Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Дослідження та розробка застосунку для автоматизації бізнес процесів з розширеними інтеграційними можливостями

Виконали: студенти VI курсу, групи СНнм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Тененський М. В.
(підпис) (прізвище та ініціали)

_____ Галюк М. В.
(підпис) (прізвище та ініціали)

Керівник _____ Никитюк В. В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Дуда О. М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Михалик Д. М.
(підпис) (прізвище та ініціали)

Рецензент _____ Михалик Д. М.
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«13» травня 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Галюку Миколі Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та розробка застосунку для автоматизації бізнес процесів з розширеними інтеграційними можливостями

Керівник роботи Никитюк Вячеслав Вячеславович, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 24 » листопада 2023 року № 4/7-1100

2. Термін подання студентом завершеної роботи 13 травня 2024р.

3. Вихідні дані до роботи Наукові статті та публікації про бізнес процеси, управління ними та їх автоматизацію, в тому числі засобами застосунків із розширеними інтеграційними можливостями.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Дослідження бізнес процесу. 2 Дослідження застосунків для управління та автоматизації бізнес процесів. 3. Проектування та розробка застосунку для автоматизації бізнес процесів. 4. Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Перелік джерел. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульна сторінка. 2 Тема, мета, об'єкт та предмет дослідження. 3 Завдання дослідження. 4 Поняття бізнес процес. 5 Методи покращення бізнес процесів. 6 Управління бізнес процесами.7 Типи застосунків для управління бізнес процесами та їх вплив. 8 Огляд та аналіз сучасних застосунків. 9 Архітектурі особливості оглянутих застосунків 10 Вимоги до розроблюваного застосунку. 11 Архітектура розроблюваного застосунку. 12. Вибір технологій для розробки. 13. Проектування структури даних 14. Розробка серверної частини застосунку та користувацького інтерфейсу.15 Інтеграція через API. 16. Інтеграція через брокери повідомлень. 17 Висновки. 18 Завершальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання 24 листопада 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	25.11.2023	*Виконано*
2.	Підбір наукових джерел про бізнес процеси, керування	27.11.2023-01.12.2023	*Виконано*
	ними та їх автоматизацію		
3.	Опрацювання наукових публікацій та збір даних по	04.12.2023-12.01.2024	*Виконано*
	темі кваліфікаційної роботи		
4.	Виконання дослідження згідно мети та поставленої	15.01.2024-16.02.2024	*Виконано*
	задачі кваліфікаційної роботи		
5.	Оформлення розділу «Дослідження поняття бізнес	19.02.2024-01.03.2024	*Виконано*
	процес»		
6.	Оформлення розділу «Дослідження сучасних	04.03.2024-15.03.2024	*Виконано*
	застосунків для управління та автоматизації бізнес		
	процесів»		
7.	Оформлення розділу «Проектування та розробка	18.03.2024-05.04.2024	*Виконано*
	власного застосунку для автоматизації бізнес		
	процесів»		
8.	Виконання завдання до підрозділу «Охорона праці»	08.04.2024-12.04.2024	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	15.04.2024-19.04.2024	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	22.04.2024-26.04.2024	*Виконано*
11.	Нормоконтроль	29.04.2024-03.05.2024	*Виконано*
12.	Перевірка на плагіат	03.05.2024	*Виконано*
13.	Попередній захист кваліфікаційної роботи	05.05.2024	*Виконано*
14.	Захист кваліфікаційної роботи	29.05.2024	

Студент _____ Галюк М. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Никитюк В. В.
(підпис) (прізвище та ініціали)

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«13» травня 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Тененському Максиму Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та розробка застосунку для автоматизації бізнес процесів з розширеними інтеграційними можливостями

Керівник роботи Никитюк Вячеслав Вячеславович, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 24 » листопада 2023 року № 4/7-1100

2. Термін подання студентом завершеної роботи 13 травня 2024р.

3. Вихідні дані до роботи Наукові статті та публікації про бізнес процеси, управління ними та їх автоматизацію, в тому числі засобами застосунків із розширеними інтеграційними можливостями.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Дослідження бізнес процесу. 2 Дослідження застосунків для управління та автоматизації бізнес процесів. 3. Проектування та розробка застосунку для автоматизації бізнес процесів. 4. Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Перелік джерел. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульна сторінка. 2 Тема, мета, об'єкт та предмет дослідження. 3 Завдання дослідження. 4 Поняття бізнес процес. 5 Методи покращення бізнес процесів. 6 Управління бізнес процесами.7 Типи застосунків для управління бізнес процесами та їх вплив. 8 Огляд та аналіз сучасних застосунків. 9 Архітектурі особливості оглянутих застосунків 10 Вимоги до розроблюваного застосунку. 11 Архітектура розроблюваного застосунку. 12. Вибір технологій для розробки. 13. Проектування структури даних 14. Розробка серверної частини застосунку та користувацького інтерфейсу.15 Інтеграція через API. 16. Інтеграція через брокери повідомлень. 17 Висновки. 18 Завершальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання 24 листопада 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	25.11.2023	*Виконано*
2.	Підбір наукових джерел про бізнес процеси, керування	27.11.2023-01.12.2023	*Виконано*
	ними та їх автоматизацію		
3.	Опрацювання наукових публікацій та збір даних по	04.12.2023-12.01.2024	*Виконано*
	темі кваліфікаційної роботи		
4.	Виконання дослідження згідно мети та поставленої	15.01.2024-16.02.2024	*Виконано*
	задачі кваліфікаційної роботи		
5.	Оформлення розділу «Дослідження поняття бізнес	19.02.2024-01.03.2024	*Виконано*
	процес»		
6.	Оформлення розділу «Дослідження сучасних	04.03.2024-15.03.2024	*Виконано*
	застосунків для управління та автоматизації бізнес		
	процесів»		
7.	Оформлення розділу «Проектування та розробка	18.03.2024-05.04.2024	*Виконано*
	власного застосунку для автоматизації бізнес		
	процесів»		
8.	Виконання завдання до підрозділу «Охорона праці»	08.04.2024-12.04.2024	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	15.04.2024-19.04.2024	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	22.04.2024-26.04.2024	*Виконано*
11.	Нормоконтроль	29.04.2024-03.05.2024	*Виконано*
12.	Перевірка на плагіат	03.05.2024	*Виконано*
13.	Попередній захист кваліфікаційної роботи	05.05.2024	*Виконано*
14.	Захист кваліфікаційної роботи	29.05.2024	

Студент _____ Тененський М. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Никитюк В. В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дослідження та розробка застосунку для автоматизації бізнес процесів з розширеними інтеграційними можливостями // Кваліфікаційна робота освітнього рівня «Магістр» // Галюк Микола Васильович // Тененський Максим Васильович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНнм-61 // Тернопіль, 2024 // С. 108, рис. – 40, табл. – 4, кресл. – 18, додат. – 5, бібліогр. – 97.

Ключові слова: автоматизація, архітектура ПЗ, бізнес процес, інтеграційні можливості, BPM, EDA, UAF.

Кваліфікаційна робота присвячена дослідженню сучасних застосунків для автоматизації бізнес процесів та розробці власного MVP рішення із розширеними інтеграційними можливостями на базі фреймворку UAF.

В першому розділі кваліфікаційної роботи описано поняття бізнес процес, подано докладну класифікацію бізнес процесів, розглянуто методи покращення бізнес процесів, визначено переваги та аргументовано необхідність їх впровадження.

В другому розділі кваліфікаційної роботи досліджено сучасні рішення для управління та автоматизації бізнес процесів, наведено статистику їх використання, проведено порівняльний аналіз найбільш популярних засобів для автоматизації бізнес процесів, описано їх переваги та недоліки, а також досліджено типові архітектурні підходи та вимоги до їх розробки.

В третьому розділі кваліфікаційної роботи аргументовано та докладно описано процес розробки серверної частини та користувацького інтерфейсу власного рішення з розширеними інтеграційними можливостями, подано опис використаних програмних засобів та наведено дані про структуру розробленого рішення.

Об’єктом дослідження є оптимізація різногалузевих задач шляхом використання застосунків для автоматизації бізнес процесів.

Предметом дослідження є сучасні застосунки для автоматизації бізнес процесів із розширеними інтеграційними можливостями.

ANNOTATION

Research and development of an application for automating business processes with advanced integration capabilities // The educational level "Master" qualification work // Haliuk Mykola // Tenenskyi Maksym // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, SNnm-61 group // Ternopil, 2024 // P. 108, fig. – 40, tables – 4, posters – 18, annexes – 5, ref. – 97.

Key words: automation, software architecture, business process, integration capabilities, BPM, EDA, UAF.

The qualification work is devoted to the study of modern applications for business process automation and the development of an MVP solution with advanced integration capabilities based on the UAF framework.

The first section of the qualification work describes the concept of business process, provides a detailed classification of business processes, considers methods of improving business processes, identifies the benefits and argues the need for their implementation.

The second section of the qualification work examines modern solutions for managing and automating business processes, provides statistics on their use, conducts a comparative analysis of the most popular tools for automating business processes, describes their advantages and disadvantages, and also explores typical architectural approaches and requirements for their development.

In the third chapter of the qualification work, the process of developing the server part and user interface of the own solution with advanced integration capabilities is argued and described in detail, a description of the software used is given, and data on the structure of the developed solution is provided.

The object of the study is the optimization of diverse tasks by using applications for business process automation.

The subject of the study is modern applications for business process automation with advanced integration capabilities.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

БД – База даних.

БП – Бізнес процес.

ІТ – Інформаційні технології.

ПЗ – програмне забезпечення.

Фреймворк – ПЗ, покликане полегшити розробку шляхом поєднання в собі різноманітних можливостей.

API (англ. Application Programming Interface) – набір визначень взаємодії різнотипного ПЗ.

BPA (англ. Business Process Automation) – оптимізація бізнес процесів.

BPM (англ. Business Process Management) – дисципліна, спрямована на аналіз, моделювання, оптимізацію та автоматизацію бізнес-процесів.

BPR (англ. Business Process Reengineering) – реінжиніринг бізнес процесів.

BPMS (англ. Business Process Management System) – технологічне ПЗ для управління, аналізу, оптимізації тощо бізнес-процесів.

DAO (англ. Data Acess Object) – об’єкт, який надає інтерфейс для доступу до даних з БД.

EDA (англ. Event Driven Architecture) – подієво-орієнтована архітектура.

MVP (англ. Minimum Viable Product) – продукт з мінімальним функціоналом, що може застосовуватись користувачами для рішення поставлених задач.

PDA (англ. Process Driven Architecture) – процесно-орієнтована архітектура.

SPA (англ. Single Page Application) – застосунок, користувацький інтерфейс якого вміщується на одній сторінці з метою забезпечення кращого досвіду користування.

UAF (англ. Unicorn Application Framework) – фреймворк для створення веб-застосунків, розроблений однойменною компанією.

ЗМІСТ

ВСТУП

Актуальність теми. Використання новітніх інформаційних технології в сфері управління є одним з ключових факторів розвитку та конкурентоспроможності підприємств. Так, завдяки цифровізації багатьох галузей людської діяльності, можливим стало проведення оптимізації та автоматизації бізнес процесів, що позитивно впливає на виробничі процеси та надання послуг [1].

Варто зважати на те, що з кожним роком відбувається ріст складності та динамічності бізнес процесів, що, в свою чергу, кидає нові виклики в сфері взаємодії людини з програмним забезпеченням. Оптимізація та автоматизація бізнес процесів дозволяють зробити цю взаємодію більш передбачуваною та прозорою, а також надають змогу відчутно зменшити витрати виробничих циклів. Завдяки цьому підприємства мають змогу ефективно, швидко та без значних витрат реагувати на зміну займаного ними ринку, що стає визначальним фактором успіху в наш час.

Існує безліч застосунків, спрямованих на автоматизацію бізнес процесів. Їх правильний вибір відповідно до актуальних вимог, розуміння принципів функціонування та коректне впровадження в сферу управляння процесами є критично важливим для підприємств, які прагнуть досягнути значного розвитку та лідерства на ринку [2].

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Магістр» є дослідження застосунків для автоматизації бізнес процесів, проведення аналізу їх технічних аспектів та, відповідно до одержаних даних, розробка власного рішення з розширеними інтеграційними можливостями. Для досягнення поставленої мети потрібно виконати ряд важливих завдань, зокрема:

– провести дослідження поняття бізнес процес, вказати його види та історію виникнення (Тененський М. В.);

– розглянути та описати основні методи вдосконалення та охарактеризувати вплив бізнес процесів (Тененський М. В.);

– дослідити сферу управління бізнес процесами, вказати життєвий цикл BPM (Галюк М. В.)

– дослідити та проаналізувати сучасні застосунки для управління та автоматизації бізнес процесів (Тененський М. В.);

– дослідити вимоги та архітектурні підходи реалізації сучасних застосунків для автоматизації бізнес процесів (Тененський М. В.);

– проаналізувати інтеграційні можливості розроблюваного застосунків для автоматизації бізнес процесів (Тененський М. В.);

– сформувати та аргументувати вимоги до власного розроблюваного застосунку для автоматизації бізнес процесів (Галюк М. В.);

– провести аналіз технологій та засобів, використаних для розробки власного застосунку для автоматизації бізнес процесів (Тененський М. В.);

– дослідити фреймворк UAF в контексті розробки власного рішення для автоматизації бізнес процесів (Галюк М. В.)

– спроектувати структури сутностей в БД та розробити серверну частину власного рішення для автоматизації бізнес процесів (Тененський М. В.);

– спроектувати та розробити клієнтський інтерфейс власного рішення для автоматизації бізнес процесів (Галюк М. В.);

– дослідити та зазначити перспективні ідеї щодо покращення розроблюваного застосунку для автоматизації бізнес процесів (Галюк М. В.);

Об’єкт дослідження. Об’єктом дослідження є оптимізація різногалузевих задач шляхом використання застосунків для автоматизації бізнес процесів

Предмет дослідження. Предметом дослідження є сучасні застосунки для автоматизації бізнес процесів із розширеними інтеграційними можливостями.

Наукова новизна одержаних результатів. Наукова новизна одержаних результатів полягає в створенні інноваційного MVP рішення в сфері автоматизації бізнес процесів, якісно відмінного від досліджуваних застосунків та систем, а також в оптимізації процесу розробки завдяки використанню UAF.

Практичне значення одержаних результатів. Дослідження сучасних застосунків для автоматизації бізнес процесів дозволяє визначити їх

найважливіші аспекти, що є сприятливим фактором в стратегії вибору та впровадження таких застосунків та систем.

Апробація результатів магістерської роботи. Основні результати проведених досліджень обговорювались на декількох наукових конференціях, зокрема:

– VI Міжнародна студентська науково-технічна конференція «Природничі та гуманітарні науки. Актуальні питання», 2023.

– XI Науково-технічна конференція «Інформаційні моделі, системи та технології», 2023.

– XII Міжнародна науково-практична конференція молодих учених та студентів «Актуальні задачі сучасних технологій», 2023.

Текст наукових тез із проведеними дослідженням подано в додатку А.

Публікації. Основні результати кваліфікаційної роботи опубліковано в шести працях, поданих на наукові конференції (див. додаток А).

Структура й обсяг кваліфікаційної роботи. Дана кваліфікаційна робота освітнього рівня «Магістр» складається зі вступу, чотирьох розділів, висновків, списку літературних джерел із 97 найменувань та 5 додатків. Загальний обсяг кваліфікаційної роботи складає 108 сторінки, з них 69 сторінки основного тексту, який містить 38 рисунків.

1 ДОСЛІДЖЕННЯ БІЗНЕС ПРОЦЕСУ

Відповідно до наведеної аргументації щодо актуальності обраної теми кваліфікаційної роботи освітнього рівня «Магістр», бізнес процеси є невід'ємною складовою діяльності будь якої сучасної компанії чи використовуваного нею ПЗ. В цьому розділі розглянуто визначення поняття бізнес процес, наведено основні його види, подано вичерпну класифікацію бізнес процесів та вказано на необхідність проведення їх автоматизації.

1.1 Основні засади, характеристики та види бізнес процесів

Існує безліч визначень до поняття бізнес процес. Проте, найбільш точним буде наступне – це певна сукупність взаємопов'язаних завдань чи видів діяльності, виконуваних конкретною особою, групою зацікавлених сторін чи обладнанням, що виробляють продукт або послугу для одного конкретного або багатьох клієнтів [3]. Також зустрічається думка, що бізнес процес – це сукупність послідовних дій та задач, які, будучи успішно виконаними, сприяють досягненню заздалегідь поставленої мети. За стандартом ISO 9000-2001 бізнес процес – це набір взаємопов'язаних заходів, що перетворюють входи на виходи. Для скорочення поняття бізнес процес використовують абревіатуру БП.

Бізнес процеси можуть бути як очевидними, так і прихованими для кінцевого клієнта чи замовника. Зазвичай, бізнес процеси моделюються у вигляді послідовності дій з кроками прийняття рішень, або у вигляді матриці дій [4]. Поруч із моделюванням використовують так званий метод симуляції бізнес процесів. Він дозволяє ефективно який виявляти потенційні причинно-наслідкові зв'язки та значною мірою полегшує оптимізацію бізнес процесів [5]. Серед основних характеристик бізнес процесів прийнято виділяти наступне:

- Повторюваність БП.
- Вартість та тривалість виконання БП.
- Узгодженість кроків БП.
- Якість кінцевого продукту чи послуги.

Саме ж моделювання, зазвичай, здійснюється із використанням BPMN – системи умовних позначень та описів бізнес процесів (див. рис. 1.1). Справедливим буде твердження, що відсутність проведеного моделювання унеможливлює подальшу оптимізацію та автоматизацію бізнес процесу [1].

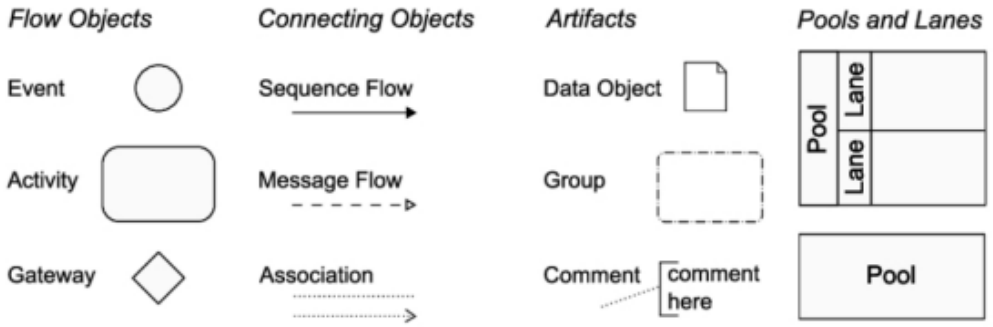


Рис. 1.1 – Приклад основних елементів BPMN [6]

Відповідно до вищезазначеної інформації розуміємо, що бізнес процеси складаються з чотирьох компонентів, детальний опис яких подано в таблиці 1.1. Так, в основі будь-якого бізнес процесу лежить певна мета чи задача, виконання якої постійно контролюється і фактично є досягненням поставленої бізнес цілі. Варто розуміти і те, що часто закінчення одного бізнес процесу є тригером для початку іншого спорідненого процесу [7].

Таблиця 1.1 – Складові бізнес процесів

Назва складової	**Характеристика складової**
Вхід	До даної складової можна віднести тригер чи подію, яка ініціює виконання бізнес процесу.
Вихід	Дана складова є результатом виконання бізнес процесу. Це може бути як кінцева продукція чи послуга, так і певна подією, яка запустить виконання спорідненого бізнес процесу.
Механізми виконання	Інструменти, в тому числі люди, ПЗ чи машини, які виконують дії відповідно до входу і виходу бізнес процесу.
Контрольні точки	Обов’язкові обмеження, інструкції та вимоги, які визначають дії над вхідними даними бізнес процесу.

Відповідно до інформації із джерела [8], загальновживаною практикою є наявність так званого власника бізнес-процесу – особи, яка відповідальна за

проектування та виконання кроків БП. Існування такої ролі в наш продиктовано умовами, необхідними для успішного завершення того чи іншого бізнес процесу, а також дотриманню чітких правил його перебігу на будь-якому етапі виконання. Іноді власник бізнес-процесу є тією ж особою, хто його виконує.

В свою чергу, найчастіше бізнес процеси поділяють за ознакою формування результату. Це дозволяє краще зрозуміти специфіку їх застосування. Окрім того, в залежності від типу, можуть використовуватись різні методи та інструменти автоматизації процесів з метою покращення кінцевого продукту, який вони створюють. Загалом прийнято виділяти три види бізнес процесів (див. табл. 1.2).

Таблиця 1.2 – Поділ бізнес процесів за ознакою формування результатів

Вид процесу	**Характеристика виду процесу**
Основний процес	Даний вид, як зрозуміло з назви, є критичним для функціонування бізнесу, адже саме він безпосередньо генерує кінцевий продукт чи послугу.
Процес підтримки	Даний тип процесів є досить важливим в контексті підтримки ефективного та безпомилкового функціонування основних бізнес процесів.
Процес управління	Бізнес процеси цього виду зосереджені на внутрішньому та зовнішньому моніторингу бізнес функцій, аналізу можливостей до оптимізації та вдосконалення усіх наявних процесів.

Інколи процеси підтримки розділяють на два підвиди: власне процеси підтримки та забезпечувальні процеси. Популярним є поділ бізнес процесів й за ознаками, відмінними від наведених в таблиці 1.2 [8]. Розглянемо декілька прикладів:

– За характером кінцевого продукту (поділяють на адміністративні та виробничі).

– За відношенням до клієнтів підприємства (поділяють на внутрішні та зовнішні).

– За рівнем деталізації (поділяють на бізнес процеси верхнього рівня, а також детальні та елементарні БП).

– За ступенем зв’язності (поділяють на локальні та інтеграційні).

– За напрямом руху (поділяють на горизонтальні та вертикальні).

– За локалізацією (поділяють на локальні та крос-організаційні)

Така класифікація бізнес процесів є необхідною, адже без неї побудувати складну послідовність взаємопов’язаних процесів, забезпечуючи при цьому ефективність їх функціонування, було б просто неможливо [8].

Так чи інакше, не рідко бізнес процеси є настільки складними та заплутаними, що варто виконати їх розбиття на достатню кількість структурних підпроцесів, яким притаманними будуть як загальні, так і певні унікальні властивості. Такий крок за дійсної необхідності позитивно впливає на досягненні успішного кінцевого результат [9]. Проте варто зважати і на те, що така декомпозиція, проведена без дотримання чітких норм та без наявності відповідного програмного забезпечення, на практиці може викликати колізії, що є очевидним негативним фактором.

Відповідно до наведеної вище інформації розуміємо, що на основі декомпозиції бізнес процесу та аналізу його окремих задач бізнес аналітики можуть приймати раціональні управлінські рішення. Робиться це на основі певних показників, відхилення від яких потребує детального огляду [5]. Такі показники або визначаються для кожного бізнес процес окремо, в залежності від його специфіки, або використовують більш узагальнені показники, серед яких: часові, фінансові та технічні. Детальніше про це описано в розділі 1.3.

Бізнес процеси часто зображають у вигляді блок-схем, які показують його контрольні точки, а також послідовність дій та задач, що виконуються бізнес процесом [10]. Бізнес процеси, відповідно до їх блок-схем, поділяють на три умовні типи:

– Послідовні бізнес процеси: мають чітко описані початкову та кінцеві точки.

– Бізнес процеси, керовані статусом: не мають чітко визначених початкових чи кінцевих точок, а завершення їх виконання може настати на будь-якому етапі, в залежності від змін в БП [10].

– Паралельні бізнес процеси: виконують декілька дій одночасно, при чому вони часто мають бути завершеними до початку наступного етапу.

Існує багато концептів опису термінів, пов’язаних з управлінням бізнесом, при чому бізнес процес – це лише одне із них [10]. Прийнято виділяти також такі речі, як бізнес процедура та бізнес функція. Бізнес процедурою вважається чітко обумовлений спосіб виконання бізнес процесу, в якому детально специфікації виконання БП та відповідальних за кожний окремий етап [10]. В свою чергу, бізнес функція – це певна організаційна одиниця в середині підприємства, яка має набір власних обов’язків та завдань, які вона повинна виконувати для підтримки бізнесу та досягненню його цілей [10]. Хоча ці всі терміни описують концепції, необхідні для функціонування підприємств та бізнесу, все ж варто розуміти, що вони не є взаємозамінними.

1.2 Історія виникнення та економічний вплив на бізнес процес

Відповідно до інформації із джерела [11], в 1776 році шотландський економіст Адам Сміт дав перше визначення бізнес процесам, запровадивши їх використання на підприємстві з виробництва шпильок. За його порадами процес виготовлення було поділено на 18 окремих операцій із залученням відповідної кількості спеціалізованих працівників. Такий розподіл робіт дозволив значно підвищити загальну ефективність підприємства та призвів до збільшення продуктивності на неймовірних 24000%. Тобто, із впровадженням бізнес процесу, робітники почали виготовляти в 240 разів більше шпильок [12].

Ще одним яскравим прикладом застосування бізнес процесу можна вважати конвеєрну модель виробництва автомобіля «Ford Model T», започатковану Генрі Фордом в 1907 році. Було вирішено, що краще навчити ремісників виконувати одну з 84 простих та повторюваних робіт, ніж щоб один робітник самостійно збирав авто [13]. Таким чином було започатковано одне з найбільш ефективних виробництв на той момент, а час збирання однієї машини скоротився з 12 до 2.5 годин [12].

В якості ще одного прикладу успішного використання бізнес процесів можна назвати введення так званого «чек-листа» Атулом Гаванде в лікарні імені Джона Хопкінса [11]. Це рішення стало найбільш ефективним серед усіх

запропонованих. Так, з введенням даного бізнес процесу, 78% працівників медичного персоналу помітили, що використання «чек-листу» запобігає виникненню багатьох помилок [11].

З плином часу бізнес процеси ускладнювалися, а з появою інтернету та розвитком IT індустрії можливим стало використання спеціалізованого ПЗ для їх покращення та управління ними. Так, справедливим буде твердження, що бізнес процесам варто постійно проходити шлях від стабільності до вдосконалення, що включає в себе автоматизацію та реінжиніринг (детальніше про ці поняття сказано в розділі 1.3) [14]. Підприємства, які зважають на ці обставини, здатні випереджувати своїх конкурентів, йдучи в ногу із сучасними світовими тенденціями та досягаючи кращих результатів при зменшенні витрат на функціонування чи виробництво товарів та послуг [14].

Фактично чим більшою є компанія чи підприємство, – тим більше їх діяльність та залежать від використання бізнес процесів, зокрема автоматизованих [2]. Щодо економічного впливу бізнес процесів та BPMS, відповідно до джерела [12], можна виділити наступні фактори:

– Зменшення витрат: оптимізовані бізнес процеси зменшують кількість зайвих дій чи персоналу, що дозволяє зменшити витрати та зробити бізнес більш рентабельним. Окрім того, оптимізовані бізнес процеси часто приносять прямі фінансові вигоди підприємствам.

– Підвищення продуктивності: із автоматизацією бізнес процесів відбувається перехід на більш ефективне використання наявних ресурсів, що позитивно позначається на продуктивності [15].

– Забезпечення конкурентоспроможності: налагоджені бізнес процеси надають можливість до швидкого реагування на ринкові зміни, полегшуючи тим самим впровадження нових чи покращення наявних продуктів та послуг, а також скорочують час виходу на ринок [16].

– Зручність до аналізу та контроль підприємства: моніторинг життєвого циклу бізнес процесів надає змогу вчасно виявляти та реагувати на ризики чи проблеми [2].

– Можливість до масштабування: зростаючи, підприємство потребує проведення певної адаптації та масштабування свої процесів. Розвиток та використання БП в цьому контексті забезпечує збереження ефективності підприємства в умовах зростаючих обсягів та складнощів створення продукції чи надання послуг [16].

– Управління інформацією та даними: ефективні бізнес процеси здатні забезпечувати основу для управління даними та інформацією з метою прийняття кращих рішень [16].

– Продуктивність працівників: використання чітких бізнес процесів допомагає працівникам краще зрозуміти свою роль та обов'язки, що призводить до їх більшої залученості в роботу та продуктивності праці [16].

Відсутність налагоджених бізнес процесів, окрім очевидних фінансових та оперативних ризиків, може призвести до трагічних наслідків. Так, в 1865 році, через відсутність БП, зіштовхнулись два потяги [10]. Ця трагедія стала причиною того, що залізниця Північної Пенсильванії відкоригувала свої процеси таким чином, щоб два потяги ніколи більше не рухались в двох різних напрямках однією і тією ж самою колією.

1.3 Оптимізація, реінжиніринг та автоматизація бізнес процесів

Як вже було зазначено в розділі 1.2, удосконалення бізнес процесів фактично є невід'ємною частиною їх впровадження та використання. Удосконалення полягає в реформації бізнес процесу з метою підняття його ефективності. Зазвичай ця операція відбувається із використанням спеціалізованого ПЗ чи за рахунок розробки власного рішення на основі потрібних технологій. Будь яка зміна бізнес процесу потребує наявності відповідних ресурсів, інструментів та процедур.

Також, відповідно до інформації з джерела [17], завжди рекомендується зважати на ряд показників, за якими можна провести експрес оцінку бізнес процесу, розрахунок коефіцієнтів для якої подано в додатку Б. Після розрахування цих показників обчислюється загальний інтегральний показник. У

випадку, якщо його значення є меншим за 2 – такий бізнес процес вважається ефективним, інакше бізнес процес потребуватиме певних змін [17]. Ба більше, відповідно до інформації з джерела [11], погані чи застарілі бізнес процеси є причиною неефективності компаній в понад 44% випадків. Забігаючи на перед зазначимо, що для покращення більшості з цих показників варто поетапно впроваджувати цифровізацію та автоматизацію БП.

Серед основних методів удосконалення прийнято виділяти власне оптимізацію бізнес процесів, а також їх реінжиніринг та автоматизацію [5]. Розглянемо ці поняття дещо детальніше.

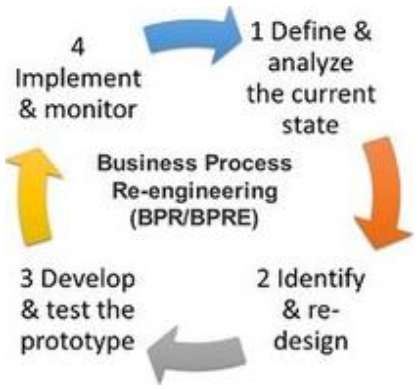


Рис. 1.2 – Життєвий цикл BPR [18]

Реінжиніринг бізнес процесів (англ. Business process re-engineering, скорочено BPR) – це стратегія вдосконалення БП, яка полягає в аналізі існуючих та проектуванні нових бізнес процесів. Її головною метою є фундаментальне переосмислення виконуваних робіт, а також їх етапів, задля покращення часу виконання та скорочення операційних витрат [15].

До BPR варто вдаватись лише за крайньої необхідності, коли без радикальної зміни бізнес процесів неможливо обійтись. Відповідно до рисунку 1.2, реінжиніринг бізнес процесів є доволі комплексною та ресурсозатратною задачею, а його проведення значною мірою залежить від наявних технологічних інновацій, в тому числі розробок IT галузі [18]. Для успішного проведення реінжинірингу бізнес процесів, відповідно до джерела [11], варто звернути увагу на такі аспекти:

– Визначаючи процеси, що підлягають реінжинірингу, варто вибрати ті, які є найважливішими та викликають найбільше конфліктів з бізнес цілями чи ті, що неможливо оптимізувати звичними методами.

– Перш ніж почати проводити реінжиніринг, варто зробити дослідження метрик існуючих бізнес процесів, щоб не повторити помилок під час проектування нових.

– Завжди варто надавати перевагу технологічним рішенням, здатним заощадити ваші ресурсі та підвищити ефективність бізнес процесів.

– Варто розглядати бізнес процес в ролі прототипу. Фактично це означає, що його можна використовувати як тренувальний шаблон доти, доки не буде досягнуто результату, який найкраще би підходив бізнес цілям.

Оптимізація бізнес процесів (англ. Business process optimization, скорочено BPO) – це широкий спектр заходів, спрямованих на досягнення позитивних в якісних та кількісних показниках бізнес процесу [19]. Оптимізація завжди сприяє підвищенню загальної результативності та адаптивності бізнес процесів в конкурентних умовах сьогодення. Також BPO часто розглядається як вагомий аспект розвитку, який спрямований на покращення кінцевого продукту чи послуги [9].

Автоматизація бізнес процесів (англ. Business process automation, скорочено BPA) – процес, головна мета якого полягає в проведенні автоматизації складних, наскрізних БП за допомогою сучасних технологічних засобів, часто відмінних від тих, що використовуються для BPO. Саме ж поняття автоматизації означає застосування технологій та систем для виконання певних завдань, операцій чи процесів без необхідності або мінімізації втручання людини. Інакше кажучи, бізнес процес, виконання якого відбувалось ручними методами, після проведеної BPA буде здійснюватися за допомогою різноманітних автоматизованих систем, застосунків, засобів тощо [20].

Так, автоматизація бізнес-процесів корисна для підприємств будь-якого розміру. Вона допомагає перейти від повільних та громіздких ручних БП до безперебійних цифрових БП, дозволяючи підприємствам еволюціонувати в цифровий бізнес [21]. Ось чому BPA інколи називають цифровою

трансформацією – вона здатна значною мірою спросити перебігу процесу, підвищуючи при цьому якість кінцевих результатів [1].

Зазвичай, автоматизація БП дозволяє зменшити витрати на його виконання за рахунок інтеграцій із сучасними технологіями, при цьому ефективно розподіляючи бюджети, матеріли, ресурси тощо, тим самим дозволяючи сконцентрувати увагу на основних операціях, що сприяють зростанню бізнесу та задоволенню його клієнтів [22].

Більше того, відповідно до проведених досліджень, автоматизація БП забезпечує точність і своєчасність виконання тих чи інших дій та зменшує ризик виникнення помилок, які негативно впливають на досвід клієнтів [21]. Етапи виконання автоматизації бізнес процесів подано на рисунку 1.3. Відповідно до нього справедливим є твердження, що перевага має надаватись швидше проведенню BPA, ніж BPR.

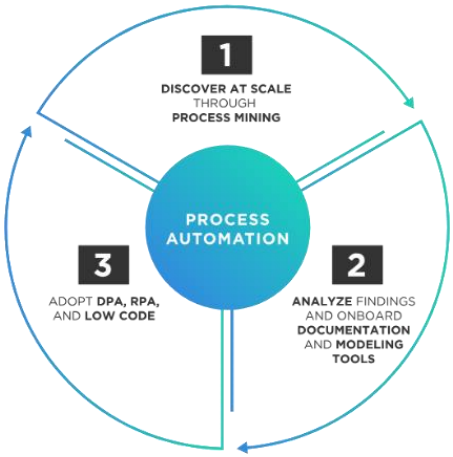


Рис. 1.3 – Життєвий цикл BPA [23]

В загальному чим менше частка автоматизованих процесів, тим гіршою є ефективність підприємства. Відповідно до інформації із джерела [24], виділяють наступні типові бізнес процеси, які найбільш часто прийнято автоматизувати:

– Процеси управління продажами.
– Комунікативні процеси з клієнтами.
– Балансування електроенергії.
– Адміністрування та функціонування веб-ресурсів.
– Процеси, пов’язані з бухгалтерським обліком та фінансами.

– Документообіг.

Одним із способів автоматизації бізнес процесів можна вважати розробку власного або придбання готового ПЗ, яке дозволяє спроектувати та виконувати певну послідовності наперед визначених дій – так званих підпроцесів, або кроків БП [25]. Хоча, як показує практика, не рідко такі застосунки не можуть повністю покрити виконання всіх етапів бізнес процесів. Ось чому, відповідно до інформації із джерела [11], виник ще один підхід BPA, який полягає в залученні як спеціалізованого ПЗ, так і виконання певних дій людиною. Він нівелює вищезгадану проблему, але, в свою чергу, може виявитись більш складним та ресурсозатратним.

Проте автоматизація бізнес процесів не позбавлена певних недоліків. Серед них можна зазначити складні для розуміння алгоритми, часові та фінансові затрати, які можуть значно вирости в разі відсутність досвіду та навичок впровадження ПЗ для автоматизації БП [23].

Так чи інакше, варто розуміти, що не для кожного бізнес процесу BPA буде доцільним рішенням. Найкраще для цього підходять часозатратні та повторювані БП. Також періодично слід проводити аналіз автоматизованих процесів, досліджувати питання безпеки, ефективності та доцільності їх функціонування [23]. Окрім того, не варто забувати, що BPA є складовою частиною стратегії управління БП – так званого BPM [21]. І хоча ці поняття часто плутають або ототожнюють, все ж їх не варто їх розрізняти. Спільним є те, що BPA та BPM покликані допомогти в удосконаленні бізнес процесів. Проте BPM, на відміну від BPA, може не включати автоматизацію БП, оскільки він використовує найрізноманітніші методи для аналізу, зміни чи оптимізації БП.

1.4 Управління бізнес процесами

Управління бізнес процесами – це сукупне визначення для узагальнених задач, пов’язаних з аналізом, моніторингом, покращенням, оптимізацією та автоматизацією бізнес-процесів. Відповідно до цього, будь-яку комбінацію методів, використаних для управління БД, можна віднести до BPM [2]. Оскільки

самі бізнес процеси можуть як структурованими, так і неструктурованими або змінними чи повторюваними, BPM часто застосовується в поєднання з допоміжними технологіями.

Рис. 1.4 – Життєвий цикл BPM [26]

Одна з головних задач BPM – це зміна БП таким чином, щоб привести їх до такого стану, який збігався би із конкретними цілями організації. Інакше кажучи, кожен бізнес процес повинен бути налаштований таким чином, щоб результати виконання процесу приводили до досягнення поставлених бізнес цілей [8]. Також, відповідно до джерела [26], виділяють наступні цілі BPM:

– Формалізація та аналіз бізнес процесів.

– Виконання оптимізації бізнес процесів шляхом проведення циклічного аналізу.

– Скорочення терміну розробки та введення нових бізнес процесів.

– Скорочення терміну оновлення чи реінжинірингу наявних бізнес процесів.

– Зменшення залежності від стандартних дій, що виконуються персоналом.

– Максимізація розробки та введення автоматизованих бізнес процесів.

– Скорочення загальної вартості управління бізнес процесами.

Розглянемо більше детально кожний з кроків життєвого циклу BPM, наведеного на рисунку 1.4. Так, розробка дизайну БП охоплює як вдосконалення існуючих процесів, так і можливість до проектування нових. Основну увагу на

даному етапі рекомендується приділяти власне розробці більш ефективного дизайну з урахуванням попереднього досвіду та усіх факторів, що впливають або матимуть вплив на перебіг бізнес процесу. Запропоноване вдосконалення може стосуватись як процесів типу “людина-людина”, так і “людина-система” або “система-система” [28].

Моделювання, в свою чергу, використовується як для відображення поточного, так і для проектування можливого стану бізнес процесу під впливом довільних факторів [26]. Описувати бізнес процес можна в різних стандартах, серед яких найбільш популярним є згаданий раніше BPMN. Також на цьому етапі зазначаються найбільш важливі показники БП. Як результат - визначаються слабкі та сильні сторони розробленого на попередньому кроці дизайну.

Виконання означає впровадження змодельованого на минулих етапах бізнес процесу. Воно може відбуватись як в автоматичному, так і в ручному режимах. Ручні БП керуються безпосередньо людино, а автоматичні – спеціалізованим ПЗ [19].

Моніторинг покликаний збирати та відображати інформацію про стан бізнес-процесів та різноманітні статистичні дані: тривалість циклу, загальна продуктивність, кількість помилок тощо. Це може бути корисним як при роботі з клієнтом, так і для наступних ітерацій покращення бізнес-процесу [15].

Оптимізація є останнім крок життєвого циклу BPM, під час якого відбувається обробка отриманої з попередніх етапів інформації про бізнес процес, виявлення потенційних вад процесу, а також можливостей до його подальшого покращення [26]. Вдало проведена оптимізація може допомогти у виявленні потенційних ризиків, ліквідації неефективних етапів процесу, впровадженні якісних змін у функціонування БП. Також вона дозволяє краще зрозуміти аспекти, які потребують автоматизації [15].

Може скластись помилкове враження, що BPM – це те саме, що і управління проектами, але це не так. Ключова відмінність між ними полягає в повторюваності і передбачуваності [8]. Так, в управлінні бізнес процесами послідовність робіт часто може відрізнятися в залежності від ринкових умов, бізнес правил тощо. Проте передбачуваність – це головний аспект BPM: яким

розгалуженим не був би бізнес процес, умови, за якими відбувається його виконання, повинні бути зрозумілими з плином часу [8].

Варто розуміти, що вдосконалення одного окремого етапу бізнес процесу не можна вважати повноцінним BPM. Займаючись управлінням процесами варто розуміти, що кожне вдосконалення чи управлінська дія повинна розглядатись в контексті усього бізнес процесу [29]. Особливістю BPM є погляд на бізнес процеси в якості активів організації, управління і розвиток яких дозволить надавати клієнтам більш якісні продукти чи послуги. Ось чому BPM часто плутають з іншими методологіями, спрямованими на вдосконалення та управління БП.

Також BPM інколи ототожнюють з BPMS, адже управління бізнес процесами часто розглядають з точки зору людської діяльності, або з точки розу застосування спеціалізованих технологічних засобів. Відповідно до інформації з джерела [30], BPM – дисципліна, якою займаються люди, а BPMS – набір технологічних інструментів або корпоративні системи, розроблені для BPM спеціалістів з метою допомогти їм в досягненні поставлених цілей, зокрема автоматизації бізнес процесів, яка є одним із аспектів BPM.

1.5 Висновки до першого розділу

В першому розділі даної кваліфікаційної роботи було розглянуто поняття бізнес процесу та визначено актуальність, а також необхідність щодо впровадження бізнес процесів. Так, найбільш точним буде визначення, що бізнес процес – це набір пов’язаних кроків чи задач, виконання яких створює послугу чи продукт або розпочинає виконання нового бізнес процесу.

Варто зазначити, що існує декілька видів класифікації бізнес процесів. Розуміємо, що це є необхідністю, адже без вказаних класифікацій неможливо було б повноцінно досліджувати та покращувати бізнес процеси в життєвому циклі BPM.

Власне щодо удосконалення бізнес процесів, можна зробити висновок, що це цілком логічна та послідовна частина їх застосування. Виділяють основні три

методи їх удосконалення: реінжиніринг, оптимізацію та автоматизацію. Усі вони мають на меті відчутно покращити ефективність та прозорість бізнес процесів. Найбільш актуальним серед них є автоматизація, яка дозволяє перейти від повільних та громіздких ручних БП до цифрових, безперебійно повторюваних бізнес процесів.

Також варто зробити висновок, що впровадження та використання бізнес процесів є невід'ємними атрибутами будь-якої сучасної успішної компанії, бізнесу та підприємства. Так, відповідно до наведених в даному розділі історичних прикладів розуміємо, що впровадження БП здатне відчутно пришвидшити виконання завдань підприємства при тій самій або меншій кількості використаних ресурсів.

2 ДОСЛІДЖЕННЯ ЗАСТОСУНКІВ ДЛЯ УПРАВЛІННЯ ТА АВТОМАТИЗАЦІЇ БІЗНЕС ПРОЦЕСІВ

Відповідно до наведеної в першому розділі даної кваліфікаційної роботи інформації, управління та автоматизація бізнес процесів є невід'ємною складовою розвитку бізнесу. В цьому розділі розглянуто ряд сучасних рішень проведено їх порівняння, вказано на їх технологічні аспекти та особливості.

2.1 Типи застосунків та система для управління автоматизації бізнес процесів

Існує безліч різновидів ПЗ, призначеного для управління бізнес процесами. Усі вони покликані пришвидшити та полегшити процес їх автоматизації, а також позитивно впливають на виявлення потенційних ризиків та нівелювання їх впливу [7]. Крім того, сучасні інструменти дозволяють краще зрозуміти заплутані бізнес процеси, швидко знайти усі їх сильні та слабкі сторони, побудувати на їх основі кращу бізнес модель тощо [31]. Їх різноманіття вражає – від роботизованої автоматизації процесів до так званих low code платформ. Розглянемо даний поділ більш детально.

Системи роботизованої автоматизації бізнес процесів (англ. Robotic process automation, скорочено RPA) – революційна технологія, покликана автоматизувати повторювані завдання, імітуючи при цьому людські дії. Завдяки програмному забезпеченню RPA, бізнес має змогу оптимізувати свої операції, зменшити кількість помилок, що допускаються при ручній діяльності, підвищити загальну ефективність тощо [22]. Визначною особливістю RPA систем є те, що їх можна легко впровадити кінцевим користувачам, без потреби залучення спеціалістів із глибокими навичками розробки. RPA особливо ефективна для автоматизації бізнес процесів, які потребують введення чи вилучення даних, створення звітів та базуються на чітких правилах. Замість того, щоб витрачати години на ці рутинні завдання, RPA системи можуть впоратися з ними за відносно короткий проміжок часу. Із залученням RPA до автоматизації

бізнес процесі можна успішно скоротити витрати на виконання бізнес процесів, пришвидшити час обробки даних, зменшити потенційну кількість помилок [22].

Платформи з низьким рівнем написання коду (так звані low code рішення) – це ПЗ, яке надає можливість користувачам автоматизувати бізнес процеси з мінімальними знаннями програмування або взагалі без них. З відси випливає їх ключова перевага – а саме здатність до прискорення часу впровадження та інтеграції. Завдяки попередньо підготованим компонентам і функціям, користувачі можуть швидко створювати та налаштовувати бізнес процеси. Це не лише економить час, але й дозволяє компаніям швидко реагувати на мінливі вимоги ринку [22]. Ще однією значною перевагою платформ з низьким рівнем написанням кодує в контексті автоматизації бізнес процесів є їх масштабованість. Така гнучкість дозволяє бізнесу не відставати від мінливих очікувань клієнтів та ринкових тенденцій.

Цифрова автоматизація процесів (англ. Digital process automation, скорочено DPA) – технологія, яка дозволяє значно скоротити або повністю уникнути ручних задач в бізнес процесі, тим самим усуваючи ризики людського фактору. Ключова особливість DPA полягає в здатності ефективно автоматизувати великі та розгалужені бізнес процеси шляхом безперешкодного з’єднання різних етапів їх виконання та автоматизації відповідних ділянок [22]. Впровадження цифрової автоматизації дозволяє значно збільшити ефективність бізнесу за рахунок оптимізації бізнес процесів та усуненню вузьких місць. Як наслідок – відбувається пришвидшення виконання БП та збільшується його надійність, що позитивно відзначається на економії коштів. Так, використовуючи DPA, організації можуть випереджати конкурентів і стимулювати власне зростання в сучасному цифровому світі.

BPM застосунки (також зустрічається використання терміну BPMS) – потужний інструмент, який покликаний оптимізувати бізнес процеси, підвищувати їх ефективність та рівень автоматизації. Однією з ключових переваг BPMS, як і в DPA, можна вважати здатність до автоматизації великих та розгалужених БП. Проте BPMS забезпечують більш високий рівень усунення ручних задач, надають аналітичні дані щодо перебігу бізнес процесів в режимі

реального часу [32]. Це дозволяє швидше та ефективніше виявляти ризики, впроваджувати покращення та адаптувати бізнес процеси до мінливих умов.

Також BPM застосунки дозволяють легко розширити автоматизовані бізнес процеси новими етапами, не порушуючи при цьому роботу існуючого функціоналу [33]. Така гнучкість та адаптивність надають значної конкурентної переваги тим підприємствам та бізнесу, які впровадили та використовують BPMS. Варто розуміти і те, що існує декілька типів BPM застосунків за призначенням, інформацію про які подано в таблиці 2.1.

Таблиця 2.1 – Типи BPM застосунків за призначенням

Тип	**Характеристика типу**
Документо-орієнтовані	Автоматизовані рішення, що створюють документи на основі даних, отриманих від кількох людей чи систем. Найчастіше даний тип BPMS використовують для генерації рахунків, умов використання, контрактів тощо.
Людино-орієнтовані	Даний тип забезпечує охоплення бізнес процесів, які потребують неодмінного контролю зі сторони працівників. Найчастіше дане рішення може використовуватись для обслуговування клієнтів, управління скаргами тощо.
Інтеграційно-орієнтовані	Таке рішення інтегрується в наявні системи для передачі між ними даних про бізнес процеси. Це зменшує кількість помилок, підвищує ефективність управління та швидкість виконання БП.

Усі ці системи нерідко взаємодіють між собою, утворюючи при цьому комплексну платформу для підтримки всіх аспектів бізнес процесу. Однак особливу увагу варто приділити BPMS, адже, як було сказано раніше, вони дозволяють не лише автоматизувати бізнес процеси, але й постійно моніторити їх виконання, що полегшує подальше вдосконалювати БП. Дослідження сучасних BPM застосунків наведено в наступному розділі 2.3.

2.2 Економічний вплив BPM застосунків

BPM застосунки є однією з інноваційних технологій, які широко використовуються в сфері управління бізнес-процесами. Багаточисленні

дослідження неодноразово відмічали позитивний ефект від інтеграції BPM застосунків та систем [1]. Наприклад, вони часто застосовується для оптимізації бізнес процесів, результатом якої, зазвичай, є зменшений час виконання відповідних етапів, а також підвищена якість кінцевого продукту чи послуг, що виробляються даним БП [34]. Не варто недооцінювати вплив BPM застосунків на фінансові показники підприємства. Як показує практика, використання такого ПЗ після проведення вдалої інтеграції, особливо в питаннях автоматизації, позитивно впливає як на прибуток, так і дозволяє зменшити витрати на виконання бізнес процесів [1].

Також BPMS надають бізнес аналітикам будь-якого професійного рівня зручний та ефективний інструментарій для прийняття рішень, аналізу ризиків, проектування бізнес процесів довільної складності з можливістю побудови розгалужень з урахуванням додаткових вимог чи правил. Це допомагає підготувати чіткі регламенти, метою яких можна вважати як проведення уніфікації всієї компанії, так і окремих її підрозділів або використовуваного нею програмного забезпечення [2].

Особливо корисними можна вважати ті BPM застосунки чи системи, які впроваджують BPMN – умовну система позначень для проектування БП. Перевагою BPMN є те, що вона зрозуміла як професійним аналітикам, так і початківцям в даній сфері. З цього випливає можливість використання BPM застосунків як великими, так і малими компаніями, що, в свою чергу, дозволяє їм заощаджувати час проектування бізнес процесів та зосереджуватись на інших важливих задачах.

Відповідно до джерела [1], в якості прикладу успішного використання та інтеграції BPMS можна навести відому американську компанію FedEx. Так, завдяки BPM застосунку, в неї з’явилась можливість автоматизувати планування логістичних маршрутів постачання та зробити цей процес більш ефективним. Дане рішення зменшило не тільки час доставки товарів замовникам, але й знизило витрати на послуги логістичних спеціалістів та пальне.

Не варто забувати і про раніше загадану компанію Toyota, яка використовує BPMS для контролю за виробничими процесами на своїх

фабриках. Це дозволяє більш ефективно управляти запасами сировини та виробничими процесами. Так, показники компанії щодо вироблення авто щороку або залишаються на доволі високому рівні, або зростають, при чому зберігається висока якість кінцевого продукту [1].

Докладний аналіз використання BPMS надає стаття Grand View Research [32], в якій стверджується, що світовий ринок застосунків для управління та автоматизації бізнес процесів оцінювався в $9.54 млрд, а в період між 2023 та 2030 роками прогнозується середньорічний приріст на рівні 33%, досягнувши при цьому рівня в $86.63 млрд. Таке стрімке зростання ринку пояснюється дедалі більшою потребою в автоматизації та цифровізації корпоративних бізнес процесів шляхом впровадження нових технологічних рішень в дану сферу.

Певний відбиток наклала і пандемія COVID-19, яка стимулювала прискорення в процесі інтеграції BPM застосунків в сферу управління БП. Очікується, що оптимізація бізнес процесів, підвищена економічна ефективність та доцільне використання ресурсів сприятимуть постійному розвитку та зростанню конкуренції на світових ринках [**Помилка! Джерело посилання не знайдено.**]. Відповідно до цього, виникає гостра необхідність до технологічної трансформації, що сприятиме подальшому розвитку та поширенню BPMS рішень. Варто також розуміти, що ця криза почала процес відсіювання тих компаній, які виявились неспроможними до вдосконалення та залучення новітніх технологій з метою автоматизації та оптимізації власних БП [**Помилка! Джерело посилання не знайдено.**].

Окрім того, відповідно до статті з ресурсу Solutions Review [36], BPM застосунки є причиною підняття рентабельності бізнесу на понад 15% в 80% випадків їх застосування. Також автором стверджується, що так чи інакше, BPMS в середньому збільшують продуктивність виконання задач на 30-50%, при чому найбільш популярними є ті рішення, які мають декілька способів до інтеграції.

Очікується, що оптимізація бізнес-процесів, підвищена економічна ефективність та доцільне використання ресурсів, сприятимуть постійному розвитку та зростанню конкуренції. Відповідно до цього, виникає гостра

необхідність до технологічної трансформації, що сприятиме подальшому розвитку BPM застосунків та систем [36].

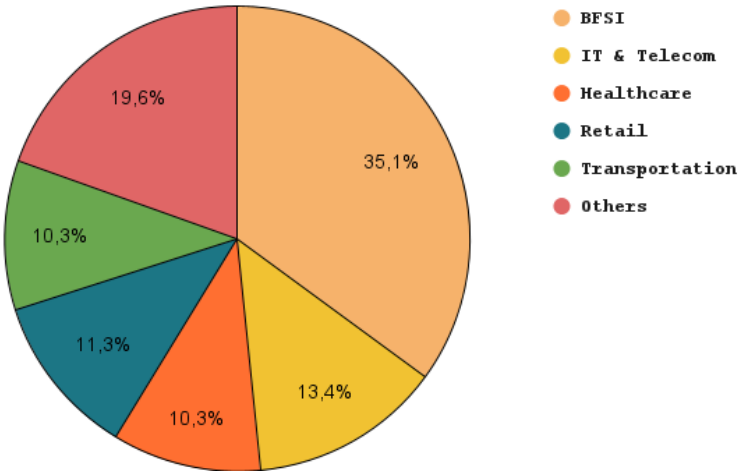


Рис. 2.1 – Сфери поширення BPMS по галузям у відсотках

На рисунку 2.1 наведено глобальний розподіл систем управління та автоматизації бізнес процесів між різними сферами. Найбільш популярними є банківська сфера – 35.1%, сфера IT та телекомунікацій – 13.4%, торгівля – 11.3% сфера охорони здоров'я та логістика – по 10.3%, а 19.6% припадають на інші сфери. Прогнозується, що сегмент IT та телекомунікацій зростатиме на 35% щорічно до 2030 року, що означатиме пришвидшення розвитку BPMS та все більше їх поширення [32].

Відповідно до інформації з джерела [37], для оцінки ефективності BPM можна використовувати декілька методів. Серед них окремо варто виділити наступні:

– SWOT-аналіз: за допомогою нього можна систематизувати інформацію про зовнішні та внутрішні ризики, що впливають на виконання бізнес процесу. Також цей метод підійде для якісної оцінки функціонування BPMS з метою внесення коректив. До недолік цього методу можна віднести його суб'єктивність.

– Імітаційне моделювання: дозволяє виявити так звані вузькі та проблемні місця бізнес процесу перед його автоматизацією за допомогою спеціалізованого ПЗ. До недоліків відносять складність цього методу та високу вартість на його виконання.

Відповідно до вищенаведеної інформації розуміємо, що переваги BPM застосунків полягають в зменшенні витрат на виконання та модифікацію процесів, а також в автоматизації та підвищенні рівня загальної продуктивності [30]. Проте дані застосунки не позбавлені ряду недоліків, найголовнішим з яких можна вважати складність інтеграції з наявними інформаційними системами, в результаті чого на даному етапі нерідко виникають затримки та непередбачені витрати [2].

Також робота із BPMS, зазвичай, вимагає проведення навчання персоналу, що працюватиме з даним ПЗ. Це, в свою чергу, теж викликає певні часові та грошові витрати. В будь якому випадку варто розуміти, що дані недоліки є незначними в контексті наведених переваг та перспектив, які привносить інтеграція та використання BPM застосунків [1].

2.3 Дослідження та аналіз сучасних BPM застосунків

В наш час ринок BPM застосунків може запропонувати надзвичайно великий вибір ПЗ для вирішення найрізноманітніших задач. Існують як браузерні, так і завантажувальні рішення. Схематичне зображення функціонування типової BPMS подано на рисунку 2.2.

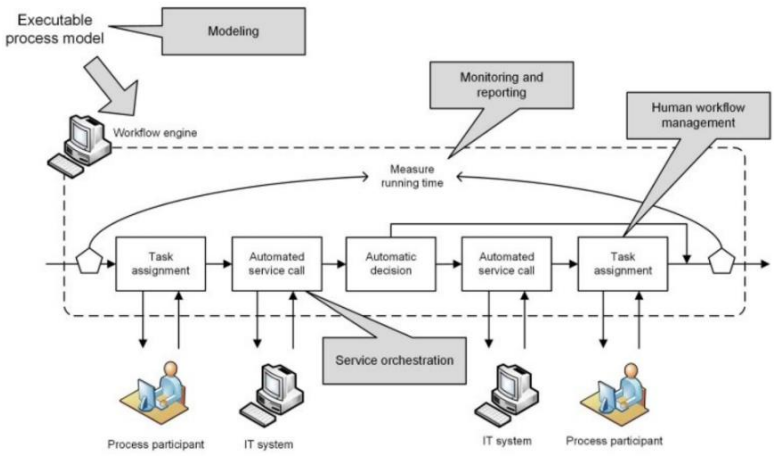


Рис. 2.2 – Схематичне зображення принципу функціонування BPMS [27]

Також популярністю користуються ті застосунки, які є достатньо гнучкими, мають здатність до масштабованості та відрізняються своєю

адаптивністю. Не менш важливим фактором є можливість інтеграції BPSM. Так, вибраний застосунок повинен легко інтегруватись в наявні системи, забезпечуючи при цьому безперебійність потоку даних та автоматизацію бізнес процесів [38].

Також, відповідно до інформації із джерела [39], при виборі того чи іншого BPM застосунку варто обирати не конкретний бренд, а конкретне рішення, яке найкраще відповідатиме поставленим задачам. Власне це і є причиною існування значної кількості BPMS – усі вони пропонують та надають засоби для вирішення великого різноманіття завдань. Так чи інакше, функціонал таких застосунків може дещо відрізнятись, як і ціна придбання чи підписки. Розглянемо найбільш популярні BPMS.

2.3.1 Monday.com

Monday.com – досить відома BPM веб-платформа, яка допомагає не великим організаціям керувати власними бізнес процесами. Відповідно до наявної інформації в джерелі [40], станом на 2020 рік даним BPM застосунком користувалися понад 100 000 організацій.

Monday.com надає можливість координувати роботу команди; керувати як проектами, так і окремими оперативними задачами; створювати візуалізацію бізнес процесів [41]. Також її можливості включають в себе понад 100 доступних шаблонів автоматизації та засоби інтеграцій з понад 50 популярними системами та сервісами. Також можлива інтеграція через API. Дане рішення поширюється за системою підписки, ціна на яку стартує від $9 на місяць за користувача.

Як інструмент управління бізнес процесами – це досить просте у використанні рішення, що пропонує зрозумілий інтерфейс (див. рис. 2.3), освоєння якого не займе багато часу [42].

Також користувачі мають змогу зберігати налаштовану фільтрацію для зручності перегляду інформації про бізнес процеси та його активності. Серед інших переваг варто відмітити кросплатформеність Monday.com, наявність зручного мобільного застосунку, використання можливостей штучного

інтелекту в задачах управління бізнес процесами, інструментарій для швидкого масштабування робочих процесів відповідно до вимог користувача.

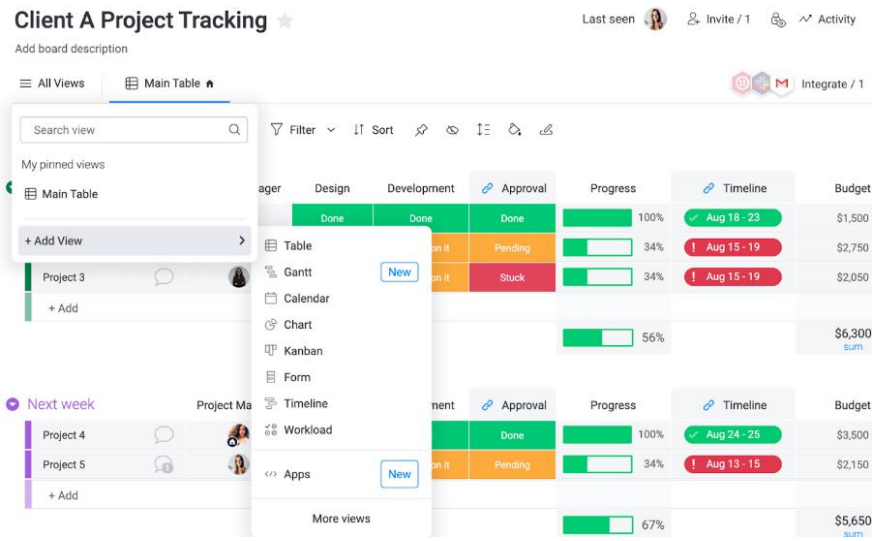


Рис. 2.3 – Користувацький інтерфейс Monday.com

Проте Monday.com не позбавлений недоліків, серед яких можна вважати обмеженість функціоналу, фактичну неможливість повністю автоматизувати виконання усіх етапів бізнес процесів та складність інтеграції у власні системи [41]. Ці недоліки і є причиною того, що Monday.com переважно використовує дрібний, рідше середній бізнес.

2.3.2 Zapier

Zapier – це яскравий приклад сучасної BPMS, реалізованої в якості веб-сервісу. Він дозволяє автоматизувати прості бізнес процеси, пов’язані з функціонуванням та взаємодією декількох веб-застосунків [43].

З архітектурної точки зору Zapier базується на EDA підході, детальніше про який інформацію подано в розділі 2.4. Така система працює наступним чином: як тільки в одному із сервісів чи застосунків виникає певна подія, вона автоматично викликає відповідну дію-реакцію в будь якому заздалегідь визначеному сервісу, в залежності від конфігурації налаштованого бізнес процесу. Процес інтеграції зі сторонніми системами, зазвичай, займає всього декілька хвилин.

Окрім того, перевагою є відсутність необхідності до наявності навичок програмування для використання цього BPMS. Власне, завдяки цьому і забезпечується простота інтеграції та використання Zapier. Більше того, користувачі можуть інтегрувати з цим BPMS велику кількість популярних сервісів одночасно.

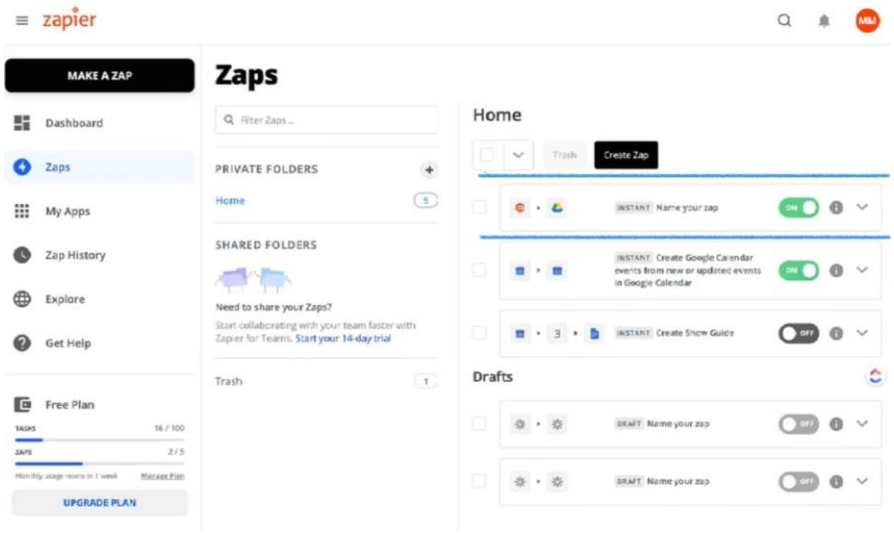


Рис. 2.4 – Користувацький інтерфейс Zapier

Щодо ціни на дане ПЗ, то вона коливається від $20 до $125 на місяць в залежності від обраного тарифного плану. Також існує пробна версія, проте вона значно обмежена в функціональних можливостях. До недоліків Zapier можна віднести умовно бідний з функціональної точки зору та фактичну неможливість до автоматизації надто складних і розгалужених бізнес процесів [41].

2.3.3 Appian та Corezoid

Appian – це BPMS, яка чудово підійде бізнесу та підприємствам усіх розмірів. Його використання допомагає в здачах, пов’язаних із розробкою, виконанням, управлінням та автоматизацією бізнес процесів. Існує безкоштовна пробна версія даного ПЗ із обмеженим періодом користування. Щодо типового функціоналу, потрібного для управління та автоматизації БП, в середньому, ліцензія на нього коштує близько $90 на місяць [44]. Варто звернути увагу і на безліч схвальних відгуків від користувачів даної BPMS, в яких стверджується,

що Appian, попри свою складність в освоєнні, є достатньо гнучким інструментом для управління та автоматизації бізнес процесів.

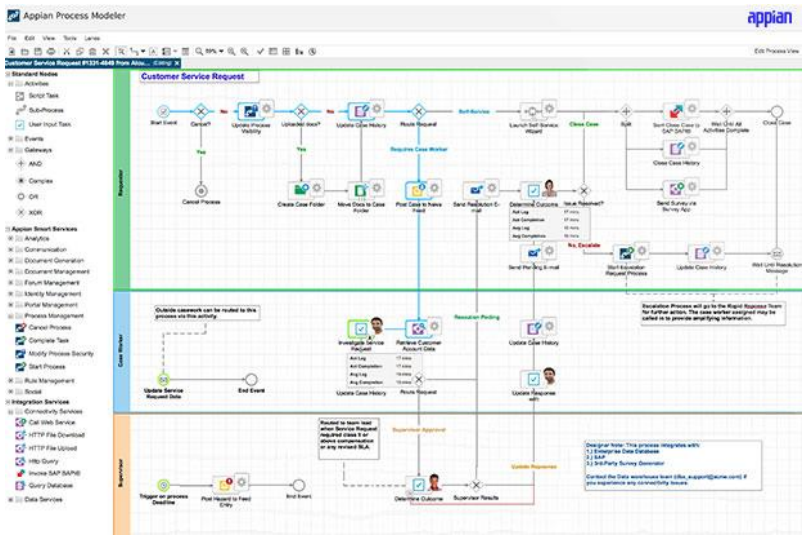


Рис. 2.5 – Користувацький інтерфейс Appian

Appian може інтегруватися до вже наявних систем, тим самим удосконалюючи її. Ключовою особливістю даної BPMS є так званий low code підхід, суть якого полягає в тому, що технічні спеціалісти та бізнес-аналітики отримують змогу управляти та автоматизувати бізнес процеси з мінімальними вимогами до написання коду. Досягається це за допомогою високо-функціонального користувацького інтерфейсу, конструкторів, шаблонних скриптів тощо (див. рис. 2.5).

З практичної точки зору, low code рішення часто є ефективнішим, ніж no code BPMS, до яких належить вищезгаданий Zapier. Причиною цьому є те, що low code підхід значно підвищує адаптивність та гнучкість в інтеграції та застосування такого BPM застосунку [44]. Також Appian впроваджує drag-and-drop функціонал, який дозволяє легко маніпулювати кожним кроком бізнес-процесу на етапі його розробки.

Характерною особливістю Appian можна вважати залучення технологій штучного інтелекту для аналізу, розробки та автоматизації бізнес-процесів. Також ця BPMS дозволяє виявляти вузькі та малоефективні місця бізнес процесів з метою їх подальшої оптимізації. Серед недоліків виділяють високу вартість

ліцензії, складнощі в інтеграції з CRM або ERP системами, а також високий поріг входження навіть для досвідчених користувачів [44].

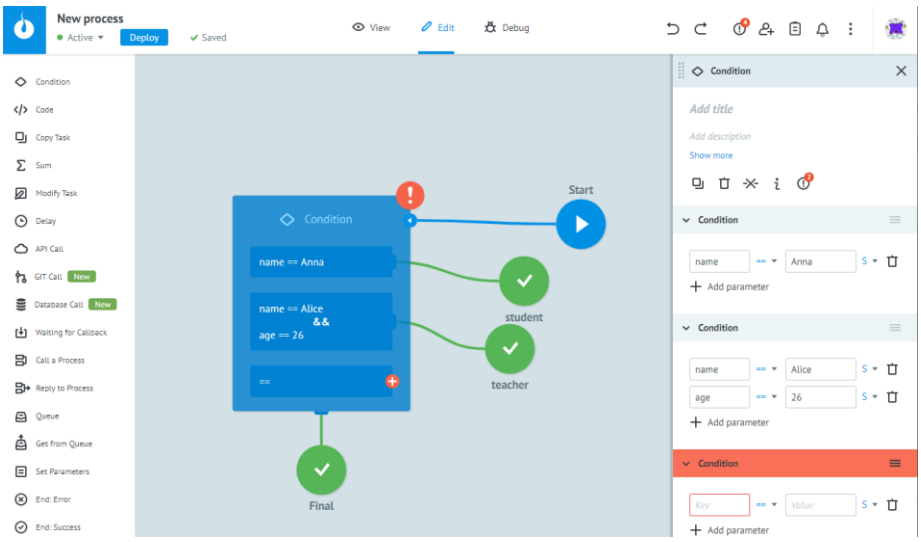


Рис. 2.6 – Користувацький інтерфейс Corezoid

Схожу характеристику можна дати іншому схожому BPM застосунку – Corezoid, який теж є low code рішенням, при чому користувачі мають змогу писати власний код прямо в тіло процесу або використовувати вже готові блоки коду, що позитивно позначається на його адаптивності та спектру виконуваних задач. Варто відмітити високі інтеграційні можливості, що дозволяє залучати Corezoid до платіжних та банківських систем [41].

Як видно з рисунку 2.6 – Corezoid пропонує більш інтуїтивно зрозумілий інтерфейс, ніж Appian. Проте, справедливим буде твердження, що він також може бути складним для нових користувачів. Окрім того, серед недоліків варто виділити загальну складність роботи з даним BPM рішенням для недосвідчених користувачів.

2.3.4 Kissflow

Ця BPM платформа є чудовим рішенням для великих підприємств та бізнесу. Відповідно до інформації із джерела [41], на вибір користувачам пропонується декілька тарифних планів: від $100 до $1000 на місяць. Як і більшість вищеописаних BPM рішень, Kissflow теж притримується low code

парадигми. Дана платформа надає велику кількість вбудованих функцій та інструментів для управління бізнес процесами.

Щодо користувацького інтерфейсу, то він є достатньо простим та інтуїтивно зрозумілим (див. рис 2.7). З його допомогою можна виконувати візуальне проектування бізнес процесів, а також управляти ними та проводити їх оптимізацію [45].

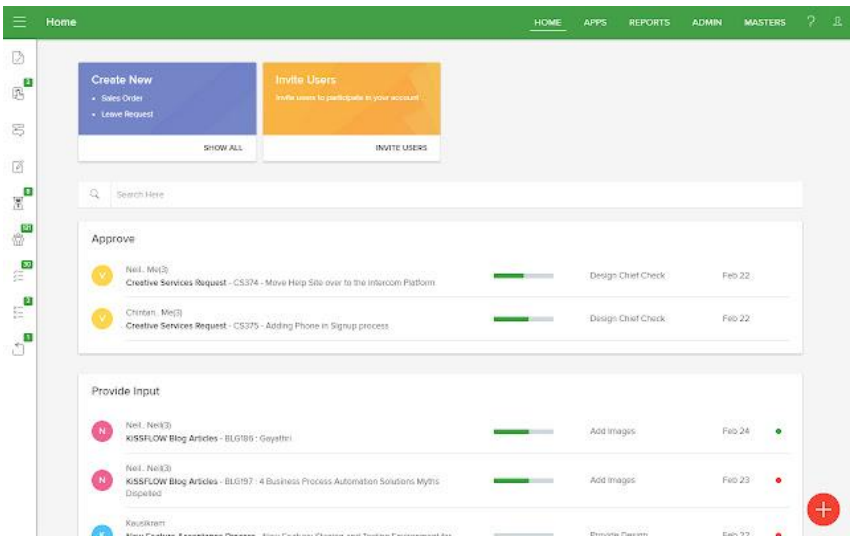


Рис. 2.7 – Користувацький інтерфейс Kissflow

Kissflow має велику кількість дашбордів, які дозволяють аналізувати БП, налаштовувати звітність, отримувати статистику щодо перебігу БП в режимі реального часу Варто відмітити інтеграційні можливості цієї BPMS з іншими системами. Відповідно до інформації із джерела [45] До прикладу, існує можливість до інтеграції із згаданим раніше Zapier за допомогою API.

2.3.5 Studio Creatio

Studio Creatio – одна з найбільш популярних BPMS, якою користуються середні та великі підприємства, що бажають автоматизувати свої бізнес процеси. Користувачам пропонується двотижнева пробна версія з доступом до всіх основних функцій, а найдешевша ліцензія коштує від $25 на місяць [44]. Популярність Studio Creatio фактично можна пояснити відсутністю необхідності щодо написання власного програмного коду, а також її композитною

архітектурою, яка чудово відповідає сучасним ринковим вимогам до BPM застосунків [41].

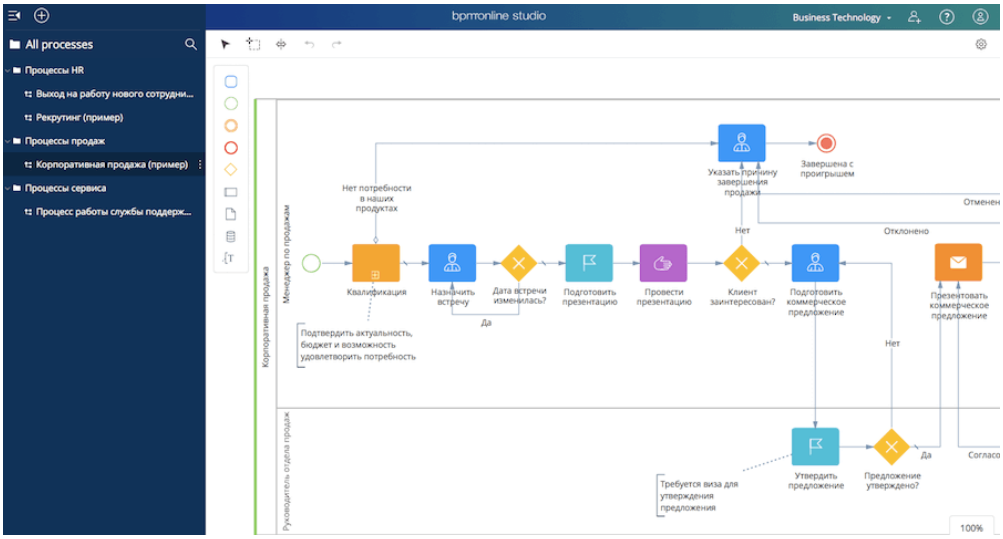


Рис. 2.8 – Користувацький інтерфейс Studio Creatio

Відповідно до рисунку 2.8 можна зробити висновок, що користувацький інтерфейс цієї BPMS є дещо схожим до вже вищезгаданих. Проте, завдяки відсутності необхідності писати код, навіть новачки без відповідних технічних знань та вмінь можуть успішно керувати бізнес процесами, проводити їх аналіз та автоматизацію із допомогою Studio Creatio. Також даний BPM застосунок має ряд готових шаблонів та розширень, що робить його використання більш гнучким та адаптивним.

Варто відмітити також і те, що Studio Creatio забезпечує функціонал для аналізу БП за допомогою штучного інтелекту, що, в свою чергу, прискорює налаштування процесів, а моніторинг процесів в режимі реального часу є корисним для аналітиків в питаннях, пов’язаних у виявленні вузьких місць та проведенні на основі аналітичних даних оптимізації процесів. Інтеграція з іншими системами можлива тільки за допомогою API [45].

Недоліки даної BPMS теж дуже типові. Серед них можна виділити високу ціна за доступ до всього передбаченого розробником функціоналу, дещо заплутаний користувацький інтерфейс (користувачі масово скаржаться, що прості по своїй суті операції вимагають здійснення багатьох зайвих кроків та дій [44]), а також обмеженість інтеграції з іншими системами.

2.3.6 Camunda BPM

Camunda BPM – це популярне серед розробників ПЗ для управління та автоматизації бізнес процесів. Існує як безкоштовна, так і платна версії цього застосунку, різниця між якими полягає в доступному до використання функціоналі [41].

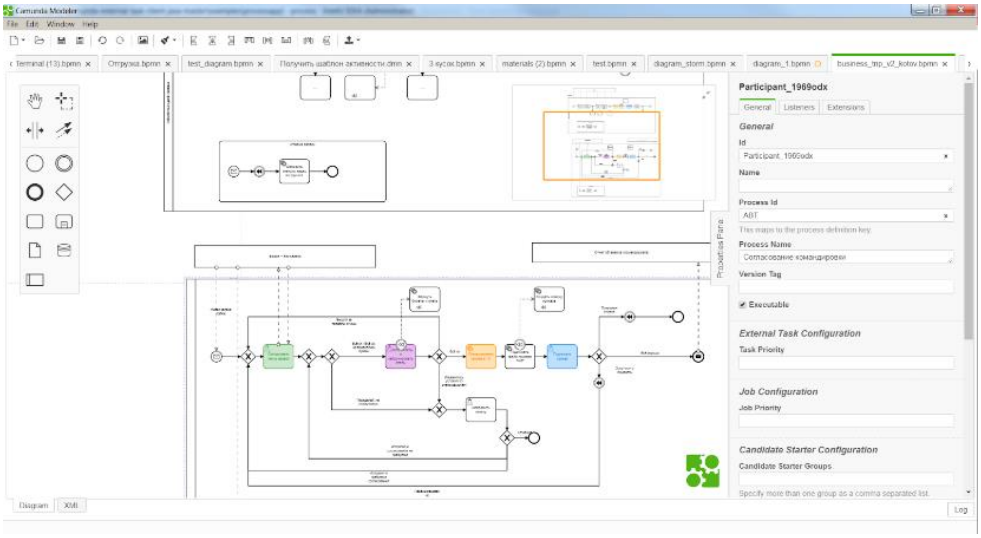


Рис. 2.9 – Користувацький інтерфейс Camunda BPM

В загальному Camunda BPM підійде для використання як малими, так і середніми й великим компаніям в будь-яких сферах. Для оптимізації та дизайну бізнес процесів Camunda BPM використовує такі стандарти, як BPMN та DMN, англ. Decision Model and Notation [46]. Яскравим показником надійності є те, шо найчастіше Camunda BPM застосовується в банківській сфері, де з її допомогою автоматизовують процеси, пов’язані зі створенням довільних заявок, оформленням банківських карток, збору фінансової статистики тощо.

Серед недоліків варто виокремити складність до налаштування платформи попри широкі можливості до цього, необхідність знання мови програмування Java для розробників, що інтегруватимуть Camunda BPM до інших систем, а також не достатньо швидку та якісну підтримку. Варто також відмітити застарілий користувацький інтерфейс. Окрім того, Camunda є одним із найбільш комплексних BPMS рішень серед розглянутих в даній кваліфікаційній роботі, тому на її освоєння може витрачено велику кількість часу [46].

2.4 Типові архітектурні рішення оглянутих BPM застосунків

Варто почати з того, що розробка застосунків для автоматизації бізнес процесів стикається з рядом складнощів, серед яких завжди фігурують вимоги до гнучкості, масштабованості та швидкості функціонування кінцевого продукту тощо. Дані показники на пряму залежать від типу обраної архітектури ПЗ чи застосунку.

В загальному виділяють багато видів архітектури, серед яких окрему уваго варто присвятити монолітній та мікросервісній архітектурам, адже на них базуються оглянуті BPMS. Так, мікросервісний підхід набув широкого поширення за останні п’ять років. Натомість використання монолітного підходу, хоч він це все ще і вважається стандартною моделлю створення ПЗ, поступово йде на спад [47].

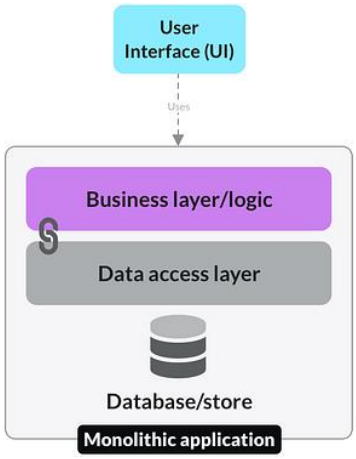


Рис. 2.10 – Схематичне зображення структури монолітного BPM застосунку [47]

Відповідно до парадигм монолітної архітектури, застосунок будується як один великий самодостатній функціональних блок, не залежний від інших застосунків [48]. Наочний приклад структури монолітного BPMS подано на рисунку 2.10. Серед переваг такого підходу варто, перш за все, виділити швидкість розробки завдяки простоті створення ПЗ завдяки наявності однієї спільної кодової бази та легкість первинного розгортання такої BPMS. Але варто зважати на те, що для внесення змін до коду необхідно буде оновити увесь стек

коду, що робить покращення існуючого коду чи розробку нових функціональних задач певною мірою обмеженими та трудомісткими задачами [47].

Розуміємо, що для компонентів монолітних BPM застосунків характерною є висока зв'язаність між собою, що може стати причиною значних технічних проблем. Проте моноліти можуть бути достатньо ефективними та швидкими, оскільки в таких застосунках відсутня комунікація між окремими сервісами через API.

Відповідно до інформації з джерела [48], мікросервісна архітектура певною мірою є альтернативою до монолітної. В основі даного підходу лежить використання ряду незалежних сервісів, які розгортаються окремо один від одного [47]. Ці незалежні сервіси обов'язково повинні виконувати різні задачі та мають власні бази даних.

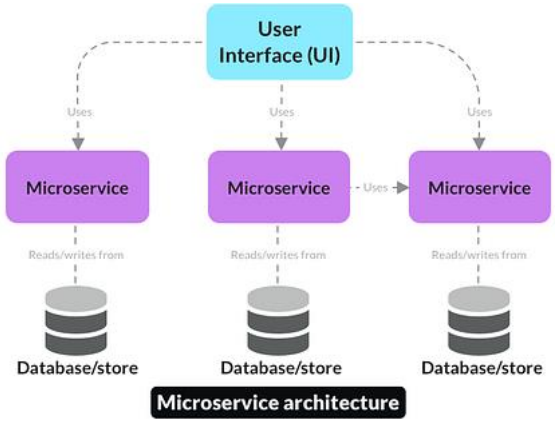


Рис. 2.11 – Схематичне зображення структури мікросервісного BPM застосунку [47]

Відповідно до рисунку 2.11 розуміємо, що мікросервіси розділяють виконання певних бізнес операцій на окремі «зони відповідальності», що дещо підвищує складність розробки BPMS, проте значно полегшує процес вирішення технічних проблем в майбутньому [47]. Також варто відмітити, що для мікросервісних BPMS можна динамічно регулювати кількість виділених ресурсів для того чи іншого мікросервісу в залежності навантаження, що припадає на них. Це дозволяє покращити загальну ефективність системи, а також якісно підняти її продуктивність [47].

Варто зазначити також і те, застосунок для автоматизації бізнес процесів може бути монолітним, проте водночас виступати в ролі мікросервісу після інтеграції в існуючу систему. Також, за відсутності потреби до масштабування застосунку, перехід від монолітного до мікросервісного підходу є просто зайвим. Відповідно, якщо BPM застосунок є простим та легким – не потрібно розбивати його на мікросервіси. З іншої сторони, мікросервісний підхід частіше використовують великі застосунки з розгалуженою бізнес логікою [48].

Щодо функціональних можливостей, відповідно до проведеного дослідження, BPM застосунки повинні використовувати архітектуру, здану до обміну даними через API. Таке рішення дозволяє взаємодіяти з необмеженою кількістю сторонніх систем, а також забезпечує надійний спосіб до обміну інформацією між ними [19]. Можна виділити два основні архітектурні принципи, які використовуються в функціонуванні розглянутих BPMS: процесно-орієнтований та подієво-орієнтований, схематичне зображення яких подано на рисунку 2.12.

Процесно-орієнтована архітектура – це підхід, відповідно до якого процес ставиться у центр функціонування тієї чи іншої системи, тобто розробка та впровадження ПЗ обертається навколо бізнес процесу. Цей підхід дозволяє в певній мірі підвищити ефективність BPM застосунків, забезпечуючи їх адаптацію до змін [49]. І хоча такий підхід добре працює за більшості сценаріїв, не рідкісними є випадки, коли ПЗ на його базі не може справитись із підтримкою складних та динамічних бізнес-процесів.

Подієво-орієнтована архітектура – це підхід, який використовується для розробки сучасних сервісів, метою яких є управління наскрізними бізнес процесами. Такі системи складаються з емітерів подій, їх обробників та каналів. Так, емітери створюють певну подію, не знаючи жодної інформації щодо її обробників [50]. Останні, в свою чергу, несуть відповідальність за обробку події та реагування на неї.

Беззаперечним фактом є те, що більшість з існуючих BPM систем не мають змоги працювати зі станами чи подіями, адже вони спрямовані на роботу із даними, отриманими з БД.

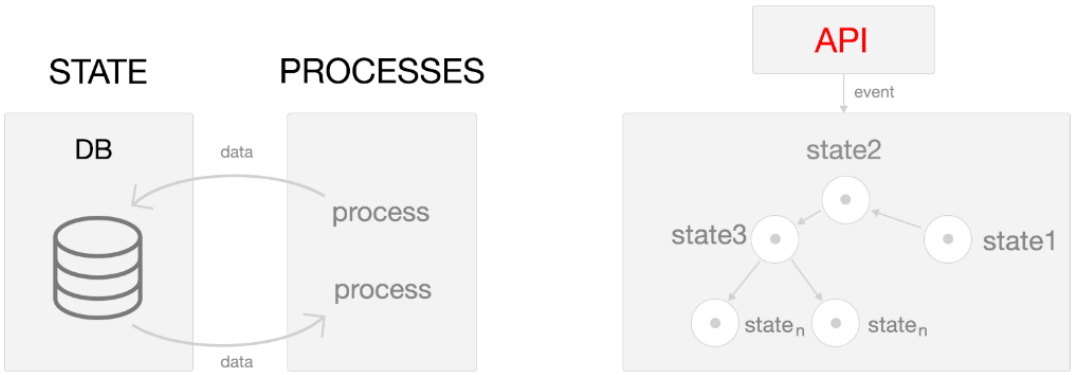


Рис. 2.12 – Схематичне зображення PDA ліворуч та EDA праворуч [51]

Розглянемо наведені раніше BPMS – Appian та Corezoid. Так, перший застосунок для отримання даних щодо стану процесу та керування ним має робити надзвичайну велику кількість запитів до БД та обробляти їх. Варто розуміти те, що чим більше запитів виконує система, тим повільніше вона працюватиме при більшому навантаженні на сервер.

На відміну від Appian та інших схожих BPMS, Corezoid оперує подіями та станами. Завдяки EDA він не потребує численних викликів до БД, при цьому він краще працює з великим та складними процесами. Загалом Corezoid та аналогічні йому застосунки працюють наступним чином: події та стани генерують процеси, процеси генерують дані, які, в свою чергу, спричиняють появу нових подій та станів, і так далі [49].

2.5 Технологічний стек сучасних BPM застосунків

Більшість розглянутих раніше застосунків та систем для управління і автоматизації бізнес процесів використовують досить простий та типовий стек технологій. Перш за все, огляд варто почати із баз даних, адже вони є одним з найбільш вагомих факторів в розробці в контексті постійного збільшення обсягу та структурності даних, якими повинні оперувати BPMS. Також не менш вагомим є необхідність захисту цих даних. Найбільш поширеними рішеннями є застосування таких БД, як MongoDB та PostgreSQL [52].

Фактично, ці дві БД присутні в списку ТОП-5 найбільш використовуваних рішень (див. рис. 2.11). Так, MongoDB характерна для BPMS, для яких необхідне

забезпечення гнучкості структури інформації чи необхідність до проведення масштабування бази даних. З іншої сторони PostgreSQL використовувати при розробці BPSM, функціонування яких вимагає дотримання цілісності даних та виконання складних агрегаційних запитів [52]. Варто зазначити, що серед інших популярних SQL рішень, відповідно до інформації з рисунку 2.13, можна віднести MySQL та Oracle. Так чи інакше, не рідко зустрічаються BPMS, які використовують ці чи інші БД одночасно для покриття як різносторонніх технічних, так і бізнес потреб.

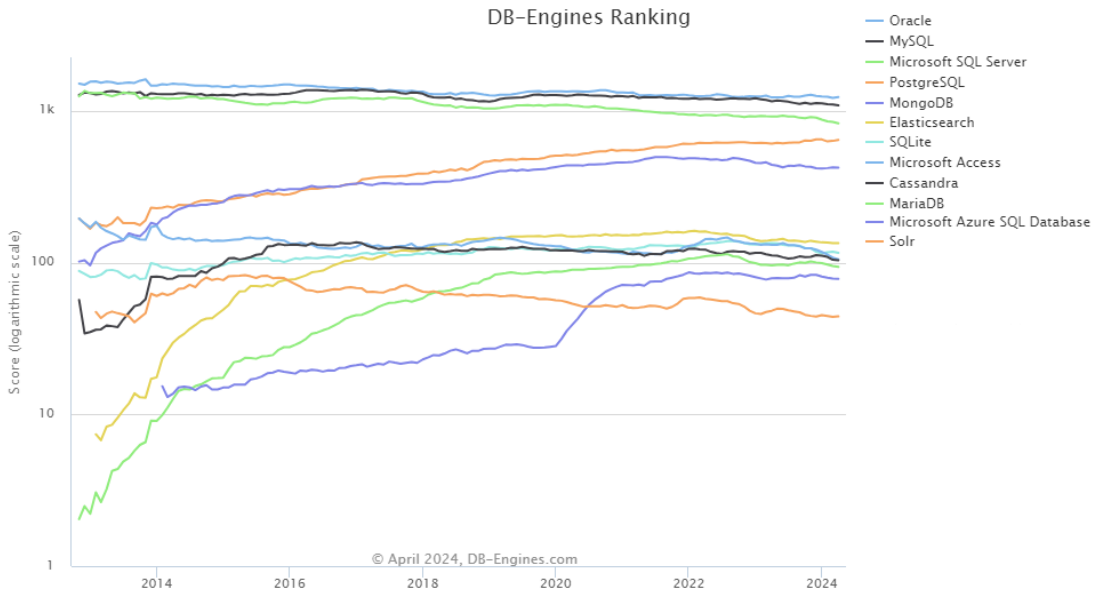


Рис. 2.13 – Графік популярності БД станом на березень 2024 року [53]

Розробка клієнтської сторони більшості оглянутих BPMS виконується із використанням бібліотеки React. Вона, разом з мовою JavaScript, дозволяє створювати оптимізовані та зручні інтерфейси. Дана бібліотека є чудовим рішенням, адже вона надає можливості для створення власних компоненті різної складності та ефективної маніпуляції даними з серверної сторони тощо. Також зустрічаються поєднання сучасного фреймворку Angular з мовою програмування TypeScript, яка, на відміну від JavaScript, додає можливості до перевірки типів даних. Це полегшує розробку та робить систему більш надійною [54].

Щодо серверної сторони BPM застосунків, найчастіше в розробці використовують Node.js разом із фреймворком під назвою Express.

Справедливим дане твердження є для відносно нових, створених протягом декількох останніх років систем [55]. Так, Node.js забезпечує платформу для виконання коду, а Express, в свою чергу, допомагає налаштувати сервер та залучається для обробки запитів з клієнтської сторони чи сторонніх систем. Таке поєднання технологій характерне для високонавантажених та швидкодіючих BPMS. Також частина деяких BPMS реалізована за допомогою мови програмування Java. Вибирають її переважно для великих рішень. Так, для BPM застосунків на базі Java якісно відрізняються своєю надійністю та продуктивністю, хоча процес розробки при цьому стає більш довгим та ресурсозатратним [56].

2.6 Вимоги до сучасних BPM застосунків

Аналізуючи найбільш відомі рішення для управління автоматизації бізнес процесів слід пам’ятати, що далеко не кожна BPMS стає популярною та прибутковою. Відповідно до проведеного дослідження, можна зробити висновок, що на ринку є досить не багато BPM застосунків для невеликих компаній, робота яких ґрунтувалась би на принципах та засадах, характерних для EDA, що надавали б зручний та функціональний інтерфейс з можливістю повної автоматизації бізнес процесів.

Варто розуміти, що наявні рішення можуть бути як масовими, так і створеними для конкретної компанії [38]. Вибір залежить від вимог та задач, які повинне покрити BPM рішення. Так чи інакше, перед вибором чи розробкою BPM застосунку необхідно скласти певний перелік вимог, який називається технічним завданням [57]. Такий підхід до впровадження чи розробки є необхідним з наступних причин:

– Ясність та зрозумілість: проведення детального опису вимог дозволяє чітко зрозуміти суть проекту для розробника та замовника BPM застосунку, шо допоможе уникати непорозумінь.

– Планування ресурсів: дослідження аспектів BPM застосунків дозволяє визначити можливі ризики та спланувати ресурси для реалізації проекту.

– Контроль якості: чітко поставленні вимоги до розроблюваного чи впроваджуваного BPM застосунку є основою для тестування та перевірки кінцевої якості отриманого продукту.

– Масштабованість та адаптивність: врахування можливості до змін розроблюваного чи кастомізації наявного рішення для управління та автоматизації бізнес процесів дозволяє застосунку залишатись гнучким та адаптивним в сучасних ринкових умовах.

Розробляючи концепт вимог до BPMS, варто вивчити рішення, що використовують конкуренти. Так, переважна більшість застосунків, як вже було сказано раніше, хоч і мають якісь особливі риси, все ж імплементують досить схожий за своєю суттю функціонал [41]. Перш за все варто виділити функціональні та нефункціональні вимоги.

Так, функціональні вимоги описують вимоги до ПЗ, що стосуються внутрішньої роботи системи, зокрема поведінку застосунку, правила обробки даних та різноманітні специфічні функції. Інакше кажучи, функціональні вимоги визначають те, що система повинна робити [58]. Розглянемо функціональні вимоги до сучасних BPM застосунків:

– Автоматизація бізнес процесів: застосунок повинен забезпечувати гнучкий функціонал для автоматизації БП, як повної, так і часткової.

– Інтеграція з іншими системами: будь-який сучасний BPM застосунок повинен мати можливість до інтеграції через API. Також перевагою буде наявність розширених можливостей до інтеграції з іншими системами та застосунками.

– Моделювання процесів: застосунок повинен надавати інструменти для візуалізації та моделювання бізнес процесів, що дозволить користувачам та бізнес аналітикам легко та швидко моделювати чи оновлювати бізнес процеси.

– Моніторинг та аналітика: наявність в BPM застосунків засобів для моніторингу виконання та аналітики бізнес процесів є корисними в контексті виявлення можливостей до оптимізації та реінжинірингу бізнес процесів.

В свою чергу, нефункціональні вимоги задають певні критерії, за якими проводиться оцінка якості роботи ПЗ. Можна стверджувати, що функціональні

вимоги визначають те, якою система має бути [59]. Розглянемо нефункціональні вимоги до сучасних BPM застосунків:

– Технологічний стек: визначення технологій, необхідних для розробки та впровадження серверної та клієнтської частин застосунку дозволить визначити апаратні та програмні вимоги до BPMS.

– Сумісність: сучасні BPM застосунки повинні функціонувати на різних операційних системах та пристроях. У випадку веб-застосунків – працювати в різних браузерах.

– Інтерфейси: дизайн інтерфейсу ПЗ, а також апаратного та комунікаційного інтерфейсів дозволить краще розробити комунікаційні аспекти застосунку. Окрім того, важливою є вимога до розробки чи використання користувацького, який відповідав би усім сучасним UX та UI стандартам.

– Операційні вимоги: в цей перелік закладають вимоги до безпеки та конфіденційності оброблюваних застосунком даних, вимоги до керування помилками та правилами перевірки вхідної та вихідної інформації системи, а також визначають очікуваний рівень продуктивності [59].

Важливою також є вимога до правильної декомпозиції сутностей BPM застосунку [60]. З огляду на проведене дослідження найбільш популярних рішень, варто використовувати такі сутності, як process, activity, processInstance та activitInstance.

Так, сутність process повинна виступати певним узагальненням, що містить типову інформацію про бізнес процес, а також об’єднує в собі декілька activity, які є відображенням опису послідовних дій, що виконуються бізнес процесом. В свою чергу, processInstance та activityInstance не просто описують процес чи його етап, а власне і є ними. Так, вони відображають стан виконання бізнес процесу, містять не типу інформацію про нього. Інакше кажучи, process та activity – це опис правил функціонування БП, а processInstance та acitivityInstace є відображенням реального бізнес процесу та його етапів.

Така декомпозиція системи є досить раціональною вимогою. З її дотриманням бізнес-аналітики отримують змогу забезпечувати автоматизоване функціонування нових бізнес процесів не залежно від змін, які могли негативно

вплинути на вже існуючі БП [60]. Окрім того, варто заначити типові вимоги до типів активностей сучасних BPM застосунків (див. табл. 2.2).

Таблиця 2.2 – Типи активностей бізнес процесів в сучасних застосунках

Тип	Характеристика типу
Стандартний	Виконання активностей цього типу залежить від інших. Так, для їх запуску повинні виконатись усі наперед визначені умови або, принаймні, одна із них, в залежності від конфігурації.
Запланований	На відміну від стандартного чи ручного типу, виконання цих активностей є запланованим на певний момент часу від початку запуску бізнес процесу.
Ручний	Дані активності можуть бути запущені лише користувачами шляхом виклику відповідного API чи за допомогою користувацького інтерфейсу.

Розуміння та дотримання вищеописаних правил і вимог гарантуватиме те, що BPM застосунок буде розроблений та (чи) розгорнутий з урахуванням всіх функціональних, нефункціональних вимог та технічних аспектів, необхідних для його ефективної та надійної роботи [61].

2.7 Висновок до другого розділу

В другому розділі кваліфікаційної роботи даному розділі було детально досліджено види та вплив BPM застосунків, а також наведено інформацію про ринок автоматизації бізнес процесів. Так, згідно зі статистики використання BPM застосунків, найчастіше вони присутні в банківській сфері та сфері фінансів. Також, відповідно до наведеної інформації, прогнозується стрімкий приріст сфери застосунків для управління та автоматизації БП на рівні 33% до 2030 року [32]. Відповідно до цього можна зрозуміти, що сфер, в яких будуть застосовуватись BPM рішення, ставатиме дедалі більше, а самі BPM застосунки будуть невпинно створюватись та розвиватись.

Окрім того, було наведено та проаналізовано ряд найбільш поширених станом на сьогодні BPM застосунків та систем, якими користуються як дрібні компанії, так і комерційні гіганти. Усі вони надають широкий інструментарій для

управління бізнес процесами, їх автоматизації та аналізу [41]. Є як малі й дешеві, так і дорогі й великі рішення, які відрізняються швидкодією, способами інтеграції з іншими системами, якістю та зручністю графічного інтерфейсу, можливостями до додаткових конфігурацій тощо. Серед усіх наведених застосунків, відповідно до наявної та вказаної в даному розділі інформації, найкращими рішеннями можна вважати Appian, Kissflow та Corezoid.

Також на прикладі вищезгаданих BPM систем, а саме Appian та Corezoid, було розглянуто два архітектурні підходи – PDA та EDA. Обидва з них мають як ряд значних переваг, так і певних недоліків. Проте, відповідно до наведеної в розділі інформації, можна зробити висновок, що саме EDA застосунки є більш ефективними та надійними, особливо якщо їх архітектура ґрунтується на мікросервісному підході, який полегшує підтримку, масштабування та подальший розвиток системи для управління та автоматизації бізнес процесів.

Щодо стеку використовуваних технологій в розглянутих застосунках для управління та автоматизації бізнес процесів, то найбільш поширеними рішеннями є поєднання Node.js, React та MongoDB в більш сучасних та простіших рішеннях. В той самий час, більш комплексні застосунки розробляють із використанням Java, Angular та реляційних БД, на кшталт PostgreSQL. Можна зробити висновок, що дана різниця полягає в можливості компанії-розробника, а також в поставлених перед BPMS задачах та цілей.

Також в даному розділі було докладно описано ряд функціональних та нефункціональних вимог, аргументовано найкращу композицію сутностей системи, наведено три основні типи активностей будь-якого сучасного BPM застосунку. Можна зробити справедливий висновок, що дотримання зазначених вимог гарантуватиме якісний процес розробки та інтеграції рішення для управління та автоматизації бізнес процесів.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ БІЗНЕС ПРОЦЕСІВ

Відповідно до вищенаведеної інформації розуміємо, що здатність підприємства до управління та автоматизації бізнес процесів є ключовим аспектом конкурентоспроможності. Особливо актуальним це стає за умови постійного розвитку IT технологій, зміни вимог клієнтів та глобальній цифровізації [14]. Ось чому впровадження BPM застосунку є дуже важливим аспектом.

Спираючись на проведене дослідження описаних раніше BPM застосунків та систем, можна сміливо стверджувати, що створення власного рішення на базі EDA із розширеними інтеграційними можливостями та зручним інтерфейсом, до функціоналу якого буде входити автоматизація БП, їх наочне відображення, можливість до проектування тощо, є дуже актуальним. Так, Особливу увагу слід приділити інтеграції з іншими системами через API, в тому числі з екосистеми компанії Unicorn, чий фреймворк UAF доцільно використати наявні ресурси для розробки власного BPM застосунку.

Розробляючи власний застосунок для автоматизації бізнес процесів, необхідно враховувати ряд факторів, які визначатимуть ефективність та зручність у користуванні та користування кінцевою системою успішність цього процесу та його вплив на роботу підприємства. У даному розділі, відповідно до проведеного дослідження, буде розглянуто ключові аспекти та обґрунтовано вибраний стек технологій та програмних засобів, необхідних для реалізації власного BPM застосунку та власне описано процес проектування і розробки такого застосунку.

3.1 Вимоги до розроблюваного BPM застосунку

Більшість вимог до розроблюваного застосунку для автоматизації бізнес процесів з розширеними інтеграційними можливостями є такими ж самими, як і

вимоги для типових сучасних BPM застосунків, описані в ході дослідження в розділі 2.6.

Так чи інакше, перш за все потрібно визначити головні вимоги проекту та перелік конкретних функції і можливостей, що будуть реалізовані в розроблюваному застосунку. Відповідно до проведеного раніше дослідження розуміємо, що на ринку мало ефективних та не перенавантажених додатковим функціоналом BPM застосунків [41], функціонування яких відповідало би концепції EDA – подієво-орієнтованої архітектури.

Варто також розуміти і те, що розроблюване рішення буде так званими MVP – продуктом з мінімально достатнім для використання першими клієнтами функціоналом [62]. Фактично це означає, що процес розробки буде проводитись з уникання зайвих задач. Натомість, додатковий функціонал можна буде розробити в наступних версіях застосунку (див. рис. 3.1). Також MVP рішення прийнято використовувати з метою дослідження певних бізнес цілей та концептів щоб зрозуміти, чи створена ідея та її реалізаціє є життєздатними в умовах сучасного ринку [63]. Оскільки відбувається тестування бізнес концепції, даний підхід до розробки власного рішення для автоматизації БП буде досить ефективним.

Рис. 3.1 – Наочне зображення філософії MVP підходу [62]

Інакше кажучи, MVP застосунок повинен мати мінімально достатній набір основних функцій для первинного розгортання. Ця стратегія спрямована на те, щоб заощадити доступні для розробки ресурси та не створювати не потрібний майбутнім клієнтам функціонал [62]. Серед інших переваг можна виділити наступне:

– Прискорення навчання щодо використовуваних технологій.

– Зменшення необхідної кількості робочих годин розробників та програмних інженерів.

– Можливість надати продукт першим клієнтам якомога швидше.

– Швидке створення нового бренду.

Щодо прямих вимог до розроблюваного застосунку для автоматизації бізнес процесів, окрім типових вимог, згаданих раніше, варто виділити наступне:

– Застосунок повинен реалізовувати функціонал для зручного перегляду, створення та оновлення бізнес процесів та активностей, що виступатимуть в ролі кроків БП.

– Компоненти користувацького інтерфейсу повинні бути гнучкими та динамічно змінювати свій вигляд чи функціональні можливості в залежності від тієї чи іншої події.

– Усі операції з процесами, відповідно до EDA підходу, мають бути асинхронними, що дозволить значно збільшити продуктивність розроблюваного застосунку.

– Застосунок повинен реалізовувати способи до інтеграції з іншими застосунками та системами.

– Масштабованість розроблюваного рішення повинна бути на достатньо високому рівні.

В загальному описаний застосунок зможе використовуватись в багатьох сферах, як для малого, так і для середнього бізнесу, або навіть застосовуватись в якості мікросервісу завдяки можливості інтеграції через API. Окрім того, відповідаючи описаним вимогам, розроблюваний застосунок зможе зайняти нішу дешевих та ефективних EDA BPMS для малого та середнього бізнесу Причиною тому, як вже було сказано раніше, є те, що наявні рішення або занадто дорогі, або не ефективні чи взагалі не підтримують можливості до повної автоматизації БП [19].

3.2 Вибір та дослідження використовуваного для розробки технологічного стеку

Як вже було сказано раніше, вибір стеку технологій для розробки BPM застосунку залежить від багатьох факторів, серед яких можна виділити вимогу

до швидкої майбутньої системи, її мінімально необхідній функціонал, визначити інтеграційні можливості, особливості користувацького інтерфейсу тощо. Варто переконатись і в тому, що технології з обраного стеку постійно розвиваються, забезпечені технічною підтримкою та мають активну спілку розробників. Це дозволить працювати з стабільними і актуальними версіями, що забезпечить полегшення процесу, спрямованого на оптимізацію та масштабування BPM застосунку в майбутньому.

Власне, з огляду на вищезазначену інформацію, для розробки власного застосунку для автоматизації бізнес процесів з розширеними інтеграційними можливостями доцільно буде скористатись MERN стеком, назва якого є абревіатурою [64]. Так, до складу MERN стеку входять наступні засоби:

– Нереляційна база даних MongoDB.
– Серверна платформа Node.js.
– Серверний фреймворк Express.
– Бібліотека для розробки користувацького інтерфейсу React.

Для початку варто визначити, чому серед наявних рішень обрано саме нереляційну базу даних MongoDB. Перш за все, це дуже популярне типове NoSQL рішення, що використовується в багатьох застосунках та системах різного призначення і складності [65]. Також дана БД чудово підходить відповідно до поставлених вимог перед розроблюваним застосунком. Вона дозволяє зберігати дані у вигляді JSON-подібних документів, що полегшує процес розробки та підтримки застосунку, а кодування та групування даних стає легшим процесом, ніж в реляційних БД. Окрім того, MongoDB здатна до горизонтального розширення, що буде корисним у випадку збільшення обсягу оброблюваних даних та навантаження на БД, а також за потреби до проведення масштабування застосунку [65].

Якщо порівнювати MongoDB із типовими SQL БД, варто зазначити, що MongoDB не потребує визначення схем даних та може зберігати довільний тип даних в документі [66], що дозволяє легко змінювати та розширювати його в ході розробки додаткового функціоналу застосунку чи проведенні його масштабування.

Окрім того, клієнт MongoDB підтримує ,велику кількість мов програмування [65], що значною мірою спрощує підбір технологій, необхідних для розробки власного застосунку для автоматизації бізнес процесів з розширеними інтеграційними можливостями. Варто відміти і швидкодію операцій з управління даними в MongoDB та підтримку транзакційних запитів, що є корисним для безпечного оновлення даних про бізнес процеси та пов’язану з ними інформацію. Також дана БЗ є умовно безкоштовною, що дозволяє зменшити витрати на розробку BPM застосунку, проте використання додаткового функціоналу, в тому числі збору статистики, вимагатиме придбання платної ліцензії [66]. Відповідно до наведеної інформації можна зробити висновок, що нереляційна база даних MongoDB є чудовим рішення для розробки власного BPM застосунку.

Щодо серверної частити застосунку, як вже було сказано раніше, варто розглянути Node.js в поєднанні з фреймворком Express. Це досить популярне та поширене рішення для розробки сучасних застосунків. Так, Node.js забезпечує платформу для виконання коду, а Express, в свою чергу, допомагає в налаштування серверу та обробці запитів з клієнтської сторони чи сторонніх систем, в тому числі при інтеграції [67]. Таке поєднання технологій дозволяє створювати високонавантажені та швидкодіючі застосунки на прототипно-орієнтованій мові програмування JavaScript. Ця мова програмування відома своєю легкістю використання та динамічною типізацією [68]. JavaScript часто використовуються в небраузерних середовищах за допомогою фреймворка Electron. Окрім того, JavaScript підтримує імперативну та функціональну парадигми програмування [68].

Завдяки тому, що Node.js є асинхронним рішенням, розроблений на його базі застосунок для автоматизації бізнес процесі зможе обробляти велику кількість одночасних запитів без блокування потоку виконання. Інакше кажучи, це дозволить обробляти дані багатьох автоматизованих бізнес процесів одночасно [67]. Окрім того, Node.js споживає відносно не багато апаратних ресурсів, не завантажує процесор та оперативно пам’ять (див. рис. 3.2). Це дозволить заощадити на розгортанні системи [69].

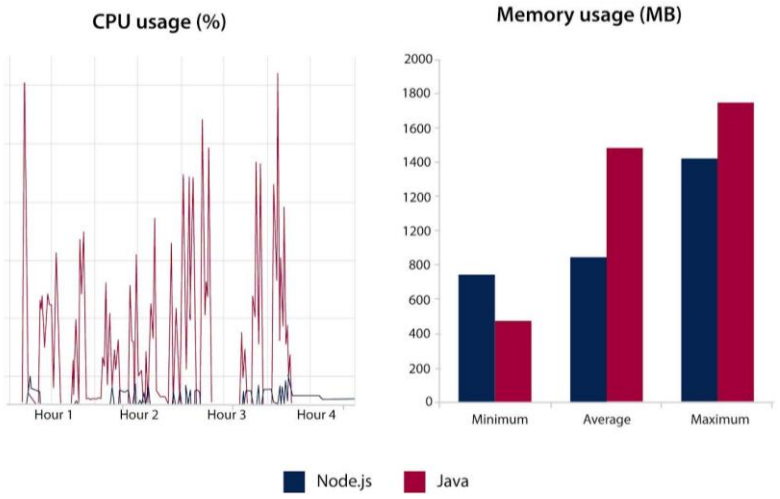


Рис. 3.2 – Порівняння навантаження системи Node.js та Java [69]

Якщо порівнювати Node.js із іншим популярним рішенням для розробки BPM застосунків, а саме Java, варто зважати на наступні аспекти. Node.js сам по собі є швидшим за Java, адже він побудований на подієвій архітектурі [70]. Так, Node.js швидше справляється з виконанням легких задач, наприклад, запити в БД чи обчислення даних [69]. Хоча варто зазначити, що виконання Post запитів платформою Node.js, відповідно до рисунку 3.3, є дещо повільнішим, ніж Java, хоча за середніми показниками Post та Get запитів Node.js є відчутно швидшим рішенням. Окрім того, відповідно до рисунку 3.1, Node.js потребує менших апаратних ресурсів, аніж Java [69].

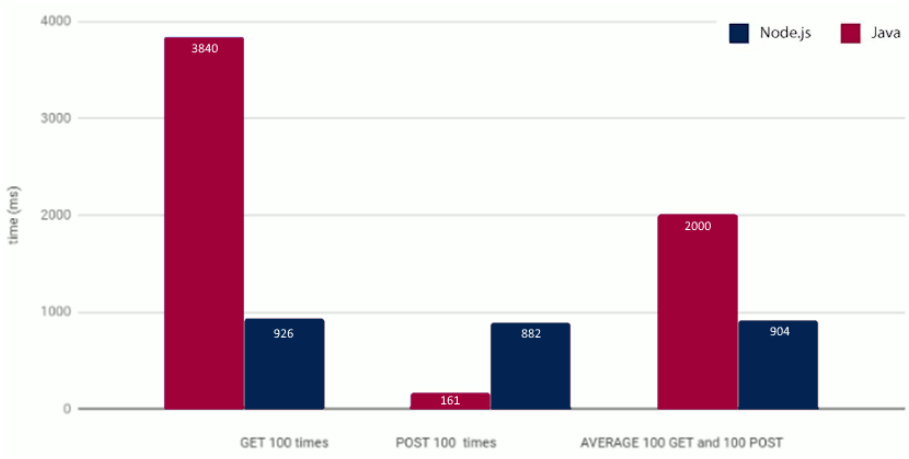


Рис. 3.3 – Порівняння часу виконання 100 запитів Node.js та Java [70]

Щодо швидкості розробки, то в поєднанні з Express відкривається можливість до швидкої розробки роутів застосунку, обробників та емітерів

подій. Важливим фактором вибору є активна спільнота розробник та надзвичайно великий вибір додаткових бібліотек з найрізноманітнішим функціоналом, що дозволяє заощадити час на розробці, використовуючи вже перевірені та готові рішення [67]. Власне, Node.js та Express чудо підходять для реалізації EDA з врахуванням усіх додаткових особливостей, що позитивно позначиться як на розробці, так і на кінцевому результаті [67].

Реалізація клієнтської сторони є не менш важливим питанням в розробці застосунку для автоматизації бізнес процесів. Для цього найдоцільніше використовувати популярну бібліотеку React. Вона, разом з мовою JavaScript, дозволяє створювати оптимізовані та зручні клієнтські інтерфейси. Дана бібліотека є чудовим рішенням, адже вона дозволяє створювати власні компоненті різної складності, ефективно маніпулювати даними з серверної сторони тощо [71]. Це буде корисним для відображення виконання бізнес процесу, а також статусу кожного з його етапів.

Щодо інтерактивності користувацької сторони, то React власне і було створено для розробки інтерактивних динамічних компонентів з власним станом, серед яких можна назвати графіки, форми даних, списки тощо. React компоненти можуть взаємодіяти з серверною частиною застосунку за допомогою HTTP (англ. Hyper Text Transfer Protocol) запитів або технології WebSocket [71].

За допомогою бібліотеки React можна налаштовувати маршрутизацію сторінок, що робить навігацію між різними дашбордами більш зручним. Таке рішення також дає змогу реалізувати SPA [72]. Фактично SPA – це односторінковий застосунок, який динамічно змінює свій вигляд в залежності від дій користувача. Серед переваг SPA, відповідно до інформації з джерела [72], можна виділити наступне:

– Доступність до функціоналу.

– Універсальність компонентів.

– Швидкість розробки.

– Можливість до використання фреймворків для покращення користувацького досвіду.

Для реалізації принципів EDA необхідно також обрати меседж брокер та планувальник задач. Почати варто із брокера. Так, він допомагає розробити застосунок чи систему таким чином, щоб компоненти, замість того, щоб надсилати одне одному асинхронні запити та очікувати відповіді, будуть надсилати повідомлення в брокер [73]. Інші сервіси, застосунки чи їх компоненти отримують це повідомлення з брокеру та обробляють його. Також за рахунок цього забезпечується не тільки швидкодія системи, але й декомпозиція та ізольованість її частин, що позитивно впливає на відмовостійкість та безпеку даних [73].

Одним з найбільш поширених меседж брокерів є RabbitMQ. Від підтримує декілька стандартних протоколів, включаючи AMQP 1.0 та MQTT 5 [74]. Його основними перевагами можна вважати легкість інтегрування в застосунок, можливість до поглинання сплеску навантаження на систему, здатність до проведення технічних робіт з обслуговування без переривання функціонування всього застосунку чи сервісу [74]. Також RabbitMQ включає в себе менеджер черг для управління взаємодії між декількома сервісами із забезпеченням консистентності даних та персистентності процесу обміну інформацією.

Щодо планувальника задач, то свою увагу варто звернути на Node-schedule. З його допомогою можна задати часову умову виконання певної бізнес логіки застосунку [75]. Так, для прикладу, можна з певною періодичністю збирати інформацію про стан перебігу бізнес процеси та його активностей. Node-schedule якісно відрізняється можливістю до гнучкого планування завдань за підходом cron. Серед недолік можна відзначити те, що планувальник працюватиме доти, доки працюватиме скрипт. За потреба запланувати завдання, що виконуватиметься при незапущеному скрипті, варто буде розробити додатковий механізм [75].

3.3 Дослідження фреймворку UAF для розробки BPM застосунку

Виходячи з вище зазначених вимог, особливостей та технологій, справедливим буде твердження, що для розробки власного застосунку для

автоматизації бізнес процесів з розширеними інтеграційними можливостями доцільно буде використати комплексний фреймворк, який дозволить пришвидшити процес розробки та полегшить інтеграцію застосунку з іншими системами. Проаналізувавши багато різних ринкових рішення, найбільш підходящим для реалізації BPM застосунку можна вважати фреймворк на базі Express від компанії Unicorn – UAF. Його використання дозволить провести розробку застосунку без зайвих часових та грошових витрат.

Unicorn Application Framework – умовно безкоштовний та досить потужний інструмент для розробки застосунків, який базується на концепції модульності, гнучкості та продуктивності. UAF дозволяє створювати застосунки, які легко масштабуються, тестуються та підтримуються [76]. Він поєднує в собі усі вищезгадані технології та концепції з різним додатковим функціоналом, що дозволяє приділити більше уваги на проектуванні, дизайні та імплементації бізнес логіки застосунку [76].

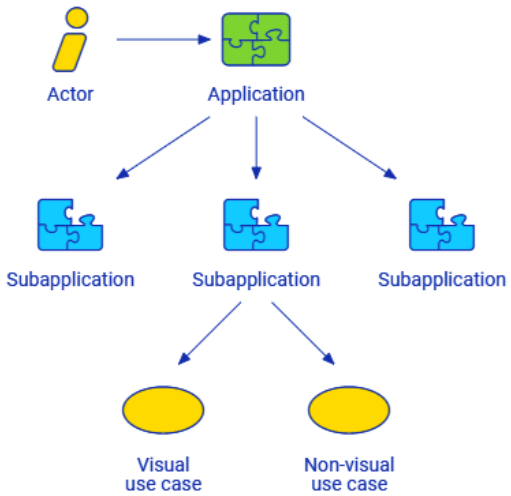


Рис. 3.4 – Схематичне зображення структури UAF застосунків [76]

Оскільки UAF використовує модульний підхід, то кожний застосунок, написаний на ньому, складається з незалежних компонентів, які можна легко змінювати, додавати або видаляти. Відповідно до рисунку 3.4, кожен компонент, в свою чергу, має власну логіку, інтерфейс та перелік залежності [76]. Це спрощує розробку та якісно відображається на чистоті та оптимізованості програмного коду. Компонентний підхід сприяє повторному використанню

коду, адже один і той самий елемент застосунку можна використовувати в різних сценаріях, зокрема для обробки та автоматизації бізнес процесів [76].

Також UAF пропонує різноманітні бібліотеки, які дозволяють створювати додаткові функції та методи, зокрема для роботи з базами даних, аутентифікації та авторизації користувачів, проведення кешування даних тощо. Також UAF має вбудований інструментарій для швидкого створення RESTful API, що дозволяє достатньо оперативно розгортати застосунок та додавати новий функціонал [76]. Однією з переваг модульного підходу UAF також є те, що дозволяє розробник застосунків можуть використовувати тільки ті програмні модулі, які їм потрібні для покриття бізнес задач, знижуючи таким чином навантаження на сервер та покращуючи продуктивність застосунку [77].

Крім того, UAF має вбудовану систему для записування логів, яка дозволяє відслідковувати та аналізувати помилки в програмному коді застосунку. UAF також підтримує використання шаблонізаторів, для прикладу Handlebars чи EJS, що дозволяє розробникам швидко створювати сторінки та відображення даних [76]. UAF дозволяє створювати як SPA застосунки, так і прості чи складні веб-сайти чи системи, що робить дане рішення досить універсальним. Серед інших переваг даного фреймворку, відповідно до джерела [76], варто зазначити наступне:

– Гнучке управління конфігурацією застосунку.

– Підтримка абстракції баз даних та міграції даних, що дозволяє змінювати структуру документів без необхідності ручного оновлення схем даних.

– Підтримка тестування, включаючи використання модульних, інтеграційних та функціональних тестів.

– Впровадження багаторівневої архітектури, відповідно до якої застосунок розбивається на окремі компоненти, яким легко оперувати.

– Засоби міграції версій фреймворку та дотичних до нього бібліотек, що дозволяє завжди підтримувати актуальну версійність ПЗ.

– Можливість інтеграції з іншими бібліотеками, які не входять в екосистему UAF.

– Наявність багатьох готових рішень та стандартів, які дозволяють легко розробляти додаток з високим рівнем якості.

– Впровадження інструментів для детальної документації та підтримки спільноти розробників, що дозволяє швидко знайти відповіді на питання та вирішити технічні проблеми.

Крім того, UAF дозволяє виконувати паралельні запити, використовуючи при цьому різні асинхронні операції [76]. Це збільшує пропускну здатність та загальну ефективність застосунку. Особливо важливим це буде для розробки застосунку для автоматизації бізнес процесів. Також фреймворк гарантує високий рівень безпеки та захисту даних завдяки вбудованій системі авторизації, аутентифікації та контролю доступу. Це дозволяє тримати конфіденційну інформацію в безпеці та зменшити ризики витоку чи крадіжки операційних даних [76]. Варто зазначити, що UAF підтримує інтеграцію з багатьма БД та мовами програмування, проте все ж типовими рішенням для нього є описаний раніше MERN стек.

Щодо розробки користувацького інтерфейсу, то UAF пропонує парадигму, згідно з якою компоненти можуть бути використані повторно в різних проектах, що дозволяє значно зменшити час розробки застосунку та спростити його подальшу підтримку. Бібліотека для розробки клієнтської сторони називається UU5 [76]. Вона поставляється разом з типовими вбудованими компонентами: кнопки, форми, списки, обробники помилок тощо. Крім того, UU5 надає власну систему тем і стилів, з допомогою якої можна гнучко налаштовувати вигляд та поведінку візуального інтерфейсу застосунків. Не менш важливим є те, що UU5 має ряд вбудованих рішень, спрямованих на оптимізацію завантаження додаткових бібліотек.

3.4 Опис інтеграційних можливостей розроблюваного застосунку

Для того, щоб розроблюваний застосунок для автоматизації бізнес процесів якісно відрізнявся від представлених на ринку рішень та став дійсно універсальним інструментом, варто сконцентрувати увагу на проектуванні та

розробці можливостей до інтеграції застосунку з іншими застосунками та системами. Також важливою причиною на користь інтеграційних можливостей є проблеми, пов’язані з стрімким створенням та споживанням даних [78]. Так, засоби інтеграції застосунків можуть вирішити питання з надмірністю та ізольованістю даних, надавши клієнтам застосунку розширені можливості до маніпулювання та аналізу інформації [78].

В загальному інтеграція застосунків – це процес, який дозволяє окремим сервісам працювати разом задля досягнення певної мети, що сприяє підвищенню операційної ефективності [79]. Інтеграція корисна практично в будь-якому програмному забезпеченні в будь-якій галузі [80]. Об’єднуючи та оптимізуючи робочі процеси між декількома застосунками, можна досягти модернізації інфраструктури та підтримувати гнучкість бізнес операцій [79]. Існує декілька типових інтеграційних методів, які можна застосовувати в розроблюваному застосунку для автоматизації бізнес процесів. Розглянемо їх детальніше.

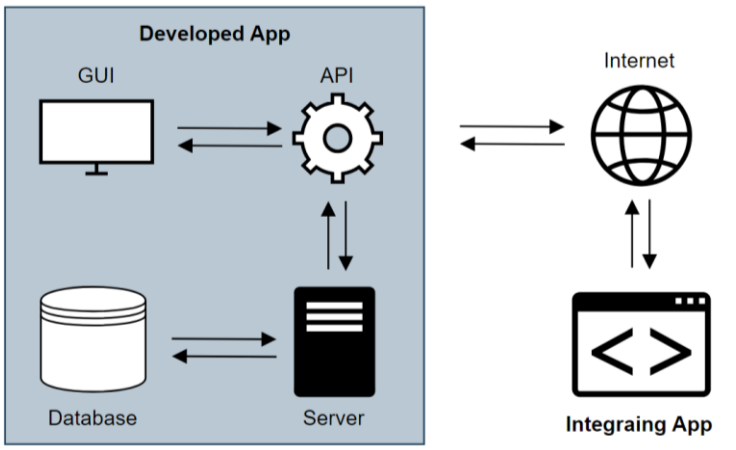


Рис. 3.5 – Схематичне зображення інтеграційних можливостей розроблюваного застосунку через API

Інтеграція через API: як вже було неодноразово сказано раніше в даній кваліфікаційній роботі, API – це набір певних функцій та процедур, які визначають правила взаємодії програмних компонентів [79]. Завдяки цьому інші застосунки та системи зможуть швидко та легко отримати доступ до функціональних можливостей розроблюваного застосунку (див рис. 3.5). Цей метод інтеграції став популярним за останні декілька років та найчастіше

використовується для інтеграцій в мікросервісне ПЗ [79]. Проте такий тип інтеграцій є дуже залежним до оновлення функціоналу та набору даних, що потребує чи повертає виклик API в розроблюваному застосунку.

Інтеграції через обробників подій: даний метод інтеграції можна розглядати через призму EDA підходів [79]. Працює він аналогічним чином – система, що бажає інтегруватись до розроблюваного застосунку для автоматизації бізнес процесів, має створити подію-повідомлення та надіслати його в меседж брокер в спеціально виділену чергу [74]. Розроблюваний застосунок реалізує обробник подій, що підписується на загадану раніше чергу в меседж брокері, та, отримавши з неї дані, обробляє їх та, за потреби, надсилає відповідь [80] (див. рис. 3.6).

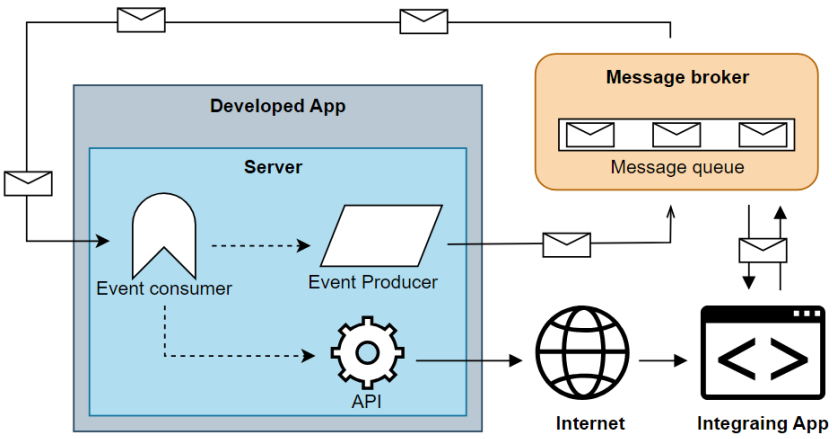


Рис. 3.6 – Схематичне зображення інтеграційних можливостей розроблюваного застосунку події та меседж брокер

Варто зазначити, що відповідно до рисунку 3.6, обробник подій для інтеграцій з іншими системами може не тільки сприймати та обробляти дані, але й викликати генератор подій для надання відповіді іншому застосунку чи навіть надсилати її через API. Такий підхід є досить зручним та дозволяє гнучко налаштувати інтеграційні аспекти розроблюваного застосунку.

Інтеграція в екосистему UAF: як і будь-який великий корпоративний фреймворк, UAF надає змогу до швидкої інтеграції застосунків, розроблених з його використанням. Перш за все, це інтеграція через внутрішню систему під назвою uuBusinessTerritory [76], яка відповідає за мапінг викликів та додаткову

аутентифікацію користувачів. Також в якості інтеграційної можливості можна зібрати всю бізнес логіку застосунку в серверну бібліотеку за допомогою інструментарію UAF та імпортувати її в іншій системи. Ба більше, через так звані startup listeners можна переписати нативний програмний код (див. рис. 3.7).

```
class RegisterProcessExpander {
    async onStartup() {
      ProcessValidatorExpander.registerEnsureProcessStateForProcessingExpanderFn(
        this._customFunction
      );
    }
    _customFunction(process){
      return process.state === "planned"
    }
  }

```

Рис. 3.7 – Інтеграція модифікованого коду в розроблюваний застосунок іншою UAF системою

Окрім вищеописаних інтеграційних можливостей, варто передбачити засоби для інтеграції з системами управління доступу на кшталт OAuth. Це протоколом, спроектований для делегування обмеженого доступу до ресурсів, що не розкриває при цьому вихідні облікові дані користувача [81]. За допомогою OAuth користувачі мають авторизуватись в системі із використанням власного Google акаунту або AppleID.

3.5 Формування структури та архітектури розроблюваного застосунку

Відповідно до проведеного аналізу предметної області та визначеними раніше вимогами, можна зробити висновок, що для розробки власного BPM застосунку для автоматизації бізнес процесів найкраще підійде структура на базі трирівневої архітектури (див. рис. 3.8).

Умовно застосунок, в основі якого лежить трирівнева клієнт-серверна архітектура, можна поділити на три основні складові: рівень клієнта, структурно-логічний рівень та рівень даних [82].

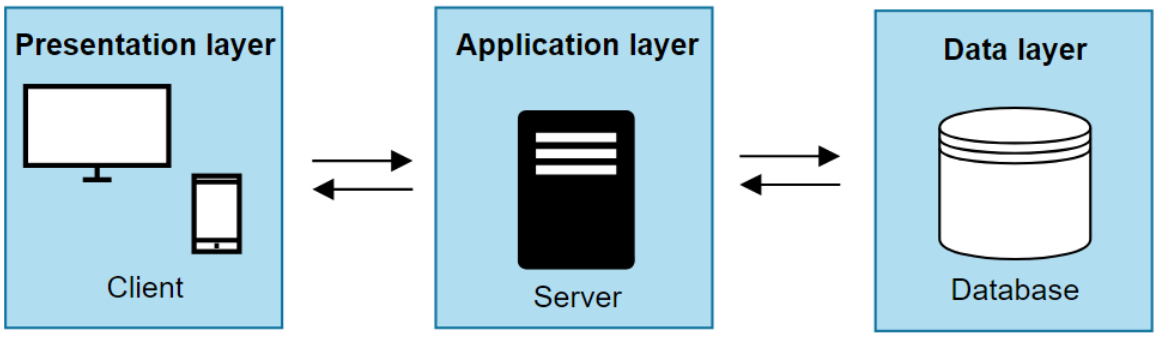


Рис. 3.8 – Схематичне зображення трирівневої клієнт-серверної архітектури

Клієнтський рівень відповідає за взаємодію користувача із системою. Даний рівень, як правило, не повинен мати комплексної бізнес логіки. Також він не повинен на пряму взаємодіяти із рівнем даних [82].

Структурно-логічний рівень представлений основним робочим ядром – це більшість функціональних засобів, службові модулі, компоненти, що підписані на події, або обробляють їх, використовувані бібліотеки тощо. Також саме на цьому рівні присутній код, пов’язаний із створенням сутностей бізнес процесів та їх активностей [82].

Рівень даних, як можна було зрозуміти з вимог до розроблюваного застосунку, представлено нереляційною базою даних MongoDB, яка зберігає дані колекцій BPM застосунку та впроваджує типовий функціонал, що дозволяє ефективно та зручно управляти ними, зокрема для створення чи оновлення даних, їх вибірки за критеріями, виконання агрегаційних запитів, налаштування індексації колекцій тощо [82].

Для роботи з застосунком для автоматизації бізнес процесів користувач використовуватиме звичайний браузер, що позбавляє його необхідності до завантаження та інсталяції на свій ПК спеціалізованого ПЗ [83]. За допомогою зручного інтерфейсу користувач зможе взаємодіяти з серверним рівнем застосунку та надсилати до нього HTTP запити. В свою чергу, серверна частина обробляє ці запити та викликає певні програмні модулі [83]. Також сервер виконує операції над даними, що зберігаються в БД. Зокрема, він може виконувати вибірку даних з БД за певними критеріями, здійснювати її оновлення чи створення [83]. Варто зазначити, що розроблюваний застосунок, як вже було неодноразово зазначено, реалізовуватимете основні підходи EDA архітектури.

Окрім того, варто зазначити, що сучасні застосунки можуть базуватись на MVC підході – поділу коду на три логічні частини: модель, представлення та контролер (див. рис. 3.9).

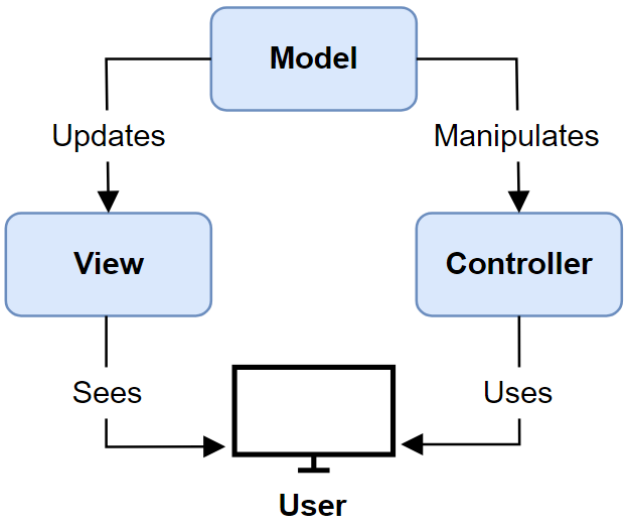


Рис. 3.9 – Схематичне зображення MVC структури

Основна задача MVC полягає у відокремленні бізнес логіки застосунку від її візуалізації. З рахунок цього можна добитись пришвидшення розробки застосунку. Також MVC підхід чудово вписується у вимоги, які постають перед застосунком в разі використання трирівневої клієнт-серверної архітектури [84].

3.6 Проектування структури бази даних

Проектування бази даних – типовий процес, що є частиною розробки будь-якого застосунку чи системи. Це дозволяє попередити дублювання та надмірності даних Відповідно до поставлених опису вимог розуміємо, що необхідно створити чотири колекції даних: process, activity, processInstance та activityInstance.

Окрім того, відповідно до патернів фреймворку UAF, самим фреймворком буде створено ряд системних колекцій, необхідних для правильного його функціонування [76]. Зв’язки між чотирма основними колекціями та їх вміст подано на рисунку 3.10.

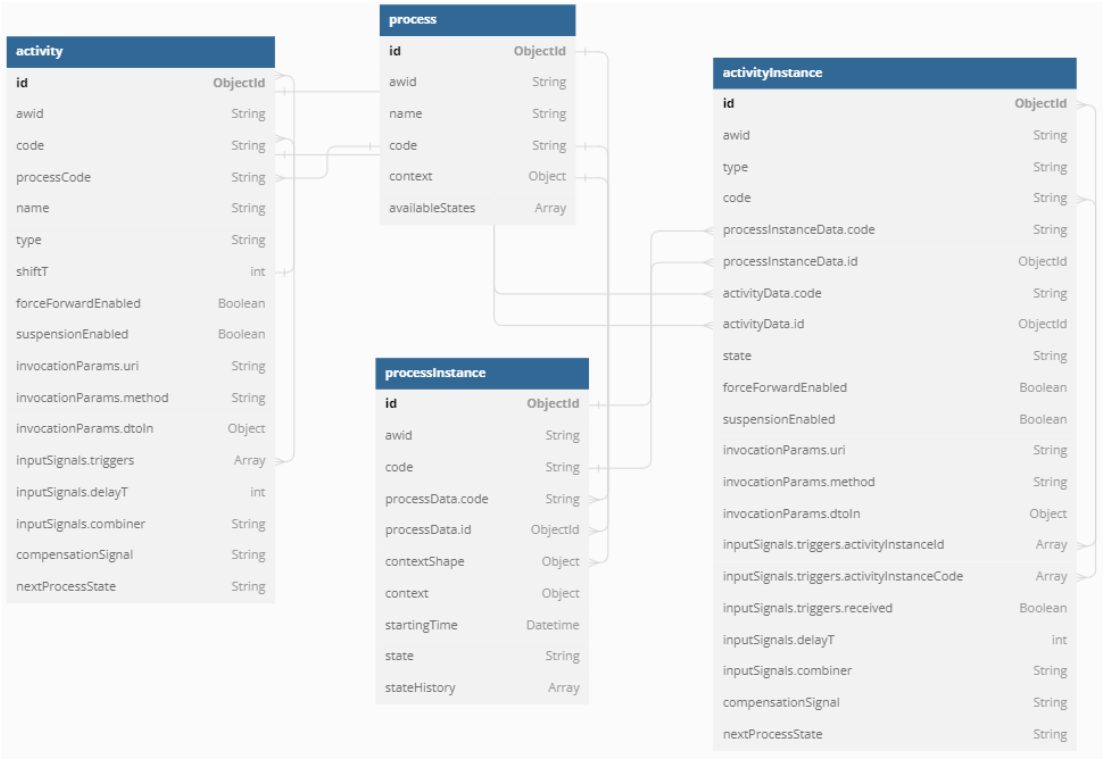


Рис. 3.10 – Схема структури основних колекцій та зв’язків між ними

Як видно із рисунку 3.10, структура даних відносно не складна. Відповідно до неї, об’єкти process та activity варто сприймати як правила створення processInstance та activityInstance відповідно, які містять інформацію про них, а також все, що стосується виконання бізнес-процесу.

```
"uuSubAppDataStoreMap": {
    "primary": "mongodb://127.0.0.1:27017/simeBpmPrimary"
},
```

Рис. 3.11 – Вказання адреси під’єднання до бази даних MongoDB

Розглянемо детальніше структуру колекції activity, базуючись на її DAO (англ. Data access object) [85]. На відміну від звичайного використання Node.js та Express, застосунок на базі фреймворку UAF автоматично створює з’єднання з базою даних під час запуску застосунку (див. рис. 3.11), а також генерує індекси до кожної із описаних колекцій.

Окрім того, бібліотеки, надані UAF, впроваджують надзвичайно зручний функціонал щодо створення запитів до бази даних та обробки отриманої

інформації [76]. Зокрема це можна побачити на рисунку 3.12 (повний код схеми розроблюваного застосунку подано в додатку В). Так, метод createSchema створює в БД відповідну колекцію із вказаними індексами.

В загальному індексація – дуже важливий аспект в розробці, яким не можна нехтувати, адже Вона допомагає більш ефективно та швидко отримувати дані із БД під час запитів до неї [66]. Метод insertOne відповідає за створення запису activity в БД, а find – за пошук відповідно до заданого фільтру.

```
async createSchema() {
    await super.createSchema();
    await super.createIndex({ awid: 1, id: 1 });
    await super.createIndex({ awid: 1, code: 1, processInstanceId: 1 },
      { unique: true, name: "activity_code_and_process_instance_id_idx_1" });
    await this.deleteUnusedIndexes();
  }

  async create(uuObject) {
    return await super.insertOne(uuObject);
  }

  async listStandardWithEmptyInputSignals(awid, processInstanceId) {
    return await super.find({ awid, processInstanceId, type: Activity.types.standard, inputSignals: null });
  }
```

Рис. 3.12 – Приклад DAO методів, реалізованих із використанням UAF

Також за допомогою DAO методів можна проводити агрегації – специфічні запити для отримання чи модифікації даних із використанням зарезервованих ключових слів [85].

Наприклад, для виконання бізнес логіки потрібно знайти усі відображення активностей за певними критеріями та позначити відповідне поле received в об’єкті масиву triggers як true. Задача сама по собі не складна, але якщо не знати принципів побудови агрегаційних запитів, вирішити її буде не буденною справою.

Так, доведеться спочатку отримати список усіх activityInstance, потім застосувати операцію фільтрації по певній умові. Коли отримаємо відфільтрований список, необхідно буде для кожного з його елементів модифікувати відповідне значення received в об’єкті масиву triggers як true та виконати операцію оновлення. В свою чергу, агрегація дозволяє це зробити за один виклик (див. рис. 3.13).

```
async markReceivedTrigger(awid, triggerActivityInstanceId, activityInstanceIdList) {
    let filter = {
      awid,
      id: { $in: activityInstanceIdList }
    };

    return super.updateMany(
      filter,
      { $set: { "inputSignals.triggers.$[x].received": true } },
      { arrayFilters: [{ "x.activityInstanceId": triggerActivityInstanceId }] }
    );
  }
```

Рис. 3.13 – Приклад агрегаційного запиту в БД

Отож, як видно з рисунку 3.13, спершу в коді формується фільтр для пошуку необхідних записів в БД. Позначка $in означає, що значення відповідного поля повинно бути еквівалентним до одного із переданого масиву значень (в даному випадку – activityInstanceIdList). Опісля викликається метод updateMany, головна мета якого – оновити декілька записів в БД базуючись на фільтрі та додаткових опціях за один виклик.

В якості додаткових опцій можна розглядати операцію $set – один з операторів оновлення, який дозволяє змінити значення окремих полів в документах БД без необхідності повного заміщення цілого документу [65]. Також оператор $set дозволяє встановлювати нові значення для одного або декількох полів в документі одночасно. Завдяки даному оператору можна зменшити обсяг даних, які потрібно зберігати та передавати [66]. Використання $set зменшує кількість мережевого трафіку та зменшує використання ресурсів на сервері бази даних, оскільки він оновлює лише необхідні поля в документі.

В свою чергу, параметр arrayFilters, використання якого зображено на рисунку 3.13 є фільтром, який застосовується в операції оновлення масиву в MongoDB. Вони дозволяють точніше задавати критерії оновлення для конкретних елементів масиву, забезпечуючи більшу гнучкість при роботі з ними. Також даний оператор може бути корисним при оновленні документів з вкладеними масивами або об'єктами, коли потрібно змінити певні значення в цих масивах або об'єктах за допомогою більш точних критеріїв фільтрації, ніж пропонують звичайні фільтри.

3.7 Розробка роутів застосунку для автоматизації бізнес процесів

Перш за все варто дати визначення поняттю роут, або, інакше кажучи, – маршрут. Фактично це поняття можна інтерпретувати як посилання в адресному рядку, яке притаманне будь-якому веб-застосунку чи сайту без виключень [86]. Так, відповідно до обраної модель застосунку, запити (також часто в документаціях вони позначаються англіцизмом «ріквести») надсилаються із користувацького інтерфейсу частити до серверу, де їх обробку здійснюватиме Node.js із використанням усіх необхідних додаткових бібліотек, завантажених за допомогою пакетного менеджера NPM [64]. Вхідне місце на сервері, як було зазначено раніше, несе назву API. Окрім користувацького інтерфейсу, сервер можуть викликати інші застосунки за допомогою REST запитів [50]. Зокрема, в ході розробки застосунку для його тестування було використано Insomnia – REST клієнт, який дозволяє тестувати API застосунків шляхом надсилання до них Post або Get запитів з тілом та іншими необхідними параметрами.

На відміну від чистого Node.js в поєднання з Express, застосунки на базі UAF мають значно легший та швидший процес налаштування та реєстрації роутів [76]. Так, не потрібно більше власноруч прописувати код app.use(…) у вхідному файлі. Відповідно до рисунку 3.14, Достатньо лише заповнити два конфігураційних json файли, а застосунок під час своєї ініціалізації зробить все за розробників. Всі конфігурації роутів розроблюваного застосунку для автоматизації бізнес процесів подано в додатку Д. Перелік необхідної інформації для коректного запуску складається з наступного:

– Назва команди, яка може складатись з довільної кількості слів, розділених між собою скісно рискою.

– Посилання на відповідний контролер та його метод, що має бути викликаним, коли на вказане API прийде запит.

– Список дозволених станів застосунку, при яких запит буде виконано. Тобто, якщо на разі застосунок перебуває в стані restricted, а в конфігураційному файлі вказано active – користувач чи інший застосунок, що надіслали цей запит, отримають в якості відповіді відповідну помилку.

– Список профілів користувачів, які мають право на виконання команди.

Рис. 3.14 – Приклад конфігурації команди process/create засобами UAF в конфігураційному файлі profiles.json

Як видно із рисунку 3.14, виконання команди process/create доступно лише за умови, що стан застосунку є active, а користувач чи система, що викликатимуть його, матимуть профіль або Authorities, або Executives. Щодо конфігурації типу команди та відповідного контролера, то дану інформацію подано на рисунку 3.15.

Рис. 3.15 – Приклад конфігурації команди process/create засобами UAF в конфігураційному файлі mapping.json

Так, на рисунку 3.15 можна побачити, що розробник застосунку, використовуючи фреймворк UAF, може вказати тип команди – Post чи Get, а також відповідний контролер та його метод, які прив’язуються до виклику даної команди та запускатимуть відповідну ABL (англ. Application Business Logic).

3.8 Розробка обробників подій

Оскільки в основі розроблюваного застосунку для автоматизації бізнес процесів з розширеними інтеграційними можливостями лежить використання

EDA, логічно припустити, що в його структурі повинні бути наявними обробники подій [49]. Так, за допомогою них можна здійснювати обробку основної логіки, пов’язаної з ходом перебігу бізнес процесу та виконання дій, запланованих для його активностей. Відповідно до поставленої задачі, прийнято рішення реалізувати чотири обробника подій:

– StartInitialActivitiesHandler: запускається разом із командою process/start. Даний обробник шукає всі незалежні активності, які повинні стартувати разом із процесом, та генерує подію activityStart.

– StartActivityHandler: даний обробник відповідає за перевірку можливості запуску активності. Окрім того, якщо activityInstance мала викликати зовнішню систему – даний обробник дочекається результату і, відповідно до заданої логіки його обробки, оновить стан самої activityInstance та її processInstance. У випадку, якщо такого виклику не передбачено, або він не закінчився помилкою, буде згенеровано подію activityFinished.

– ActivityFinishedHandler: відповідає за встановлення кінцевого стану activityInstance та пошуку усіх залежних від неї активностей. За умови, що всі зазначені умови для кожної окремої активності виконуються, відбудеться генерація події activityStart. Даний обробник є найбільш комплексним зі сторони реалізації бізнес логіки.

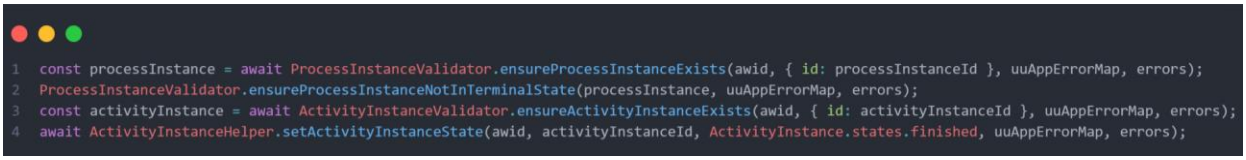
– ChangePiStateHandler: даний обробник оновлює стан processInstance, знайденого за вхідними параметрами. Створення подій для нього відбувається в багатьох місцях вищезгаданих обробників подій.

Відповідно до рисунку 3.16, створюється певна подія, відповідний запис про яку поміщається в MongoDB. Паралельно із цим, сконфігурований компонент scheduler в режимі реального часу перевіряє записи в колекції events та здійснює запуск тих, чий час старту рівний або більший від поточного [49]. Такий підхід дозволяє забезпечити надійність та безперебійність роботи застосунку, а також гарантує збереження консистентності даних, що обробляються описаними вище обробниками подій. Повний програмний код усіх обробників подій розроблюваного застосунку для автоматизації бізнес процесів подано в додатку Е.

```
produceCommonEvent(awid, eventCode, startingTime, payload, uuAppErrorMap, errors) {
    return this._produceEvent(awid, eventCode, startingTime, payload, this.eventDao, uuAppErrorMap, errors);
  }

  producePiUpdateEvent(awid, eventCode, startingTime, payload, uuAppErrorMap, errors) {
    return this._produceEvent(awid, eventCode, startingTime, payload, this.piUpdateEventDao, uuAppErrorMap, errors);
  }

  _produceEvent(awid, eventCode, startingTime, payload, dao, uuAppErrorMap, errors) {
    const validationResult = dao.validate(`${eventCode}_payload_dtoInType`, payload);

    let eventErrorMap = ValidationHelper.processValidationResult(payload, validationResult, `${errors.UC_CODE}unsupportedKeys`, errors.InvalidDtoIn);
    Object.assign(uuAppErrorMap, eventErrorMap);

    const createDtoIn = EventHelper.prepareCreateDtoIn(awid, eventCode, startingTime, payload);

    let createdEvent;
    try {
      createdEvent = dao.create(createDtoIn);
    } catch (error) {
      throw new errors.CreateEventFailed({ uuAppErrorMap }, createDtoIn, error);
    }

    return { ...createdEvent, uuAppErrorMap };
  }
```

Рис. 3.16 – Програмний код менеджера створення подій

Розглянемо детальніше імплементацію коду для обробника подій під назвою ActivityFinishedHandler. Спершу він перевіряє, чи існує відповідний processInstance, чи він не в термінальному стані та чи існує документ в БД для activityInstance (див. рис. 3.17).



```
const processInstance = await ProcessInstanceValidator.ensureProcessInstanceExists(awid, { id: processInstanceId }, uuAppErrorMap, errors);
ProcessInstanceValidator.ensureProcessInstanceNotInTerminalState(processInstance, uuAppErrorMap, errors);
const activityInstance = await ActivityInstanceValidator.ensureActivityInstanceExists(awid, { id: activityInstanceId }, uuAppErrorMap, errors);
await ActivityInstanceHelper.setActivityInstanceState(awid, activityInstanceId, ActivityInstance.states.finished, uuAppErrorMap, errors);
```

Рис. 3.17 – Початкові перевірки обробника подій ActivityFinishedHandler

Опісля, якщо активність повинна змінити стан процесу, відбудеться створення відповідної події. Подальшим кроком буде пошук усіх залежних активностей. Для кожної з них (якщо вони взагалі існують) буде оновлено відповідний об’єкт – trigger, чий параметр received набуде значення true.

Якщо ж активності можуть бути запущені (значення комбінатора AND і всі тригери отримані або значення комбінатора OR і отримано хоча б один тригер) – відбудеться запуск відповідних активностей через генерацію події activityStart. Також, як видно із вищенаведеного рисунку, ActivityInstanceHandler використовує типову логіку, винесену в окремі класи-валідатори та класи-хелпери. Така декомпозиція дозволяє не одноразово перевикористовувати однакову логіку без зайвого дублювання коду, що відповідає принципу

програмування DRY (англ. Don’t repeat yourself), украї нський переклад якого звучить як «Не повторюйся».

3.9 Розробка користувацького інтерфейсу

Для розробки користувацького інтерфейсу власного BPM застосунку використано підхід, який базується на концепції архітектури UAF. Він дозволяє побудувати інтерфейс таким чином, щоб той максимально відповідав потребам користувачів та забезпечував якнайбільшу зручність та ефективність роботи із застосунком [87].

Для початку визначимо, які сторінки буде мати застосунок (див. рис. 3.18). Виходячи з вище зазначених вимог застосунок буде мати декілька сторінок:

– Список процесів (processList): включає набір описів процесів, які у майбутньому будуть запускатися, також містить форму для створення процесів і їхнього оновлення.

– Відображення деталей процесу (processDetail): перейти на цю сторінку можна із списку процесів, адже вона відображає вибраний процес. Дана сторінка включає набір активностей вибраного процесу, форму для їх створення чи оновлення.

– Список екземплярів процесу (processIntanceList): відображає список запланованих процесів, а при виборі одно з них відображає активності, які відносяться до нього.

– Дашборд (dashboard): надає загальну інформацію про процеси, їх стан, список виконаних активностей тощо.

Одним з важливих аспектів реалізації підходу UAF є дотримання чіткої структури папок, яка допомагає розподілити компоненти інтерфейсу за функціональними групами та зробити проект більш організованим [76].

Окрім того, така файлова структура дозволяє ефективно керувати кодом, а також зробити його більш зрозумілим та допомагає зменшити кількість помилок під час розробки чи підтримки застосунку.

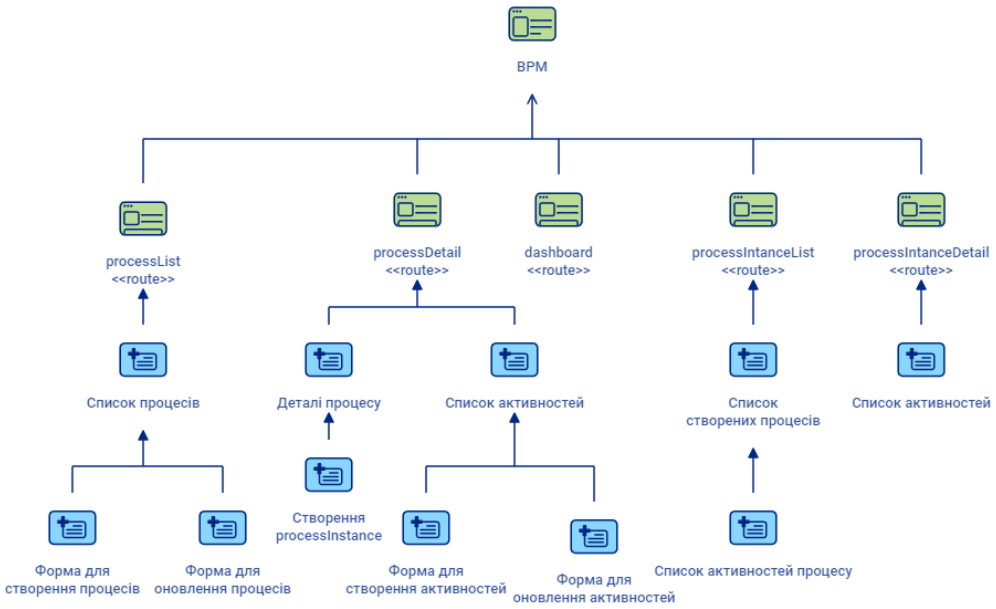


Рис. 3.18 – Наочне зображення структури користувацького інтерфейсу

Основною точкою входу користувацького інтерфейс є компонент під назвою SPA [87], який завантажує основні дані про застосунок. Цей компонент дозволяє користувачам швидко та зручно отримувати доступ до необхідної інформації, працювати з даними та взаємодіяти з системою.

```
const Spa = createVisualComponent({
  //@@viewOn:statics
  uu5Tag: Config.TAG + "Spa",
  //@@viewOff:statics

  //@@viewOn:propTypes
  propTypes: {},
  //@@viewOff:propTypes

  //@@viewOn:defaultProps
  defaultProps: {},
  //@@viewOff:defaultProps

  render() {
    //@@viewOn:render
    return (
      <Plus4U5.SpaProvider initialLanguageList={["en"]} skipAppWorkspaceProvider>
        <SimeBpm.Provider>
          <SimeBpm.PermissionProvider>
            <Plus4U5.RouteDataProvider>
              <SpaView />
            </Plus4U5.RouteDataProvider>
          </SimeBpm.PermissionProvider>
        </SimeBpm.Provider>
      </Plus4U5.SpaProvider>
    );
    //@@viewOff:render
  },
});
```

Рис. 3.19 – Реалізація компоненту SPA засобами UAF

Відповідно до рисунку 3.19 варто зазначити, що на нижньому рівні архітектури компоненту розміщуються два основні провайдери –

SimeBpmProvider та PermissionProvider. Перший відповідає за завантаження конфігурації застосунку, включаючи налаштування BPM рушія та розміщення інших сервісів, а другий визначає права доступу користувачів до різних функціональних можливостей застосунку, включаючи доступ до певних екранів та операцій [87]. Також він є корисним в контексті дотримання безпеки даних та запобіганню неприпустимим діям від несанкціонованих користувачів.

Так, управління правами доступу в застосунку є дуже важливим аспектом. Неправильне налаштування прав доступу може призвести до проблем з безпекою даних, витоку конфіденційної інформації чи виникненню помилок в процесі управління БП [88]. Тому важливо забезпечити, щоб права доступу були належно налаштовані та контрольовані в застосунку.

Наступним кроком після створення SPA компоненту та підключення відповідних провайдерів є розробка навігації для забезпечення зручності користувачів у взаємодії з застосунком. Для цього можна використовувати компонент RouterBar (див. рис. 3.20).

```
const RouteBar = createVisualComponent({
  //@@viewOn:statics
  uu5Tag: Config.TAG + "RouteBar",
  //@@viewOff:statics

  render() {
    //@@viewOn:private
    const [, setRoute] = useRoute();
    const session = useSession();
    const appActionList = [
      {
        children: <Lsi lsi={LsiData.processList} />,
        onClick: () => setRoute(Config.Routes.PROCESS_LIST),
        icon: Config.Icons.PLAY_LIST,
        collapsed: false,
      },
      {
        children: <Lsi lsi={LsiData.aboutMenu} />,
        onClick: () => setRoute(Config.Routes.ABOUT),
        icon: Config.Icons.INFORMATION,
        collapsed: true,
      },
    ];
    //@@viewOn:render
    return session.state === "authenticated" ? <Plus4U5App.RouteBar appActionList={appActionList} /> : null;
    //@@viewOff:render
  },
});
```

Рис. 3.20 – Реалізація компоненту RouterBar засобами UAF

Наступним кроком після створення компоненту навігації є розробка сторінок застосунку. Кожна сторінка відображає одну сутність. Наприклад,

процес чи його активність. Також на сторінці може бути форма для створення нового чи редагування існуючого запису або перегляду даних про ту чи іншу сутність [87].

Щодо форми для введення даних, то вони можуть містити різні типи елементів управління, такі як текстові чи числові поля, списки вибору, чекбокси тощо. Для валідації введених даних зазвичай використовуються різні правила, серед яких є, наприклад, обов'язковість заповнення певних полів, перевірка на унікальність та тип значення тощо [88].

Так чи інакше, створення зручного та зрозумілого інтерфейсу сторінок застосунку є дуже важливим етапом розробки, оскільки від цього залежить продуктивність взаємодії користувача із системою. Варто зазначити, що компоненти використовуваної бібліотеки UU5 компоненти можна конфігурувати залежно від потреб проекту: налаштовувати їх зовнішній вигляд, поведінку на дії користувача тощо. Наприклад, UU5 має компоненти для відображення даних у вигляді таблиць, списків, каруселей тощо, а також для валідації форм, взаємодії з сервером та багатьох інших завдань.

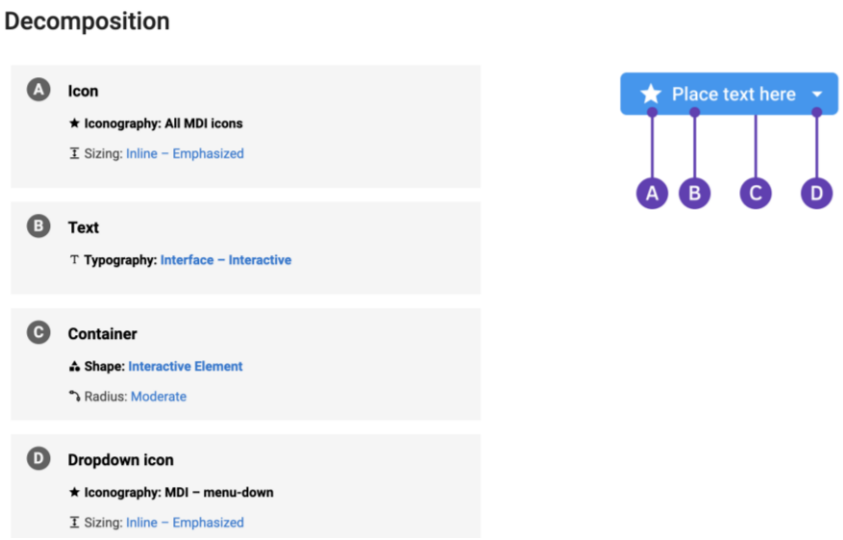


Рис. 3.21 – Приклад декомпозиції UU5 компонентів

Крім того, UU5 дозволяє створювати власні унікальні компоненти. Разом із наявністю великого набору готових компонентів з підтримкою конфігурації та можливістю до створення власних розробка користувацького інтерфейсу на базі UAF є швидким та відносно простим процесом [76]. Також цей фреймворк

реалізує ряд власних хуків, на кшталт useDataObject або useDataList. Вони дозволяють завантажити з серверу об’єкт або список об’єктів.

Ці хуки, як і багато інших, що надає фреймворк UAF, містять у собі handlerMap об’єкт, який надає методи обробки чи завантаження цих даних. Зазвичай ці методи викликаються автоматично при запиті на дані зі сторінки, але їх також можна викликати явно в коді [87].

3.10 Аналіз можливостей до покращення розроблюваного застосунку

З огляду на те, що розроблюваний застосунок є так званим MVP рішенням, існує безліч можливостей до його подальшого вдосконалення та масштабування. Зокрема, можна виділити наступні аспекти покращення застосунку для автоматизації бізнес процесів з розширеними інтеграційними можливостями.

В контексті покращення користувацького інтерфейсу можна виділити наступні аспекти:

– Провести UX та UI оптимізацію відповідно до відгуків перших користувачів застосунку.

– Додати використання анімованих компонентів для надання користувацькому інтерфейсу більш сучасного вигляду.

– Мінімізувати використання надлишкових бібліотек.

В контексті покращення серверної частини застосунку можна виділити наступні аспекти:

– Виконати оптимізацію запитів до бази даних та максимально можливо зменшити їх кількість шляхом розробки агреційних запитів.

– Реалізувати кешування даних засобами фреймворку UAF.

– Розробити механізм моніторингу та логування операцій задля швидшого пошуку потенційних помилок.

– Покращити реалізацію синхронізації із зовнішніми системами.

Проведення такого аналізу з певною періодичністю дозволяє постійно вдосконалювати розроблюваний застосунок, робити його більш ефективним та зручним у використанні.

3.11 Висновок до третього розділу

В третьому розділі даної кваліфікаційної роботи аргументовано рішення щодо необхідності розробки власного рішення для автоматизації бізнес процесів. Відповідно до проведеного дослідження розуміємо, що існує достатньо мало застосунків , що реалізують EDA підхід та надають розширені інтеграційні можливості.

Власне, щодо інтеграційних можливостей, можна зробити висновок, що особливу увагу, як найбільш поширеному підходу, слід приділити інтеграції з іншими системами через API. Окрім того, як показує практика, додатковою перевагою розроблюваного рішення над існуючими буде наявність можливості до інтеграції з розроблюваним застосунком через меседж брокер шляхом надсилання відповідних повідомлень.

Щодо вимог до застосунку, то можна зробити висновок, що вони не дуже відрізняються від типових вимог, описаних в другому розділі даної кваліфікаційної роботи. Спираючись на наявну інформацію, було прийнято рішення, що розроблюваний застосунок має бути так званим MVP рішенням – продуктом з мінімально достатнім для використання першими клієнтами функціоналом [62]. Це дозволить значно скоротити час розробки та розгортання застосунку.

Щодо вибору технологічного стеку, дослідивши популярні концепти та рішення, було вибрати використання MERN стеку разом із фреймворком UAF, який надає надзвичайно великий об’єм готового функціоналу та компонентів користувацького інтерфейсу. Відповідно можна зробити висновок, що використання фреймворку UAF дозволить більше часу та ресурсів приділити розробці бізнес логіки.

Також в третьому розділі було визначено ряд можливостей та аспектів для подальшого розвитку розроблюваного застосунку. Це допоможе покращити його продуктивність, розробити нові функціональні можливості та задовільнити користувачів.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Пожежна безпека в навчальних закладах

Пожежна безпека в навчальних закладах України регулюється правилами пожежної безпеки, затвердженими Міністерством освіти і науки від 15 серпня 2016 року Міністерством освіти та науки (наказ № 974), зареєстрованими в Міністерстві юстиції України 8 вересня 2016 року за номером 1229/29359 [89]. Ці правила спрямовані на захист та рятування учнів та персоналу навчальних закладів від ряду пожежних ризиків. Так, у випадку виникнення пожежі, дії персоналу мають бути спрямовані на забезпечення безпеки та ефективного проведення евакуаційних заходів [90].

Перед початком нового навчального року всім освітнім установам та закладам необхідно отримати схвалення від відповідної комісії, до якої входять представники органів державного контролю за пожежною безпекою [89].

Варто зазначити, що в приміщеннях дитячих дошкільних закладів діти мають бути розташовані так, щоб молодші з них знаходяться на нижніх поверхах. У багатоповерхових навчальних будівлях та школах-інтернатах класи також мають бути розміщені на нижніх поверхах [89].

Окрім того, в приміщеннях, де перебувають діти, підлога має бути закріплена та мати помірну здатність до димоутворення. Дитячі заклади, які працюють цілодобово, повинні бути забезпеченні чергуванням нічного персоналу. Зал очікування має бути забезпечений телефонним зв'язком [90]. Також черговий такого закладу має забезпечити особовий склад, на випадок виникнення пожежі, засобами індивідуального захисту органів дихання, комплектом ключів від евакуаційних дверей, портативним ліхтарем, а також знати точну кількість дітей, які ночують в закладі, знати їх місцезнаходження та мати здатність зателефонувати до найближчого пожежно-рятувального в разі потреби [89].

В загальноосвітніх навчальних закладах (за винятком тих, що призначені для дітей з розумовими та фізичними вадами) можливе створення дружин

молодих пожежників-рятувальників. У закладах і установах, де учні проживають цілодобово, обов'язково встановлюються обов'язки персоналу нічної служби, серед яких заборона на сон під час зміни [89]. Зал очікування має бути забезпечений телефонним зв'язком. Черговий персонал має окремо оснастити фільтруючими пристроями всіх дітей та обслуговуючий персонал закладу на випадок пожежі, а також володіти комплектом ключів від евакуаційних виходів та воріт, автомобільних під'їздів тощо.

Відповідно до інформації з джерела [89], в будівлях навчальних установ забороняється:

– Розміщення людей на горищах чи поверхах без наявності двох евакуаційних виходів.

– Проводити перепланування будівлі без дотримання визначених законом правил та будівельних норм.

– Встановлення ґрат або інших пристроїв на вікнах приміщень, де перебувають учасники навчального процесу, за винятком приміщень з обладнанням, що несе матеріальну цінність. Якщо ґрата все ж встановлено, то вони повинні бути розсувними або мати можливість бути відкритими.

– Видалення дверних полотен у виходах з коридорів та дверей евакуації.

– Використання саморобних нагрівальних пристроїв для опалення.

– Готувати їжу в приміщеннях, не призначених для цього.

– Допускати блокування шляхів евакуації.

– Створювати будь-які перешкоди на шляху евакуації або встановлювати на цьому шляху дзеркала чи фальш-двері.

– Проводити електрозварювальні або інші пожежонебезпечні роботи в присутності учасників навчального процесу.

– Використовувати свічки, газові ламп чи газові ліхтарі для освітлення.

– Допускати наявність відкритого вогню для нагрівання труб систем опалення, водопостачання тощо.

– Зберігання використаних обтиральних матеріалів у робочих місцях або кишенях одягу.

– Підключення електроприладів до джерел живлення без нагляду.

При дотриманні усіх цих вимог ризик виникнення пожежі в начальному закладі будь-якого типу значно зменшується [90]. Ба більше, навіть у випадку виникнення пожежі, дотримання вищезазначених вимог дозволяє значно полегшити проведення евакуації та мінімізувати наслідки пожежі.

4.2 Пожежна безпека в складських приміщеннях

Пожежна безпека в складських приміщеннях є критично важливою та потребує ретельного планування. Відповідно до цього необхідне створення конструкцій, які можуть витримати пожежу протягом певного часу, обмежити поширення вогню та диму, а також забезпечення ефективну та безпечну евакуацію людей [91]. Особлива увага приділяється специфіці товарів та речей, що зберігаються на складі, а також використанню відповідних систем протипожежного захисту.

Зберігання матеріалів, товарів та речей на складах має відповідати всім чинним вимогам щодо пожежної безпеки. Відповідно до інформації з джерела [**Помилка! Джерело посилання не знайдено.**], умови та можливість сумісного зберігання небезпечних і особливо небезпечних речовин і матеріалів повинні визначатися відповідно до ДСТУ 8828:2019 «Пожежна безпека. Загальні положення». Класифікація цих речовин повинна відповідати ДСТУ 4500-3:2008 «Вантажі небезпечні. Класифікація» і вказуватися у відповідних документах [93]. Небезпечні і особливо небезпечні речовини повинні зберігатися у спеціально призначених складах з вогнестійкими характеристиками.

Відповідно до вищезазначених особливостей пожежної безпеки та інформації з джерела [91], складські споруди мають розроблюватись, будуватись та експлуатуватись з врахуванням наступних аспектів:

– Збереження стійкості будівельних конструкцій протягом визначеного періоду.

– Обмеження можливостей до виникнення та поширення вогню та диму всередині будівлі.

– Проектування та використання приміщень таким чином, щоб мінімізувати розповсюдження пожежі на сусідні будівлі.

– Забезпечення безпечної евакуації людей та можливості до їх рятування.

– Врахування потреб та безпеки пожежно-рятувальних служб.

Оскільки в складських приміщеннях зазвичай зберігаються різноманітні матеріали та речовини, то їх розміщення має відбуватися з урахуванням усіх фізико-хімічних властивостей, сумісності, а також характеристик речовин, що використовуються для гасіння пожежі [**Помилка! Джерело посилання не знайдено.**].

Ефективне планування складського комплексу має велике значення для забезпечення пожежної безпеки. При розміщенні кількох будівель на території необхідно провести їх розділення на чіткі зони з однаковими вимогами щодо протипожежного захисту. Будівлі, де зберігаються матеріали з підвищеною небезпекою, повинні бути розташовані з підвітряного боку відносно інших [91]. Окрім того, між складськими приміщеннями необхідно забезпечити протипожежні розриви відповідно до встановлених норм [**Помилка! Джерело посилання не знайдено.**].

Операції з розпакування, перевірки та дрібного ремонту товарів чи речей, що зберігаються в складських приміщеннях, повинні відбуватися у спеціально виділених для цього місцях [93]. Також у складських приміщеннях та будинках різного рівня вогнестійкості дозволяється використовувати стелажі з горючих матеріалів висотою до 3 метрів за умови, що між ними та стінами залишається простір для проходу шириною не менше 1 метра [93]. Якщо в складі відсутні системи опалення, стелажі дозволяється розміщувати вздовж стін. Також стелажі, вищі за 3 метри, повинні бути зроблені з негорючих матеріалів [93].

Якщо застосовується безстелажний метод зберігання, товари слід укладати у штабелі, при цьому проходи навпроти дверей мають бути шириною, що рівна ширині дверей, але не меншою від 1 метра. У складі шириною понад 10 метрів потрібно створити поздовжній прохід завширшки не менше 2 метрів. Проходи між штабелями мають бути не менше 1 метра [93]. Складські приміщення, що знаходяться у підвальних або цокольних поверхах, повинні мати виходи

розміром 0,9 х 1,2 метра з приямками для випуску диму під час пожежі (допускається їх відсутність в разі наявності системи для димовидалення) [**Помилка! Джерело посилання не знайдено.**].

Також в складських приміщеннях допускається розміщувати робочі місця комірників з використанням скляних перегородок заввишки не менше 1,8 метра за умови, що вони не перешкоджають евакуації. Дерев'яні конструкції будинків та навісів складів повинні бути оброблені вогнезахисними засобами [91]. Для обігріву складських приміщень слід використовувати централізоване опалення, адже використання обігрівальних приладів з відкритими нагрівальними елементами заборонено [**Помилка! Джерело посилання не знайдено.**]. Освітлювальні прилади на складах мають бути розміщені над вільним від стелажів простором. Автоматичні вимикачі для освітлення повинні бути встановлені поза складськими приміщеннями [**Помилка! Джерело посилання не знайдено.**].

4.3 Система управління охороною праці як складова частина управління виробництвом

Управління охороною праці входить до складу загальної системи державного керівництва. Ця система розроблена для захисту прав працівників, що гарантовані законами України, та для виконання ними своїх обов'язків у сфері охорони праці на всіх рівнях управління: державному, регіональному, галузевому та виробничому [93].

Під поняттям «система» в контексті управляння охороною праці розуміють певну сукупність об'єктів управління (наприклад, машини, підприємства, галузі промисловості тощо) та елементів управління, які співпрацюють між собою для досягнення конкретної мети [94].

Справедливим буде твердження, що СУОП є ефективним засобом управління, метою якого є скорочення витрат підприємства та усунення наслідків недотримання визначених норм стосовно охорони праці.

Основна ідея впровадження СУОП полягає в запобіганні ймовірним виробничим аваріям та травмуванню персоналу. Окрім того, відповідно до інформації з джерела [94], СУОП визначає встановлює стандартний набір процедур для керівництва на всіх рівнях організації з метою дотримання безпечних умов праці. Ця мета досягається шляхом відповідності умов праці на кожному робочому місці вимогам чинного законодавства з охорони праці [94].

Як було зазначено раніше, СУОП складається з кількох ієрархічних рівнів: державного, галузевого, регіонального та виробничого [93]. На регіональному рівні існують кілька підрівнів згідно з адміністративно-територіальним поділом області: районний, міський, районний у містах, селищний [93]. Галузевий рівень може включати підрівні в об'єднаннях підприємств (корпорації, холдинги, акціонерні товариства тощо), які отримують певні повноваження в сфері охорони праці від підприємств [93]. На виробничому рівні зосереджені підприємства незалежно від їх форми власності та видів діяльності [93].

Відповідно до джерела [95], органи управління на регіональному рівні включають в себе обласні та районні державні адміністрації, органи місцевого самоврядування згідно з законом про місцеве самоврядування, а також регіональні (територіальні) підрозділи центральних органів виконавчої влади. їх повноваження включають управління, нагляд і контроль у сфері охорони праці, пожежної безпеки, техногенно-екологічної безпеки та управління надзвичайними ситуаціями (див. рис. 4.1).

Рис. 4.1 – Спрощена блок-схема СУОП

Відповідно до рисунку 4.1, фактори, що впливають на стан охорони праці на об'єкті управління, включають внутрішні та зовнішні чинники: умови праці, методи роботи та зовнішні загрози. Система управління охороною праці є циклічною, де інформація про стан безпеки та виконання рішень є основою для подальших дій і прийняття нових рішень [93].

Варто також зазначити, що наявність інформаційної бази є необхідною складовою для коректного функціонування всієї системи охорони праці на всіх її рівнях. Ця база формується з внутрішніх та зовнішніх джерел інформації і включає законодавчі акти, проектну та технічну документацію, звітні документи тощо [93]. Стосовно конкретних осіб на робочому місці, це стосується поточної інформації про події та засоби виробництва, необхідної для негайних дій, а також знань та вмінь, які зберігає людина у своїй пам'яті [94].

4.4 Вимоги до організації робочого місця користувача ПЕОМ

Організація робочого місця користувача ПЕОМ повинна відповідати нормативам щодо ергономіки, встановленим у відповідних документах та стандартах, зокрема ГОСТ 12.2.032. ССБТ, НПАОП 0.00-1.28-10 та ДСАНПІН 3.3.2.007-98, з урахуванням особливостей трудової діяльності [93]

Розташування монітора ПЕОМ має забезпечувати безпеку та комфорт робочого процесу, зокрема, верхній край екрану повинен бути на рівні очей користувача. Клавіатура повинна розміщуватись на відстані від 100 мм до 300 мм від краю столу або висувної полиці, а кут нахилу клавіатури має бути в межах від 5 до 15 градусів [96]. Клавіші клавіатури ПЕОМ повинні бути зручними та м'якими при натисканні, їх хід має бути однаковим з мінімальним опором натискання 0,25 Н та максимальним опором не більше 1,5 Н [93].

Крім того, конструкція робочого місця повинна сприяти підтримці оптимальної робочої позиції користувача. Відповідно до інформації з джерела [93], оптимальна поза для роботи перед комп'ютером передбачає наступне:

– Ноги повинні стояти на підлозі або на підставці для ніг, особливо, якщо ноги не досягають підлоги при правильній висоті стільця.

– Стегна мають бути розташовані горизонтально.

– Передпліччя мають бути вертикально.

– Лікті повинні бути згинатися під кутом 70-90 градусів відносно вертикальної площини.

– Зап'ястя повинні бути зігнуті під кутом не більше 20 градусів відносно горизонтальної площини.

– Голову слід нахиляти на кут 15-20 градусів відносно вертикальної площини.

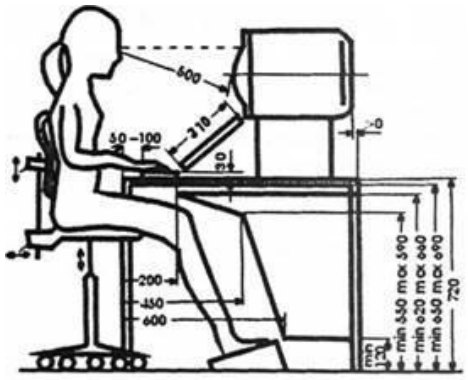


Рис. 4.2 – Схематичне зображення місця роботи користувача ПЕОМ

Відповідно до рисунку 4.2, робоче місце з ПЕОМ має бути комфортним для користувача. Крісло, стіл та інші елементи меблів повинні відповідати його потребам. Наприклад, невідповідне крісло може призвести до розвитку різних захворювань при довготривалій роботі. Також рекомендованою є наявність підставки для ніг. Вона ніг повинна мати можливість регулювання висоти на відстані до 150 мм та нахилу опорної поверхні в межах 20 градусів [96].

При конфігурації робочого місця слід дотримуватися вимог безпеки щодо відстані від світлових прорізів (вікон), розміщення робочих місць між собою, а також розташування екранів комп'ютерів та розміри робочих столів. Щоб забезпечити зручне розташування комп'ютерних елементів, слід розмістити їх на оптимальній відстані від очей користувача, з урахуванням розміру екрана та

символів на ньому [93]. Так чи інакше, важливо створити такі умови роботи, які б забезпечували комфорт для користувача ПЕОМ протягом робочого часу.

4.5 Вимоги до режиму праці й відпочинку при роботі з ПЕОМ

Враховуючи характер професійної діяльності, відповідно до джерела [93], можна виділити три групи працівників за класифікатором ДК-003-95:

– Розробники програм: їх робота вимагає великої концентрації та напруги зору, яка поєднується з нервово-емоційною напругою через тривале перебування у робочій позиції та маніпуляції з руками. Вони працюють у вільному темпі, постійно перевіряючи програми на помилки в умовах обмеженого часу.

– Оператори ПЕОМ: їх завданням є обробка та введення інформації за запитом. Робота відбувається з перервами різної тривалості, пов'язаними з іншими завданнями. Також їх роботу характеризує напруга зору, відносно легкі фізичні зусилля та середня нервова напруга, і вона виконується у вільному темпі.

– Оператори комп'ютерного набору: їх завдання полягає в введенні даних з високою швидкістю. Ця робота супроводжується підвищеним навантаженням на руки та напругою зору через фіксацію погляду на документах.

Державними санітарними правилами встановлено регламент роботи та відпочинку з ПЕОМ для кожної групи працівників при 8-годинних та 12-годинних робочих змінах [96]. У разі неможливості використання регламентованих перерв, тривалість безперервної роботи не повинна перевищувати чотирьох годин.

Для зменшення напруги та поліпшення фізичного та психологічного стану рекомендується використовувати активний відпочинок та психологічне розвантаження під час перерв або наприкінці робочого дня [93].

Для забезпечення здоров'я користувачів ПЕОМ, уникнення професійних захворювань та підтримки працездатності необхідно встановлювати регламентовані перерви для відпочинку протягом робочої зміни. Під час виконання робіт з ПЕОМ, що становлять не менше половини тривалості робочого дня, передбачаються такі перерви: обідня перерва для відпочинку та

їжі, перерва для особистих потреб, додаткові перерви, які вводяться для окремих професій з урахуванням особливостей трудової діяльності [96].

Варто зазначити, що у випадках, коли користувачі постійно скаржаться на зорову втому при роботі з ПЕОМ, допускається індивідуальний підхід до обмеження тривалості роботи та зміни змісту роботи або чергування з іншими видами діяльності [93].

Відповідно до джерела [97], при 8-годинному робочому дні встановлюються наступні режими праці та відпочинку в залежності від характеру роботи:

– Розробники програм із застосуванням ПЕОМ мають право на перерву тривалістю 15 хвилин кожну годину роботи.

– Оператори ПЕОМ мають право на перерву тривалістю 15 хвилин кожні дві години роботи.

– Оператори комп’ютерного набору мають право на перерву тривалістю 10 хвилин кожну годину роботи.

Якщо виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи із ПЕОМ не повинна перевищувати 4 години [93].

Для зменшення нервово-емоційного напруження, втоми зорового аналізатора та підвищення мозкового кровообігу, користувачам ПЕОМ рекомендується використовувати перерви для психофізіологічного розвантаження [93]. Психофізіологічне розвантаження також рекомендується проводити в кінці робочого дня в спеціально обладнаних приміщеннях - кімнатах психологічного розвантаження.

4.6 Охорона праці як система заходів та засобів, спрямованих на збереження здоров’я і працездатності користувачів комп’ютерів

Важливість надійності системи «людина – комп’ютер» визначається не лише функціональним станом самого комп’ютера, але й станом людини, яка ним керує. Психофізіологічні та емоційні навантаження, а також втома оператора

можуть призвести до помилок у комп’ютеризованих системах керування, що може призвести до серйозних економічних втрат [93].

У сфері адміністративного управління помилки працівників, які використовують комп’ютери, можуть мати серйозні наслідки. Тому важливо визначати та досліджувати фактори, що впливають на функціональний стан користувачів, такі як напруженість, втома та стрес, а також приймати відповідні профілактичні заходи [93].

Фізичні фактори виробничого середовища, такі як електромагнітні хвилі, шум, параметри мікроклімату, також можуть впливати на функціональний стан користувачів комп’ютерів. Таке виробниче середовище може викликати професійні захворювання та зумовити значні економічні втрати [93].

Урахування всіх вищеописаних факторів дозволяє розробити заходи для забезпечення безпеки, підвищення працездатності та збереження здоров’я користувачів комп’ютерів [95]. Варто звернути увагу на три основні групи діяльності професіоналів, які використовують комп’ютери:

– Перша група включає діяльність, при якій виконуються прості, але багаторазово повторювані операції, зокрема до цієї групи можна віднести роботу операторів комп’ютерного набору.

– Друга група займається виконанням логічних операцій, що постійно повторюються, наприклад, у сфері інженерії.

– Третя група діяльності передбачає прийняття рішень в умовах, коли заздалегідь не відомий алгоритм, зокрема до цього можна віднести роботу програмістів чи диспетчерів.

Наукові дослідження показують, що користувачі комп’ютерів частіше зазнають негативного впливу на здоров'я, особливо щодо зору. Професійні користувачі ПЕОМ можуть відчувати різноманітні негативні симптоми: втому, біль в очах, а також різні порушення зору [95]. Відповідно до цього, враховуючи високу поширеність користування комп’ютерами в сучасному світі, важливо розробляти стратегії збереження здоров'я для користувачів цих технологій.

4.7 Висновок до четвертого розділу

В четвертому розділі даної кваліфікаційної роботи було описано досить важливу інформацію. Зокрема, зазначено актуальні правила пожежної безпеки в навчальних закладах та в складських приміщеннях. Наведено нормативні документи, що регулюють правила пожежної безпеки та вказано основні її аспекти. Також було оглянуто правила роботи користувачів ПЕОМ, зокрема поради і вимоги до організації робочого місця та вимоги до режиму праці та відпочинку. Відповідно до наведеної інформації, особливу вагу також слід приділити засобам та заходам, спрямованим на збереження здоров’я користувачів комп’ютерів. Можна зробити висновок, що дотримання цих правил дозволяє не тільки вберегти здоров’я користувачам та операторам ПЕОМ, але й збільшити ефективність та надійність виконуваних ними робіт. Також було описано основні аспекти СУОП в якості складової частини управління виробництва.

ВИСНОВКИ

В результаті виконання даної кваліфікаційної роботи освітнього рівня «Магістр» було проведено дослідження сучасних застосунків для автоматизації бізнес процесів, оглянуто їх характерні особливості, а також спроектовано та розроблено якісно нове MVP рішення з розширеними інтеграційними можливостями, що базується на задах подієво-орієнтованої архітектури. Використання таких підходів зумовлено інформацією, отриманою в ході дослідження поняття бізнес процес та застосунків для його автоматизації. Варто зазначити, що використання для розробки MERN стеку разом із фреймворком UAF дозволило створити ефективний та оптимізований застосунок з розширеними інтеграційними можливостями за умовно не великий проміжок часу.

В першому розділі даної кваліфікаційної роботи було проведено дослідження поняття бізнес процес та описано основні аспекти, пов’язані з його управлінням. Так, найбільш точним визначенням можна вважати наступне: бізнес процес – це набір кроків чи задач, виконання яких створює послугу чи продукт або розпочинає виконання нового бізнес процесу. Вперше він був застосований в 1776 році економістом на ім’я Адам Сміт задля оптимізації робіт на фабриці по виробництву шпильок [11]. Відповідно до наведеної інформації можна зробити висновок, що впровадження та використання бізнес процесів значно збільшує ефективність та конкурентоспроможність підприємства. Також в цьому розділі було описано три основні методи покращення бізнес процесу: оптимізацію, реінжиніринг та автоматизацію. Справедливим буде твердження, що автоматизація – найбільш актуальний та популярний метод в наш час. Окрім того, в першому розділі було подано ряд класифікацій бізнес процесу, розуміння яких дозволяє провести глибший аналіз та краще використовувати методи покращення БП.

В другому розділі даної кваліфікаційної роботи було проведено докладне дослідження економічного впливу застосунків для автоматизації бізнес процесів та розглянуто найбільш поширені та популярні рішення. Так, світовий ринок

таких застосунків оцінюється в $9.54 млрд, а між 2023 та 2030 роками очікується щорічний приріст на 33% [32]. Відповідно можна зробити висновок, що сфер поширення застосунків та систем, як і їх самих, ставатиме дедалі більше з кожним роком. Щодо розглянутих рішень, то всі вони покривають певний спектр задач, що залежить від ряду функціональних та нефункціональних вимог. Найбільш цікавими та доскональними з технічної точки зору рішеннями, відповідно до отриманої в ході дослідження інформації, можна вважати Appian, Kissflow та Corezoid. Окрім того, в другому розділі було описано архітектурі аспекти досліджуваних застосунків, серед яких варто виділити EDA та PDA підходи. Також було досліджено найбільш популярні програмні засоби для розробки застосунків для автоматизації БП, серед яких для розробки серверної частини використовують Node.js або Java, для клієнтського інтерфейсу – React або Angular, а в якості баз даних прийнято використовувати MongoDB чи PostgreSQL.

В третьому розділі даної кваліфікаційної роботи було проведено проектування та розробку власного застосунку для автоматизації бізнес процесів. Аргументацією до такого рішення може слугувати те, що попри всі переваги оглянутих застосунків, ніша BPMS для малих підприємств та бізнесу на EDA підході з можливістю до повної автоматизації БП, інформативним та зручним користувацьким інтерфейсом та розширеними інтеграційними можливостями фактично пустує. Власне, щодо інтеграційних можливостей, завдяки проведеному дослідженню розуміємо, що окрему увагу слід приділити інтеграціям через API та меседж брокери як найбільш універсальних та надійних з технічної точки зору підходах. Також в даному розділі було визначено переваги використовуваного для розробки MERN стеку та фреймворку UAF, за допомогою яких вдалось відносно швидко розробити власне MVP рішення з великим потенціалом до подальшого покращення.

В розділі «Охорона праці та безпека в надзвичайних ситуаціях» було розглянуто ряд критичних аспектів, пов’язаних з дотриманням правил пожежної безпеки в навчальних закладах та складських приміщеннях та подано приклади нормативних документів, що регулюють ці правила. Також в цьому розділі

наведено інформацію про правила робити із ПЕОМ. Можна зробити висновок, що дотримання описаних правил, вимог та рекомендацій дозволять не тільки зберегти здоров’я користувачам ПЕОМ, але й зробить їх роботу більш ефективно, надійною та швидкою.

ПЕРЕЛІК ДЖЕРЕЛ

1 Тененський М., Галюк М. Аналіз використання BPM застосунків. Природничі та гуманітарні науки. Актуальні питання: матеріали VI Міжнар. студ. наук.-техн. конф. (м. Тернопіль, 27–28 квіт. 2023 р.). Тернопіль: ТНТУ, 2023. С. 181–182.

2 Галюк М., Тененський М. Вплив BPM застосунків на ефективність підприємства. Природничі та гуманітарні науки. Актуальні питання: матеріали VI Міжнар. студ. наук.-техн. конф. (м. Тернопіль, 27–28 квіт. 2023 р.). Тернопіль: ТНТУ, 2023. С. 128–129.

3 Business process management. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Business_process_management

4 Що таке бізнес-процеси та як їх описувати? Дія. Бізнес. URL: https://business.diia.gov.ua/handbook/sistematizacia-biznes-procesiv-2/so-take-biznes-procesi-ta-ak-ih-opisuvati

5 Business Process Management / ed. H. Volzer, P. Soffer, S. Sadiq. Berlin: Springer International Publishing, 2012. 434 p.

6 Eid-Sabbagh R.-H. Business Process Architectures: Concepts, Formalism, and Analysis. ResearchGate. August 2015. URL: https://www.researchgate.net/publication/283514454_Business_Process_Architectures_Concepts_Formalism_and_Analysis

7 Панцир П. Імплементація цифрових інструментів в структуру бізнес-процесів організації. Інформаційні системи та технології: матеріали VII наук.-техн. конф. (м. Тернопіль, 10–11 груд. 2019 р.). Тернопіль: ТНТУ, 2019. С. 81.

8 Panagacos T. The Ultimate Guide to Business Process Management: Everything you need to know and how to apply it to your organization. Seattle: Amazon Digital Services LLC, 2012. 186 p.

9 Бізнес-процес (Business Process). Вікіпедія. Вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/Бізнес-процес

10 Pratt Mary K., Roy M., McLaughlin E. Business process. TechTarger. URL: https://www.techtarget.com/searchcio/definition/business-process

11 Brandall B., Henshall A. The Complete Guide to Business Process Management. URL: https://www.process.st/wp-content/uploads/2017/03/The-Complete-Guide-to-Business-Process-Management.pdf

12 The history of business process managment methodologies. Holtz consulting. URL: https://www.holtz.sa/the-history-of-business-process-managment-methodologies/

13 Wyatt Frank J. A brief history of process management to the modern day. Medium. February 13, 2018. URL: https://medium.com/business-process-management-software-comparisons/a-brief-history-of-process-management-to-the-modern-day-2f90d12d8e99

14 Демиденко Д. Економічний зміст та необхідність бізнес-процесів у конкурентоспроможності підприємства. Сучасні підходи до управління підприємством: збірник тез доповідей VIII Всеукр. наук.-практ. конф. (м. Київ, 6 квіт. 2017 р.). Київ: КПІ, 2017. С. 67.

15 Business process. Wikipedia, the Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Business_process

16 Doglio F. What is a Business Process and Why Should You Care. Camunda. February 29, 2024. URL: https://camunda.com/blog/2024/02/what-is-a-business-process-why-important/

17 Тігарєва В. А., Станкевич І. В. Аналіз існуючих підходів та методів оцінювання бізнес-процесів підприємств та організацій. Вісник Кременчуцького національного університету ім. Михайла Остроградського. 2016. № 3 (1). С. 113–122.

18 Business process re-engineering. Wikipedia, the Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Business_process_re-engineering

19 Rosing M., Scheel H., Scheer A.-W. The Complete Business Process Handbook: Body of Knowledge from Process Modeling to BPM. Morgan Kaufmann, 2014. Vol. 1. 776 p.

20 Морозова К. Автоматизація бізнес-процесів: сучасний тренд чи спосіб підвищення ефективності бізнесу? EBA. URL: https://eba.com.ua/

avtomatyzatsiya-biznes-protsesiv-suchasnyj-trend-chy-sposib-pidvyshhennya-efektyvnosti-biznesu/

21 Business Process Strategy to Automate End-to-End Business Process. URL: https://kissflow.com/workflow/bpm/business-process-strategy/

22 How to Automate Business Processes In 6 Simple Steps (+ Tools List). Process.st. URL: https://www.process.st/how-to-automate-business-processes/

23 What is process automation? TIBCO. URL: https://www.tibco.com/glossary/what-is-process-automation

24 Автоматизація бізнес-процесів: все про автоматизацію бізнесу. BX–master. URL: https://bx-master.com/ua/news/biznes-protsessy/avtomatizatsiya-biznes-protsessov-vsye-ob-avtomatizatsii-biznesa

25 Lawton G., Tucci L. Business process automation (BPA). TechTarger. URL: https://www.techtarget.com/searchcio/definition/business-process-automation

26 Strauss L. What is business process management (BPM), and why should you care? Zapier. October 3, 2023. URL: https://zapier.com/blog/business-process-management/

27 Березін В. Управління бізнес-процесами підприємства. SMART business. 23 жовт., 2019. URL: https://www.smart-it.com/uk/2019/10/business-process-management/

28 What is Process Automation? TIBCO. URL: https://www.tibco.com/reference-center/what-is-process-automation

29 What is BPM? Workflow Management Coalition. URL: https://wfmc.org/what-is-bpm/

30 Schmelzer R. From Process Automation To Autonomous Process. Forbes. February 14, 2020. URL: https://www.forbes.com/sites/cognitiveworld/2020/02/14/from-process-automation-to-autonomous-process/?sh=40e0306863b6

31 The Extensive Guide to Business Processes for 2024. Kissflow. March 20, 2024. URL: https://kissflow.com/workflow/bpm/business-process/#types

32 Workflow Management System Market Size, Share & Trends Analysis Report By Component (Software, Services), By Software, By Service, By Deployment, By Vertical, By Region, And Segment Forecasts, 2023-2030. Grand View Research.

URL: https://www.grandviewresearch.com/industry-analysis/workflow-management-systems-market

33 Adams R. What is the business process management (BPM) lifecycle? EPC. September 30, 2019. URL: https://www.epc.co.uk/media-centre/blog/item/what-is-the-business-process-management-bpm-lifecycle

34 Drucker P. Management Challenges for the 21st Century. New York: Taylor & Francis, 2012. 208 p.

35 Information Information technology platform for the selection and analytical processing of information on COVID-19 / O. Duda, N. Kunanets, S. Martsenko, V. Nykytyuk**,** V. Pasichnyk. 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT) (Lviv, Ukraine 22-25 Sept. 2021). Lviv, 2021. Vol. 2. P. 231-328. DOI: 10.1109/CSIT52700.2021.9648839.

36 Gavala G. 5 Business Process Management Statistics You Should Know. Solutions Review. URL: https://solutionsreview.com/business-process-management/business-process-management-statistics-you-should-know/

37 Артюх О., Чернишова Л. Оцінка результативності бізнес-процесів на підприємствах роздрібної торгівлі: огляд підходів. Економіка та суспільство. 2022. № 40. URL: https://economyandsociety.in.ua/index.php/journal/article/view/1443

38 Іздебський В. BPM-система (BPMS): як вибрати найкраще рішення для оцифрування бізнесу. WEZOM. URL: https://wezom.com.ua/ua/blog/bpm-sistema-bpms-kak-vybrat-luchshee-reshenie-dlja-otsifrovki-biznesa

39 Як обрати BPM систему: 7 кроків. URL: https://pmb.com.ua/uk/blog/kak-vybrat-bpm-sistemu-chek-list-iz-7-shagov/

40 Monday.com. Wikipedia, the Free Encyclopedia. URL: https://uk.wikipedia.org/wiki/Monday.com

41 Esam A. Top 10 Process Management Software to Drive Business Growth. beSlick. June 2023. URL: https://beslick.com/bpm-software/

42 Lambden D., Macey J. Monday.com Review: Features, Pros and Cons. tech.co. January 23, 2024. URL: https://tech.co/project-management-software/monday-review#what

43 Takahara J. Zapier: What it is and how to use it. Pipedrive. April 15, 2024. URL: https://support.pipedrive.com/article/zapier-what-it-is-and-how-to-use-it

44 Top 10 BEST Business Process Management Software: BPM Tools Of 2023. Software Testing Help. August 21, 2023. URL: https://www.softwaretestinghelp.com/bpm-tools/

45 Панцир П. П. Імплементація цифрових інструментів в структуру бізнес-процесів організації: дипломна робота / Тернопільський національний технічний університет ім. Івана Пулюя. Тернопіль: ТНТУ, 2020. 112 с. URL: https://elartu.tntu.edu.ua/handle/lib/31749

46 Camunda Platform. Reviews and Ratings. TrustRadius. URL: https://www.trustradius.com/products/camunda-bpm/reviews?qs=pros-and-cons#overview

47 Никитюк В. В., Галюк М. В., Тененський М. В. Аналіз використання та порівняння мікросервісної і монолітної архітектур. Актуальні задачі сучасних технологій: матеріали XII міжнар. наук.-практ. конф. молодих учених та студентів (Тернопіль, 6-7 груд. 2023 р.). Тернопіль: Паляниця В. А., 2023. С. 374–375.

48 IcePanel. Monolithic vs. microservices architectures. Medium. March 7, 2023. URL: https://icepanel.medium.com/monolithic-vs-microservices-architectures-e71c75b252d1

49 Никитюк В. В., Тененський М. В., Орловська А. В. Аналіз використання EDA для вирішення проблем сучасних застосунків та систем. Інформаційні моделі, системи та технології: матеріали XI наук.-техн. конф. (Тернопіль, 13-14 груд. 2023 р.). Тернопіль: ТНТУ, 2023. С. 89–90. (Інформаційні системи та технології, кібербезпека).

50 Welsh M., Brewer E. SEDA: an architecture for well-conditioned, scalable internet services. ACM SIGOPS Operating Systems Review. 2001. Vol. 35, Issue 5. P. 230–243. URL: https://dl.acm.org/doi/10.1145/502059.502057

51 Comparison of cloud OS Corezoid and traditional BPM-systems. Corezoid. September 16, 2015. URL: https://corezoid.com/blog/comparison-of-cloud-os-corezoid-and-traditional-bpm-systems/

52 Тененський М. В. Порівняння баз даних MonoDB та PostgresQL в контексті розробки сучасних веб-застосунків. Інформаційні моделі, системи та технології: матеріали XI наук.-техн. конф. (Тернопіль, 13-14 груд. 2023 р.). Тернопіль: ТНТУ, 2023. С. 184–185. (Комп’ютерні системи та мережі)

53 DB-Engines Ranking - Trend Popularity. FB-Engines. URL: https://db-engines.com/en/ranking_trend

54 Au-Yeung J. Angular Basics: A Primer for Getting Started with TypeScript in Angular. May 15, 2023. URL: https://www.telerik.com/blogs/angular-basics-primer-getting-started-typescript-angular

55 Hantoush Y. Light Business Process Management (BPM) in Nodejs / Typescript. Yasir Hantoush Blog. June 21, 2022. URL: https://yasirhantoush.hashnode.dev/light-business-process-management-bpm-in-nodejs-typescript

56 Васильєв М. Вибір технологічного стеку та баз даних для розробки програмного забезпечення підприємства. Almexoft. URL: https://almexoft.com.ua/news/choosing-technologies-for-enterprise-software-development

57 Бондаренко О., Сиротюк Н. Технічне завдання: ідеальна формула для успішного проєкту. Freelancehunt.com. 16 січня, 2024. URL: https://freelancehunt.com/blog/tiekhnichnie-zavdannia-idiealna-formula-dlia-uspishnogho-proiektu/

58 Функціональні вимоги. Вікіпедія. Вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/Функціональні_вимоги

59 Нефункціональні вимоги. Вікіпедія. Вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/Нефункціональні_вимоги

60 The trick of using an entity task layout in your BPM. URL: https://appworks-tips.com/2020/09/18/066_the_trick_of_using_an_entity_task_layout_in_your_bpm/

61 Adams T. Determining Your BPM Software Requirements. BPMinstitute.org. URL: https://www.bpminstitute.org/resources/articles/determining-your-bpm-software-requirements

62 Minimum viable product. Wikipedia, the Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Minimum_viable_product

63 Rouse M., Becker R. Minimum Viable Product. Techopedia. August 14, 2020. URL: https://www.techopedia.com/definition/27809/minimum-viable-product-mvp

64 Brown E. Web Development with Node and Express: Leveraging the JavaScript Stack. 2nd edit. Sebastopol, California: O'Reilly, 2019. 340 p.

65 MongoDB Documentation. URL: https://www.mongodb.com/docs/#welcome-to-the-mongodb-documentation

66 Bradshaw S., Brazil E., Chodorow K. MongoDB: The Definitive Guide: Powerful and Scalable Data Storage. 3rd edit. Sebastopol, California: O'Reilly Media, 2019. 511 p.

67 Brown E. Web Development with Node and Express: Leveraging the JavaScript Stack. 2nd edit. Cambridge: O'Reilly Media, 2018. 325 p.

68 Flanagan D. JavaScript. The Definitive Guide. Master the World's Most-Used Programming Language. 7th edit. USA: O'Reilly Media, 2020. 704 p.

69 Alexandra & Andrew. Node.JS vs Java: Why Compare? IntexSoft. July 15, 2019. URL: https://intexsoft.com/blog/node-js-vs-java-why-compare/

70 Smiryakhin A., Boyko B., Volkov D. Java vs NodeJS on AWS Lambda: Benchmark Survey - VARTEQ Inc. URL: https://varteq.com/java-vs-nodejs-on-aws-lambda-benchmark-survey/

71 Посібник: знайомство з React. URL: https://uk.legacy.reactjs.org/tutorial/tutorial.html

72 Прус О. ЩО таке SPA-додатки Wezon. 9 груд., 2019. URL: https://wezom.com.ua/ua/blog/chto-takoe-spa-prilozheniya

73 What is a message broker? IBM. URL: https://www.ibm.com/topics/message-brokers

74 RabbitMQ. One broker to queue them all. URL: https://www.rabbitmq.com/

75 Mawa J. Comparing the best Node.js schedulers. Logrocket. July 11, 2023. URL: https://blog.logrocket.com/comparing-best-node-js-schedulers/#node-cron

76 uuApp Framework & The Architecture. Unicorn. URL: https://docs.plus4u.net/framework

77 Unicorn Universe Process. URL: https://uuapp.plus4u.net/uu-webkit-maing02/6bb0727659ca4d0a95409bfcc959a041

78 What is application integration? Talend. URL: https://www.talend.com/resources/what-is-application-integration/

79 What is application integration? IBM. URL: https://www.ibm.com/topics/application-integration

80 What is Application Integration? AWS. URL: https://aws.amazon.com/what-is/application-integration/

81 OAuth 2.0. URL: https://oauth.net/2/

82 Terra J. What is Client-Server Architecture? Everything You Should Know. Simplilearn. August 7, 2023. URL: https://www.simplilearn.com/what-is-client-server-architecture-article

83 Клієнт-серверна архітектура. Вікіпедія. Вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура

84 Model–view–controller. Wikipedia, the Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Model-view-controller

85 Data access object. Wikipedia, the Free Encyclopedia. URL: https://uk.wikipedia.org/wiki/Data_access_object

86 Routing in Node. GeeksforGeeks. January 15, 2024. URL: https://www.geeksforgeeks.org/routing-in-node-js/

87 Roldan C. S. React 18 Design Patterns and Best Practices: Design, build, and deploy production-ready web applications with React by leveraging industry-best practices 4th edit. Birmingham: Packt Publishing, 2023. 524 p.

88 Subramanian V. Pro MERN Stack. Full Stack Web App Development with Mongo, Express, React, and Node. 2nd edit. California: Apress Berkeley, 2019. 542 p.

89 Про затвердження Правил пожежної безпеки для навчальних закладів та установ системи освіти України: наказ МОН України від 15.08.2016 р. № 974. URL: https://zakon.rada.gov.ua/laws/show/z1229-16#Text

90 Рожков А. Пожежна безпека у закладах освіти: рекомендації експерта. URL: https://www.auc.org.ua/novyna/pozhezhna-bezpeka-u-zakladah-osvity-rekomendaciyi-eksperta

91 Рожков А. Основні напрями забезпечення пожежної безпеки на складах. Охорона праці і пожежна безпека. URL: https://oppb.com.ua/news/osnovni-napryamy-zabezpechennya-pozhezhnoyi-bezpeky-na-skladah

92 The use of abstract moore automaton to control the sensors of a service-oriented alarm and emergency network / O. Kryazhych, V. Itskovych, K. Iushchenko, V. Hrytsyshyna, D. Bruvier, V. Nykytyuk, I. Bodnarchuk. Scientific Journal of TNTU. Ternopil: TNTU, 2023. Vol 109. No 1. P. 111–120.

93 Голінько В. І., Іконніков М. Ю, Лебедєв Я. Я. Охорона праці в галузі інформаційних технологій: навч. посіб. Дніпропетровськ: НГУ, 2015. 246 с.

94 Охорона праці в галузі: конспект лекцій / уклад. О. І. Полукаров; Київ. політехн. ун-т. Київ, 2014. Лекція 2: Система управління охороною праці в організації. URL: https://opcb.kpi.ua/wp-content/uploads/2014/09/Лекція_2.pdf

95 Яскілка В. Я., Олійник М. З. Охорона праці в галузі: конспект лекцій. Тернопіль: ТНТУ ім. І. Пулюя, 2015. 55 с.

96 Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98: постанова Головного державного санітарного лікаря України від 10.12.1998 р. № 7. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text

97 Кирильчук М. Роботодавці повинні надавати працівникам перерви у роботі за комп’ютером і оплачувати їх - Держпраці. ЛІГА ЗАКОН. 15 груд., 2020. URL: https://buh.ligazakon.net/news/200332_robotodavts-povinn-nadavati-pratsvnikam-pererv i-u-robot-za-kompyuterom--oplachuvati-kh--derzhprats

ДОДАТКИ

Додаток А

Публікація 1

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XI НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

13-14 грудня 2023 року

ТЕРНОПІЛЬ
2023

УДК 004.65
М.В. Тененський
Тернопільський національний технічний університет імені Івана Пулюя

ПОРІВНЯННЯ БАЗ ДАНИХ MONODB ТА POSTGRESQL В КОНЕКСТІ РОЗРОБКИ СУЧАСНИХ ВЕБ-ЗАСТОСУНКІВ

M.V.Tenenskyi

COMPARISON OF MONGODB AND POSTGRESQL DATABASES IN THE CONTEXT OF MODER WEB APPLICATIONS DEVELOPMENT

Ключові слова: Бази даних, веб-застосунки, MongoDB, PostgreSQL
Key words: Databases, MongoDB, PostgreSQL, web application

В наш час стрімкого розвитку IT індустрії розробники часто стикаються з певними труднощами при виборі бази даних для того чи іншого веб-застосунку, яка є однією з найважливіших його частиною. Особливо справедливим це твердження є з врахування факту постійного збільшення обсягу та структурності даних, а також зростання кількості загроз на кшталт фішингових атак тощо. Відповідно, щоб захистити інформацію та мати змогу до її ефективного оброблення, варто зробити вибір серед найбільш ефективних та вживаних баз даних, доступних на ринку. Такими варіантами є усім відомі MongoDB та PostgreSQL, які фактично стали сучасними титанами в сфері управління даних (див. рис. 1), пропонуючи своїм користувачам ряд унікальних особливостей [1].

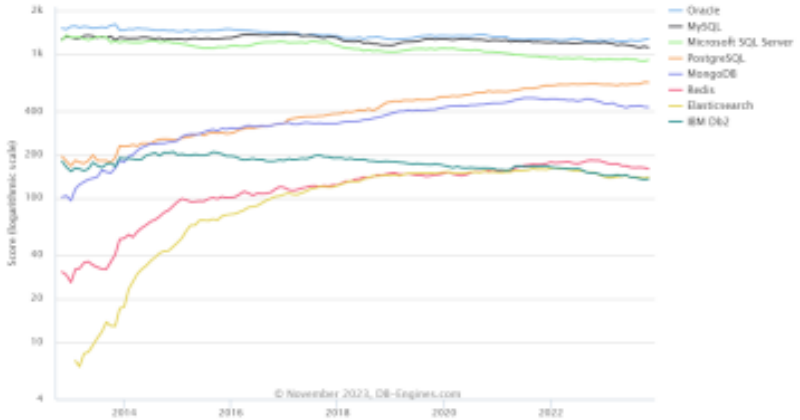


Рисунок 1 – Графік популярності БД станом на листопад 2023 року

Перш за все варто зрозуміти ключові відмінності між вищезгаданими базами даних, визначити їх сильні та слабкі сторони в контексті розробки сучасних веб-застосунків. Так, MongoDB – це високопродуктивна та гнучка нереляційна база даних, яка забезпечує ефективну обробку неструктурованих або напівструктурованих даних. Інформація зберігається у вигляді BSON (Binary JSON) документів з довільною структурою [2]. Фактично BSON – це формат серіалізації, який архівує JSON-подібні дані, завдяки чому забезпечується висока швидкість виконання запитів та є зручним для Node.js розробників. MongoDB дуже легко використовувати в розробці веб-застосунків, оскільки NoSQL бази даних простіші за своєю суттю та є легшими для розуміння. Мовою запитів в MongoDB служить так звана MQL, яка має багато схожих до SQL функцій, а також підтримує велику кількість мов програмування. Окрім того, ця база має розподілену архітектуру, що дозволяє підтримувати взаємодію між її компонентами на

різних платформах [3]. Фактично це означає необмежений потенціал до горизонтальної масштабованості за потреби, що робить її чудовим вибором для веб-застосунків із швидко зростаючими наборами даних. Окрім того, MongoDB пропонує використання шифрування на стороні клієнта за допомогою протоколів TLS та SSL, які значно зменшують вразливість даних та дозволяють шифрувати трафік [3]. З додаткових особливостей варто відмітити підтримку транзакцій та виконання агрегаційних запитів із забезпечення узгодженості та цілісності даних.

В свою чергу, PostgreSQL – це реляційна база даних, в основі якої лежить збереження документів у вигляді структурованих об'єктів з наперед визначеною структурою полів. На практиці це означає додатку як додаткові витрати часу на розробку веб-застосунків, так і збереження даних в більш керованому та читабельному форматі із їх суворої узгодженістю [2]. Окрім того, PostgreSQL підтримує традиційний SQL синтаксис та має монолітну архітектуру. Компоненти даної бази об'єднані таким чином, що масштабування може здійснюватися лише в межах та за рахунок ресурсів машини, на якій вона розгорнута. Варто зазначити також і те, що PostgreSQL чудово підходить для виконання транзакійних запитів завдяки вбудованим засобам резервування та захисту даних від різного виду збоїв. Ось чому PostgreSQL часто використовується в розробці банківських веб-застосунків.

Якщо ж порівнювати швидкодію MongoDB та PostgreSQL, то відповідно до досліджень, для яких використовувався набір запитів, що імітували реальні сценарії та являли собою просторові та часові предикати, то за відсутності індексації даних PostgreSQL буде перевершувати MongoDB. Якщо враховувати використання індексів, що є важливим аспектом швидкої будь-якого сучасного веб-застосунку, то середній час запиту MongoDB зміниться більш якісно, ніж в PostgreSQL [3].

Підсумовуючи все вищезгадане в процесі дослідження використання MongoDB та PostgreSQL в контексті розробки сучасних веб застосунків, можна зробити висновок, кожна з цих баз даних підходить для вирішення різного спектру вимог та задач, що виникають в розробці того чи іншого веб-застосунку. Так, MongoDB підходить для розробки веб-застосунків, для яких характерною особливістю є гнучкість структури інформації чи необхідність до проведення масштабування бази даних. Це ідеальне рішення для систем управлінням контенту, невеликих інтернет-магазинів чи тих веб-застосунків, де набір даних не є чітко структурованим та може мінятись з плином часу. З іншої сторони PostgreSQL варто використовувати при розробці застосунків, функціонування яких вимагає дотримання цілісності даних та виконання складних агрегаційних запитів. PostgreSQL часто використовують в розробці фінансових та медичних веб-застосунків. І хоча PostgreSQL може використовуватись в застосунках з виском рівнем масштабованості, все ж вона проявляє себе в цьому плані значно гірше, ніж MongoDB з її горизонтальною масштабованістю. Так чи інакше, не рідко зустрічаються застосунки, які використовують ці бази даних одночасно для покриття як різносторонніх технічних, так і бізнес потреб.

Література

1. DB-Engines Ranking – Trend Popularity. *DB-Engines*. URL: https://db-engines.com/en/ranking_trend

2. Dutta S. MongoDB vs PostgreSQL: Choosing the Right Database for Your Project. *Spinkle*. URL: https://www.sprinkledata.com/blogs/mongodb-vs-postgres

3. MongoDB Vs PostgreSQL: A comparative study on performance aspects / A. Makris et al. *GeoInformatica*. 2021. № 25. P. 243–268. URL: https://link.springer.com/article/10.1007/s10707-020-00407-w

Публікація 2

Матеріали XII Міжнародної науково-практичної конференції молодих учених та студентів
«АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ» – Тернопіль, 6-7 грудня 2023 року

УДК 004.415.2

В. В. Никитюк, канд. тех. наук, доц., М. В. Галюк, М. В. Тененський
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

АНАЛІЗ ВИКОРИСТАННЯ ТА ПОРІВНЯННЯ МІКРОСЕРВІСНОЇ І МОНОЛІТНОЇ АРХІТЕКТУР

V. V. Nykytyuk, Ph.D., Assoc. Prof., M. V. Haliuk, M. V. Tenenskyi

ANALYSIS OF THE USE AND COMPARASION OF MICROSERVICE AND MONOLITHIC ARCHITECTURE

Сучасна розробка застосунків та систем складається з рядом складнощів, серед яких не останнє місце посідають вимоги до гнучкості, масштабованості та швидкості функціонування кінцевого продукту тощо. Дані показники на пряму залежить від типу обраної архітектури ПЗ чи застосунку. Так, прийнято виділяти так звану монолітну та мікросервісну архітектури. Так, мікросервісний підхід набув широкого поширення за останні п'ять років [1]. Натомість використання монолітного підходу, хоч він це все ще і вважається стандартною моделлю створення ПЗ, поступово йде на спад. Розглянемо детальніші дані архітектурні підходи та визначимо їх ключові переваги та недоліки.

Так, відповідно до монолітної архітектури, застосунок будується як один великий самодостатній функціональних блок, не залежний від інших застосунків (див. рис. 1) [2]. Фактично це означає наявність великої обчислювальної мережі з єдиною кодовою базою, яка об'єднує виконання усіх бізнес задач. Серед переваг такого підходу можна виділити високу швидкість розробки завдяки простоті створення ПЗ на основі однієї кодової бази. Варто відмітити і простоту первинного розгортання монолітного застосунку. Проте для внесення змін до нього необхідно оновити увесь стек, отримавши перед цим доступ до бази коду, що робить оновлення певною мірою обмеженими та трудомісткими [3].

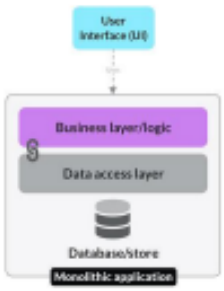


Рисунок 1. Схематичне зображення структури монолітного застосунку

Моноліти можуть бути дуже ефективними, допоки не посте питання щодо їх масштабування чи розширення функціональних можливостей. Так, внесення навіть найменшої зміни вимагатимете повне тестування усієї системи, що на пряму суперечить сучасному гнучкому підходу до розробки застосунків. Пов'язано це із високим рівнем взаємозв'язків між внутрішніми модулями системи, що часто створює багато технічних проблем. З цього випливає низька надійність монолітних систем та обмеження до впровадження використання нових технологічних засобів, через що ускладнюється їх підтримка. Ось чому для оновлення технологічного стеку чи додання комплексної бізнес логіки монолітні системи не рідко переписують ледь не з нуля [3].

Мікросервісна архітектура в певній мірі є альтернативою монолітній. В основі даного підходу лежить використання ряду незалежних один від одного сервісів, які розгортаються окремо. Також такі сервіси обов'язково виконують різні задачі та мають власні бази даних (див. рис. 2) [2]. Інакше кажучи, мікросервіси розділяють виконання бізнес логіки на окремі, незалежні бази коду. Такий підхід дещо збільшує складність розробки, проте значно полегшує вирішення різноманітних технічних проблем

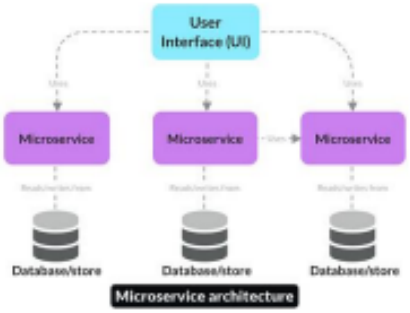


Рисунок 2. Схематичне зображення мікросервісного застосунку

Так, мікросервісний підхід дозволяє проводити оновлення, тестування, розгортання та масштабування для кожного сервісу окремо, що дозволяє швидко адаптуватися до вимог користувачів чи замовника. Окрім того, можна динамічно регулювати ресурси для конкретних мікросервісів в залежності від їх навантаження, що покращує загальну ефективність та продуктивність системи [1]. Проте з переваг мікросервісної архітектури випливають і її недоліки: довша та дорожча розробка ПЗ; складність до підтримки узгодженості і цілісності даних через велику кількість баз даних та комунікаціям між мікросервісами; високі витрати на розгортання сервісів; відсутність стандартизації або її недостатній рівень.

Підсумовуючи все вищезгадане в процесі дослідження та порівняння мікросервісної та монолітної архітектур можна зробити висновок, що монолітні застосунки можуть бути зручними на ранніх етапах свого життєвого циклу. Так, за відсутності потреби до масштабування, перехід до мікросервісної архітектури є просто зайвим та лише створить зайві витрати. І хоча даний підхід стає все менш популярним, він має свої сильні сторони. Так, якщо метою є створення простого та легкого застосунку, необхідність впровадження мікросервісів відпадає сама собою. В свою чергу, мікросервісний підхід буде більш кращим рішенням для застосунків чи систем, які охоплюють велику кількість користувачів та мають розгалужену і складну бізнес логіку. Такі системи легко підтримувати, оновлювати та масштабувати. Ось чому варто завжди подумати двічі щодо вибору між монолітною та мікросервісною архітектурами.

Література

1. Harris C. Microservices vs. monolithic architecture. URL: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith
2. IcePanel. Monolithic vs. microservices architectures. *Medium*. URL: https://icepanel.medium.com/monolithic-vs-microservices-architectures-e71c75b252d1
3. From Monolithic Systems to Microservices: A Comparative Study of Performance / F. Tapia, M. Angel Mora, W. Fuertes, H. Aules et al. *Appl. Sci.* 2020, 10(17), 5797. DOI: https://doi.org/10.3390/app10175797

Публікація 3

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ
УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XI НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

13-14 грудня 2023 року

ТЕРНОПІЛЬ
2023

УДК 004.415.2
В.В. Никитюк, канд. тех. наук, доц., М.В. Тененський, А.В. Орловська
Тернопільський національний технічний університет імені Івана Пулюя

АНАЛІЗ ВИКОРИСТАННЯ EDA ДЛЯ ВИРІШЕННЯ ПРОБЛЕМ СУЧАСНИХ ЗАСТОСУНКІВ ТА СИСТЕМ

V.V. Nykytyuk, Ph.D., Assoc. Prof., M.V. Tenenskyi, A.V. Orlovska

ANALYSIS OF EDA USAGE FOR SOLVING PROBLEMS OF MODERN APPLICATIONS AND SYSTEMS

Ключові слова: Архітектура, мікросервіси, патерни, подія, програмне забезпечення, EDA
Key words: Architecture, microservices, pattern, event, software, EDA

В наш час будь-яке ПЗ чи застосунок для широкого використання є достатньо складними, а їх розробка часто базується на застосуванні мікросервісного підходу, при якому один застосунок ділять на декілька невеликих та незалежних сервісів (мікросервісів), що спілкуються між собою, утворюючи мікросервісну систему. Відповідно до цього застосунки або їх складові повинні постійно обмінюватись інформацією між собою завдяки REST API або RPC. Такі застосунки повинні впоратися з наступними вимогам: доступність, масштабованість та швидка комунікація сервісів [2]. Доступність полягає в тому, що один екземпляр сервісу функціонував в будь-який момент часу. Тому, масштабованість значною мірою відповідає за продуктивність сервісів, проте враховуючи, що в класичних stateful застосунках її реалізація не є легкою задачею. Щодо комунікації, то в типовому сценарії "запит-відповідь" клієнт блокує виконання будь-яких дій до моменту отримання відповіді на свій запит. В мікросервісній архітектурі може призвести до так званого "Microservices Hell" – ланцюг викликів проходить послідовно через п'ять сервісів з середньою швидкістю обробки в 500 мс. За такої умови блокування клієнту триватиме 2.5 сек., що може бути критичним аспектом, особливо якщо ланцюг викликів, як і час обробки запиту кожним сервісом, буде значно довшим, збільшуючи при цьому час блокування. Саме цю проблему покликана вирішити подієво-орієнтована архітектура (англ. Event-driven architecture, скорочено EDA) [2]. Тому архітектурний шаблон проектування ПЗ, в основі якого лежить створення подій та реагування на них в режимі реального реального часу. Застосунки та їх мікросервіси, спроектовані з дотриманням патернів EDA, в основі своїй повинні складатися з трьох основних частин (див. рис. 1) [3].

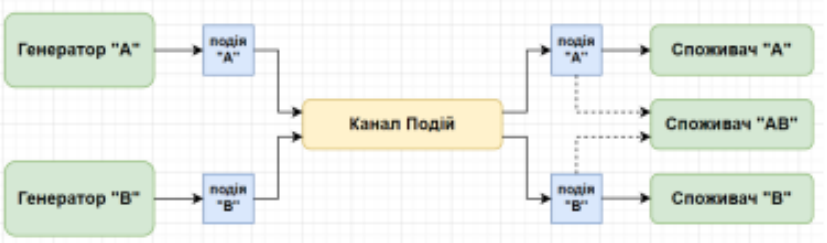


Рисунок 1 – Схематичне зображення функціонування подієво-орієнтованих застосунків

Подією може бути: від запиту користувача на зміну паролю, до аварійного вимкнення одного з сервісів застосунку. Відповідно до рис. 1, генератор подій відповідає за виявлення того чи іншого фактору, на основі якого створює відповідну подію та надсилає її в канал подій. Окрім того, генератор здатний проводити різноманітні операції над даними, трансформуючи їх у зручний для системи вигляд. Канал подій являє собою специфічний механізм, основною задачею якого є забезпечення асинхронної та

безперебійної передачі подій до відповідного заявника. Опрацювання подій відповідає за ідентифікацію події та реагування на неї [4]. Також він може комунікувати з іншими системами чи базою даних для обробки вхідних даних. Оскільки генератор події не очікує відповіді від її споживача, то усувається вищезгадана проблема – блокування виконання коду [1]. Також з рисунку 1 видно, що на одну подію може бути реалізовано декілька споживачів. Такий підхід EDA дозволяє забезпечувати доступність та надійність застосунку. Якщо той чи інший споживач тимчасово недоступний (або не встигає в повній мірі обробити вхідний потік даних) то канал подій, часто представлений сервісами-брокерами, надсилатиме події до іншого споживача, забезпечуючи надійне та стійке до збоїв функціонування системи [2].

Таким чином EDA допомагає реалізувати асинхронну комунікацію сервісів без їх блокування до виконання інших задач. Перехід до EDA означає перехід від класичної stateful моделі, орієнтованої на пряму комунікацію між сервісами, до моделі яка зводиться до комунікації через канал подій, які стають найважливішою складовою обміну інформацією між застосунками [3]. Комбінація підходів EDA з мікросервісною архітектурою дозволяє покращувати масштабованість застосунків шляхом відокремлення комунікації між сервісами від основної бізнес-логіки. Це дозволяє розробляти витривалі до непередбачуваних навантажень системи. Ідеї ПОА можуть бути доповненими принципами SOA (сервіс-орієнтованої архітектури). Дане рішення особливо корисне у випадках, коли споживачі подій розроблюваного застосунку не мають змоги з тієї чи іншої причини забезпечувати виконання власних реакцій [1].

В процесі аналізу використання EDA дозволяє спроектувати та розробити відмовостійкі та оптимізовані застосунки на основі мікросервісної архітектури. Реалізація вільного, неблокуючого зв'язку між сервісами є ключовою перевагою EDA та причиною популяризації її використання. Для підвищення ефективності та надійності застосунку, рекомендується створення більше одної операції на кожний з типів подій. Особливу увагу варто звертати і на безпеку даних, які передаються. Тому перехід до EDA, за умови дотримання усіх вищеописаних принципів, підніме ваш застосунок на якісно новий рівень. Проте не варто очікувати, що EDA вирішить абсолютно усі проблеми комунікації між застосунками та його складовими, адже в деяких випадках все ще можуть знадобитись звичайні виклики по REST API типу "запит-відповідь".

Література

1. Cameron D. Complex Event Processing and Service oriented Architecture (SOA). TMCNET NEWS. August 18, 2006. URL: https://www.tmcnet.com/usubmit/2006/08/18/1816129.htm

2. Sucaciu B. How event-driven architecture solves modern web app problems. Stackoverflow Blog. 2020, March 16. URL: https://stackoverflow.blog/2020/03/16/how-event-driven-architecture-solves-modern-web-app-problems/

3. Welsh M., Brewer E. SEDA: an architecture for well-conditioned, scalable internet services. ACM SIGOPS Operating Systems Review. 2001. Vol. 35, Issue 5. P. 230–243. URL: https://dl.acm.org/doi/10.1145/502059.502057

4. V. Nykytyuk, V. Dozorskyi, O. Dozorska, A. Karnaukhov and L. Matiichuk. The Method of User Identification by Speech Signal. The 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP-2022) Ternopil, Ukraine, November 22-24, 2022. Vol-3309 urn:nbn:de:0074-3309-1. P.225-232. ISSN 1613-0073 DOI: 10.1425/jsdtl. (Scopus).

Публікація 4

VI Міжнародна студентська науково - технічна конференція
"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ"

УДК 004.4:005.8
Тененський М. – ст. гр. СНнм-51, Галюк М. – ст. гр. СНнм-51
Тернопільський національний технічний університет імені Івана Пулюя

АНАЛІЗ ВИКОРИСТАННЯ BPM ЗАСТОСУНКІВ

Науковий керівник: старший викладач Шимчук Г.

Tenenskyi M., Haliuk M.
Ternopil Ivan Puluj National Technical University

ANALYSIS OF BPM APPLICATIONS USAGE

Supervisor: Senior Lecturer Shymchuk G.

Ключові слова: Автоматизація, бізнес процес, управління процесом, BPM
Key words: Automation, business process, process management, BPM

BPM (англ. Business Process Management) – це певний підхід до управління бізнес-процесами, що включає в себе моделювання, аналіз, виконання, моніторинг та власне оптимізацію бізнес-процесів. Основною його метою можна назвати підвищення ефективності та якості роботи бізнесу, підприємства, застосунку тощо. Так, будь-який бізнес-процес можна розглядати у якості цілеспрямованої діяльності з певною обов'язковою умовою: вона має мати завершуваний характер та приводити до будь-якого результату виконання [1]. Можна навести чимало прикладів бізнес-процесів: працевлаштування нової людини в компанію, здійснення фінансових операцій, балансування електроенергії, процес продажі товарів в мережі інтернет тощо. В загальному ж BPM здатний розбивати систему бізнес-процесів компанії чи бізнесу на довільну скінченну послідовність кроків, описує їх властивості та впроваджує автоматизацію.

BPM застосунки – це програмні засоби, що дозволяють підтримувати і виконувати бізнес-процеси в автоматизованому режимі. Вони не тільки допомагають підприємствам забезпечувати прозорість бізнес-процесів, але й дозволяють відчутно зменшувати витрати на їх виконання та підвищувати загальний рівень якості та продуктивності [2]. Також справедливим буде твердження, що сучасні BPM системи, зазвичай, складаються з певної групи програмного забезпечення, що реалізують концепцію управління бізнес-процесами, в тому числі складними та такими, що здатні динамічно розвиватись.

Застосунки такого типу, як вже було сказано, можуть використовуватись в найрізноманітніших галузях сьогодення, проте перед ними завжди буде поставлення схожа задача: функціонал BPM застосунків має включати в себе власне проектування бізнес-процесів, їх моделювання, виконання, моніторинг та загальну оптимізацію. Окрім того, за допомогою BPM можна виконати автоматизацію різних бізнес-процесів: від операцій, пов'язаних з управлінням складу, до балансування електроенергії відповідними TSO чи BSP організаціями.

Варто розуміти також і те, що інтеграція BPM застосунків та їх правильна конфігурація неодмінно призведе до підвищення якості продуктів та послуг, зменшення ризику помилок тощо. Крім того, впровадження BPM може допомогти зменшити час на виконання типових завдань, що підвищить ефективність та призведе до збільшення прибутку компанії.

Розкриваючи тему аналізу використання BPM застосунків більш детально, варто зазначити також і те, що одним з головних споживачів даних застосунків можна

назвати виробництва. Так, BPM система здатна допомогти виробничій компанії оптимізувати основні робочі процеси та збільшити ефективність обладнання чи ПЗ.

Також вищезгадані застосунки допомагають бізнес-аналітикам будь-якого професійного рівня проектувати діаграми процесів будь-якої складності, в тому числі з врахуванням довільно великої кількістю розгалужень, специфічних правил чи умов до бізнес-процесу [1]. Завдяки цьому бізнес отримує чітко прописані регламенти, основна мета яких – це проведення уніфікації як діяльність компанії загалом, так і її підрозділів, кожного окремого співробітника чи програмного забезпечення для досягнення найкращих результатів.

Для побудови бізнес-процесів існує безмежна кількість різних інструментів, визнаним стандартом серед яких вважають нотація BPMN (англ. Business Process Model and Notation) – система умовних позначень, що використовується для моделювання бізнес процесів [1]. Найголовніше перевага даного стандарту полягає в тому, що будь яка діаграма бізнес-процесу, спроектована з її допомогою, буде одночасно зрозуміла як пересічному користувачу програмного забезпечення, так і професійнім бізнес-аналітикам з багаторічним досвідом. Крім того, використання BPM на базі BPMN застосунків дозволить компанії зосередитися на важливих завданнях та покращити ефективність взаємодії зі своїми клієнтами, партнерами тощо.

Як можна зрозуміти з вищенаведеної інформації, типові переваги застосунків такого типу полягають в зниженні витрат на виконання, інтеграцію чи модифікацію бізнес-процесів, а також підвищенні рівня автоматизації та продуктивності, забезпеченні якості, зрозумілості, надійності та прозорості бізнес-процесів. Окрім того, заме BPM системи надають можливість достатньо швидко адаптувати бізнес-процеси до змін у ринкових умовах та вимогах споживачів.

Проте застосування BPM систем не позбавлене типових недоліків, одним з яких може вважати відносно високу складність їх інтеграції з існуючими системами. Проте цей недолік можна усунути за допомогою використання стандартних протоколів та інтерфейсів, таких як вищезгаданий BPMN, BPEL (англ. Business Process Execution Language) та WS-* (Web Services Specifications), які дозволяють забезпечити інтеграцію BPM з іншими системами [2].

Так чи інакше, BPM застосунки неодмінно стануть в нагоді для підвищення ефективності та якості виконання бізнес-процесів. Вони дозволяють підприємствам забезпечити прозорість та контроль за бізнес-процесами, зменшувати витрати на їх виконання та підвищити продуктивність [1]. В той самий час важливо враховувати можливі недоліки, такі як складність інтеграції таких систем чи їх налаштування, та забезпечувати відповідне навчання та підтримку співробітників, щоб максимально використати переваги BPM застосунків.

Література:

1. Weske M. (2019). Business Process Management: Concepts, Languages, Architectures. 3rd ed. 417 p. Springer-Verlag Berlin Heidelberg. ISBN 978-3-662-59431-5.
2. Panagacos T. (2012). The Ultimate Guide to Business Process Management: Everything you need to know and how to apply it to your organization. 186 p. CreateSpace Publishing. ISBN 978-1477486139.

Публікація 5

VI Міжнародна студентська науково - технічна конференція
"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ"

УДК 004.4:005.8
Галюк М. – ст. гр. СНнм-51, Тененський М. – ст. гр. СНнм-51
Тернопільський національний технічний університет імені Івана Пулюя

ВПЛИВ BPM ЗАСТОСУНКІВ НА ЕФЕКТИФНІСТЬ ПІДПРИЄМСТВА

Науковий керівник: старший викладач Шимчук Г.

Haliuk M., Tenenskyi M.
Ternopil Ivan Puluj National Technical University

THE INFLUENCE OF BPM APPLICATIONS ON THE EFFICIENCY OF THE ENTERPRISE

Supervisor: Senior Lecturer Shymchuk G.

Ключові слова: Автоматизація, бізнес процес, ефективність, управління процесом, BPM
Key words: Automation, business process, efficiency, process management, BPM

В наш час нікого не здивує твердження, що конкурентоспроможність підприємств залежить від багатьох факторів, одним з яких є використання інноваційних технологій, залучених до процесу управління. BPM-застосунки є одним з таких інструментів, допомагають бізнесу досягти більш ефективних та високих результатів в управління [1]. Існує безліч трактувань поняття BPM, але найбільш коректне – це підхід до управління бізнес-процесом, який передбачає моделювання, аналіз, виконання, моніторинг та оптимізацію даного процесу.

Так, для дослідження впливу BPM застосунків на ефективність підприємства було прийняти використовувати типові показники, що відображають результати діяльності підприємства. Серед них найбільш вагомими можна вважати прибуток, обіг, рентабельність тощо. В процесі багаточисленних досліджень неодноразово було встановлено, що впровадження BPM застосунків позитивно впливає на ефективність підприємства. Зокрема, їх використання збільшує рівень автоматизації та оптимізації процесів, зменшує час, необхідний для виконання бізнес-процесів, підвищує якість продукту [1]. Окрім того, вдала інтеграція BPM систем позитивним чином відображається показниках фінансової ефективності підприємства, зокрема збільшення прибутку, зменшення витрат на виконання бізнес-процесів тощо.

Одним з найвідоміших прикладів використання BPM застосунків прийнято вважати американську компанію FedEx, яка впровадила використання системи з відстеження вантажів в режимі реального часу для оптимізації своїх логістичних процесів. Завдяки цій BPB системі працівники FedEx мають змогу планувати маршрути доставки товарів більш ефективно, зменшуючи при цьому витрати на пальне для свої автомобілів та збільшуючи швидкість доставки одночасно [2].

В якості ще одного прикладу успішного використання BPM застосунків можна навести японського автомобільного гіганта – компанію Toyota. Вона використовує ПЗ на базі усім відомої системи Kanban для контролю за виробничим процесом на заводах. В свою чергу, це дозволяє компанії більш ефективно керувати запасами і виробничими процесами [3]. Відповідного до цього, показники вироблення автомобілі щороку лишаються на висоті, при чому вдається оптимізувати витрати на виробництво та зберігати стабільно високу якість кінцевої продукції.

Як можна зрозуміти із вищенаведеної інформації, BPM застосунки мають чималу кількість переваг, позитивно впливають на бізнес-процеси. Так, шляхом їх оптимізації та автоматизації, підприємства мають змогу зменшувати витрати, покращувати ефективність та підвищувати якість своїх товарів і послуг [3]. Окрім того, певні спеціалізовані BPM системи можуть забезпечувати швидке вирішення запитів клієнтів та розв'язання їх проблем у найкоротші терміни. В загальному це відображається на підвищенні задоволення клієнтів, збільшенні їх лояльності до продукції чи послуг підприємства, загальної популяризації бренду.

Хоча BPM застосунки зазвичай розглядають виключно як корисний інструмент для забезпечення ефективності бізнес-процесів, їх неправильне впровадження та використання може мати негативний вплив на підприємство [1]. Ось деякі можливі проблеми, які можуть виникнути при впровадженні BPM систем . Найбільш можливий негативний вплив – це так звана надмірна автоматизація. Полягає вона в тому, що використання BPM здатне за певних умов призвести до того, що бізнес-процеси будуть автоматизовані повністю, інакше кажучи працівники підприємства будуть мало зайняті, що може призвести до зниження їх продуктивності. Також не варто забувати і про складність впровадження BPM застосунків. Як показує практика, зазвичай, це дуже складним процесом, особливо якщо нове BPM рішення має бути інтегрованим в уже існуючі системи на підприємстві [2]. Як результат – це може призвести до певних часових затримок в процесі впровадження та збільшити загальні витрати бізнесу в цей час. Окрім того, використання складних та комплексних BPM рішень вимагає наявність навченого персоналу. Якщо ж такого нема – підприємству необхідно самотужки вирішувати це питання. Проте варто розуміти, що усі ці недоліки перекриваються значними перевагами після успішного впровадження BPM систем в управління бізнес-процесами.

У висновку варто зазначити, що в наш час розвитку програмного забезпечення та постійної конкуренції на ринках використання BPM застосунків є дуже важливим фактором для підтримки ефективного функціонування будь-якого бізнесу [1]. Вони допомагають автоматизувати бізнес-процеси, збільшувати продуктивність підприємств, знижувати витрати на виробництво та поліпшувати якість кінцевої продукції. Також справедливим буде твердження, що перевірені BPM системи є незамінним інструментом для автоматизації бізнес-процесів, що дозволяє зменшити ручну роботу, уникнути помилок та значно підвищити точність і якість виконання робіт [2]. Окрім того, інтеграція та розробка BPM застосунків, без перебільшення, є дуже перспективним напрямом розвитку бізнесу, оскільки вони дозволяють постійно покращувати якість кінцевого продукту та дозволяють компаніям виконувати поставлені задачі більш якісно і швидко, що позитивним чином впливає на їх кінцевих споживачів.

Література:

1. Weske M. (2019). Business Process Management: Concepts, Languages, Architectures. 3rd ed. 417 p. Springer-Verlag Berlin Heidelberg. ISBN 978-3-662-59431-5.
2. How Technology Works (2019). The facts visually explained. Dorling Kindersley. 256 p. ISBN 978-0241356289.
3. Panagacos T. (2012). The Ultimate Guide to Business Process Management: Everything you need to know and how to apply it to your organization. 186 p. CreateSpace Publishing. ISBN 978-1477486139.

Додаток Б

Таблиця розрахунку показників для експрес методу оцінки бізнес процесів

Показник	Формула розрахунку	Норма
Показник складності	$Кскл = \frac{\sum Прів}{\sum Пекз}$, де Прів – кількість рівнів системи БП, Пекз – кількість екземплярів БП.	≤ 0.66
Показник процесності	$Кпр = \frac{\sum Проз}{\sum Пкп}$, де Проз – кількість «розривів» процесів в екземплярах БП, Пкп – кількість класів БП.	< 1
Показник контрольованості	$Квідп = \frac{ВП}{\sum Пкп}$, де ВП – число власників БП, Пкп – кількість класів БП.	< 1
Показник ресурсоємності	$Кр = \frac{Р}{\sum Пвих}$, де Р – кількість використовуваних ресурсів для виконання БП, Пвих – кількість виходів БП.	< 1
Показник урегульованості	$Крег = \frac{\sum Прег}{\sum Пкп}$, де Прег – кількість нормативних документів, Пкп – кількість класів БП.	≥1
Після розрахунку цих показників ефективності розраховується інтегральний показник. Якщо його нормативне значення знаходиться в межах від $1 \leq \sum K \leq 2$, то БП вважається ефективним, інакше – вважається таким, що потребує змін		

Додаток В

Вміст файлу process-instance-mongo.js, який містить схему та описує DAO методи сутності processInstance

```
"use strict";
const UuObjectExtendedDao = require("./support/uuobject-extended-
dao");

class ProcessInstanceMongo extends UuObjectExtendedDao {
  async createSchema() {
    await super.createSchema();
    await super.createIndex({ awid: 1, id: 1 });
    await super.createIndex({ awid: 1, code: 1, startingTime: 1 },
{ name: "process_code_and_starting_time_idx_1" });
    await this.deleteUnusedIndexes();
  }

  async create(uuObject) {
    return await super.insertOne(uuObject);
  }

  async getByFilterCriteria(awid, filterCriteria) {
    return await super.findOne({ awid, ...filterCriteria });
  }

  async setState(awid, processInstanceId, nextState, timeStamp =
new Date().toISOString()) {
    let filter = { awid, id: processInstanceId };

    return super.findOneAndUpdate(filter, {
      $set: { state: nextState },
      $push: { stateHistory: { value: nextState, timeStamp: new
Date(timeStamp) } }
    });
  }
}

module.exports = ProcessInstanceMongo;
```

Вміст файлу process-mongo.js, який містить схему та описує DAO методи сутності process

```
"use strict";
const UuObjectExtendedDao = require("./support/uuobject-extended-
dao");

class ProcessMongo extends UuObjectExtendedDao {
```

```
  async createSchema() {
    await super.createSchema();
    await super.createIndex({ awid: 1, id: 1 });
    await super.createIndex({ awid: 1, code: 1 }, { unique: true,
name: "process_code_idx_1" });
    await this.deleteUnusedIndexes();
  }
  async create(uuObject) {
    return await super.insertOne(uuObject);
  }
  async get(awid, id) {
    let filter = {
      awid: awid,
      id: id
    };
    return await super.findOne(filter);
  }
  async getByFilterCriteria(awid, filterCriteria) {
    return await super.findOne({ awid, ...filterCriteria });
  }
  async listByFilterCriteria(awid, filterCriteria, pageInfo) {
    return await super.find({ awid, ...filterCriteria },
pageInfo);
  }

  async update(uuObject) {
    let filter = {
      awid: uuObject.awid,
      id: uuObject.id
    };
    return super.findOneAndUpdate(filter, uuObject, "NONE");
  }

  async remove(uuObject) {
    let filter = {
      awid: uuObject.awid,
      id: uuObject.id
    };
    return await super.deleteOne(filter);
  }
}

module.exports = ProcessMongo;
```

Вміст файлу bpm-mongo.js, який містить схему та описує DAO методи основної сутності застосунку

```
const { UuObjectDao } = require("uu_appg01_server").ObjectStore;

class BpmMongo extends UuObjectDao {
  async createSchema() {
    await super.createIndex({ awid: 1 }, { unique: true });}
```

```
  async create(uuObject) {
    return await super.insertOne(uuObject);
  }

  async getByAwid(awid) {
    return await super.findOne({ awid });
  }

  async update(uuObject) {
    let filter = {
      awid: uuObject.awid,
      id: uuObject.id
    };
    return await super.findOneAndUpdate(filter, uuObject, "NONE");
  }

  async remove(uuObject) {
    let filter = {
      awid: uuObject.awid,
      id: uuObject.id
    };
    return await super.deleteOne(filter);
  }
}

module.exports = BpmMongo;
```

Вміст файлу activity-instance-mongo.js, який містить схему та описує DAO методи сутності activityInstance

```
"use strict";
const { Activity } = require("../utils/constants");
const UuObjectExtendedDao = require("./support/uuobject-extended-
dao");

class ActivityInstanceMongo extends UuObjectExtendedDao {
  async createSchema() {
    await super.createSchema();
    await super.createIndex({ awid: 1, id: 1 });
    await super.createIndex({ awid: 1, code: 1, processInstanceId:
1 }, { unique: true, name:
"activity_code_and_process_instance_id_idx_1" });
    await this.deleteUnusedIndexes();
  }

  async create(uuObject) {
    return super.insertOne(uuObject);
  }

  async createMany() {
    return super.insertMany();
  }
```

```
  async listStandardWithEmptyInputSignals(awid, processInstanceId)
{
    return super.find({ awid, processInstanceId, type:
Activity.types.standard, "inputSignals.triggers": { $size: 0 } });
  }

  async getByFilterCriteria(awid, filterCriteria) {
    return super.findOne({ awid, ...filterCriteria });
  }

  async listDependentOnInputSignals(awid, processInstanceId,
activityInstanceId) {
    const filterCriteria = { processInstanceId,
"inputSignals.triggers": { $elemMatch: { activityInstanceId } } };

    return super.find({ awid, ...filterCriteria });
  }

  // TODO: check if it works
  async fillInTriggersData(awid, processInstanceId,
activityInstanceIdList) {
    let filter = {
      awid,
      "processInstanceData.id": processInstanceId,
      id: { $in: activityInstanceIdList }
    };

    return super.updateMany(filter, [
      {
        $addFields: {
          "inputSignals.triggers": {
            $map: {
              input: "$inputSignals.triggers",
              as: "trigger",
              in: {
                $cond: {
                  if: {
                    $eq: ["$$trigger.activityCode",
"$activityData.code"]
                  },
                  then: {
                    $mergeObjects: [
                      "$$trigger",
                      {
                        activityInstanceId: "$_id"
                      }
                    ]
                  },
                  else: "$$trigger"
                }
              }
            }
          }
        }}]);}
```

```
async fillInTriggersData(awid, processInstanceId,
activityInstanceIdList) {
    let filter = {
      awid,
      "processInstanceData.id": processInstanceId,
      id: { $in: activityInstanceIdList }
    };
   return super.updateMany(filter, [
     {
       $lookup: {
         from: "myCollection",
         localField: "code",
         foreignField: "code",
         as: "matchingObjects"
       }
     },
     {
       $set: {
         id: { $arrayElemAt: ["$matchingObjects.id", 0] }
       },
       $unset: ["matchingObjects"]
     }
   ]);
}
  async markReceivedTrigger(awid, triggerActivityInstanceId,
activityInstanceIdList) {
    let filter = {
      awid,
      id: { $in: activityInstanceIdList }
    };
    return super.updateMany(
      filter,
      { $set: { "inputSignals.triggers.$[x].received": true } },
      { arrayFilters: [{ "x.activityInstanceId":
triggerActivityInstanceId }] }
    );
  }
  async setState(awid, activityInstanceId, nextState) {
    let filter = { awid, id: activityInstanceId };
    return super.findOneAndUpdate(filter, { $set: { state:
nextState } });
  }} module.exports = ActivityInstanceMongo;
```

Вміст файлу activity-mongo.js, який містить схему та описує DAO методи сутності activity

```
"use strict";
const UuObjectExtendedDao = require("./support/uuobject-extended-
dao");

class ActivityMongo extends UuObjectExtendedDao {
  async createSchema() {
```

```
    await super.createSchema();
    await super.createIndex({ awid: 1, id: 1 });
    await super.createIndex({ awid: 1, code: 1, processCode: 1 },
{ unique: true, name: "activity_and_process_code_idx_1" });
    await this.deleteUnusedIndexes();
  }

  async create(uuObject) {
    return await super.insertOne(uuObject);
  }

  async getByFilterCriteria(awid, filterCriteria) {
    return await super.findOne({ awid, ...filterCriteria });
  }

  async listByFilterCriteria(awid, filterCriteria, pageInfo) {
    return await super.find({ awid, ...filterCriteria },
pageInfo);
  }
  async areAllRelatedActivitiesExisting(awid, processCode,
relatedActivityCodeList) {
    const storedRelatedActivitiesAmount = await super.count({
awid, processCode, code: { $in: relatedActivityCodeList } });

    return storedRelatedActivitiesAmount ===
relatedActivityCodeList.length;
  }
  async update(uuObject) {
    let filter = {
      awid: uuObject.awid,
      id: uuObject.id
    };
    return super.findOneAndUpdate(filter, uuObject, "NONE");
  }
  async updateProcessCode(awid, oldProcessCode, newProcessCode) {
    let filter = { awid, processCode: oldProcessCode };
    return super.updateMany(filter, { $set: { processCode:
newProcessCode } });
  }
  async remove(uuObject) {
    let filter = {
      awid: uuObject.awid,
      id: uuObject.id
    };
    return super.deleteOne(filter);
  }

  async removeManyByProcessCode(awid, processCode) {
    let filter = { awid, processCode };

    return super.deleteMany(filter);
  }}

module.exports = ActivityMongo;
```

Додаток Д

Вміст конфігураційного файлу profiles.json

```
{
  "{asid}": {
    "profileList":       ["AsidAuthorities",       "AsidExecutives",
"AsidLicenseOwner", "Public"],
    "useCaseMap": {}
  },
  "*": {
    "profileList":           ["Authorities",           "Executives",
"AwidLicenseOwner", "Public"],
    "useCaseMap": {
      "sys/uuAppWorkspace/init": {
        "sysStateList": ["created", "assigned"],
        "profileList": ["AwidLicenseOwner"]
      },
      "sys/uuAppWorkspace/load": {
        "sysStateList": ["active"],
        "profileList": ["Authorities", "Executives"]
      },
      "sys/uuAppWorkspace/loadBasicData": {
        "sysStateList": ["active"],
        "profileList": ["Authorities", "Executives"]
      },
      "defaultUve": {
        "profileList": ["Public"]
      },
      "sys/uuAppWorkspace/initUve": {
        "sysStateList": ["created"],
        "profileList": ["Public", "AwidLicenseOwner"]
      },
      "simeBpm/update": {
        "sysStateList": ["active"],
        "profileList": ["Authorities", "Executives"]
      },
      "process/create": {
        "sysStateList": ["active"],
        "profileList": ["Authorities", "Executives"]
      },
      "process/update": {
        "sysStateList": ["active"],
        "profileList": ["Authorities", "Executives"]
      },
      "process/get": {
        "sysStateList": ["active"],
        "profileList": ["Authorities", "Executives"]
      },
      "process/list": {
        "sysStateList": ["active"],
        "profileList": ["Authorities", "Executives"]
```

```
      },
      "process/delete": {
        "sysStateList": ["active"],
        "profileList": ["Authorities", "Executives"]
      },
      "process/start": {
        "sysStateList": ["active"],
        "profileList": ["Authorities", "Executives"]
      },
      "activity/create": {
        "sysStateList": ["active"],
        "profileList": ["Authorities", "Executives"]
      },
      "activity/update": {
        "sysStateList": ["active"],
        "profileList": ["Authorities", "Executives"]
      },
      "activity/get": {
        "sysStateList": ["active"],
        "profileList": ["Authorities", "Executives"]
      },
      "activity/list": {
        "sysStateList": ["active"],
        "profileList": ["Authorities", "Executives"]
      },
      "activity/delete": {
        "sysStateList": ["active"],
        "profileList": ["Authorities", "Executives"]
      }
    }
  }
}
```

Вміст конфігураційного файлу persistence.json

```
{
  "uuSubAppDataStore": { dffrt
    "primary": {
      "type": "uuAppObjectStore",
      "realization": "MongoDB",
      "schemaMap": {
        "simeBpm": {
          "realization": "dao/SimeBpmMongo"
        },
        "process": {
          "realization": "dao/ProcessMongo"
        },
        "activity": {
          "realization": "dao/ActivityMongo"
        },
        "processInstance": {
          "realization": "dao/ProcessInstanceMongo"
        },
```

```
        "activityInstance": {
          "realization": "dao/ActivityInstanceMongo"
        },
        "event": {
          "realization": "dao/EventMongo"
        },
        "piUpdateEvent": {
          "realization": "dao/PiUpdateEventMongo"
        }
      }
    }
  }
}
```

Вміст конфігураційного файлу mapping.json

```
{
  "{vendor}-{uuApp}-{uuSubApp}": {
    "useCaseMap": {
      "sys/uuAppWorkspace/init": {
        "realization": "api/controllers/SimeBpmController.init",
        "httpMethod": "POST",
        "type": "CMD"
      },
      "sys/uuAppWorkspace/load": {
        "realization": "api/controllers/SimeBpmController.load",
        "httpMethod": "GET",
        "type": "CMD"
      },
      "sys/uuAppWorkspace/loadBasicData": {
        "realization":
"api/controllers/SimeBpmController.loadBasicData",
        "httpMethod": "GET",
        "type": "CMD"
      },
      "defaultUve": {
        "realization": "index.html",
        "httpMethod": "GET",
        "type": "UVE"
      },
      "sys/uuAppWorkspace/initUve": {
        "realization": "index.html",
        "httpMethod": "GET",
        "type": "UVE"
      },
      "simeBpm/update": {
        "realization": "api/controllers/SimeBpmController.update",
        "httpMethod": "POST",
        "type": "CMD"
      },
      "process/create": {
        "realization": "api/controllers/ProcessController.create",
        "httpMethod": "POST",
```

```
      "type": "CMD"
    },
    "process/update": {
      "realization": "api/controllers/ProcessController.update",
      "httpMethod": "POST",
      "type": "CMD"
    },
    "process/get": {
      "realization": "api/controllers/ProcessController.get",
      "httpMethod": "GET",
      "type": "CMD"
    },
    "process/list": {
      "realization": "api/controllers/ProcessController.list",
      "httpMethod": "GET",
      "type": "CMD"
    },
    "process/delete": {
      "realization": "api/controllers/ProcessController.delete",
      "httpMethod": "POST",
      "type": "CMD"
    },
    "process/start": {
      "realization": "api/controllers/ProcessController.start",
      "httpMethod": "POST",
      "type": "CMD"
    },
    "activity/create": {
      "realization": "api/controllers/ActivityController.create",
      "httpMethod": "POST",
      "type": "CMD"
    },
    "activity/update": {
      "realization": "api/controllers/ActivityController.update",
      "httpMethod": "POST",
      "type": "CMD"
    },
    "activity/get": {
      "realization": "api/controllers/ActivityController.get",
      "httpMethod": "GET",
      "type": "CMD"
    },
    "activity/list": {
      "realization": "api/controllers/ActivityController.list",
      "httpMethod": "GET",
      "type": "CMD"
    },
    "activity/delete": {
      "realization": "api/controllers/ActivityController.delete",
      "httpMethod": "POST",
      "type": "CMD"
    }}
  }
}
```

Додаток Е

Вміст файлу start-initial-activities-handler.js, який містить програмний код для запуску початкових активностей процесу

```
"use strict";
const { DaoFactory } = require("uu_appg01_server").ObjectStore;
const { Schemas, Event, ProcessInstance }=
require("../../utils/constants");
const EventManager = require("../event-manager");
const ProcessInstanceValidator = require("../validators/process-
instance-validator");

class StartInitialActivitiesHandler {
  constructor() {
    this.activityInstanceDao=
DaoFactory.getDao(Schemas.activityInstance);
  }

  async handle(awid, event, uuAppErrorMap, errors) {
    const { processInstanceId } = event;

    // HDS 1 - Ensure processInstance exists.
    const processInstance = await
ProcessInstanceValidator.ensureProcessInstanceExists(awid, { id:
processInstanceId }, uuAppErrorMap, errors);

    // HDS 2 - Ensure processInstance is not in terminal state.

ProcessInstanceValidator.ensureProcessInstanceNotInTerminalState(p
rocessInstance, uuAppErrorMap, errors);

    // HDS 3 - Produce processInstance update event.
    await EventManager.producePiUpdateEvent(
      awid,
      Event.codes.changePiState,
      Date.now(),
      { processInstanceId, nextState:
ProcessInstance.states.running },
      uuAppErrorMap,
      errors
    );

    // HDS 4 - Load all standard activityInstances with empty input
signals.
    const { itemList:
standardActivityInstancesWithEmptyInputSignals } = await
this.activityInstanceDao.listStandardWithEmptyInputSignals(
      awid,
      processInstanceId
    );
```

```
    // HDS 5 - For each obtained standard activity: produce common
event
    for             (let             activityInstance             of
standardActivityInstancesWithEmptyInputSignals) {
      await EventManager.produceCommonEvent(
        awid,
        Event.codes.startActivity,
        Date.now(),
        {          processInstanceId,          activityInstanceId:
activityInstance.id },
        uuAppErrorMap,
        errors
      );
    }

    return { uuAppErrorMap };
  }
}

module.exports = new StartInitialActivitiesHandler();
```

Вміст файлу start-activity-handler.js, який містить програмний код для запуску непочаткових активностей процесу

```
"use strict";
const { DaoFactory } = require("uu_appg01_server").ObjectStore;
const  {  Schemas,  Event,  ActivityInstance,  ProcessInstance  }  =
require("../../utils/constants");
const EventManager = require("../event-manager");
const   ActivityInstanceClient   =   require("../clients/activity-
instance-client");
const  ProcessInstanceValidator  =  require("../validators/process-
instance-validator");
const ActivityInstanceValidator = require("../validators/activity-
instance-validator");
const  ActivityInstanceHelper  =  require("../../helpers/activity-
instance-helper");

class StartInitialActivitiesHandler {
  constructor() {
    this.activityInstanceDao                                    =
DaoFactory.getDao(Schemas.activityInstance);
  }

  async handle(awid, event, uuAppErrorMap, errors) {
    const { processInstanceId, activityInstanceId } = event;

    const            processInstance            =            await
ProcessInstanceValidator.ensureProcessInstanceExists(awid,  {  id:
processInstanceId }, uuAppErrorMap, errors);
```

```
ProcessInstanceValidator.ensureProcessInstanceNotInTerminalState(p
rocessInstance, uuAppErrorMap, errors);

    const            activityInstance            =            await
ActivityInstanceValidator.ensureActivityInstanceExists(awid,  {  id:
activityInstanceId }, uuAppErrorMap, errors);

    await     ActivityInstanceHelper.setActivityInstanceState(awid,
activityInstanceId, ActivityInstance.states.running, uuAppErrorMap,
errors);

    if (activityInstance.invocationParams) {
      try {
        await   ActivityInstanceClient.runActivityInstanceCmd(awid,
activityInstance.invocationParams);

        await ActivityInstanceHelper.setActivityInstanceState(awid,
activityInstanceId,               ActivityInstance.states.finished,
uuAppErrorMap, errors);
      } catch {
        await ActivityInstanceHelper.setActivityInstanceState(awid,
activityInstanceId,  ActivityInstance.states.error,  uuAppErrorMap,
errors);

        await EventManager.producePiUpdateEvent(
          awid,
          Event.codes.changePiState,
          Date.now(),
          {              processInstanceId,              nextState:
ProcessInstance.states.error },
          uuAppErrorMap,
          errors
        );

        // TODO: IMPL compensation signal later
      }
    } else {
      await   ActivityInstanceHelper.setActivityInstanceState(awid,
activityInstanceId,               ActivityInstance.states.finished,
uuAppErrorMap, errors);
    }

    await EventManager.produceCommonEvent(
      awid,
      Event.codes.activityFinished,
      Date.now(),
      { processInstanceId, activityInstanceId },
      uuAppErrorMap,
      errors
    );
    return { uuAppErrorMap };
  }
}
module.exports = new StartInitialActivitiesHandler();
```

Вміст файлу change-pi-state-handler.js, який містить програмний код для опрацювання активностей по зміні статусу процесу

```
"use strict";
const ProcessInstanceValidator = require("../validators/process-instance-validator");
const ProcessInstanceHelper = require("../../helpers/process-instance-helper");

class ChangePiStateHandler {
  async handle(awid, event, uuAppErrorMap, errors) {
    const { processInstanceId, nextState } = event;

    // HDS 1 - Ensure processInstance exists.
    const processInstance = await ProcessInstanceValidator.ensureProcessInstanceExists(awid, { id: processInstanceId }, uuAppErrorMap, errors);

    // HDS 1 - Ensure processInstance not in terminal states

ProcessInstanceValidator.ensureProcessInstanceNotInTerminalState(processInstance, uuAppErrorMap, errors);

    // HDS 2 - Update processInstance state
    await ProcessInstanceHelper.setProcessInstanceState(awid, processInstanceId, nextState, uuAppErrorMap, errors);

    return { uuAppErrorMap };
  }
}

module.exports = new ChangePiStateHandler();
```

Вміст файлу activity-finished-handler.js, який містить програмний код для опрацювання завершення роботи активності

```
"use strict";
const { DaoFactory } = require("uu_appg01_server").ObjectStore;
const { Schemas, Event, ActivityInstance } = require("../../utils/constants");
const EventManager = require("../event-manager");
const ProcessInstanceValidator = require("../validators/process-instance-validator");
const ActivityInstanceValidator = require("../validators/activity-instance-validator");
const ActivityInstanceHelper = require("../../helpers/activity-instance-helper");

class ActivityFinishedHandler {
  constructor() {
```

```
    this.activityInstanceDao=
DaoFactory.getDao(Schemas.activityInstance);
  }

  async handle(awid, event, uuAppErrorMap, errors) {
    const { processInstanceId, activityInstanceId } = event;

    const            processInstance            =            await
ProcessInstanceValidator.ensureProcessInstanceExists(awid,   {   id:
processInstanceId }, uuAppErrorMap, errors);


ProcessInstanceValidator.ensureProcessInstanceNotInTerminalState(p
rocessInstance, uuAppErrorMap, errors);

    const            activityInstance            =            await
ActivityInstanceValidator.ensureActivityInstanceExists(awid, { id:
activityInstanceId }, uuAppErrorMap, errors);

    await     ActivityInstanceHelper.setActivityInstanceState(awid,
activityInstanceId,               ActivityInstance.states.finished,
uuAppErrorMap, errors);

    if (activityInstance.nextProcessState) {
      await EventManager.producePiUpdateEvent(
        awid,
        Event.codes.changePiState,
        Date.now(),
        {               processInstanceId,               nextState:
activityInstance.nextProcessState },
        uuAppErrorMap,
        errors
      );
    }

    const       dependentActivityInstanceList       =       (await
this.activityInstanceDao.listDependentOnInputSignals(awid,
processInstanceId, activityInstanceId))
      .itemList;

    if (dependentActivityInstanceList.length) {
      const       updatedActivityInstanceList       =       await
ActivityInstanceHelper.markReceivedTriggerAndUpdate(
        activityInstanceId,
        dependentActivityInstanceList,
        uuAppErrorMap,
        errors
      );

      for (let activityInstance of updatedActivityInstanceList) {
        if
(ActivityInstanceValidator.ensureActivityInstanceCanStartBasedOnIn
putSignals(activityInstance, activityInstanceId)) {
```

```
          const startingTime = activityInstance.inputSignals.delayT
? Date.now() + activityInstance.inputSignals.delayT * 1000 :
Date.now(); //delayT is stored in seconds

          await EventManager.produceCommonEvent(
            awid,
            Event.codes.startActivity,
            startingTime,
            { processInstanceId, activityInstanceId },
            uuAppErrorMap,
            errors
          );
        }
      }
    }

    return { uuAppErrorMap };
  }
}

module.exports = new ActivityFinishedHandler();
```