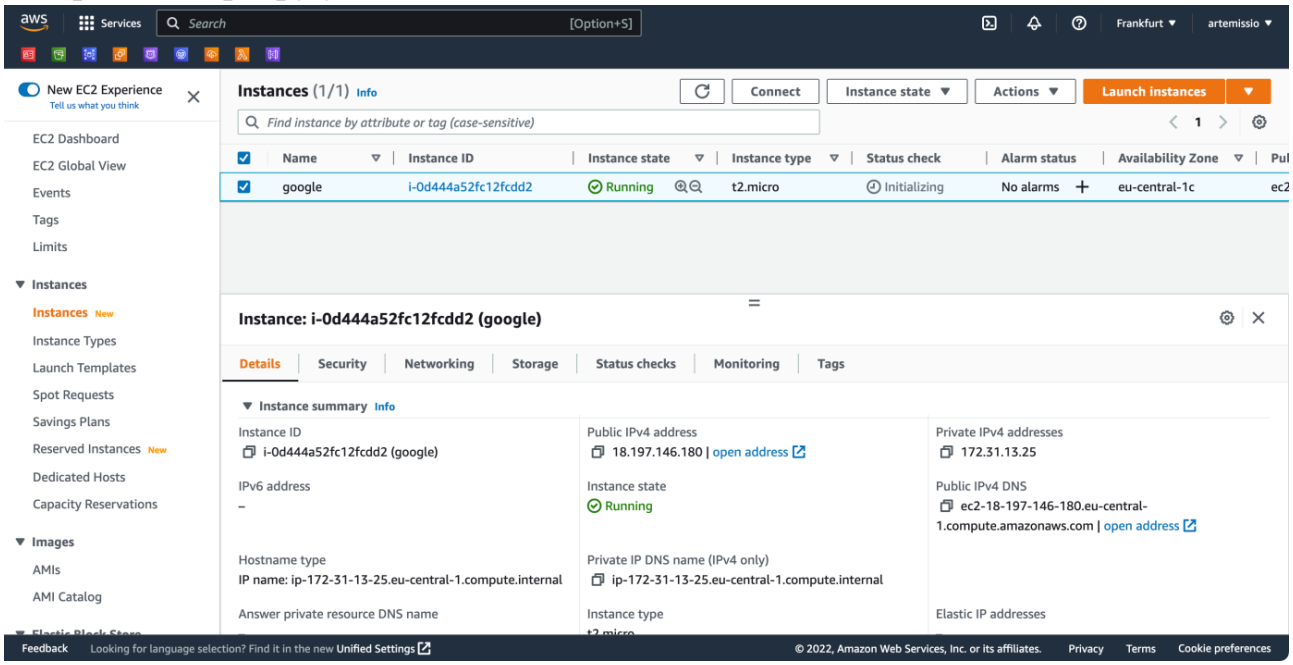


Рисунок 4.9 – Запущений веб-сервер ERP системи

– запуск балансувальника навантажень. Після успішного запуску веб-сервера відбувається запуск балансувальника навантажень із приєднаним веб-сервером (рисунок 4.10). Після ряду перевірок на роботоздатність серверу балансувальником процес переходить до наступного кроку;

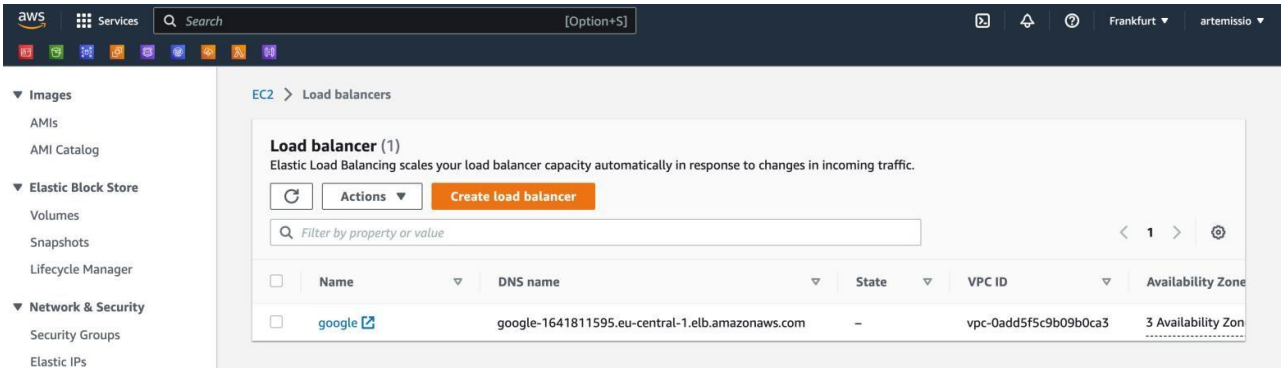


Рисунок 4.10 – Налаштований балансувальник навантаження ERP системи

– створення сабдомену і налаштування перенаправлення трафіку на створений балансувальник навантажень. На цьому процес запуску ERP системи

завершується, система приймає статус Running (рисунок 4.11), її сабдомен перенаправляє трафік на веб-сервіс (рисунок 4.12).

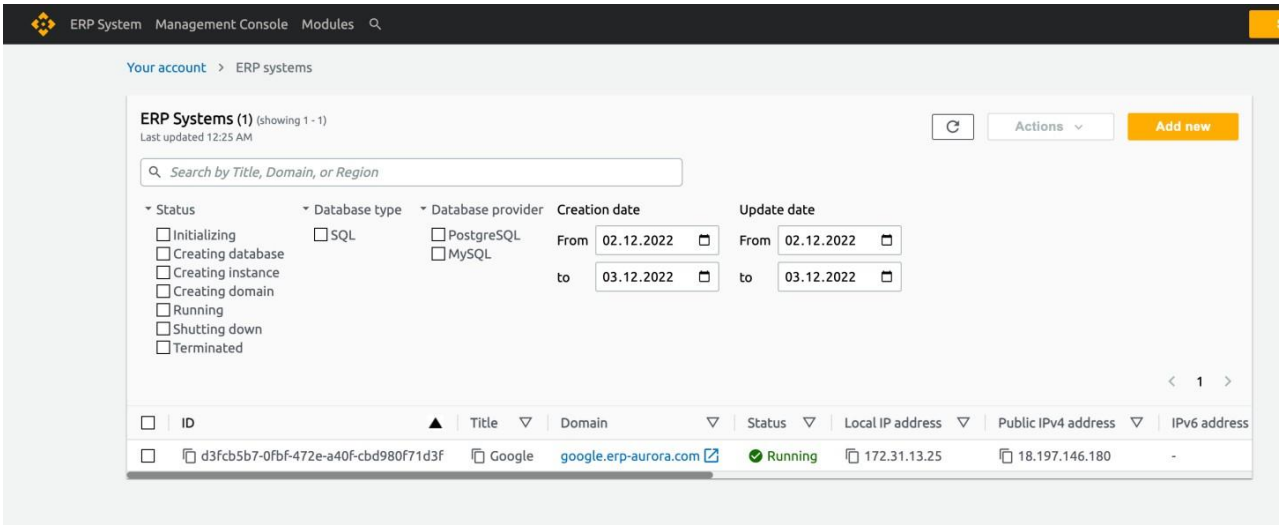


Рисунок 4.11 – Статус Running у запущеної ERP системи

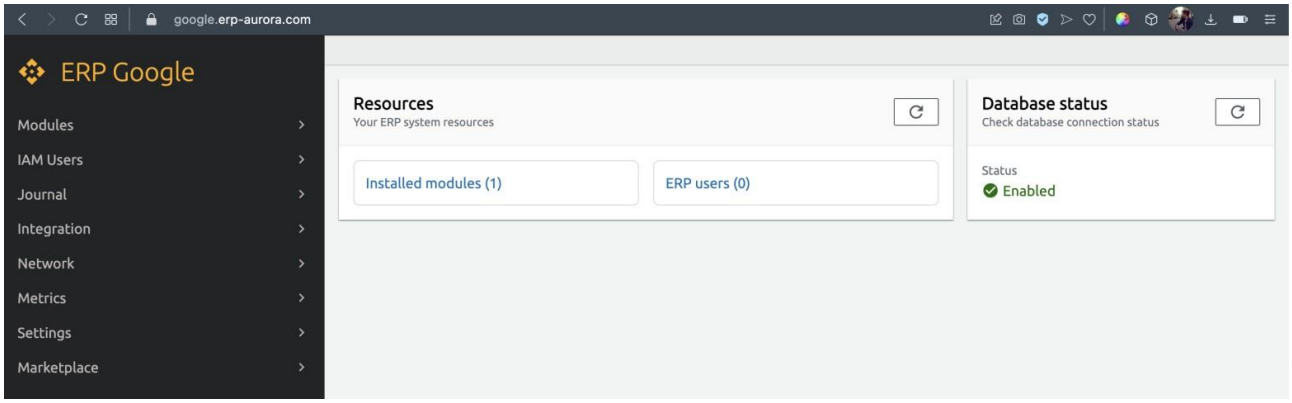


Рисунок 4.12 – Повністю запущена ERP система із глобальним Інтернет доменом

Процес видалення ERP системи є повністю зеркальним до процесу створення: система набуває статусу Shutting down, видаляється Інтернет сабдомен, щоб не можна було надсилати до системи запити під час її видалення, потім балансувальник навантажень, після цього веб-сервер, і в нарешті – БД системи.

Рисунок 4.13 – Електронний лист про успішне видалення ERP системи

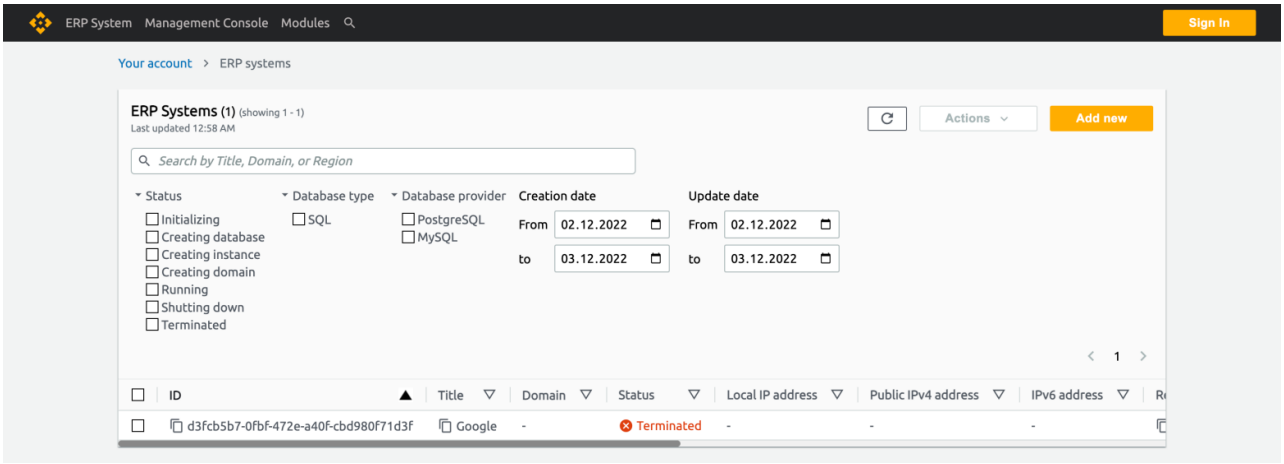


Рисунок 4.14 – Набуття ERP системою статусу Terminated

В кінці процесу користувач отримує електронного листа, який засвідчує, що вказана ERP система була успішно видалена (рисунок 4.13), а система набуває статусу Terminated (рисунок 4.14).

4.5. Розробка API

Однією з вимог до розробки було створення стабільного API фреймворку, здатного обробляти запити у будь-який момент і який буде справлятися із будь-яким навантаженням. Із такою поставленою задачею чудово справиться технологія безсерверної архітектури.

API фреймворку представляє собою набір лямбда-функцій, кожна із яких оброблятиме свій запит. Всього створено 9 лямбда-функцій:

– modules-marketplace: повертає всі невстановлені модулі у вказаній ERP системі;

– modules-describe: повертає детальну інформацію про вказаний модуль;

– modules-all: повертає всі опубліковані модулі фреймворку;

– erp-systems-list: повертає перелік ERP систем користувача;

– erp-systems-describe: повертає детальну інформацію про вказану ERP систему користувача;

– erp-systems-check-availability: повертає статус доступ назви ERP

системи. Використовується для перевірки, щоб не можна було створити декілька систем з однаковими назвами;

– erp-systems-create: запускає процес створення ERP системи;

– erp-systems-modify-module: встановлює або видаляє модуль із ERP системи;

– erp-systems-terminate: запускає процес видалення ERP системи.

Сама API є API шлюзом сервісу API Gateway, до якої приєднані дані лямбда- функції, і яка приєднана до сабдомену API із відповідним мапінгом – своєрідна маршрутизація, яка вказує, на яку API надіслати запит в залежності від шляху (рисунок 4.15).

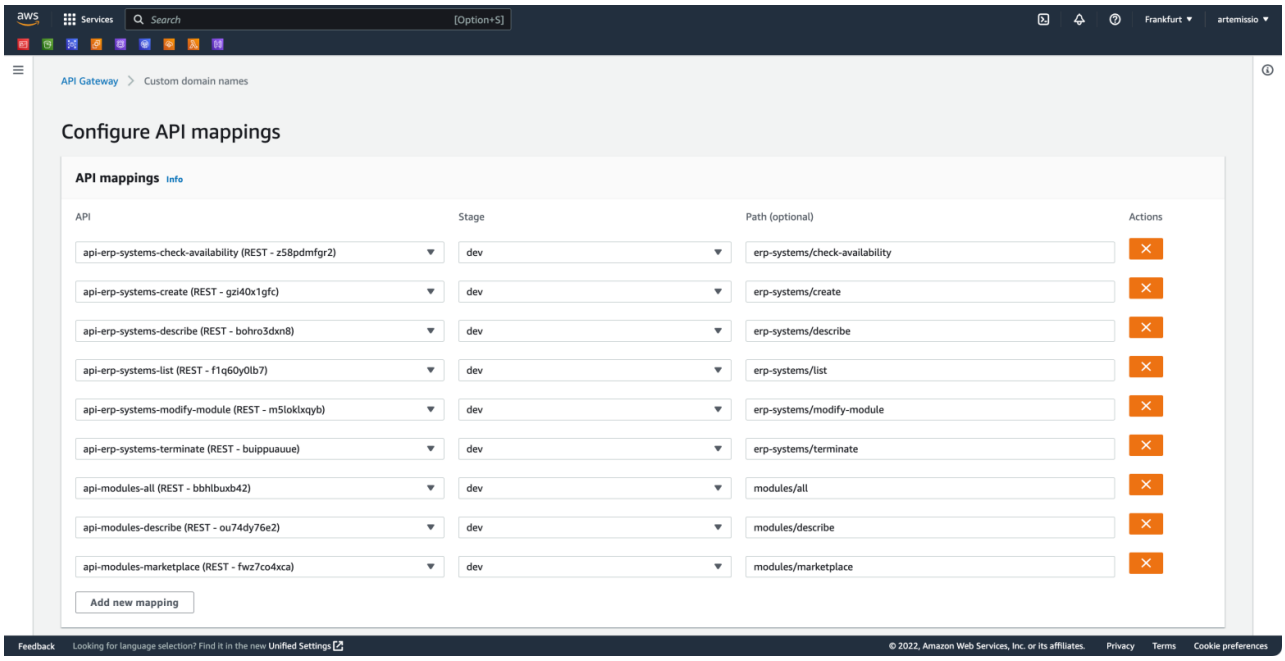


Рисунок 4.15 – Мапінг API шлюзу

Така реалізація є надзвичайно вигідною, оскільки оплата буде тільки за відпрацьовані запити, а також сама архітектура є дуже надійною – кожний запит буде оброблятися окремою функцією, тому навантаження як такого немає.

РОЗДІЛ 5.
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1. Розробка логіко-імітаційної моделі виникнення травм і аварій

Методикою оцінки рівня небезпеки робочих місць, машин, виробничих процесів та окремих виробництв передбачено пошук об'єктивного критерію рівня небезпеки для конкретного об'єкта [3]. Таким показником вибрана ймовірність виникнення аварії, травми залежно від явища, що досліджується.

Для побудови логіко-імітаційної моделі процесу, формування і виникнення аварії та травми в процесі створення мікрокліматичних умов у приміщенні оцінюють відповідні небезпечні події. Кожній їх них присвоїмо ймовірність виникнення:

Шифр	Назва події	Ймовірність
P_1	Відсутність захисного заземлення	0,02
P_2	Пошкодження захисного заземлення	0,04
P_3	Спрацювання складових захисту	0,1
P_4	Неправильна експлуатація захисту	0,02
P_5	Відсутність профілактичних заходів	0,2
P_6	Відсутність захисного щита	0,12
P_7	Недотримання правил вибору взуття	0,15
P_8	Незнання правил техніки безпеки	0,1
P_9	Відсутність засобів індивідуального захисту	0,2
P_{10}	Легковажність	0,08

На основі наведених подій будуємо матрицю логічних взаємозв'язків між окремими пунктами, графічна інтерпретація якої зображено на рис. 5.1.

Розрахуємо ймовірності виникнення подій, що формують логіко-імітаційну модель процесів створення мікрокліматичних умов. Розглянемо травмонебезпечну ситуацію, що виникає за умови роботи працівників із електронебезпекою.

Підставивши дані ймовірностей базових подій у формулу, отримаємо ймовірність події 13: $P_{13} = 0{,}2 + 0{,}4 - 0{,}2 \cdot 0{,}4 = 0{,}0592$.

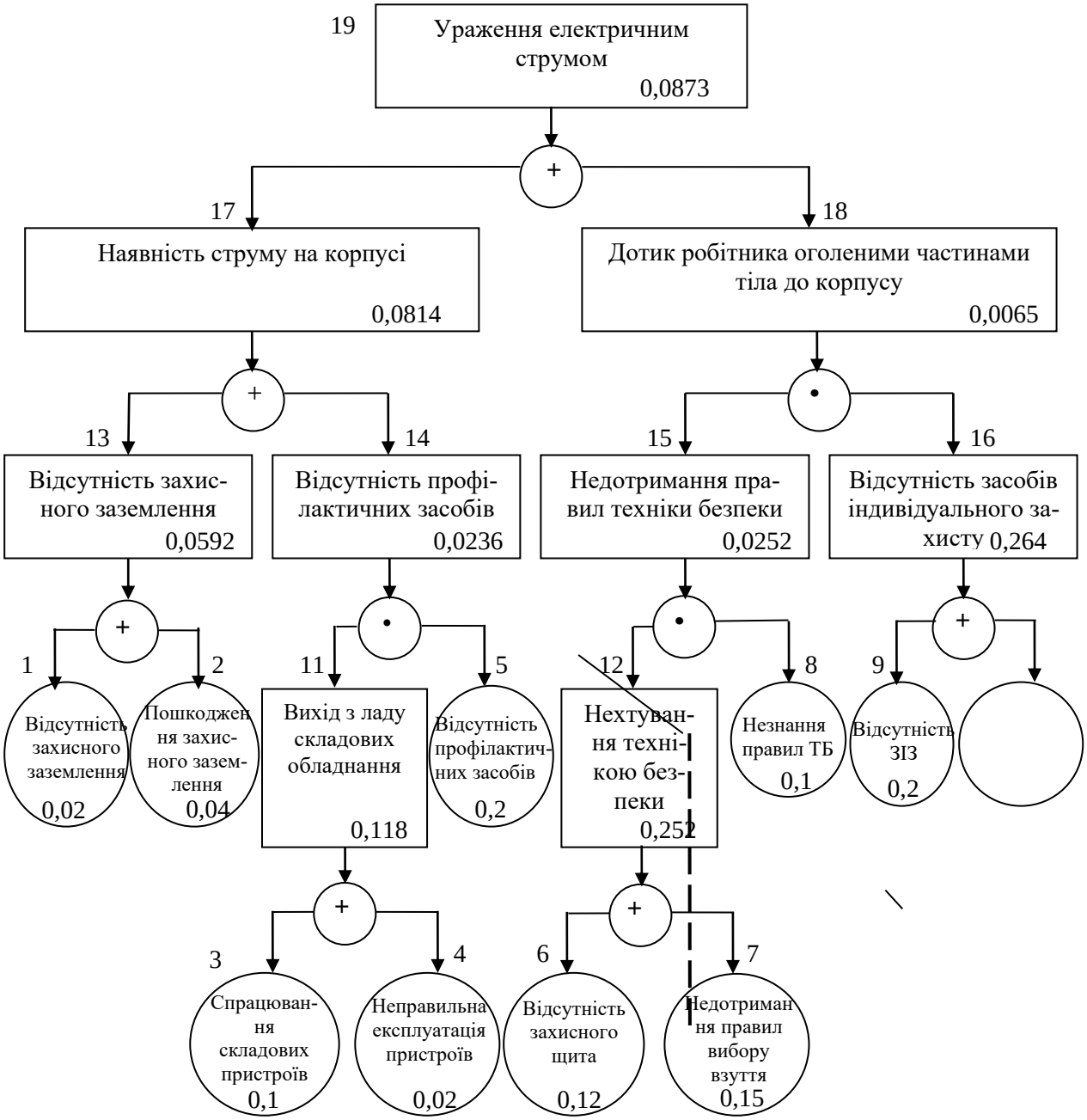


Рис. 5.1. Матриця логічних взаємозв'язків між окремими подіями травмонебезпечної ситуації [3]

Аналогічно визначаємо ймовірність інших подій:

$$P_{11} = P_4 + P_5 - P_4P_5 = 0{,}3 + 0{,}4 - 0{,}3 \cdot 0{,}4 = 0{,}118.$$

$$P_{12} = P_6 + P_7 - P_6P_7 = 0{,}3 + 0{,}5 - 0{,}3 \cdot 0{,}5 = 0{,}252.$$

$$P_{16} = P_9 + P_{10} - P_9P_{10} = 0{,}2 + 0{,}15 - 0{,}2 \cdot 0{,}15 = 0{,}264.$$

$$P_{14} = P_{11} \cdot P_5 = 0{,}118 \cdot 0{,}2 = 0{,}0236.$$

$$P_{15} = P_{12} \cdot P_8 = 0{,}252 \cdot 0{,}1 = 0{,}0252.$$

$$P_{17} = P_{13}+P_{14} - P_{13}\cdot P_{14} = 0{,}592+0{,}0236-0{,}0592\cdot 0{,}0236 = 0{,}0814.$$

$$P_{18} = P_{15}\cdot P_{16} = 0{,}264\cdot 0{,}0252 = 0{,}0065.$$

$$P_{19} = P_{17}+P_{18} - P_{17}\cdot P_{18} = 0{,}0065+0{,}0814 - 0{,}0065\cdot 0{,}0814 = 0{,}0873.$$

Таким чином, ймовірність перекидання машини та наслідкового виникнення травми працівника є досить мала і становить – $P_{19} = 0,0873$.

5.2. Планування заходів із покращення умов праці

До заходів щодо покращення умов праці належать всі види діяльності, спрямовані на попередження, нейтралізацію або зменшення негативної дії шкідливих і небезпечних виробничих факторів на працівників.

Рівень умов праці оцінюють порівнянням за фактичними і нормативними значеннями узагальнених (групових) показників.

Заходи щодо поліпшення умов праці здійснюють з метою створення безпечних умов праці шляхом:

- доведення до нормативного рівня показників виробничого середовища за елементами умов праці;
- захисту працівників від дії небезпечних і шкідливих виробничих факторів.

До показників ефективності заходів щодо поліпшення умов праці належать:

а) зміни стану умов праці:

- зміна кількості засобів виробництва, приведених у відповідність до вимог стандартів безпеки праці;
- покращання санітарно-гігієнічних показників;
- покращання психофізичних показників, зменшення фізичних і нервово-психічних навантажень, в т.ч. монотонних умов праці;
- покращання естетичних показників, раціональне компонування робочих місць і впорядкування робочих приміщень;

б) соціальні результати заходів:

- збільшення кількості робочих місць, що відповідають нормативним вимогам;
- зниження рівня виробничого травматизму;
- зменшення кількості випадків професійних захворювань;
- зменшення плинності кадрів через незадовільні умови праці;
- престиж та задоволення працею.

Отже, на покращення охорони праці потрібно виділити кошти на відновлення вентиляційних систем у ремонтних майстернях, естетично оформити приміщення офісу, відновити кабінет з охорони праці, поновити протипожежний інвентар.

5.3. Безпека в надзвичайних ситуаціях

Актуальність проблеми природно-техногенної безпеки для населення і території, зумовлена зростанням втрат людей, що спричиняється небезпечними природними явищами, промисловими аваріями та катастрофами. Ризик надзвичайних ситуацій природного та техногенного характеру невпинно зростає, тому питання захисту цивільного населення від надзвичайних ситуацій на сьогодні є дуже важливе [3].

У системі цивільної оборони окремого господарства необхідно забезпечити захист населення таким чином:

Укриття в захисних спорудах, якому підлягає усе населення відповідно до приналежності, досягається створенням фонду захисних споруд.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об’єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення його у позаміській зоні.

Медичний захист проводиться для зменшення ступеня ураження людей, своєчасного надання допомоги постраждалим та їх лікування, забезпечення епідеміологічного благополуччя в районах надзвичайних ситуацій.

Радіаційний і хімічний захист включає заходи щодо виявлення і оцінки радіаційної та хімічної обстановки, організацію і здійснення дозиметричного та хімічного контролю, розроблення типових режимів радіаційного захисту, забезпечення засобами індивідуального захисту, організацію і проведення спеціальної обробки.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об’єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення у позаміській зоні.

ВИСНОВКИ І РЕКОМЕНДАЦІЇ

1) ERP – це скорочення від Enterprise Resource Planning, тобто планування ресурсів підприємства. Головна функція ERP – об’єднання всіх бізнес-процесів (і ресурсів компанії) в одну систему, а також спрощення, прискорення й оптимізація. Цього вдається досягти завдяки автоматизації рутинних процесів та уникненню багаторазового введення тієї ж інформації виконавцями з різних відділів.

2) Система планування виробничих ресурсів (MRP II) — це інтегрована інформаційна система, яка використовується підприємствами. Планування виробничих ресурсів (MRP II) розвинулося на основі попередніх систем планування потреб у матеріалах (MRP) із додатковою можливістю інтеграції додаткових даних, таких як вимоги до робочої сили та фінансові потреби.

3) Ключовим аспектом будь-якої ERP-системи є модульність. Основою цієї концепції є поділ загальних сутностей або функціональних можливостей на окремі блоки. У той же час модульність — це тип властивості системи, яка вимірює ступінь, наскільки сильно пов’язані компоненти в системі можуть бути розділені на окремі сутності або кластери, які взаємодіють один з одним більше, ніж інші.

4) Організації всіх типів, розмірів та галузей використовують хмару для різних цілей, включаючи резервне копіювання даних, аварійне відновлення, електронну пошту, віртуальні робочі столи, розробку та тестування програмного забезпечення, аналіз великих даних та клієнтські веб-додатки. Наприклад, медичні компанії використовують хмарні технології розробки більш персоналізованих методів лікування своїх пацієнтів. Фінансові компанії використовують хмару для виявлення та запобігання шахрайству в режимі реального часу.

5) Фреймворк, що розробляється, повинен давати користувачу можливість швидко створювати веб-сервіси ERP систем у глобальному Інтернеті із глобальним Інтернет доменом. Створити та видалити ERP системи повинна

бути змога як і через головний веб-сайт, так і через запит до API даного фреймворку. При процесі створення користувач повинен мати досить різноманітний вибір налаштувань, таких як назва системи, тип бази даних, логін і пароль для доступу до БД тощо.

6) Визначений список основних технічних та функціональних вимог фреймворку -підставою для цих вимог є дані, одержані на основі огляду та аналізу існуючих рішень. Використання хмарних сервісів може заощадити багато фінансових ресурсів та значно прискорити процес розробки та впровадження нових функцій.

7) Запропоновано технічну вимогу щодо підписання запитів для забезпечення надійності захисту передачі інформації між додатками. Оскільки йдеться про корпоративні дані, крадіжка цієї інформації може призвести до значних фінансових втрат для компанії.

8) Сучасна мова програмування C# і платформа .NET роблять процес розробки швидким, простим та надійним. Володіючи загальною кодовою базою для всіх програм, Blazor Server стає зручним інструментом для швидкої розробки веб-додатків на вищезгаданих мовах програмування без необхідності вивчення JavaScript. EF Core спрощує та прискорює роботу з базами даних як у самій структурі, так і у вашій ERP- системі.

9) Розроблений фреймворк із створення веб-сервісів управління ресурсами підприємства, починаючи з розробки загальної структури системи, переліку застосунків, завершуючи реалізацією конкретних функціональних можливостей. Описано процес розробки кожного застосунку-компонента фреймворку, технології та архітектурні принципи та підходи, що були використані при його розробці, його взаємодія із іншими компонентами фреймворку.

10) Окремої уваги заслуговує публікація застосунків у хмарі: були описані та аргументовані причини використання того чи іншого хмарного сервісу, що був використаний при розробці того чи іншого застосунку.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Здобувач А. Бізнес-перспектива використання інформаційних технологій при реалізації ERP-проекту на підприємстві / А. Здобувач // Науковий вісник НЛТУ України. 2014. № 24.7. с. 344-350.

2. Карпенко М. Ю., Манакова Н. О., Гавриленко І. О. Технології створення програмних продуктів та інформаційних систем : навч. посіб. Харків : ХНУМГ ім. О. М. Бекетова, 2017. 93 с.

3. Лехман С.Д. та ін. Запобігання аварійності і травматизму у сільському господарстві / С.Д. Лехман, В.І. Рубльов, Б.І. Рябцев. К.: Урожай, 1993. 272 с.

4. Метод функціонального моделювання : веб-сайт. URL: http://ni. biz.ua/3/3_20/3_20501_metod-funktsionalnogo-modelirovaniya-SADT-IDEF.html (дата звернення: 10.12.2023).

5. СУБД. URL: https://itglobal.com/company/glossary/subd-sistema-upravleniya- bazami-dannyh/ (дата звернення: 07.11.2023).

6. Черненко М. Реінжиніринг і псевдореінжиніринг / М. Черненко // Управління компанією. 2008. № 21-22. С. 31-35.

7. Що таке MRP. URL: https://www.sap.com/cis/insights/what-is-mrp.html (дата звернення: 31.10.2023).

8. Amazon ARN. URL: https://docs.aws.amazon.com/general/latest/gr/aws-arns-and- namespaces.html (дата звернення: 11.11.2023).

9. Amazon Web Services. URL: https://aws.amazon.com (дата звернення: 10.11.2023).

10. API vs. SDK: The Difference Explained. URL: https://getstream.io/glossary/api-vs-sdk/ (дата звернення: 05.11.2023).

11. Blazor. URL: https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor (дата звернення: 07.11.2023).

12. Canonical requests. URL: https://cloud.google.com/storage/docs/authentication/ canonical-requests (дата

звернення: 12.11.2023).

13. CQRS Pattern. URL: https://habr.com/ua/post/347908/ (дата звернення: 07.11.2023).

14. C# Documentation. URL: https://docs.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 07.11.2023).

15. Entity Framework Core documentation : веб-сайт. URL: https://docs. microsoft. com/en-us/ef/core (дата звернення: 07.12.2023).

16. Entity Framework Core. URL: https://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx (дата звернення: 07.112023).

17. Epicor Kinetic. URL: https://www.epicor.com/en/industry-productivity-solutions/manufacturing/platforms/kinetic/ (дата звернення: 04.11.2023).

18. ESB. URL: https://habr.com/ru/company/pnn/blog/191600/ (дата звернення: 03.11.2023).

19. Google Cloud Platform. URL: https://cloud.google.com (дата звернення: 10.11.2023).

20. Microsoft Azure. URL: https://azure.microsoft.com/en-us/ (дата звернення: 10.11.2023).

21. Microsoft Dynamics 365. URL: https://dynamics.microsoft.com/en-us/ (дата звернення: 04.11.2023).

22. Modular ERP: advantages and disadvantages. URL: https://blog.vault-erp.com/post/2021/10/21/modular-erp-advantages-and-disadvantages (дата звернення: 31.10.2023).

23. MySQL Workbench : веб-сайт. URL: https://www.mysql.com/products/ workbench (дата звернення: 10.12.2023).

24. Oracle ERP Cloud. URL: https://en.wikipedia.org/wiki/Oracle_Cloud_Enterprise_Resource_Planning (дата звернення: 04.11.2023).

25. Oracle NetSuite. URL: https://www.netsuite.com/portal/home.shtml (дата звернення: 04.11.2023).

26. Overview of integration with ERP components. URL: https://www.ibm.com/docs/en/order-management-sw/10.0?topic=systems-overview-integration-erp-components (дата звернення: 31.10.2023).

27. Paul DuBois. MySQL Cookbook. 3rd edition. O'Reilly, 2014. 837 p.

28. PostgreSQL. URL: https://www.postgresql.org/ (дата звернення: 07.11.2023).

29. RFC 4868 HMAC SHA256. URL: https://www.rfc-editor.org/rfc/rfc4868 (дата звернення: 11.11.2023).

30. SAP S/4HANA. URL: https://www.sap.com/ukraine/products/erp/s4hana.html (дата звернення: 04.11.2023).

31. SignalR Core. URL: https://habr.com/ru/company/jugru/blog/349096/ (дата звернення: 07.11.2023).

32. Top ERP security problems and best practices. URL: https://www.netsuite.com/portal/resource/articles/erp/erp-security-best-practices.shtml (дата звернення: 31.10.2023).

33. Top 10 Programming Languages for Desktop Apps In 2021. Decipherzone : веб-сайт. URL: https://www.decipherzone.com/blog-detail/topprogramming-languages-for-desktop-apps-in-2021 (дата звернення: 07.12.2023).

34. UML Use Case Diagrams : веб-сайт. URL: https://www.umldiagrams.org/use-case-diagrams.html (дата звернення: 07.12.2023).

35. What is an Entity Diagram(ERD)? веб-сайт. URL: https://medium.com/@soni.dumitru/what-is-an-entity-relationship-diagram-erd13daee5b2a.

36. .NET Platform. URL: https://dotnet.microsoft.com/ (дата звернення: 07.11.2023).