Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна

Кваліфікаційна наукова
праця на правах рукопису

ДААС ТИМУР ІМАД АХМАД

УДК 004.05

ДИСЕРТАЦІЯ

ЗНАННЯ-ОРІЄНТОВАНІ МОДЕЛІ ТА МІКРОСЕРВІСНІ АРХІТЕКТУРИ ДЛЯ РОЗРОБКИ СИСТЕМ ІНТЕРНЕТ-БАНКІНГУ

Спеціальність 122 Комп’ютерні науки
Галузь знань 12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ **Т.І. Даас**

Науковий керівник: **Ткачук Микола Вячеславович**
доктор технічних наук, професор

Харків - 2026

АНОТАЦІЯ

Даас Т.І. Знання-орієнтовані моделі та мікросервісні архітектури для розробки систем інтернет-банкінгу. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття ступеня доктора філософії за спеціальністю 122 Комп'ютерні науки (Галузь знань 12 Інформаційні технології). –Харківський національний університет імені В. Н. Каразіна, Міністерство освіти і науки України, Харків, 2026.

Дисертація присвячена вирішенню актуальної науково-технічної задачі підвищення якості процесів розробки та супроводу систем інтернет-банкінгу (e-Banking) шляхом застосування мікросервісних архітектур та знання-орієнтованих моделей при їх проєктуванні.

У дисертації виконано аналіз поточного стану процесів розробки та супроводу систем e-Banking. Встановлено, що основними проблемами є відсутність застосування знання-орієнтованих моделей та інтелектуальних методів при проєктуванні цих систем. А також відсутні архітектурні патерни проєктування, які б були адаптовані до особливостей функціонування систем e-Banking.

Об'єкт дослідження – процеси розробки та супроводу систем e-Banking, побудованих з використанням мікросервісної архітектури (МСА), з урахуванням необхідності забезпечення відповідного рівня показників якості їх функціонування.

Предмет дослідження – знання-орієнтовані моделі та інформаційні технології для побудови e-Banking на основі МСА з урахуванням вимог якості до такого програмного забезпечення (ПЗ).

Наукова новизна отриманих результатів полягає в наступному:

вперше: запропонована знання-орієнтована доменна модель для процесу формування виписок за рахунками користувачів систем e-Banking, яка відрізняється від існуючих побудовою та застосуванням онтологічних профілів

користувачів таких систем, у поєднанні з експертними доменними правилами для обробки їх емпіричних знань, що дозволяє розширити інформаційний базис і враховувати семантичний контекст вимог до функціоналу системи e-Banking, і що, в кінцевому рахунку, сприяє підвищенню якості обслуговування клієнтів відповідного банку;

отримали подальший розвиток: методи аналізу та синтезу функціональної структури систем e-Banking за рахунок визначення та інтеграції у типову структуру таких систем додаткового інтелектуального модуля прогнозування можливих дій користувачів, що дозволяє адаптувати процеси використання системних обчислювальних ресурсів при виконанні деяких витратних і критично важливих операцій клієнтів банку, зокрема, при формуванні різних типів виписок за їх поточними рахунками;

удосконалено: процедури формування виписок за рахунками користувачів систем e-Banking шляхом комбінованого застосуванням знання-орієнтованої доменної моделі та модифікованого методу аналізу часових рядів, що дозволяє враховувати особливості реалізації функціональних вимог різних типів користувачів банківських послуг і забезпечує підвищення показників продуктивності та надійності функціонування таких систем у порівнянні з існуючими процедурами;

удосконалено: проєктні патерни для розробки програмного забезпечення (ПЗ) систем e-Banking шляхом використання переваг мікросервісної архітектури у поєднанні із розробленим переліком критеріїв і кількісних метрик для їх визначення, що забезпечує можливість отримання кращих показників якості функціонування ПЗ у порівнянні із монолітною архітектурою таких систем.

Запропоновані моделі та технологічні методи проєктування та супроводу систем e-Banking і розроблені для їх підтримки інструментальні засоби були успішно використані для вирішення цих завдань при виконанні прикладної НДР МОН України «Концептуальні моделі, методи та технології створення адаптивних інформаційних систем на основі знання-орієнтованих підходів та

засобів розробки програмного забезпечення» (№ДР: 0121U110310) на кафедрі інтелектуальних програмних систем і технологій Харківського національного університету ім. В.Н. Каразіна, а також в розробках систем e-Banking української групи компаній CS Ltd (м. Харків).

У вступі до дисертаційної роботи обґрунтовано актуальність теми дослідження, показано зв'язок дослідження з науковими темами. Сформульовано мету дослідження, визначено предмет та об'єкт дослідження. Визначено методи дослідження, які базуються на застосуванні об'єктно-орієнтованого аналізу та синтезу ПЗ із використанням уніфікованої мови моделювання UML, нотацій сімейства IDEF для формалізації процедур проєктування ПЗ систем e-Banking, кількісних метрик оцінки якості ПЗ, технологій побудови доменних моделей та еталонних системних архітектур. Описано наукову новизну та практичне значення отриманих результатів. Наведено інформацію про практичне застосування та особистий внесок здобувача, апробацію результатів дослідження та їх висвітлення у публікаціях, а також містяться відомості щодо структури та обсягу дисертаційної роботи.

У першому розділі виконано аналітичний огляд існуючих підходів до проєктування систем e-Banking. Огляд та порівняльний аналіз функціонуючих в Україні найбільш поширених систем e-Banking дозволив визначити типову функціональність систем цього класу, а також визначити основні критерії, за якими можна виконати оцінку типової системи e-Banking. Було визначено, що більшість систем досі побудовані з використанням монолітних архітектур, а також майже без застосування знання-орієнтованих моделей та інтелектуальних методів. В свою чергу використання МСА дозволяє значно ширше та якісніше використовувати модельні методи як при побудові нових систем, так і при додаванні нового функціоналу у вже існуючі. Виділено основні проблеми, з якими стикаються при розробці цих систем. Виконано постановку основних задач дослідження та їх зв'язок з очікуваними науковими та практичними результатами, а також побудовано концептуальну схему дисертаційного дослідження.

У другому розділі були представлені методологічні основи розробки та супроводу систем e-Banking. Розглянуто можливість застосування доменного моделювання в якості концептуальної основи для розробки перспективної версії системи e-Banking. Була запропонована схема розширеної функціональності для систем e-Banking, яка передбачає використання інтелектуального модуля прогнозування (ІМП). Модуль має застосовуватися для прогнозування дій користувача при побудові виписок за рахунками. Це має дозволити системі заздалегідь підготовити дані, які необхідні для формування конкретної виписки, особливо виписок великого розміру. Що в свою чергу дозволяє адаптувати процеси використання системних обчислювальних ресурсів при виконанні витратних і критично важливих операції клієнтів банку. У цьому розділі також було проведено огляд можливих підходів до побудови ІМП. На основі проведеного аналізу було аргументовано обрано метод аналізу часових рядів SARIMA в якості базового інтелектуального методу, який використовуватиме знання-орієнтовані моделі під час свого функціонування. Враховуючи визначені нефункціональні вимоги до ПЗ систем e-Banking було мотивовано запропоновано використання МСА для побудови програмної архітектури розширеної системи e-Banking.

У третьому розділі було розроблено та наведено у вигляді IDEF0-діаграми бізнес-логіку модуля ІМП, а також побудовано доменну модель для функціоналу формування виписок за рахунками користувачів у системі e-Banking. Визначено основні інформаційні сутності з їх атрибутами та зв’язки між сутностями. Запропонована доменна модель є знання-орієнтованою та відрізняється від існуючих побудовою та застосуванням онтологічних профілів користувачів систем e-Banking, у поєднанні з експертними доменними правилами для обробки їх емпіричних знань, що дозволяє розширити інформаційний базис і враховувати семантичний контекст вимог до функціоналу таких систем.

Також у цьому розділі було виконано проєктування МСА для реалізації пропонованого підходу у системі e-Banking, зокрема імплементацію ІМП.

Архітектура наведена у вигляді UML-діаграми розгортання компонентів. Провідну роль у проєктуванні МСА відіграє доменне моделювання (Domain-Driven Design – DDD), яке використовується для визначення меж сервісів та формалізації бізнес-логіки предметної області. Було визначено сервіси, на які має бути поділена система, а також протоколи для їх взаємодії. Для імплементації пропонованої МСА було розроблено концептуальні принципи її проєктування та визначено перелік архітектурних патернів, придатних для інтеграції у системи e-Banking та які мають дозволити досягти кращих показників якості ПЗ. Окремо було визначено критерії оцінки якості функціонування ІМП та пропонованого підходу в цілому. При цьому враховані як суто технічні показники, так і переваги для кінцевих користувачів системи e-Banking та/або банку.

Четвертий розділ дисертації присвячено програмній реалізації та експериментальному дослідженню запропонованого підходу та розробці модельно-технологічного інструментарію. Було мотивовано запропоновано стек технологій для реалізації МСА у системі e-Banking, який також враховує імплементацію знання-орієнтованого модуля ІМП. В процесі дослідження було розроблено архітектуру та програмно реалізовано прототип фрагменту системи e-Banking, який обмежений доменом «формування виписок за рахунком». Було виконано експериментальне дослідження розробленого прототипу та на основі визначених критеріїв та метрик виконано оцінку показників якості при застосуванні запропонованого у роботі підходу. Аналіз отриманих метрик показав, що використання МСА та ІМП дозволило покращити показники продуктивності та надійності у порівнянні з використовуваними підходами до побудови виписок за рахунками у системах з монолітними архітектурами. В середньому досягнуто зменшення навантаження на ЦП на 12–15%. Пропускна здатність була збільшена в середньому на 60%. На випсках великого розміру було досягнуто суттєве зменшення відсотку помилок. Для виписки на 500 сторінок: з 5% до 0.39%. Такі результати зумовлені як застосуванням МСА та

передового програмного стеку технологій, так і використанням знання-орієнтованих моделей та інтелектуальних методів.

Результати дисертаційного дослідження були також використані у навчальному процесі кафедри інтелектуальних програмних систем і технологій Харківського національного університету імені В.Н. Каразіна. Результати знайшли застосування для розробки та оновлення робочих програм та завдань для лекцій, лабораторних та практичних робіт студентів за ОП F3 (бакалаври) – «Комп'ютерні науки» у дисциплінах «Основи бізнес-аналізу та інженерії вимог до програмного забезпечення» та «Методи та засоби доменного моделювання варіабельних програмних систем».

У дисертаційній роботі вирішена актуальна науково-прикладна задача підвищення якості процесів розробки та супроводу систем e-Banking шляхом застосування МСА та знання-орієнтованих моделей. Запропонований у роботі підхід забезпечує покращення основних показників якості ПЗ, таких як продуктивність та надійність, а також відкриває перспективи для подальших досліджень його застосування в інших доменних областях систем e-Banking.

Ключові слова: *системи інтернет-банкінгу, доменне моделювання, програмне забезпечення, мікросервісна архітектура, знання-орієнтовані моделі, виписка за рахунком, прогнозування, методи аналізу часових рядів, метрики якості, інтелектуальний підхід, продуктивність, надійність, масштабування системи, UML-діаграми*

ABSTRACT

Daas T. Knowledge-oriented models and microservice architectures for the development of internet banking systems. – Qualification scholarly paper: a manuscript.

The dissertation submitted for obtaining the Doctor of Philosophy degree in Information Technology: Speciality 122 Computer science. V. N Karazin Kharkiv National University, Ministry of Education and Science of Ukraine, Kharkiv, 2026.

The dissertation is devoted to solving a relevant scientific and technical problem of improving the quality of development and maintenance processes for internet banking (e-Banking) systems through the application of microservice architectures and knowledge-oriented models during their design.

The dissertation provides an analysis of the current state of development and maintenance processes for e-Banking systems. It has been established that the primary challenges include the lack of knowledge-oriented models and intelligent methods in the design of these systems. Furthermore, there is a lack of architectural design patterns adapted to the functional characteristics of e-Banking systems.

The object of the study is the processes of development and maintenance of e-Banking systems built using microservice architecture (MSA), considering the necessity of ensuring appropriate quality metrics for their operation.

The subject of the study is knowledge-oriented models and information technologies for constructing e-Banking based on MSA, taking into account the quality requirements for such software.

The scientific novelty of the obtained results is as follows:

For the first time: a knowledge-oriented domain model for the process of generating account statements for e-Banking users has been proposed. It differs from existing models through the construction and application of ontological user profiles in combination with expert domain rules for processing empirical knowledge. This allows for the expansion of the information basis and accounts for the semantic

context of functional requirements for e-Banking systems, ultimately contributing to the improvement of customer service quality for the respective bank.

Have been further developed: methods for the analysis and synthesis of the functional structure of e-Banking systems. This was achieved by defining and integrating an additional intelligent module for predicting potential user actions into the standard system structure. This integration enables the adaptation of system computing resource utilization when performing certain resource-intensive and critical client operations, specifically during the generation of various types of current account statements.

Have been improved: procedures for generating account statements for e-Banking users through the combined application of a knowledge-oriented domain model and a modified time-series analysis method. This approach accounts for the specific functional requirements of different types of banking service users and ensures enhanced performance and reliability metrics compared to existing procedures.

*Have been improved: d*esign patterns for e-Banking software development. This improvement was achieved by leveraging the advantages of microservice architecture in conjunction with a developed set of criteria and quantitative metrics for their determination. This provides the capability to achieve superior software performance and quality metrics compared to the monolithic architecture of such systems.

The proposed models and technological methods for the design and maintenance of e-Banking systems, along with the instrumental tools developed for their support, were successfully utilized to solve these tasks during the execution of the applied R&D project of the Ministry of Education and Science of Ukraine, titled "Conceptual models, methods, and technologies for creating adaptive information systems based on knowledge-oriented approaches and software development tools" (No. 0121U110310) at the Department of Intelligent Software Systems and Technologies of V. N. Karazin Kharkiv National University, as well as in the development of e-Banking systems by the Ukrainian group of companies CS Ltd (Kharkiv).

The Introduction substantiates the relevance of the research topic and demonstrates the connection between the study and scientific programs. It formulates the research goal and defines the subject and object of the study. The research methods are identified, based on the application of object-oriented analysis and software synthesis using the Unified Modeling Language (UML), the IDEF family of notations for formalizing the software design procedures of e-Banking systems, quantitative software quality assessment metrics, technologies for constructing domain models, and reference system architectures. The scientific novelty and practical significance of the obtained results are described. Information is provided regarding the practical application and personal contribution of the applicant, the approbation of research results and their coverage in publications, as well as details regarding the structure and volume of the dissertation.

In the first chapter, an analytical review of existing approaches to the design of e-Banking systems is performed. A review and comparative analysis of the most widespread e-Banking systems operating in Ukraine allowed for the identification of typical functionality for systems of this class, as well as the determination of the main criteria for evaluating a typical e-Banking system. It was established that most systems are still built using monolithic architectures, with almost no application of knowledge-oriented models or intelligent methods. In turn, the use of microservice architecture allows for a significantly broader and higher-quality application of modeling methods both when building new systems and when adding new functionality to existing ones. The main challenges encountered during the development of these systems are highlighted. The formulation of the primary research tasks and their connection to the expected scientific and practical results is performed, and a conceptual framework of the dissertation research is constructed.

In the second chapter, the methodological foundations for the development and maintenance of e-Banking systems were presented. The possibility of applying domain modeling as a conceptual basis for developing a perspective version of an e-Banking system is considered. An extended functionality scheme for e-Banking systems has been proposed, which involves the use of an Intelligent Forecasting

Module (IFM). The module is intended to predict user actions during the generation of account statements. This should allow the system to pre-prepare the data necessary for generating specific statements, particularly large-scale ones. This, in turn, enables the adaptation of system computing resource utilization when performing resource-intensive and critical banking operations. This chapter also provides an overview of potential approaches to constructing the IFM. Based on the analysis, the SARIMA time-series analysis method was argumentatively selected as the core intelligent method, utilizing knowledge-oriented models during its operation. Considering the established non-functional requirements for e-Banking software, the use of Microservice Architecture (MSA) for constructing the software architecture of the expanded e-Banking system was motivated and proposed.

In the third chapter, the business logic of the IFM was developed and presented in the form of an IDEF0 diagram, and a domain model was constructed for the account statement generation functionality in the e-Banking system. Key information entities, their attributes, and the relationships between entities were defined. The proposed domain model is knowledge-oriented and differs from existing ones by the construction and application of ontological profiles of e-Banking system users, combined with expert domain rules for processing their empirical knowledge. This allows for the expansion of the information basis and accounts for the semantic context of the functional requirements of such systems.

Furthermore, this chapter covers the design of the MSA for implementing the proposed approach in the e-Banking system, specifically the implementation of the IFM. The architecture is presented as a UML deployment diagram. Domain-Driven Design (DDD) plays a leading role in the MSA design, being used to define service boundaries and formalize the business logic of the subject area. The services into which the system should be decomposed were identified, along with the protocols for their interaction. To implement the proposed MSA, conceptual design principles were developed, and a list of architectural patterns suitable for integration into e-Banking systems was defined to achieve superior software quality metrics. Separately, the criteria for evaluating the operational quality of the IFM and the proposed approach

as a whole were established. These criteria account for both purely technical indicators and benefits for the end-users of the e-Banking system and the bank.

The fourth chapter of the dissertation is devoted to the software implementation and experimental study of the proposed approach, as well as the development of model-technological tools. A technology stack for implementing the MSA within the e-Banking system was argumentatively proposed, which also accounts for the implementation of the knowledge-oriented IFM module. In the course of the study, the architecture was developed and a software prototype of an e-Banking system fragment, limited to the "account statement generation" domain, was implemented. An experimental study of the developed prototype was conducted, and based on the established criteria and metrics, an assessment of quality indicators was performed using the approach proposed in the work.

Analysis of the obtained metrics demonstrated that the use of MSA and IFM allowed for improved performance and reliability indicators compared to the approaches used for account statement generation in systems with monolithic architectures. On average, a reduction in CPU load of 12–15% was achieved. Throughoutput was increased by an average of 60%. For large-scale statements, a significant reduction in the error rate was achieved; specifically, for a 500-page statement, the error rate decreased from 5% to 0.39%. These results are attributed to both the application of MSA and a cutting-edge software technology stack, as well as the use of knowledge-oriented models and intelligent methods.

The results of the dissertation research were also utilized in the educational process of the Department of Intelligent Software Systems and Technologies at V. N. Karazin Kharkiv National University. The findings were applied to the development and updating of syllabi and assignments for lectures, laboratory, and practical works for students of the F3 (Bachelor's) Educational Program – "Computer Science" within the courses "Fundamentals of Business Analysis and Software Requirements Engineering" and "Methods and Tools for Domain Modeling of Variable Software Systems."

In the dissertation, a relevant scientific and applied problem of improving the quality of development and maintenance processes for e-Banking systems through the application of MSA and knowledge-oriented models has been solved. The approach proposed in this work ensures the enhancement of key software quality indicators, such as performance and reliability, and opens prospects for further research into its application in other domain areas of e-Banking systems.

Keywords: *e-banking systems, domain modeling, software, microservice architecture, knowledge-based models, account statement, forecasting, time series analysis methods, quality metrics, intelligent approach, performance, reliability, system scaling, UML diagrams*

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Статті у наукових фахових виданнях, що входять до міжнародних наукометричних баз Scopus та Web of Science:

1. Daas, T., Tkachuk, M. (2026). Development of intelligent model-technological tools for e-Banking systems based on microservices. Eastern-European Journal of Enterprise Technologies, 1 (2 (139)), 48-57. https://doi.org/10.15587/1729-4061.2026.351604

   (Особистий внесок здобувача: розробка архітектури, розробка програмного забезпечення, експериментальне дослідження і аналіз результатів, написання тексту роботи. Особистий внесок Миколи Ткачука: концептуалізація, перевірка тексту роботи, перевірка наукової достовірності отримуваних результатів, редагування)

Статті у наукових фахових виданнях України:

2. Даас Т.І., Ткачук М.В. Аналіз сучасного стану і перспектив розвитку у галузі розробки та супроводу систем інтернет-банкінгу. Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління. 2024. вип. 62. С.6– 18. https://doi.org/10.26565/2304-6201-2024-62-01

   (Особистий внесок здобувача: проведення аналізу систем інтернет-банкінгу, визначення перспектив розвитку. Особистий внесок Миколи Ткачука: концептуалізація, перевірка тексту роботи, редагування)

3. Даас Т.І., Ткачук М.В. Застосування методів аналізу часових рядів та доменного моделювання при розробці інтелектуального модуля прогнозування в системі інтернет-банкінгу. Системи обробки інформації. 2025. вип. 3 (182). С.34-43.

https://doi.org/10.30748/soi.2025.182.04

(Особистий внесок здобувача: побудова доменної моделі, розробка алгоритму роботи інтелектуальноо модуля прогнозування, верифікація прототипу. Особистий внесок Миколи Ткачука: концептуалізація, перевірка тексту роботи, перевірка наукової достовірності отримуваних результатів, редагування)

Наукові праці, які засвідчують апробацію матеріалів дисертації:

4. Даас Т.І., Ткачук М.В. Про один підхід до розробки банківських інформаційних систем: знання-орієнтовані моделі та мікросервісні архітектури // Збірник наукових праць міжнародної науково-технічної конференції «Комп'ютерне моделювання в наукоємних технологіях» (КМНТ-2022). м. Харків, 23-25 листопада 2022 р. – С. 54 – 57.
5. Daas T.I. Towards domain modeling approach to software development for bank information systems // Матеріали XXIII Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів «Стан, досягнення та перспективи інформаційних систем і технологій». м. Одеса, 20 – 21 квітня 2023 р. – Одеса, Видавництво ОНТУ, 2023 р. – С. 183 – 185.
6. Даас Т.І., Ткачук М.В. Застосування методів аналізу часових рядів для розробки інтелектуального модуля прогнозування у системах e-Banking. Матеріали XVII Міжнародної науково-практичної конференції «Інформаційні технології і автоматизація 2024». м. Одеса, 31 жовтня – 1 листопада 2024 р. – Одеса, Видавництво ОНТУ, 2024 р. – С. 442– 445.
7. Даас Т.І., Ткачук М.В. До питання вдосконалення алгоритмів побудови виписок за рахунками клієнтів систем e-Banking // Збірник наукових праць міжнародної науково-технічної конференції «Комп'ютерне моделювання в наукоємних технологіях» (КМНТ-2024). м. Харків, 27-29 листопада 2024 р. – С. 73 – 76.

8. Daas T.I. Design principles and tools to develop of microservice architecture for e-banking systems // Матеріали XVI Міжнародної науково-практичної конференції "Пріоритетні шляхи розвитку науки і освіти". Львів. 29-30 вересня 2025 р. – С. 51 – 55.
9. Даас Т.І., Ткачук М.В. Інтелектуальні підходи до розробки та супроводу мікросервісних архітектур: класифікація, основні виклики та досвід застосування // Збірник наукових праць міжнародної науково-технічної конференції «Інтелектуальні технології у міждисциплінарних дослідженнях» (ІТМД-2025). м. Харків, 12-14 листопада 2025 р.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

ПЗ	–	Програмне забезпечення
МСА	–	Мікросервісна архітектура
МА	–	Монолітна архітектура
API	–	Application programming interface
UML	–	Unified Modeling Language, уніфікована мова моделювання
BPMN	–	Business Process Model and Notation, модель та нотація бізнес-процесів
ФОП	–	Фізична особа-підприємець
АБС	–	Автоматизована банківська система
MQ	–	Message Queue, черга повідомлень
MVC	–	Model-View Controller
DDD	–	Domain-Driver Design, проблемно-орієнтоване проєктування
OTP	–	One-Time Password, одноразовий пароль
SLA	–	Service Level Agreement, угода про рівень надання послуг
XML	–	Extensible Markup Language, розширювана мова розмітки

ВСТУП

Обґрунтування вибору теми дослідження. Попри значний прогрес у розвитку систем e-Banking, наявні рішення все ще не забезпечують належного рівня якості та гнучкості програмного забезпечення. Це обумовлює необхідність глибшого аналізу архітектурних особливостей побудови таких систем. На практиці значна частина існуючих рішень базується на трирівневій монолітній архітектурі, що передбачає централізовану реалізацію бізнес-логіки та використання спільних ресурсів. Такий підхід ускладнює масштабування системи, підвищує ризик впливу змін на всі її компоненти та призводить до зниження продуктивності при пікових навантаженнях. Додатковою проблемою є відсутність формалізованих доменних моделей та знання-орієнтованих підходів, що ускладнює опис складної бізнес-логіки та призводить до надмірної залежності від експертного досвіду при прийнятті рішень.Розробка систем інтернет-банкінгу потребує вирішення проблеми їх проєктування з урахуванням постійного розвитку галузі та появи нових вимог до ПЗ, як функціональних, так і нефункціональних. Зокрема, актуальними є задачі забезпечення масштабованості, підвищення надійності, спрощення процесів тестування та розгортання, а також скорочення витрат на супровід і розвиток систем. Відсутність інтелектуальних механізмів підтримки прийняття рішень та адаптації до змін поведінки користувачів додатково обмежує ефективність існуючих рішень. Таким чином, побудова систем e-Banking з забезпеченням належного рівня показників якості вимагає обґрунтованого вибору еталонної системної архітектури, а також використання сучасних методів обробки знань щодо вимог користувачів. І тому, питання побудови цього класу систем є актуальною науково-технічною задачею.

Мета і задачі дослідження. Основною метою дослідження є підвищення якості процесів створення систем e-Banking шляхом використання знання-орієнтованих моделей і методів, у поєднанні з перевагами застосування мікросервісної архітектури, для розробки програмного забезпечення цих

систем. Для досягнення поставленої мети було сформульовано перелік основних задач виконання дослідження:

1. Виконати аналіз існуючих підходів, які використовуються для побудови систем класу e-Banking та аналіз сучасних проблем розробки і супроводу цих систем.
2. Визначити основні функціональні та нефункціональні вимоги до систем класу e-Banking, а також основні властивості (features) програмного забезпечення цих систем, виконати порівняльних аналіз сучасних систем.
3. Проаналізувати типову функціональну структуру існуючих систем e-Banking й запропонувати можливі шляхи її вдосконалення за рахунок розробки та впровадження додаткових інтелектуальних сервісів, зокрема, для домену «Формування виписок за рахунками клієнтів».
4. Розробити знання-орієнтовану доменну модель для подальшого застосування у процесах предметно-орієнтованого проєктування програмного забезпечення таких систем та визначити метод прогнозування можливих дій користувачів.
5. Запропонувати вдосконалення процедури формування виписок за рахунками користувачів систем e-Banking шляхом комбінованого застосуванням знання-орієнтованої доменної моделі та методу аналізу часових рядів.
6. Розробити перспективну компонентну архітектуру для програмного забезпечення систем e-Banking на основі мікросервісної архітектури та розробити проблемно-орієнтовані принципи її проєктування.
7. Побудувати колекцію патернів проєктування для розробки програмного забезпечення систем e-Banking шляхом використання переваг застосування мікросервісної архітектури.
8. Визначити систему критеріїв та розробити методику їх застосування для оцінки якості програмного забезпечення таких системи.

9. Спроектувати та реалізувати прототип фрагменту системи e-Banking, який функціонально обмежений доменом «Формування виписок за рахунками клієнтів».
10. Провести експериментальне дослідження працездатності прототипу системи e-Banking, визначити переваги та недоліки запропонованого підходу і надати практичні рекомендації щодо його можливого розвитку та застосування у промисловій експлуатації.

Об'єктом дослідження є процеси розробки та супроводу систем e-Banking, побудованих з використанням мікросервісної архітектури, з урахуванням необхідності забезпечення вимог до відповідного рівня показників якості їх функціонування.

Предмет дослідження – знання-орієнтовані моделі та інформаційні технології для побудови e-Banking на основі мікросервісної архітектури з урахуванням вимог якості до такого програмного забезпечення.

Методи дослідження. У роботі використано комплекс методів, що доповнюють результати одне одного. Для аналізу сучасного стану систем e-Banking та визначення їх основних проблем застосовано методи системного аналізу та порівняльного аналізу, що дозволило обґрунтувати доцільність використання МСА. Для побудови знання-орієнтованої доменної моделі використано методи доменного моделювання та побудови онтологій, які забезпечили формалізацію бізнес-логіки та представлення знань про предметну область. Для вдосконалення процедури формування виписок за рахунками застосовано методи аналізу часових рядів, зокрема моделі SARIMA, що дозволило реалізувати прогнозування поведінки користувачів. Пропонований алгоритм для формування виписок було наведену у нотації IDEF0. При проєктуванні архітектури системи використано методи архітектурного проєктування програмних систем, зокрема принципи та патерни МСА, а також нотацію UML. Для оцінки якості запропонованих рішень застосовано методи експериментального дослідження та статистичного аналізу результатів, що

забезпечило об’єктивність оцінювання показників продуктивності, надійності та ефективності використання ресурсів. Обґрунтований вибір зазначених методів та їх комплексне застосування забезпечують достовірність отриманих результатів і сформульованих висновків.

Наукова новизна отриманих результатів.

Вперше: запропонована знання-орієнтована доменна модель для процесу формування виписок за рахунками користувачів систем e-Banking, яка відрізняється від існуючих побудовою та застосуванням онтологічних профілів користувачів таких систем, у поєднанні з експертними доменними правилами для обробки їх емпіричних знань, що дозволяє розширити інформаційний базис і враховувати семантичний контекст вимог до функціоналу системи e-Banking, і що, в кінцевому рахунку, сприяє підвищенню якості обслуговування клієнтів відповідного банку.

Отримали подальший розвиток: методи аналізу та синтезу функціональної структури систем e-Banking за рахунок визначення та інтеграції у типову структуру таких систем додаткового інтелектуального модуля прогнозування можливих дій користувачів, що дозволяє адаптувати процеси використання системних обчислювальних ресурсів при виконанні деяких витратних і критично важливих операцій клієнтів банку, зокрема, при формуванні різних типів виписок за їх поточними рахунками.

Удосконалено: процедури формування виписок за рахунками користувачів систем e-Banking шляхом комбінованого застосуванням знання-орієнтованої доменної моделі та модифікованого методу аналізу часових рядів, що дозволяє враховувати особливості реалізації функціональних вимог різних типів користувачів банківських послуг і забезпечує підвищення показників продуктивності та надійності функціонування таких систем у порівнянні з існуючими процедурами.

Удосконалено: проєктні патерни для розробки програмного забезпечення систем e-Banking шляхом використання переваг мікросервісної архітектури у

поєднанні із розробленим переліком критеріїв і кількісних метрик для їх визначення, що забезпечує можливість отримання кращих показників якості функціонування програмного забезпечення у порівнянні із монолітною архітектурою таких систем.

Отримані результати мають значний потенціал для практичного застосування в ІТ-індустрії, зокрема в системах e-Banking, у яких одночасно працює в системі онлайн велика кількість користувачів, що робить дослідження вкрай актуальним та корисним для подальшого розвитку банківського програмного забезпечення в цілому.

Особистий внесок здобувача. Дисертаційне дослідження виконано здобувачем самостійно, усі сформульовані в ньому положення, результати та висновки зроблені на основі власних досліджень здобувача. Для аргументації окремих тез використані праці інших науковців, на які зроблені відповідні посилання.

Апробація результатів дисертації. Основні теоретичні положення, висновки і пропозиції, які містяться в дисертації, обговорювалися та були схвалені на засіданнях кафедри інтелектуальних програмних систем і технологій Харківського національного університету імені В.Н. Каразіна. Ключові положення дослідження висвітлені у доповідях на науково-технічних конференціях всеукраїнського та міжнародного рівнів:

- Міжнародній науково-технічній конференції «Комп’ютерне моделювання в наукоємних технологіях» (КМНТ-2022). м. Харків, 23-25 листопада 2022 р.
- XXIII Всеукраїнській науково-технічної конференції молодих вчених, аспірантів та студентів «Стан, досягнення та перспективи інформаційних систем і технологій». м. Одеса, 20 – 21 квітня 2023 р.
- XVII Міжнародної науково-практичної конференції «Інформаційні технології і автоматизація 2024». м. Одеса, 31 жовтня – 1 листопада 2024 р.

- Міжнародній науково-технічній конференції «Комп’ютерне моделювання в наукоємних технологіях» (КМНТ-2024). м. Харків, 27-29 листопада 2024 р.
- XVI Міжнародній науково-практичній конференції "Пріоритетні шляхи розвитку науки і освіти". Львів. 29-30 вересня 2025 р.
- Міжнародній науково-технічній конференції «Інтелектуальні технології у міждисциплінарних дослідженнях» (ІТМД-2025). м. Харків, 12-14 листопада 2025 р.

Публікації. Основні наукові результати дисертаційної роботи у повній мірі викладено в 3 публікаціях, з яких: 2 статті опубліковано в фахових виданнях категорії «Б» МОН України, 1 з них опубліковано англійською мовою та проіндексовано в наукометричній базі SCOPUS у журналі Eastern-European Journal of Enterprise Technologies, 1 (2 (139)), Kharkiv, Ukraine, ISSN 1729-3774.

Структура та обсяг дисертації. Дисертаційна робота складається зі вступу, чотирьох розділів, загальних висновків та списку використаних літературних джерел, який містить 97 найменувань. Загальний обсяг дисертаційних досліджень викладено на 151 сторінці друкованого тексту, де обсяг основного тексту – 114 сторінок. Дисертація включає 22 рисунка, 8 таблиць та 3 додатки.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ ПРОБЛЕМ ПРИ РОЗРОБЦІ СИСТЕМ E-BANKING

1.1 Актуальність проблем розробки програмного забезпечення, що використовується в системах e-Banking

Сучасні світові тенденції розвитку зумовлюють необхідність впровадження все більшої кількості електронних послуг у сфері банківської діяльності, зокрема у системах інтернет-банкінгу (e-Banking). Системи e-Banking є тим класом банківських інформаційних систем, з якими найчастіше взаємодіє клієнт банку. І саме клієнт є для будь-якої банківської установи основним джерелом прибутку. Перелік таких електронних послуг та складність їх бізнес-логіки зростає щоденно. Бізнес-вимоги користувачів вимагають від розробників програмного забезпечення (ПЗ) швидкого впровадження нового функціоналу у системи e-Banking, забезпечення мінімізації, а краще, відсутності впливу нових модулів на якість та логіку роботи вже впровадженого функціоналу, а також забезпечення відмовостійкості (robustness) при пікових навантаженнях на систему. У свою чергу, банки, як кінцеві власники продукту, також висувають свої вимоги до ПЗ систем e-Banking, серед них можна виділити:

- можливість усунення помилок в роботі певного функціонала у мінімальний строк з забезпеченням "недоторканності" інших функцій системи
- мінімізація ресурсів при горизонтальному масштабуванні
- впровадження вимог регулятора (для України – це НБУ) у відведені строки
- мінімізація складнощів супроводу ПЗ ІС
- можливість моніторингу робочого стану ІС
- швидке усунення вразливостей (vulnerability), знайдених у системі чи сторонніх компонентах [1].

Останнє є вкрай важливим в умовах значного росту кількості кібератак на критичну інфраструктуру держави [2]. В умовах, коли надання банком послуг своїм клієнтам має здійснюватися у найкоротші строки та з мінімальною кількістю відвідувань відділень банку, системи e-Banking, які забезпечують дистанційне обслуговування клієнтів, грають роль одного з ключових інструментів продажу банківських продуктів і ведення банком своєї бізнес діяльності загалом. Якість функціонування таких систем є одним із важливих факторів, що сприяють залученню нових клієнтів для будь-якого банку [3]. Більшість систем e-Banking, які застосовуються в українських банках, і досі побудовані із застосуванням монолітних архітектур (МА) через певну множину організаційно-технічних чинників. Проте за багатьма показниками МА вже не встигають за динамікою розвитку бізнес-потреб користувачів і власників програмного продукту, що зумовило необхідність пошуку нових методів розробки та архітектур. Такі архітектури, як і МА, мають забезпечувати розподілену обробку даних, але водночас мають надавати кращі можливості для виконання вимог, які висуваються до ПЗ банківських інформаційних систем (БІС). Саме тому в ІТ-галузі все більшої популярності набувають мікросервісні архітектури (МСА), які забезпечують гнучкість ІТ-інфраструктури, незалежність компонентів (сервісів) один від одного, можливість адаптивного масштабування та високу здатність до швидкої модернізації ПЗ [4].

БІС, побудовані з використанням МА, як правило, починали розроблятися ще у минулому десятилітті. Це відобразилося на підходах, що були застосовані під час проєктування інформаційних систем. Більшість таких систем створювалися як інструмент для автоматизації та спрощення певних типових задач, зокрема міжбанківської взаємодії (передача платежів між фінансовими установами), контролю залишків на рахунках тощо. Подібні системи не передбачали значної кількості зовнішніх користувачів і зазвичай виконували переважно функції «підрахунків». З точки зору модельного підходу вони були доволі простими, тому застосування модельних методів під час їх проєктування та розробки фактично не потребувалося. Із розвитком електронних технологій

еволюціонували й БІС, які поступово почали охоплювати все більш різноманітний функціонал. Особливо це стосувалось функціоналу, що вимагав взаємодії з великою кількістю кінцевих користувачів. Однак, оскільки майже всі тодішні БІС були створені з використанням МА, новий функціонал зазвичай додавався у вигляді чергового «шару» до вже існуючої структури системи. Тому це не передбачало широкого використання моделей, зокрема доменних та знання-орієнтованих. Часто будувались лише UML-діаграми, причому переважно для відображення особливостей функціоналу, який було необхідно додати, і без урахування взаємодії з іншими сталими компонентами системи в цілому. Така ситуація пояснювалася тим, що використання доменних моделей передбачало необхідність відносно частого рефакторингу як архітектури ІС, так і програмного коду, що потребувало додаткових ресурсів. Водночас знання-орієнтовані моделі, для отримання якісних результатів у кінцевому продукті, вимагали залучення широкого кола експертів, яких через новизну предметної області було недостатньо [1]. У сукупності ці чинники, а також взаємна залежність компонентів у МА, зумовили потребу у пошуку такого типу архітектури, який дозволив би для підвищення якості ПЗ БІС та їх подальшого швидкого розвитку використовувати під час проєктування і розробки також науковий підхід. Зокрема, застосовувати знання-орієнтовані та доменні моделі, експертні методи пошуку рішень (зокрема метод Дельфі та метод аналізу ієрархій), методи об’єктно-орієнтованого аналізу та синтезу ПЗ із використанням мови UML (Unified Modeling Language), мови BPML (Business Process Modeling Language) і графічних нотацій сімейства IDEF.

МСА прийшла на зміну МА, яка тривалий час була основним архітектурним стилем для розробки більшості клієнт-серверних систем. МА характеризується чітко визначеною структурою застосунку, повною залежністю його компонентів між собою, використанням єдиного стеку технологій і фреймворків, а також залежністю від платформ розгортання. Натомість МСА подається як сукупність невеликих програмних сервісів, що розміщуються незалежно один від одного. Кожен із таких сервісів реалізує певний завершений

функціонал, має власний процес виконання, а взаємодія з ним здійснюється за допомогою чітко визначеного протоколу обміну даними. При цьому сервіс не має жорсткої залежності від мови програмування чи технології, за допомогою яких він реалізований, платформи, на якій розгорнутий, а також апаратних засобів, які використовуються для його функціонування [4, 5].

У МСА також наявна можливість використання різних механізмів оркестрації та контейнеризації. Це забезпечує масштабування із мінімальними витратами ресурсів, а також моніторинг робочого стану ІС. На сьогодні найпоширенішим засобом контейнеризації є Docker, тоді як оркестрацію найчастіше забезпечує Kubernetes. Це дозволяє мінімізувати ресурси, які витрачаються на розгортання додаткових екземплярів мікросервісів, оскільки розгортається лише той мікросервіс, на який зросло навантаження під впливом зовнішніх факторів. На противагу, у МА у разі потреби масштабування необхідно розгортати цілі сервери, у яких реалізовано значний об’єм функціоналу системи, включно з тим, необхідність використання якого не збільшилася. Для систем e-Banking це має особливе значення, оскільки вони характеризуються нерівномірним використанням різних функціональних можливостей ІС з боку клієнтів, а також існує певна закономірність у часі виникнення пікових навантажень на систему. Технології оркестрації дають змогу досить гнучко забезпечувати таке масштабування, а також, що є не менш важливо, ефективно керувати утилізацією ресурсів у періоди зниження навантаження [6]. Використання МСА дозволяє значно ширше і ефективніше застосовувати модельні методи як під час розробки нових систем, так і при розширенні функціоналу для вже існуючих. Оскільки МСА передбачає наявність великої кількості мікросервісів, кожен з яких відповідає за окремий функціонал, у разі необхідності додавання нового мікросервісу з новими можливостями немає потреби перебудовувати всі раніше створені моделі та визначені онтології. Водночас розробка і використання ПЗ, побудованого на основі МСА, породжує низку проблем. До них можна віднести додаткові витрати на обмін даними між мікросервісами, потребу підтримувати

спадкоємність інтерфейсів під час розробки нового функціоналу, необхідність забезпечення коректної роботи мікросервісу у випадку збою іншого компонента ІС, а також необхідність залучення кваліфікованого персоналу для супроводу системи [4, 5].

У теперішній час значна увага приділяється питанням вдосконалення підходів до розробки та оцінювання якості програмного забезпечення систем e-Banking, які проєктуються і розробляються з використанням МСА. Так, наприклад, у роботі [7] на основі аналізу сучасних публікацій, як закордонних так і вітчизняних, наводяться такі показники якості розробки та впровадження МСА, як середній обсяг коду мікросервісу та ступінь зв’язаності, причому як окремих мікросервісів, так і програмного коду кожного сервісу окремо. Ці метрики дозволяють кількісно оцінити якість архітектурного рішення, однак залишають поза увагою зовнішні фактори, зокрема інфраструктуру, стек технологій тощо. У роботі [8] описано відмінності між програмним забезпеченням, побудованим на основі МА та МСА, а також наведено приклад реалізації конкретної системи з визначенням основних показників якості. Проте при цьому практично не розглядаються більш складні питання вдосконалення методологічних підходів до адаптивної розробки БІС.

Отже, попри значний прогрес у розвитку систем e-Banking, наявні рішення все ще не забезпечують належного рівня якості та гнучкості програмного забезпечення. Це обумовлює необхідність глибшого аналізу архітектурних особливостей побудови таких систем, що й стане предметом розгляду у підрозділі 1.2.

1.2 Аналіз особливостей побудови та функціонування систем e-Banking

Перш за все, для аналізу особливостей побудови та функціонування систем класу e-Banking необхідно взагалі визначити їх місце та призначення у загальній структурі систем категорії БІС. Системи класу e-Banking відносяться до типу “платіжних”, що детальніше наведено на рисунку 1.1, і можуть також тісно взаємодіяти з системами інших типів на рівні інтеграційних інтерфейсів

[9]. З точки зору використовуваних архітектур, системи e-Banking зазвичай побудовані з використанням або МА, або МСА, як і в цілому інші ІС банків, про що було зазначено у попередньому підрозділі. Але варто відзначити, що в процесі розвитку системи e-Banking, які побудовані з використанням МА, починають поступово впроваджувати МСА, як більш технологічно зрілу еталонну системну архітектуру, яка, попри наявні складності її використання впровадження, має все ж таки суттєві переваги порівняно з МА.

Рис. 1.1. Місце і призначення системи e-Banking у загальній структурі БІС

Яскравим прикладом таких складнощів є необхідність при побудові ПЗ систем e-Banking, з використанням МСА, звертати особливу увагу на процес поділу функціоналу на мікросервіси, оскільки в ІС цього класу завжди присутній службовий функціонал, який може використовуватися як один з етапів бізнес-логіки для великої частини кінцевого функціоналу, який виділений кожний в окремий мікросервіс. Це створює ситуацію, коли неможливо повністю реалізувати одну з вимог МСА – «незалежність» [4].

Прикладом такого функціоналу є робота з електронним підписом (ЕП) чи OTP-кодами, які використовуються як підтвердження авторизації в системі, платіжних документів чи неплатіжних операцій. Реалізувати його можна двома способами: в кожному мікросервісі, який обробляє платежі/операції або в окремому мікросервісі, до якого за потреби звертатимуться інші мікросервіси. Перший варіант призводить до дублювання логіки, і таке дублювання зростає зі зростанням кількості мікросервісів, а другий варіант підвищує залежність мікросервісів між собою. Тому вибір варіанту для реалізації повинен залежить від функціональних особливостей ІС, та виконується лише після розрахування кількісних та якісних показників, до яких можна віднести: кількість строк коду, незмінність функціоналу, швидкодія, тощо [9].

Для можливості обґрунтованого внесення власних пропозицій щодо вдосконалення процесів проєктування та розробки систем e-Banking було зроблено, на підставі аналізу інформаційних джерел, порівняльний огляд деяких існуючих таких систем, які представлені в Україні [10-17].

Усі існуючі системи e-Banking можна умовно розділити на два класи:

- розвинені комерційні рішення;
- системи власної розробки.

Варто відзначити, що для систем e-Banking відсутній такий клас, як “безкоштовні програмні продукти” (open source). Це обумовлено тим, що системи e-Banking мають відповідати міжнародним стандартам, таким як наприклад PCI DSS, а також постійно впроваджувати нові вимоги національного регулятора [9].

До програмних комерційних рішень із групи (1) належать, наприклад, такі системи як iFOBS, яка розроблена компанією CS, і iBank2, яка розроблена компанією ДБО Софт. Перший продукт є одним з лідерів на ринку і є більш популярним серед юридичних осіб та ФОП, аніж для фізичних осіб. Абсолютна більшість банків, при наданні своїх послуг для юридичних осіб, використовують саме систему iFOBS. Система iBank2 також більше

використовується банками при наданні послуг юридичним особам, але є менш популярною через меншу кількість функціоналу та інтеграційних можливостей.

Програмні продукти групи (2) також використовуються досить широко, і зазвичай розробляються власними командами програмістів найбільш успішних банків, які краще представлені у сегменті фізичних осіб. Це обумовлено необхідністю швидкого впровадження оновлень для реакції на зміни у суспільних процесах, а також бажанням мати власний унікальний дизайн застосунку. Характерними представниками цього класу є системи e-Banking таких банків як Приватбанк, Монобанк та А-Банк [9].

Отримані результати аналізу опубліковані в статті Дааса Т.І. та Ткачука М.В. «Аналіз сучасного стану і перспектив розвитку у галузі розробки та супроводу систем інтернет-банкінгу» у «Віснику Харківського національного університету імені Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». 2024. вип. 62, с.6–18.

Нижче стисло розглянемо характеристики деяких найбільш поширених систем e-Banking, з метою побудови та аналізу, в подальшому, схеми їх типової функціональності.

1.2.1. Система iFOBS

Ця система є однією з найбільш популярних систем на українському ринку, оскільки є комерційним продуктом, який має велику різноманітність функціоналу та можливості для інтеграції з core-системами банку, такими як, наприклад, автоматизовані банківські системи(АБС) чи CRM-системи. Система представлена у двох системних архітектурах: монолітній та мікросервісній.

Система надає для клієнтів Web-інтерфейс, Windows-додаток і мобільні додатки під платформи IOS та Android, а також інтерфейс для адміністрування. Додатково система підтримує для клієнтів відкриті інтерфейси для зовнішньої інтеграції (Open API) [10, 11].

Серед найбільш цікавих особливостей цього продукту можна виділити:

- наявність окремих інтерфейсів для клієнтів фізичних осіб, юридичних осіб, а також ФОП. Кожен інтерфейс розроблений з урахуванням специфіки роботи і необхідного функціоналу кожної з груп користувачів
- наявність гнучкої рольової системи як для користувачів, так і для адміністраторів
- широкий спектр підтримуваних засобів авторизації та їх комбінації
- висока ступінь масштабованості системи у версії з мікросервісною архітектурою

1.2.2. Система iBank2

Ця система також є комерційним продуктом, але вона менше представлена на ринку через гіршу різноманітність функціоналу, а також гіршу продуктивність при високому навантаженні. Система наразі представлена лише у монолітній архітектурі, але все одно експлуатується декількома банками національного рівня, такими як, наприклад, Райффайзен Банк чи Таскомбанк.

Система надає для клієнтів лише Web-інтерфейс і мобільні додатки під платформи IOS та Android, а також інтерфейс для адміністрування [12, 13].

Особливостями даної системи є:

- наявність окремих інтерфейсів для клієнтів фізичних осіб, юридичних осіб, а також ФОП
- наявність сертифікованих криптографічних бібліотек власного виробництва, а також центру сертифікації ключів(ЦСК) [14, 15]
- можливість двофакторної авторизації за допомогою додатку Google Authenticator

Серед недоліків системи можна виділити погані можливості для масштабування, які обумовлені використаною архітектурою, а також клієнтський інтерфейс, який не відповідає вимогам до сучасного UI/UX-дизайну.

1.2.3. Система Privat24

Система Privat24 є власною розробкою Приватбанку – найбільшого українського банку. Ця система є найбільш використовуваною за кількістю користувачів. Як і попередня, ця система надає для клієнтів лише Web-інтерфейс і мобільні додатки під платформи IOS та Android, а також інтерфейс для адміністрування [16].

Серед особливостей системи можна виділити:

- наявність окремих інтерфейсів для клієнтів фізичних осіб та юридичних осіб
- найбільшу різноманітність функціоналу
- наявність засобів детального моніторингу дій користувача
- гарні можливості масштабування системи при зміні в середовищі функціонування

Система повністю наразі побудована з використанням мікросервісної архітектури, і є яскравим прикладом плавного переходу від монолітної архітектури до мікросервісної.

1.2.4. Система Monobank

Система Monobank є власною розробкою однойменного банку. Ця система була введена в експлуатацію у 2017 році і є однією з найновіших e-Banking систем, які діють на українському банківському ринку. Через новизну система є єдиним представником, серед наведених в цьому документі, яка з самого початку була побудована та розроблена з використанням МСА. Але наразі, на відміну від попередніх перерахованих систем, ця система представлена лише у вигляді мобільних додатків під платформи IOS та Android, і має інтерфейси лише для клієнтів фізичних осіб [17]. Це можна вважати основними недоліками порівняно з аналогами.

Серед інших особливостей виділяються:

- наявність засобів детального моніторингу дій користувача

– гарні можливості масштабування системи при зміні в середовищі функціонування
– найшвидше впровадження нового функціоналу в продуктивне середовище

1.2.5. Порівняльний аналіз наведених систем

Для проведення порівняльного аналізу властивостей вищезгаданих систем e-Banking, синтезу загальної схеми їх типової функціональності, а також для розробки мотивованої пропозиції щодо її можливого подальшого вдосконалення, у цій роботі використана методика, яка була запропонована у [18] для опрацювання аналогічної проблематики у галузі розробки складних систем управління IT-інфраструктурою підприємств.

Результат порівняння розглянутих в даному розділі систем наведено у табл. 1.1. Оцінки виставлялися групою експертів у сфері розробки банківського програмного забезпечення за 5-ти бальною шкалою.

Таблиця 1.1. Порівняння систем e-Banking

Критерії	iFOBS	iBank2	Privat24	Monobank
Використовувана архітектура	Частково мікросервісна	Монолітна	Мікросервісна	Мікросервісна
Масштабованість	4	3	5	5
Гнучкість конфігурування	5	4	5	5
Супроводжуваність	4	4	4	5
Інтерфейс	4	3	4	5
Різноманітність функціоналу	5	4	5	3
Безпека	4	4	4	4
Продуктивність	5	4	5	4

Можливості для інтеграції зовнішніх систем	4	4	5	4

Проведений порівняльний аналіз цих продуктів дозволяє визначити основні критерії, за якими можна виконати оцінку типової системи e-Banking. В наступних розділах буде побудована типова функціональність існуючих в Україні систем e-Banking, а також буде запропонований шлях її вдосконалення за рахунок впровадження знання-орієнтованого підходу у процес проєктування та розробки цього класу банківських інформаційних систем. Пропонований підхід включає також використання мікросервісної архітектури, яка забезпечує кращі показники якості функціонування цих систем [9].

1.2.6. Типова функціональність систем e-Banking

Типова функціональність систем e-Banking має забезпечувати весь спектр послуг, які банк, як фінансова установ, повинен надавати клієнтам у дистанційному вигляді. Як було зазначено у попередніх розділах, системи класу e-Banking є такими що динамічно розвиваються, і, відповідно, формалізована схема типової функціональності систем e-Banking, яка наведена на рисунку 1.2 у вигляді UML-діаграми пакетів, може розглядатися як певна концептуальна модель “ідеальної системи”, але лише станом на зараз. В процесі розвитку можуть додаватися нові функції, але зазвичай це будуть функції 3-го рівня, згідно наведеної діаграми [9].

Операції клієнтів. Функції цієї групи мають стратегічне значення для кожної системи e-Banking, оскільки саме вони забезпечують безпосередню взаємодію банку та його клієнтів, кількість та рівень задоволення яких відіграє головну роль в успішності функціонування банку як бізнесу. Саме у цій групі функцій основну роль відіграє дизайн, швидкодія та відмовостійкість, оскільки клієнта банку не цікавить внутрішня складова системи, її архітектура чи стек технологій. Для клієнта важливо отримати якісну послугу, без збоїв в роботі

системи та з виконанням якомога меншої кількості дій у системі. Для функцій цієї групи обов'язково має бути розроблений та підтримуватися певний рівень обслуговування (Service Level Agreement – SLA).

Система адміністрування. Ця група процесів відповідає за роботу співробітників банку, які безпосередньо взаємодіють з клієнтами: працюють у відділеннях або у відділі підтримки. Система адміністрування повинна забезпечувати повний спектр функції, які потрібні для створення та редагування облікових записів користувачів системи, в тому числі і видачі нових прав чи ролей користувачам, для виконання операції авторизації різноманітних дій користувача та/або іншого адміністратора в системі, в тому числі і в режимі “maker-checker” (або 4-eyes), для фіксації та перегляду усіх успішних та неуспішних дій будь-якого користувача, особливо критичних фінансових операції та авторизації.

Забезпечення безпеки. Функції цієї групи є одними з головних оскільки в умовах постійного розвитку методів для здійснення кібератак, банківська сфера є однією з найпріоритетніших для здійснення цих атак, оскільки обробляє цінний актив – гроші. До функцій цієї групи відносять модулі, що відповідають за вияв шахрайських та аномальних операцій, блокування їх чи накладання додаткового фактору для підтвердження, а також за сповіщення відповідального підрозділу банку про наявність підозрілих платежів у конкретного клієнта чи групи клієнтів. Для захисту від відомих та найбільш загрозливих атак на веб-додатки мають бути впроваджені функції аналізу вхідних запитів до системи для запобігання обробки модулями бізнес-логіки системи запитів, які мають ознаки атаки на веб-додаток. Перелік актуальних вразливостей підтримується учасниками відкритого проєкту захисту веб-додатків (OWASP) [19], в рамках якого також розробляється керівний документ, який де-факто є стандартом, захисту веб-додатків при проєктуванні та розробці OWASP ASVS [20]. Необхідність керуватися документами розробленими OWASP є наразі вимогою Національного банку України [21].

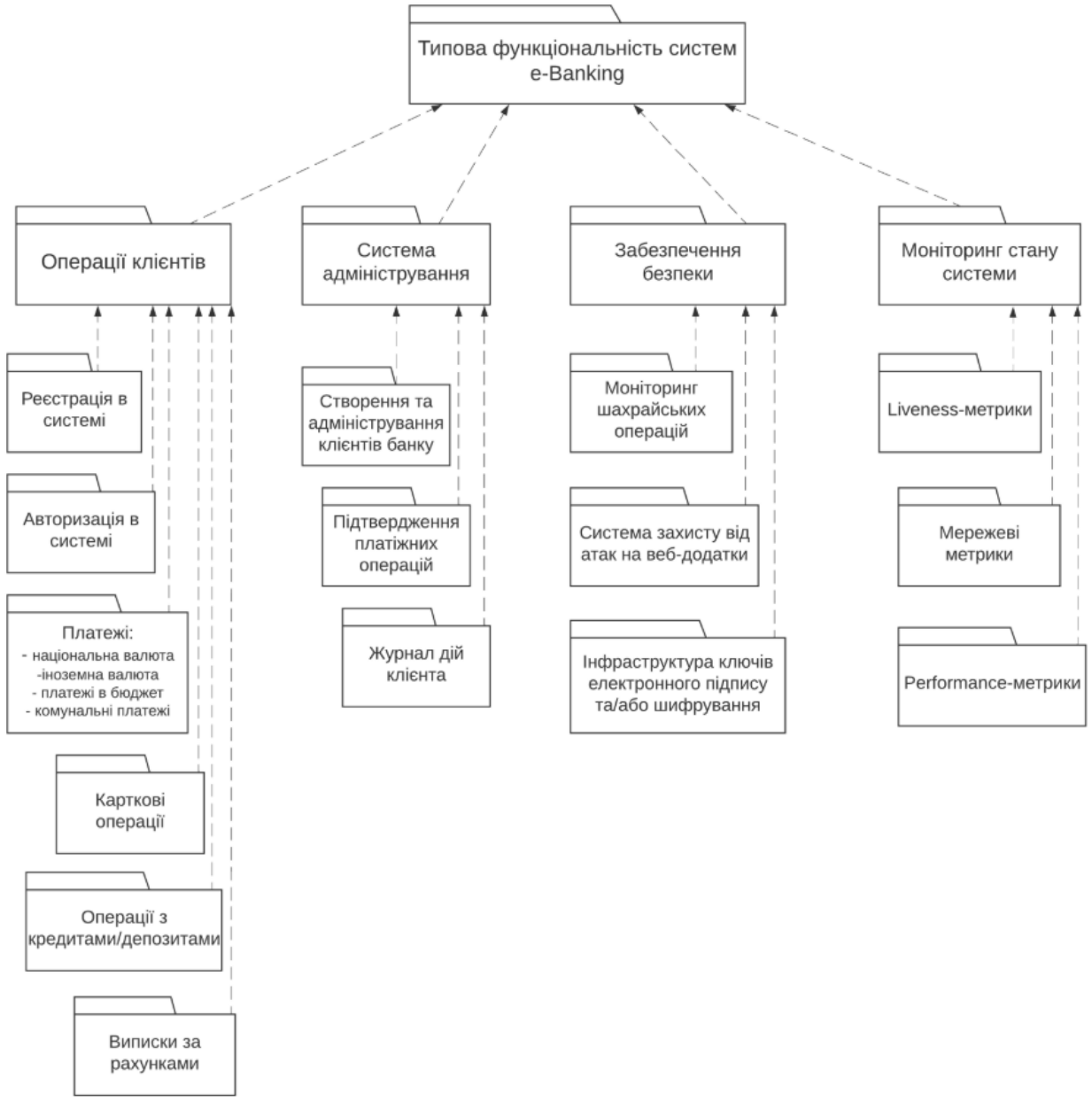

Рис. 1.2. Типова функціональність систем e-Banking

Моніторинг стану системи. Функції моніторингу забезпечують ІТ-відділам банку, які супроводжують систему, можливість оперативного реагування на позаштатні інциденти в системі, а також розслідування цих подій. До основних метрик, які повинні бути забезпечені системами e-Banking відносять внутрішні показники стану компонентів системи, які також називають liveness-метриками, мережеві показники як при взаємодії різних компонентів системи між собою, так і при взаємодії з сторонніми системами, а також метрики продуктивності

ключових бізнес-функцій системи. Для зберігання та накопичення цих метрик може бути як розроблений інтегрований в систему e-Banking модуль, так і використовуватися корпоративні системи обробки, накопичення та візуалізації метрик системи, які вже впроваджені в банку як корпоративний стандарт. До таких систем можна віднести, наприклад, Grafana, Prometheus, Zipkin, Jaeger [22-25].

Таким чином, аналіз архітектурних рішень засвідчив, що жоден із підходів ані монолітний, ані мікросервісний не є універсальним для сучасних систем e-Banking [9]. Це підкреслює важливість не лише вибору архітектури, але й застосування відповідних методів розробки та оцінки якості програмного забезпечення, що розглядатиметься у підрозділі 1.3.

1.3 Стислий аналіз підходів до розробки та оцінки якості програмного забезпечення систем e-Banking

На сучасному етапі розвитку інформаційних технологій моделювання предметної області стає все більш використовуваним стилем при побудові складних розподілених ІС. Основним підходом є проєктування, кероване предметною областю (Domain-driven design – DDD), яке було запроваджено на початку 2000-х років. Підхід був розроблений для спрощення взаємодії та співпраці експертів предметної області та розробників програмного забезпечення на всіх етапах життєвого циклу продукту. DDD – це підхід до розробки програмного забезпечення, який розділяє розробку на предметні області, які, в свою чергу, повинні проєктуватися та розроблятися максимально незалежно одна від одної. Основним принципом DDD є обмежений контекст, який полягає в чіткому визначенні меж використання моделі [26, 27].

Однією з переваг DDD є відсутність необхідності використовувати певні архітектури. Це дозволяє будувати системи, використовуючи МА або використовуючи мікросервіси. І якщо до переваг МА можна віднести простоту та швидкість розробки, особливо на початкових фазах, то недоліками цього рішення є зв'язність коду, складність горизонтального масштабування та

концентрація всіх обмежених контекстів в одному місці без будь-яких фізичних меж між ними, що є прямою небезпекою для підходу збереження DDD. У свою чергу, MCA не має перелічених недоліків, а що ще важливіше, вона сама базується на принципах обмеженого контексту, коли кожен мікросервіс реалізує лише певну виділену бізнес-логіку і нічого більше [27].

Найпопулярнішими архітектурними патернами DDD є Multi-level, Hexagon та Onion. Особливості кожного з шаблонів показані на рисунку 1.3. Усі ці шаблони легко комбінуються для забезпечення принципів MCA. При впровадженні IC на базі MCA рівень інфраструктури може включати: Discovery Service, Config Service, Gateway, Load Balancer, Database, MQ Broker та інші сервіси. Рівень додатків включатиме використовуваний фреймворк, який надає MVC-контролери та інструменти для обробки різних форматів вхідних даних, що реалізовані в системі. Рівень домену представлений у вигляді сервісів, репозиторіїв, об'єктів та агрегаторів [28].

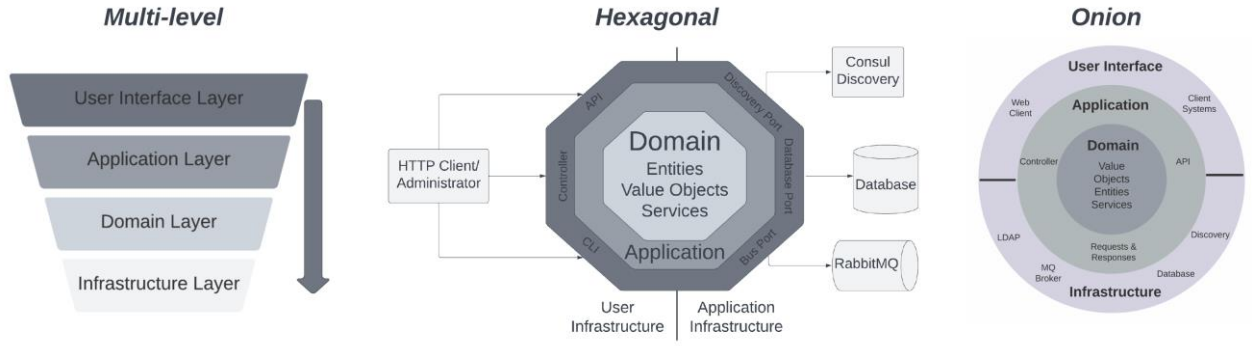

Рис. 1.3. Архітектурні шаблони DDD

Під час міграції з MA на MCA принцип DDD може бути використаний для розкладання системи на сервіси, особливо для складних розподілених систем, які потребують виділення обмежених контекстів. При цьому також потрібно вибрати основний принцип обміну даними між контекстами. Існує два основних принципи: прямий виклик сервісу та асинхронний обмін повідомленнями, наприклад, через MQ Broker. При цьому рівень домену в

кожному мікросервісі не повинен залежати від обраного принципу обміну даними, оскільки цей обмін має виконуватися на рівні інфраструктури (для відправника) та програми (для одержувача) [29].

Набагато складнішою з точки зору побудови доменів є система e-Banking, яка повинна реалізувати всі функції для взаємодії клієнта з банком. Домени, межі яких можна чітко визначити, це: користувачі та їхні права, автентифікація в системі, підтвердження транзакцій та дій у системі (електронний підпис, OTP-коди тощо) та робота з особистими та системними налаштуваннями. Такі домени мають обмежений контекст і можуть бути досить просто розділені на окремі мікросервіси. Але водночас існують певні домени, які через їх суттєвий зв'язок досить важко розділити всередині контекстів, що, у свою чергу, викликає труднощі при поділі на мікросервіси та організації процедур обміну даними між мікросервісами. Прикладами таких доменів є платежі та обмін документами. По-перше, платежі поділяються за типами, обробка яких може бути різною, що робить необхідним виділити кожен тип в окремий домен і, відповідно, мікросервіс. А по-друге, певні типи платежів вимагають вкладення різних документів, обробка яких, у свою чергу, впливає на процес обробки самого платежу. Але оскільки домен обміну документами є окремим обмеженим контекстом, його необхідно винести в окремий мікросервіс, а тому в домені конкретного типу платежу на рівні сервісу слід передбачити «компенсаційні заходи» [27].

Для формалізації результатів доменного моделювання при побудові різних складних систем часто застосовують різні нотації, які дозволяють описати знання про предметну області з максимально різних сторін. До таких нотацій, що забезпечують модельні підходи при розробці можна віднести:

- Нотації сімейства UML [30]
- Нотації сімейства IDEF
- BPMN (Business Process Model and Notation) [31]

У сучасній науковій літературі та практиці розробки програмних систем поширеним є використання міжнародних стандартів оцінки якості. Зокрема,

класичні підходи ґрунтуються на моделі ISO/IEC 9126, яка визначає базові характеристики якості програмного забезпечення, та її наступниці — ISO/IEC 25010, що деталізує вісім ключових атрибутів: функціональність, надійність, зручність використання, ефективність, сумісність, безпека, придатність до супроводу та переносимість. Ці моделі є універсальними і широко застосовуються для оцінки корпоративних і фінансових систем. Варто зазначити, що визначені у ISO/IEC 25010 дуже сильно перетинаються з атрибутами якості, які визначає інший популярний стандарт оцінки якості ПЗ – а саме SWEEBOK. Але якщо SWEEBOK більше визначає та описує основні показники якості ПЗ, то ISO/IEC 25010 зосереджений на тому, щоб описати методи, підходи та моделі саме до оцінювання визначених показників якості. З цього можна зробити висновок, що ці два стандарти доцільно використовувати разом, але на різних етапах дослідження.

Для систем e-Banking найбільш критичними є показники безпеки, надійності, ефективності(продуктивності) та зручності використання. Користувачі очікують миттєвого доступу до послуг, високого рівня захисту персональних даних і безперебійної роботи сервісу незалежно від навантаження. Тому оцінка якості таких систем повинна враховувати як технічні параметри (наприклад, час відгуку чи відмовостійкість), так і користувацькі аспекти (інтерфейс, простота взаємодії, довіра до системи) [27]. На практиці, для оцінювання якості ПЗ використовуються як кількісні метрики (середній час відгуку, кількість відмов, рівень доступності, кількість виявлених дефектів), так і якісні експертні методи (опитування користувачів, експертні оцінки, аудит відповідності стандартам). Для фінансових систем особливо важливим є поєднання формальних технічних метрик із суб’єктивними показниками довіри та зручності, що відображає комплексний характер оцінки якості в домені e-Banking. Попри наявність розвинених стандартів і метрик, існуючі методи оцінки не враховують динамічний характер поведінки користувачів та швидку еволюцію бізнес-вимог у сфері e-Banking. Зокрема, традиційні моделі якості фіксують лише статичний зріз характеристик системи,

тоді як реальні сценарії використання потребують врахування історичних даних, поведінкових патернів та прогнозних моделей. Це обмеження підкреслює необхідність пошуку нових методологічних підходів до оцінки та розвитку програмного забезпечення для e-Banking [27].

Як показав аналіз, наявні методи моделювання та оцінки якості програмного забезпечення мають обмеження, що не дозволяють у повній мірі відобразити специфіку та динаміку систем e-Banking. Це створює підґрунтя для постановки дослідницької задачі, спрямованої на пошук нових методологічних засад і рішень, які обґрунтовуються у підрозділі 1.4.

1.4 Постановка задачі дослідження дисертаційної роботи

Виходячи з зазначених висновків та визначених проблем, об’єктом дослідження є процеси розробки та супроводу систем e-Banking, побудованих з використанням МСА, з урахуванням необхідності забезпечення вимог до відповідного рівня показників якості.

Предмет дослідження – знання-орієнтовані моделі та інформаційні технології для побудови e-Banking на основі МСА з урахуванням вимог якості до такого ПЗ.

Мета дослідження – підвищення якості процесів створення систем e-Banking шляхом використання знання-орієнтованих моделей і методів, у поєднанні з перевагами застосування мікросервісної архітектури (МСА), для розробки програмного забезпечення цих систем. Для досягнення зазначеної мети та отримання очікуваних результатів в процесі виконання дослідницької роботи необхідно вирішити наступні задачі:

- виконати аналіз існуючих підходів, які використовуються для побудови систем класу e-Banking та аналіз сучасних проблем розробки і супроводу цих систем;
- визначити основні функціональні та нефункціональні вимоги до систем класу e-Banking, а також основні властивості (features) ПЗ цих систем, виконати порівняльних аналіз сучасних систем e-Banking;

- проаналізувати типову функціональну структуру існуючих систем e-Banking й запропонувати можливі шляхи її вдосконалення за рахунок розробки та впровадження додаткових інтелектуальних сервісів, зокрема, для домену «Формування виписок за рахунками клієнтів»;
- розробити знання-орієнтовану доменну модель для подальшого застосування у процесах предметно-орієнтованого проєктування (DDD approach) ПЗ таких систем та визначити метод прогнозування можливих дій користувачів;
- запропонувати вдосконалення процедури формування виписок за рахунками користувачів систем e-Banking шляхом комбінованого застосуванням знання-орієнтованої доменної моделі та методу аналізу часових рядів;
- розробити перспективну компонентну архітектуру для ПЗ систем e-Banking на основі МСА та розробити проблемно-орієнтовані принципи її проєктування;
- побудувати колекцію патернів проєктування для розробки ПЗ систем e-Banking шляхом використання переваг застосування МСА;
- визначити систему критеріїв та розробити методику їх застосування для оцінки якості ПЗ таких системи;
- спроєктувати та реалізувати прототип фрагменту системи e-Banking, який функціонально обмежений доменом «Формування виписок за рахунками клієнтів»;
- провести експериментальне дослідження працездатності прототипу системи e-Banking, визначити переваги та недоліки запропонованого підходу і надати практичні рекомендації щодо його можливого розвитку та застосування у промисловій експлуатації.

Концептуальна схема виконання запланованого дослідження представлена на рисунку 1.4 у вигляді блок-схеми, яка визначає існуючі проблеми розробки та супроводу ПЗ систем e-Banking і негативні наслідки та зображає послідовний

зв’язок. Також на схемі наведено пропоновані підходи до вирішення цих проблем і очікувані науково-практичні результати, які надалі можна буде використовувати при проєктуванні та розробці систем e-Banking та/або певних модулів цих систем, які забезпечують відокремлений функціонал [1].

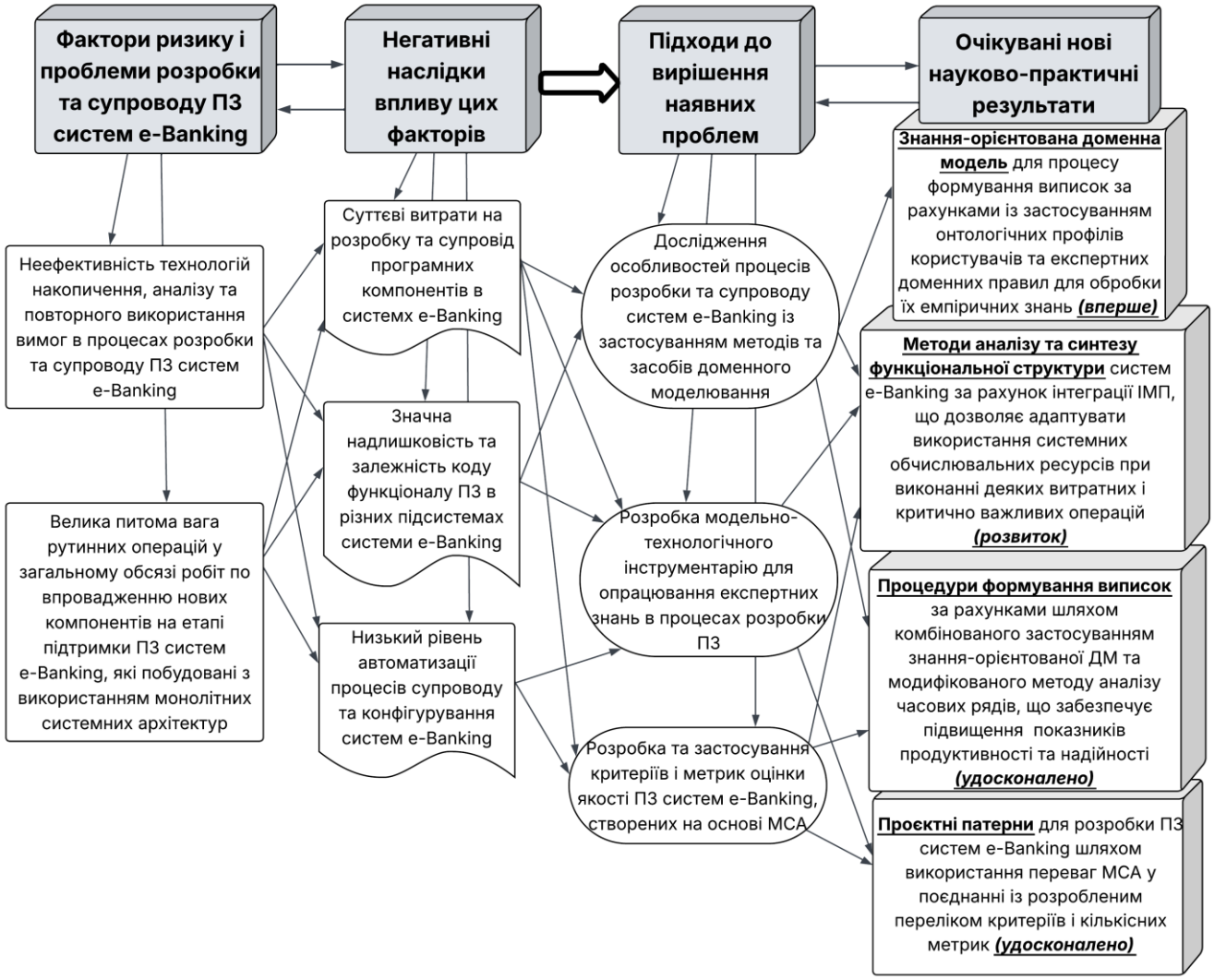

Рис. 1.4. Концептуальна схема дисертаційного дослідження

Сформульовані мета та завдання дослідження визначають необхідність розробки узагальненої концептуальної основи для побудови систем e-Banking нового покоління, яка б відповідала сучасним вимогам до розробки банківського програмного забезпечення. Узагальнені висновки щодо проведеного аналізу та постановки задач дослідження наведено у підрозділі 1.5.

1.5 Висновки до розділу 1

У даному розділі було здійснено комплексний аналіз актуальних проблем, що виникають під час розробки та супроводу програмного забезпечення для систем електронного банкінгу (e-Banking). Проведене дослідження дозволило окреслити як зовнішні чинники, пов’язані зі зростанням вимог користувачів і фінансових інституцій до масштабованості, надійності та безпеки, так і внутрішні проблеми існуючих програмних архітектур та методів моделювання предметної області.

У підрозділі 1.1 було показано, що актуальність проблематики зумовлена стрімким розвитком цифрових фінансових послуг, зростанням кількості клієнтів систем e-Banking, а також підвищеними вимогами до якості, швидкодії та захисту даних. Наявні рішення часто виявляються недостатньо ефективними через обмеження традиційних підходів, що безпосередньо впливає на конкурентоспроможність фінансових установ.

У підрозділі 1.2 було проведено аналіз існуючих в Україні систем e-Banking та встановлено, що сучасні банківські системи функціонують на базі двох ключових архітектурних парадигм — монолітної та мікросервісної. Монолітна архітектура історично була базовою для більшості банківських систем, однак вона має істотні обмеження щодо масштабованості, надійності та супроводжуваності. Натомість мікросервісна архітектура забезпечує більшу гнучкість і адаптивність, але вимагає вирішення питань узгодження сервісів, оптимізації продуктивності та підтримання цілісності даних. Це визначає потребу у системному підході до вибору архітектури в контексті майбутніх досліджень.

У підрозділі 1.3 було здійснено аналіз існуючих підходів до моделювання та оцінки якості програмного забезпечення. Було виділено DDD, як основний архітектурний паттерн, використовуваний для розробки складних систем. Показано, що традиційні методи (UML, IDEF, BPMN тощо) забезпечують формалізований опис бізнес-процесів, проте не дозволяють ефективно відобразити знання, необхідні для прогнозування поведінки користувачів і

динамічної адаптації систем. Таким чином, актуалізується завдання створення знання-орієнтованих моделей, які б інтегрували методи аналізу часових рядів та інтелектуальні алгоритми у процеси розробки та функціонування систем e-Banking.

У підрозділі 1.4 було сформульовано постановку задачі дисертаційної роботи, яка полягає у пошуку методологічних основ, що забезпечать поєднання мікросервісної архітектури та знання-орієнтованих моделей для створення перспективної системи e-Banking із розширеною функціональністю. Зокрема, увага акцентується на необхідності розробки інтелектуального модуля прогнозування (ІМП), здатного на основі історичних даних користувача передбачати його наступні дії у домені “побудова виписок за рахунками”.

Отже, результати аналізу підтверджують, що існуючі підходи до побудови систем e-Banking не повною мірою відповідають сучасним вимогам ринку та потребам користувачів. Це обґрунтовує доцільність подальших досліджень у напрямі створення методологічних засад для інтеграції знання-орієнтованих моделей та мікросервісної архітектури. Таким чином, проведений аналіз стає підґрунтям для розробки методологічних основ побудови й супроводу систем e-Banking, що є предметом розгляду у наступному розділі.

РОЗДІЛ 2
МЕТОДОЛОГІЧНІ ОСНОВИ РОЗРОБКИ ТА СУПРОВОДУ СИСТЕМ E-BANKING ПОБУДОВАНИХ ІЗ ЗАСТОСУВАННЯМ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ

2.1 Доменне моделювання як концептуальна основа для розробки перспективної версії системи e-Banking

В цілому, проєктування МСА для систем e-Banking ґрунтується на певній множині методів та підходів. Вони дозволяють виконати декомпозицію предметної області, спроєктувати якісну взаємодію між виділеними компонентами та забезпечують можливість інтеграції інтелектуальних механізмів подальшої підтримки пропонованих архітектурних рішень. Базовим принципом є декомпозиція системи на сервіси з чітко визначеними межами відповідальності (bounded context). Етап проєктування системи є одним із перших згідно типового життєвого циклу програмного забезпечення. Застосування такого підходу на етапі проєктування дозволяє мінімізувати залежності між сервісами, підвищити їх автономність функціонування та спростити майбутнє масштабування системи. Для домену e-Banking це дає змогу забезпечити відокремлення функціональності, пов’язаної з обробкою транзакцій, управлінням рахунками, веденням профілів клієнтів, формуванням виписок та іншими критично важливими процесами. Що у свою чергу забезпечує одну з важливих сучасних вимог до систем цього класу: відсутність впливу нових модулів на якість та логіку роботи вже впровадженого функціоналу [32].

Провідну роль для цього у проєктуванні МСА відіграє доменне моделювання (Domain-Driven Design – DDD), яке використовується для визначення меж сервісів та формалізації бізнес-логіки предметної області. В розрізі систем e-Banking це означає побудову моделей, що відображають ключові інформаційні сутності домену: користувачів, рахунки, транзакції, виписки, платіжні інструменти; а також принципи і порядок їх взаємодії у

системі. DDD допомагає забезпечити відповідність пропонованих архітектурних рішень заданій предметній області та зменшити ризик дублювання функціональності між різними сервісами [27]. Важливою складовою доменного моделювання є формалізація структури предметної області. Зокрема, доменна модель може бути представлена як сукупність множин сутностей, відношень та обмежень, що визначають допустимі операції у системі. Такий підхід дозволяє чітко описати бізнес-логіку та забезпечити її узгоджене використання у процесі розробки програмного забезпечення.

Однією з особливостей застосування доменного моделювання є формування єдиної узгодженої термінології, яка використовується як розробниками, так і експертами предметної області. Для цього обов’язково формується повноцінний глосарій та система бізнес-правил домену. Це дозволяє зменшити кількість неоднозначностей при інтерпретації вимог та забезпечує більш точне відображення бізнес-процесів у програмній реалізації. У випадку систем e-Banking, де складність предметної області є високою через наявність фінансових операцій, регуляторних обмежень та вимог до безпеки, використання єдиної мови опису домену є критично важливим фактором успішності розробки. Крім того, доменне моделювання дозволяє виділити агрегати (aggregates) та визначити їх межі узгодженості, що є важливим для забезпечення цілісності даних у розподілених системах. У контексті e-Banking це може проявлятися у визначенні агрегатів для рахунку клієнта, який включає інформацію про баланс, історію транзакцій та пов’язані обмеження. Такий підхід дозволяє чітко визначити правила оновлення стану системи та забезпечити узгодженість бізнес-логіки навіть у випадку асинхронної взаємодії між мікросервісами.

Застосування принципів доменного моделювання також створює передумови для інтеграції інтелектуальних компонентів у систему. Зокрема, чітке виділення бізнес-сутностей та їх взаємозв’язків дозволяє формалізувати дані, які можуть бути використані для подальшого аналізу поведінки користувачів. Таким чином, доменне моделювання виступає ключовим

елементом концептуальної основи розробки перспективних систем e-Banking, оскільки забезпечує формалізацію предметної області, обґрунтовану декомпозицію системи на мікросервіси та створює умови для інтеграції інтелектуальних модулів. Його використання дозволяє підвищити узгодженість архітектурних рішень, спростити супровід системи та забезпечити відповідність сучасним вимогам до якості програмного забезпечення.

2.2 Розробка схеми розширеної функціональності системи e-Banking із інтелектуальним модулем прогнозування дій користувача

Однією з основних функцій системи e-Banking є побудова виписок за рахунками. Ця операція є однією з найчастіше використовуваних функцій типової системи e-Banking [33]. Визначення, структура, призначення виписок та особливості їх побудови наведені у [34]. За своєю суттю побудова виписки за рахунком є операцією час виконання якої, а також необхідні ресурси, залежать від вхідних умов, таких як: кількість обраних рахунків, кількість усіх операцій за рахунком, обраний період виписки, формат виписки (наприклад, PDF, XLS, XML, тощо), необхідність сортування, тощо. Всі ці чинники призводять до двох проблем:

- неможливо однозначно оцінити час виконання операції і створити певні SLA-критерії
- потрібен великий об’єм ресурсів, а особливо спільних, таких як робота бази даних, в процесі виконання операції побудови виписки з великим обсягом платежів

Друга проблема негативно впливає на функціонування системи в цілому, оскільки призводить до того, що модулі, які виконують інші операції в системі, очікують звільнення ресурсу бази даних. Особливо це відчутно у періоди, які є найбільш типовими для побудови виписок багатьма клієнтами системи e-Banking. Яскравим прикладом є побудова річної виписки в останніх числах грудня, коли такі виписки будують і найбільші за обсягом платежів клієнти банку. При цьому, системи, які побудовані на основі МА, через використання

спільних ресурсів та середовища усіма модулями системи відчувають більший негативний ефект, ніж системи на основі МСА.

Саме тому доцільно розглянути деякі можливості знання-орієнтованого підходу до проєктування та розробки систем e-Banking, який дозволив би виконувати аналіз поведінки та попередніх дій користувача у системі при побудові будь-яких виписок та використати його для прогнозування майбутніх дій цього користувача пов’язаних з побудовою виписки. Отримані прогнози пропонується використати для можливості системи “діяти на упередження” і готувати дані для “прогнозованої виписки” заздалегідь і, що найбільш важливо, у час найменшої завантаженості системи. Наприклад, якщо користувач кожен останній день місяця, на протязі вже пів року, будує місячну виписку за своїм гривневим рахунком, то є велика ймовірність того, що й наступного місяця він також буде будувати місячну виписку за цим рахунком. Одним з використовуваних підходів до такого прогнозування є метод аналізу часових рядів, який дозволяє для описаної моделі, на основі раніше зібраних даних, виконати прогнозування [35]. Метод відноситься до категорії формалізованих математичних методів, що базується на статистиці [36]. Складові частини прогнозу наведені у [37] та схематично зображені у вигляді блок-схеми на рисунку 2.1.

Вхідними параметрами можуть бути поточний час та стан рахунків користувача. Історичними даними є усі минулі операції з побудови виписок цього конкретного користувача, а також інших користувачів системи, в тому числі і метаінформація операції: обраний період виписки, обрані рахунки, час формування виписки, формат виписки, тощо. До моделей та методів прогнозування можна віднести семантичні правила обробки та аналізу збереженої історичної інформації, методи зберігання цієї інформації, побудовану доменну модель операції, а також метод отримання “прогнозу” в конкретний момент часу.

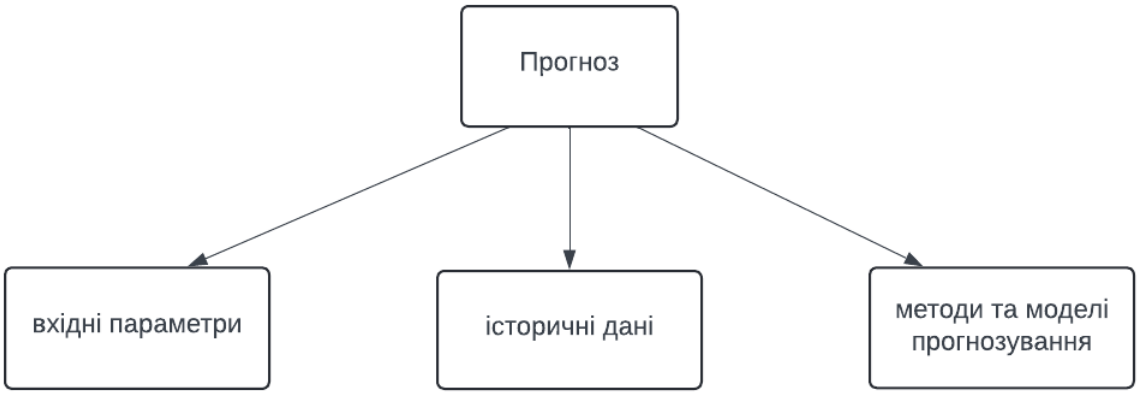

Рис. 2.1. Блок-схема складових частин прогнозу

Слід окремо зазначити, що методи аналізу часових рядів також застосовуються при розробці сучасних рекомендаційних систем (recommender system) різного призначення (див., наприклад у [38, 39, 40]), що також є одним з чинників мотивації для обрання запропонованого підходу до розширення функціональності перспективної системи e-Banking.

Рис. 2.2. Розширена функціональність системи e-Banking

З урахуванням цих міркувань пропонується розширити типову функціональність систем e-Banking новим модулем, як зазначено на рисунку

2.2. На цій схемі групу функцій “операції клієнтів” доповнено “інтелектуальним модулем прогнозування” (ІМП). В контексті піддомену виписок цей модуль відповідає за обробку історичної інформації, прогнозування дій користувача в системі при побудові виписок за рахунками, а також повинен створювати у розрахований ним час системні задачі по побудові “прогнозованої виписки”, які вже далі обробляються модулем побудови виписок з урахування прогнозованих параметрів цієї виписки.

Використання ж МСА дозволить отримати тримати кращі показники якості для запропонованого підходу. Так як МСА передбачає розділення системи на окремі незалежні сервіси, то відповідно є можливість відокремлення нового модулю в окремий мікросервіс, який не матиме впливу на стабільність роботи інших клієнтських операцій та системи в цілому [41]. Також впровадження мікросервісної архітектури забезпечить набагато більш гнучке масштабування системи з мінімальними ресурсами при пікових навантаженнях, що забезпечує підвищення продуктивності системи e-Banking в цілому, оскільки максимально знижує вплив операції формування виписок на інші функції системи, особливо в порівнянні з монолітною архітектурою, в якій існують спільні ресурси [42, 43]. В контексті цієї роботи це в першу чергу стосується мікросервісу, який відповідатиме в системі за доменну область “формування виписок за рахунками”, оскільки саме на нього покладатиметься корисне навантаження по побудові прогнозованих виписок.

Для виявлення існуючих проблем в побудові та роботі систем e-Banking, у період лютий-квітень 2024 року було проведено опитування співробітників відділів супроводу та розробки української компанії CS [44], яка є розробником системи iFOBS [10, 11]. Було виявлено, що операція побудови виписки за рахунком є найбільш довгою операцією в системі, а також, через необхідність оперування великим об’ємом даних при виконанні, створює найбільш негативний вплив на швидкодію інших операцій користувачів в системі. Одним з прикладів такого впливу було зниження у 4 рази продуктивності операцій з проведення гривневих платежів через заповнення кешованої області пам’яті у

СУБД Oracle даними, які отримувалися сервером для формування річної виписки одного з найбільших клієнтів банку, який є корпорацією національного масштабу та має досить велику кількість рахунків і проведених за цими рахунками платежів. Зниження продуктивності спостерігалося в проміжок часу, впродовж якого формувалася вищезгадана виписка.

Наразі в системі iFOBS побудова виписки за рахунком виконується синхронно та в момент запиту користувача. Побудова річної виписки займає від 1-2 секунд для користувачів із категорії малого/середнього бізнесу, до декількох десятків секунд для корпорацій національного масштабу. У табл. 2.1 наведена статистика формування виписок за рахунками у форматі pdf, який наразі є найбільш популярним серед користувачів, в залежності від кількості сторінок сформованої виписки. Процедура формування виписки складається з двох основних частин:

- підготовка даних про платежі на основі вхідних параметрів запитуваної виписки
- формування PDF-файлу за допомогою бібліотеки JasperReports [45]

Таблиця 2.1. Статистика формування виписок за рахунками

Кількість сторінок	Підготовка даних (мс)	Формування PDF-файлу(мс)	Усього (мс)
1	40	120	160
5	60	220	280
10	70	350	420
20	80	600	680
50	200	1500	1700
100	350	3100	3450
150	800	4200	5000
360	2500	12000	14500

500	4500	15000	19500
850	5500	40000	45500

Отримана статистика виконання операцій формування виписки користувачами системи показує, що є певні часові закономірності таких операцій та їх параметрів. Все це дозволяє зробити висновок, що впровадження (ІМП) для операцій побудови виписок за рахунками є цілком доцільним і має допомогти в досягненні основної мети роботи, а саме підвищенні продуктивності роботи системи e-Banking. Пропоноване рішення в першу чергу направлене на банки всеукраїнського масштабу, які серед своїх клієнтів мають як корпорації національного масштабу, для яких потрібно забезпечити можливість обробки великих об’ємів платіжних даних, так і велику кількість користувачів з категорії фізичних осіб та/або малого-середнього бізнесу, які через свою велику кількість постійно створюють навантаження на систему і потребують певний рівень якості наданих банком дистанційних послуг впродовж усього часу роботи системи.

В розділі 2.3 буде розглянуто питання можливих підходів до розробки ІМП, в тому числі і вибір знання-орієнтованих моделей та методів, які відповідатимуть бізнес-вимогам, що висуваються до модуля, та будуть використані при побудові доменної моделі і розробці алгоритму бізнес-логіки ІМП.

2.3 Огляд можливих підходів для розробки інтелектуального модуля прогнозування та вибір методу аналізу часових рядів для його реалізації

У попередньому розділі була запропонована перспективна функціональна схема для типової системи e-Banking, яка використовує переваги знання-орієнтованого підходу до проєктування та розробки систем e-Banking. Ця схема пропонує використовувати ІМП для аналізу поведінки та попередніх дій користувача у системі при побудові виписок за рахунками, операції яка є

однією з найбільш використовуваних та ресурсоємних. Використання ІМП дає можливість системі “діяти на упередження” та готувати дані для “прогнозованої виписки” заздалегідь і, що найбільш важливо, у час найменшої завантаженості системи. Також було запропоновано для прогнозування використовувати методи аналізу часових рядів у поєднанні з семантичними правилами. В цьому розділі буде розглянуто існуючі методи аналізу часових рядів та мотивовано обрано найбільш підходящий метод, який би враховував усі особливості доменної області e-Banking Systems.

При розробці ІМП та виборі найбільш підходящого методу для аналізу часових рядів, необхідно враховувати такі вимоги:

- при прогнозі ІМП має враховувати як накопичену інформацію про дії користувача при побудові виписок за рахунками, так і можливі зміни у вимогах до процесу формування виписок. Прикладом таких змін є будь-які нові вимоги законодавства, наприклад: “фізичні особи-підприємці (ФОП) мають надавати кожного року надавати до пенсійного фонду річну виписку певної форми”
- можливість банку впливати на результат прогнозу за допомогою “додавання нових експертних знань”, які можуть бути отримані, наприклад, з результатів сторонніх аналітичних досліджень, які проведені банком
- розрахунок прогнозу не повинен бути складною операцією ні математично, ні зі точки зору обчислювальних ресурсів

Вимоги 1 та 3 стосуються в першу чергу методів аналізу часових рядів, а вимога 2 має бути реалізована за рахунок додаткового використання експертних доменних правил, вимоги до яких буде розглянуто в наступних розділах. Виходячи з наведених вимог далі буде виконано огляд та вибір для майбутнього застосування конкретного методу аналізу часових рядів [46, 47].

Для початку наведемо узагальнене визначення методів аналізу часових рядів: це набір статистичних та математичних методів, спрямованих на дослідження даних, зібраних у вигляді послідовності спостережень у часі. Ці

методи дозволяють виявляти закономірності, тренди, сезонність, і навіть прогнозувати майбутню поведінку часового ряду з урахуванням його минулих значень. Основні завдання аналізу часових рядів включають ідентифікацію моделей, прогнозування, згладжування даних та виявлення аномалій [48].

Усі методи аналізу часових рядів можна поділити на такі категорії [49]:

- традиційні методи прогнозування
  - з лінійними моделями
  - з нелінійними моделями
- методи прогнозування з використання моделей машинного навчання

До лінійних моделей відносяться: авторегресивні модель (AR), модель ковзного середнього (MA), авторегресивна модель ковзного середнього (ARMA) та авторегресивна інтегрована модель ковзного середнього (ARIMA). Окремо виділяють метод SARIMA – це розширення моделі ARIMA, яке враховує сезонні тренди при побудові прогнозів[50]. Також до лінійних моделей відносяться усі 3 варіації експонентного згладжування(ES): SES(single ES), DES (double ES) та TES(triple ES).

Нелінійні моделі включають в себе авторегресивні моделі умовної гетероскадичності (ARCH) та узагальнені авторегресійні моделі умовної гетероскадичності (GARCH).

Методи прогнозування, які використовують машинне навчання, стали активно використовуватися з початком розвитку нейронних мереж, які стали основою для більшості методів цього класу, характерними представниками якого є штучні нейронні мережі (ANN), рекурентні нейронні мережі (RNN) та мережі довгострокової короткочасної пам’яті (LSTM).

Нижче, у табл. 2.2, наведено коротку характеристику основних методів, особливості їх застосування, переваги та недоліки [49-52].

Таблиця 2.2. Порівняння основних методів аналізу часових рядів

Назва методу	Характеристики/переваги/недоліки
ARIMA	- підходить для нестаціонарних даних із тенденціями

	та/або сезонністю. - моделі ARIMA обробляють як короткострокові, так і довгострокові залежності. - добре підходить для невеликих об’ємів даних - низькі обчислювальні витрати
SARIMA	- є розширенням методу ARIMA, яке дозволяє врахувати сезонність про прогнозуванні
TES	- включає в себе сезонність на додачу до рівня та тенденції - використовує три коефіцієнти згладжування - підходить для даних з тенденціями та сезонністю
GARCH	- спеціалізується на моделюванні кластеризації волатильності - фіксує зміну волатильності на проміжку часу - включає минулу дисперсію для прогнозування майбутньої волатильності - зазвичай використовується на фінансових ринках - добре підходить для великих об’ємів даних
ANN	- здатність фіксувати нелінійні залежності - стійкість до зашумлених даних - здатність самонавчатися - вимагає значних обчислювальних ресурсів
RNN	- здатність добре оброблювати дані з низькою залежністю - існує проблема вибуху градієнта та зникнення градієнта
LSTM	- істотно вдосконалена версія RNN - здатність добре оброблювати дані з тривалою залежністю - здатність будувати глибокі нейронні мережі - вимагає значних обчислювальних ресурсів - повільний розрахунок прогнозу

Обираючи найбільш підходящий метод, необхідно виходити з визначених раніше вимог до ІМП та того факту, що у системах e-Banking виписки за рахунками зазвичай не будуються користувачем при кожному вході у систему і навіть не кожен день, але мають ознаки тенденційності та сезонності. По-перше, є цілий сегмент користувачів, до яких відносяться юридичні особи та ФОПи, які мають чітко визначений графік необхідності побудови виписок, в першу чергу для ведення бухгалтерського обліку та сплати податків. По-друге,

інший сегмент користувачів, фізичні особи, зазвичай мають тенденцію при побудові виписок, наприклад, виписка будується перед отриманням заробітної платні чи після сумарної витрати певної суми за проміжок часу для аналізу ретроспективи власних витрат.

Серед наведених у роботі методів аналізу часових рядів, метод SARIMA є таким, що відповідає усім вимогам, оскільки він враховує сезонність та поточні тренди, обробляє як довгострокові так і короткострокові залежності у вибірці даних, краще підходить для невеликих об'ємів даних, не вимагає значних обчислювальних ресурсів для формування прогнозу. Як було вже зазначено у табл. 2.2, метод SARIMA є розширенням методу ARIMA, але який дозволяє враховувати сезонність. Модель SARIMA можна описати за допомогою формули [46, 47, 50]:

$$SARIMA = ARIMA(p, d, q)\ (P, D, Q)_S,$$

Модель, створювану за допомогою методу SARIMA, яка дозволяє зробити прогноз наступного значення ряду, можна описати за допомогою формул [46, 47, 52, 53]:

$$\phi_p(B)\ \Phi_P(B^s)\ (1 - B)^d\ (1 - B^s)^D y_t = \Theta_Q(B^s)\ \theta_q(B)$$

$$\phi_p(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p$$

$$\Phi_P(B^s) = 1 - \Phi_1 B^s - \Phi_2 B^{2s} - \ldots - \Phi_P B^{Ps}$$

$$\theta_q(B) = 1 + \theta_1 B + \theta_2 B^2 + \ldots + \theta_q B^q$$

$$\Theta_Q(B^s) = 1 + \Theta_1 B^s + \Theta_2 B^{2s} + \ldots + \Theta_Q B^{Qs}$$

де y_t – значення часового ряду у момент часу t, змінні p, d та q – це несезонні коефіцієнти моделі ARIMA, параметри P, D та Q – відповідні сезонні коефіцієнти конкретного періоду, а s – довжина періоду у сезоні. У свою чергу коефіцієнти для моделі ARIMA визначаються таким чином:

– коефіцієнт p, який визначає порядок авторегресії (кількість минулих значень, які використовуються для прогнозування поточного), визначається за допомогою функції часткової автокореляції (PACF). Має бути цілим невід'ємним числом;

- коефіцієнт q, який визначає порядок ковзної середньої (кількість минулих помилок прогнозу, які враховуються в моделі), визначається за допомогою функції автокореляції (ACF). Має бути цілим невід'ємним числом;
- коефіцієнт d, який визначає ступінь диференціювання, визначається за допомогою тесту Дікі-Фуллера, який дозволяє визначити стаціонарність ряду, зазвичай дорівнює 1. Значення 1 означає, що часовий ряд має тренд [54, 55].

Оскільки виписки будуються в першу чергу для фінансової звітності, а у бухгалтерській діяльності, як і в повсякденному житті, найбільш очевидним періодом, який має ознаки повторюваності, є календарний рік, то коефіцієнт s доцільно обирати рівним кількості виписок побудованих користувачем за останній рік. З аналогічних міркувань для коефіцієнта p доцільно також встановити значення рівне кількості виписок побудованих користувачем за останній рік, для того аби усі побудовані виписки за сезон були враховані. Значення коефіцієнта q рекомендується вказувати у діапазоні від 0 до 3, і корегувати в залежності від точності попередніх прогнозів: при збільшенні точності прогнозів – зменшувати значення q [56].

Варто зазначити, що при проєктуванні модуля ІМП варто дати функціональну можливість системним адміністраторам банку задавати значення за замовчуванням для вище наведених параметрів на рівні конфігурації системи. Це дозволить без оновлення системи впливати на створювану модель прогнозу і покращувати якість прогнозування, оскільки адміністратори банку можуть додатково аналізувати графіки функцій ACF та PACF для знаходження пікових точок функцій автокореляції. Також для визначення коефіцієнтів може бути застосовані критерії AIC та BIC[52].

2.4 Аналіз альтернативних варіантів побудови програмної архітектури розширеної системи e-Banking та мотивація доцільності використання мікросервісної архітектури

Сучасні системи e-Banking характеризуються високим рівнем складності, що обумовлено постійним зростанням кількості користувачів, обсягів транзакцій та функціональних можливостей. У таких умовах особливої актуальності набуває питання вибору ефективної програмної архітектури, яка здатна забезпечити виконання як функціональних, так і нефункціональних вимог, зокрема масштабованості, продуктивності, надійності та доступності системи. Традиційним підходом до побудови інформаційних систем, у тому числі й систем e-Banking, є використання монолітної архітектури. Такий підхід передбачає реалізацію всієї бізнес-логіки у вигляді єдиного програмного комплексу з централізованим управлінням даними та спільним використанням ресурсів. Монолітні системи можуть бути ефективними на початкових етапах розробки, однак зі зростанням складності вони стикаються з низкою суттєвих обмежень. Зокрема, внесення змін у будь-яку частину системи потребує повторного розгортання всього додатку, що ускладнює супровід і знижує гнучкість. Крім того, масштабування монолітної системи здійснюється шляхом дублювання всього застосунку, що призводить до неефективного використання ресурсів. У контексті систем e-Banking це особливо критично, оскільки різні підсистеми мають різний характер навантаження [57].

Альтернативою монолітному підходу є сервіс-орієнтована архітектура (SOA), яка передбачає розподіл системи на окремі сервіси, що взаємодіють між собою через стандартизовані інтерфейси. Такий підхід дозволяє підвищити повторне використання компонентів та забезпечити певний рівень гнучкості. Водночас SOA часто передбачає використання централізованих механізмів інтеграції, що може призводити до виникнення вузьких місць та зниження продуктивності при високих навантаженнях. Крім того, сервіси в SOA зазвичай мають більший розмір і нижчий рівень ізоляції порівняно з мікросервісами, що ускладнює їх незалежний розвиток. У відповідь на зазначені обмеження

сучасною тенденцією є використання МСА, яка передбачає декомпозицію системи на незалежні сервіси з власними життєвими циклами. Кожен мікросервіс реалізує окрему бізнес-функцію та може розроблятися, розгортатися і масштабуватися незалежно від інших компонентів системи. Такий підхід дозволяє забезпечити високу гнучкість, спростити внесення змін та оптимізувати використання ресурсів [32]. Разом з тим, проєктування МСА для систем e-Banking пов’язане з рядом специфічних труднощів. Банківські системи повинні обробляти значні обсяги транзакцій, забезпечуючи при цьому їх коректність і цілісність. Це створює складні умови для реалізації розподіленої архітектури. Зокрема, застосування принципу незалежного зберігання даних для кожного сервісу (database per service) ускладнює забезпечення узгодженості даних, особливо у випадках, коли одна бізнес-операція охоплює кілька сервісів. Ще одним важливим аспектом є організація взаємодії між сервісами. У розподілених системах можливе використання як синхронних, так і асинхронних механізмів комунікації. Синхронні виклики дозволяють отримувати результат у режимі реального часу, однак при високих навантаженнях вони можуть призводити до збільшення затримок і зниження продуктивності. Асинхронні механізми, у свою чергу, дозволяють зменшити навантаження на систему за рахунок розподілу обробки у часі, проте ускладнюють контроль за виконанням операцій і вимагають додаткових засобів координації транзакцій [32, 58]. Таким чином, ефективне застосування МСА потребує врахування як технічних, так і доменних особливостей системи. Одним із ключових принципів проєктування МСА є декомпозиція системи на сервіси з чітко визначеними межами відповідальності (bounded context), що дозволяє мінімізувати залежності між компонентами та підвищити їх автономність. Іншим важливим аспектом є поєднання різних способів взаємодії між сервісами залежно від функціональних вимог системи, що дозволяє досягти балансу між швидкодією та стійкістю до навантажень.

У процесі проєктування МСА широко використовуються типові архітектурні патерни, які описують узагальнені підходи до вирішення

поширених задач та забезпечують повторювані рішення для типових задач інтеграції та управління сервісами. Нижче наведені сучасні архітектурні патерни, які доцільно застосовувати для вирішення поставлених задач цього дослідження.

API Gateway. Забезпечує централізовану маршрутизацію усіх запитів до системи та всередині неї, що також дозволяє легко та якісно реалізувати різноманітні контролі доступу для забезпечення відповідного рівня для показника захищеності. Функціонуючи як єдина точка входу в систему, якою є множина мікросервісів, цей патерн інкапсулює внутрішню топологію мікросервісів від зовнішніх клієнтів, виступаючи захисним проксі-шаром. Це дозволяє винести спільну функціональність таку як розшифрування SSL-трафіку, автентифікація через за допомогою токенів, логування та моніторинг на рівень інфраструктурного шлюзу. Це дозволяє не дублювати її в кожному мікросервісі. Завдяки механізмам агрегації запитів, API Gateway здатен зменшити кількість мережевих викликів з боку клієнта, збираючи дані з декількох джерел в одну відповідь, що суттєво знижує мережеві затримки. Крім того, така реалізація забезпечує стабільність системи шляхом впровадження політик Rate Limiting [59, 60]. Основний принцип зображений на рисунку 2.3.

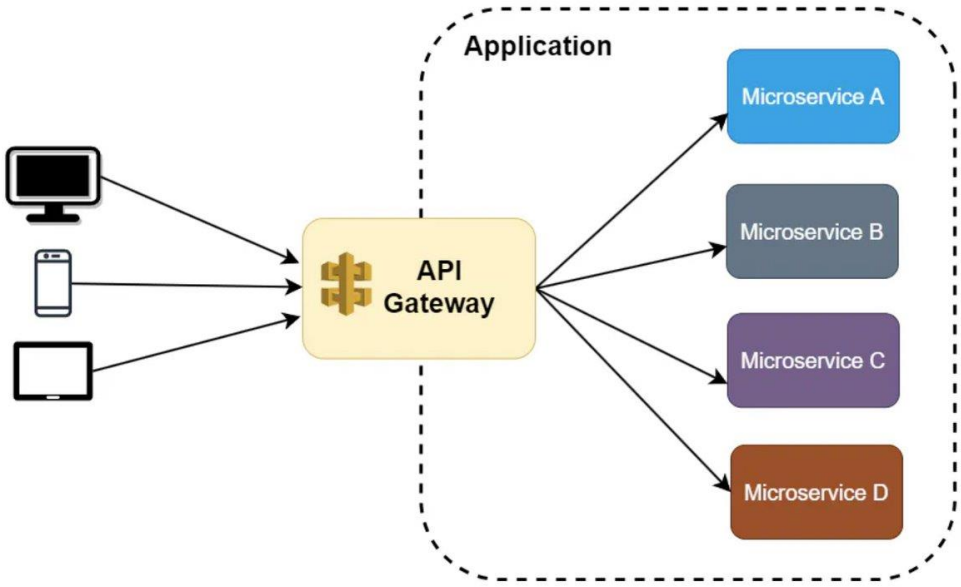

Рис. 2.3. Принцип роботи патерну API Gateway

Database per Service. Передбачає виділення окремого сховища даних для кожного мікросервіса, що забезпечує автономність зберігання даних. Такий підхід гарантує повну ізоляцію сервісів на рівні домену, унеможливлюючи виникнення жорстких зв’язків через спільні таблиці чи схеми бази даних. Це дозволяє реалізувати принцип інкапсуляції, де доступ до даних здійснюється виключно через публічний API відповідного сервісу, що спрощує незалежне розгортання та модифікацію структур даних. Окрім того, розділення баз даних мінімізує негативний вплив на систему у разі збоїв у роботі сховища, підвищуючи загальну відмовостійкість використовуваної архітектури. Проте реалізація патерну вимагає особливої уваги до забезпечення цілісності даних у розподілених транзакціях та складної агрегації інформації з декількох джерел [59]. Основний принцип зображений на рисунку 2.4.

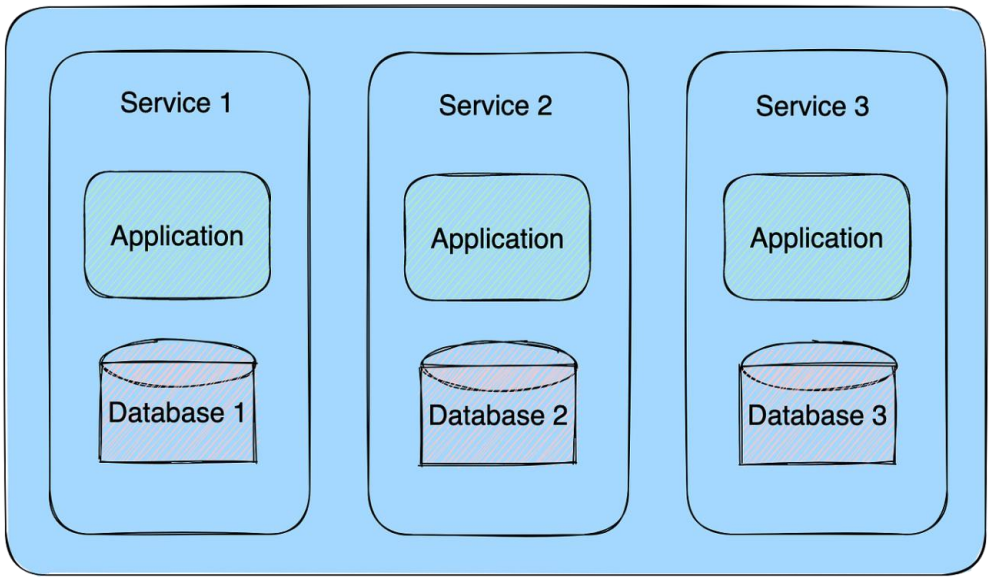

Рис. 2.4. Принцип роботи патерну Database per Service

Saga Pattern. Відповідає за управління розподіленими транзакціями у МСА і допомагає вирішити питання відкату транзакцій при різних збоях, що є однією о основних складностей МСА у порівнянні з іншими еталонними системними архітектурами. Оскільки в розподілених системах класичний two-phase commit часто є неефективним через високу затримку та ризики блокування ресурсів,

Saga пропонує модель, яка забезпечує цілісність шляхом виконання певної послідовності локальних транзакцій. Кожен крок такої послідовності оновлює дані в межах одного конкретного сервісу та формує подію чи повідомлення, яке ініціює наступний крок у ланцюжку компенсуючих транзакцій. У випадку виникнення помилки на будь-якому етапі, патерн передбачає виконання компенсуючих транзакцій, які в зворотному порядку нівелюють результати попередніх успішних операцій, забезпечуючи кінцеву узгодженість виконуваних операцій в системі. Реалізація даного патерну можлива двома способами: хореографія, де кожен сервіс самостійно реагує на події породжені іншими сервісами, або оркестрація, де вже централізований вузол керує логікою виконання компенсуючих транзакцій [61]. Основний принцип зображений на рисунку 2.5.

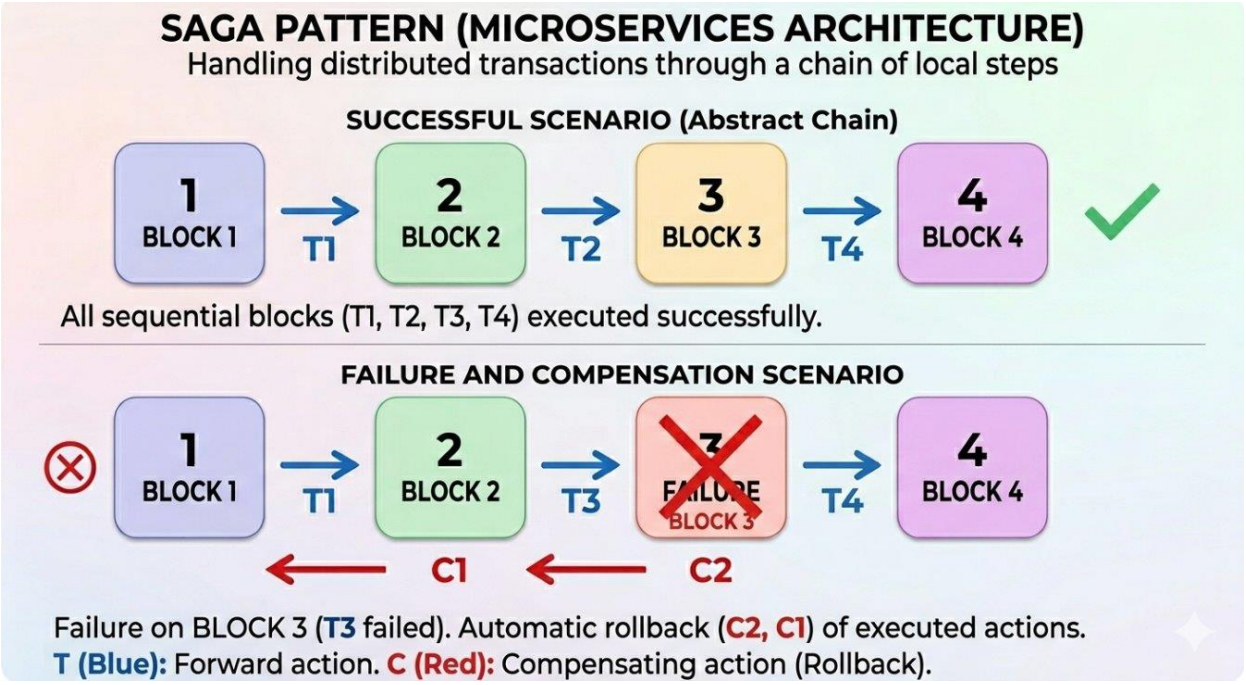

Рис. 2.5. Принцип роботи патерну SAGA

Circuit Breaker. Допомагає вирішити питання раннього сповіщення про недоступність та/або непрацездатність інших сервісів системи, що допомагає підвищити надійність системи та реалізувати механізми ізоляції збоїв. Патерн функціонує на основі трьох станів: Closed, Open та Half-Open. Якщо кількість

помилкових запитів до певного сервісу перевищує встановлений поріг, система припиняє подальші виклики до нього, миттєво повертаючи помилку. Це запобігає каскадним збоям та вичерпанню ресурсів потоків у сервісах, які ініціюють запити, через тривалі очікування таймаутів на отримання відповіді. Після певного періоду очікування механізм автоматично перевіряє доступність цільового сервісу, пропускаючи обмежену кількість тестових запитів для прийняття рішення про повне відновлення трафіку [59]. Основний принцип зображений на рисунку 2.6.

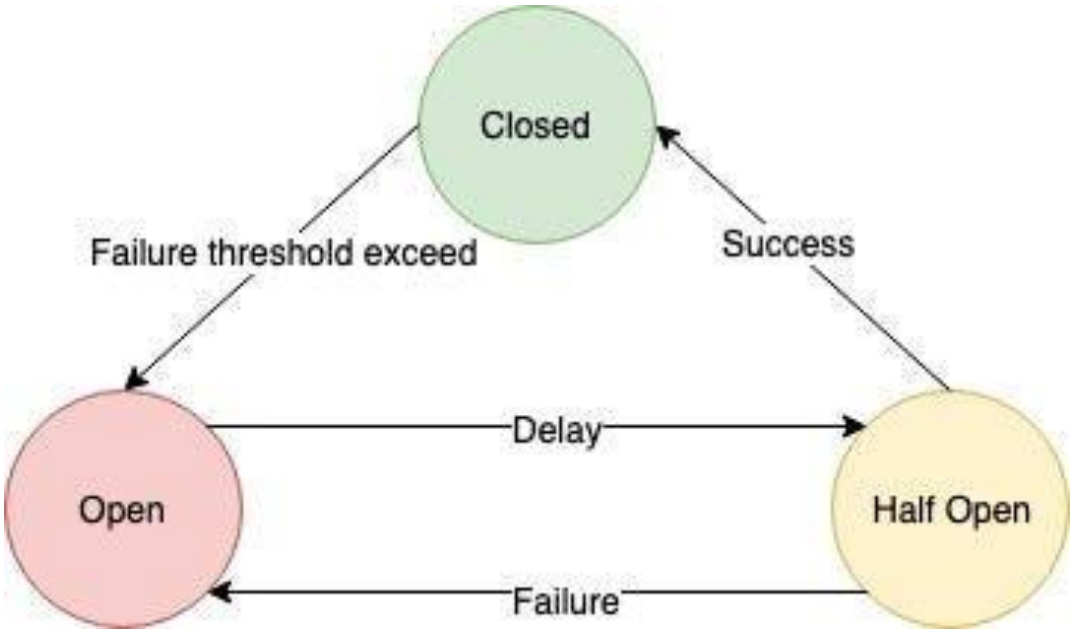

Рис. 2.6. Принцип роботи патерну Circuit Breaker

Застосування таких патернів дозволяє підвищити гнучкість і надійність розподілених систем, однак їх ефективність значною мірою залежить від конкретного контексту використання. Важливою складовою сучасних архітектурних підходів є використання асинхронної обробки даних та подієво-орієнтованих моделей взаємодії. Передача інформації у вигляді подій дозволяє зменшити зв’язність між компонентами системи та забезпечити більш ефективне використання ресурсів. Такий підхід є особливо актуальним для задач, які не потребують негайного виконання і можуть бути відкладені у часі [62]. Поряд із цим, важливим фактором є забезпечення масштабованості

системи. МСА дозволяє здійснювати незалежне масштабування окремих компонентів відповідно до їх навантаження, що сприяє більш ефективному використанню обчислювальних ресурсів. Це є суттєвою перевагою порівняно з монолітними системами, де масштабування здійснюється для всієї системи в цілому [63].

Таким чином, проведений аналіз альтернативних підходів до побудови програмної архітектури систем e-Banking дозволяє зробити висновок про доцільність використання МСА як основи для побудови розширеної системи. МСА забезпечує необхідний рівень гнучкості, масштабованості та адаптивності до змін, що є критично важливими для сучасних систем e-Banking. Водночас її ефективне застосування потребує врахування специфіки предметної області та використання відповідних методологічних підходів, що і розглядаються у подальших розділах дисертаційної роботи.

2.5 Висновки до розділу 2

У даному розділі було проаналізовано та наведено методологічні основи для розробки та супроводу систем e-Banking побудованих із застосуванням МСА.

У підрозділі 2.1 було показано, що доменне моделювання є невід'ємною складовою при побудові складних проблемно-орієнтованих систем, а для систем класу e-Banking взагалі є концептуальною основою для їх проєктування та розробки. З використанням доменного моделювання можна легко декомпозувати складну з точки зору набору функціональності систему на множину відносно незалежних доменів, розробка та супровід яких далі може розглядатися як незалежні один від одного процеси. Крім того, побудова та підтримка актуальності доменної моделі системи дозволяє підвищувати якість розробки та супроводу системи в цілому. Чітке розділення доменів дозволяє краще ізолювати розробку в межах конкретного домену та значно зменшити вплив змін у функціональності одного домену на стабільність функціонування

інших доменів, що є однією з ключових вимог, які висуваються до систем класу e-Banking.

У підрозділі 2.2 було запропоновано розширену схему функціональності для системи e-Banking. Було запропоновано використання ІМП для можливості прогнозування дій користувача у системі. Прогнозування пропонується застосувати до операції побудови виписок за рахунками користувача, яка є однією з найчастіше використовуваних функцій типової системи e-Banking. Побудова виписок є такою, що потребує миттєвого виділення значних ресурсів для свого виконання і, відповідно, є такою, що має значний вплив на показники надійності системи та її продуктивності. Особливо це стосується побудови великих виписок. З іншого боку, операція побудови виписок часто виконується у певні періоди та з передбачуваними вхідними параметрами. Вище перераховане дозволяє мотивовано запропонувати виконувати прогнозування дій користувача пов’язаних з побудовою виписки та дати можливість системі “діяти на упередження” і готувати виписку заздалегідь у періоди простою системи, тобто коли виділені системі ресурси простоюють.

У підрозділі 2.3 було проведено огляд можливих підходів до побудови запропонованого модуля ІМП. Було визначено, що оскільки побудова виписок є певною подією на часовому проміжку і історія попередніх операцій з побудови виписок користувачем є основою для визначення закономірності і прийняття рішення щодо прогнозу, то найкращим знання-орієнтованим методом для такого прогнозування є методи аналізу часових рядів. Для вибору конкретного методу, який має бути використаний у ІМП було проведено порівняльний аналіз характеристик основних відомих методів цього класу. При аналізі враховувались: математична складність методу, необхідні обчислювальні ресурси, швидкодія, точність прогнозу. З урахуванням вимог, які висуваються та характеристик порівнюваних методів, було обрано метод SARIMA. Цей метод враховує сезонність, є математично нескладним, не потребує значних обчислювальних ресурсів та має високу швидкодію, що є доволі важливим для якісної роботи ІМП.

У підрозділі 2.4 було розглянуто варіанти побудови програмної архітектури для системи e-Banking, у яку додатково інтегровано модуль ІМП. Серед використовуваних архітектур для цієї задачі було мотивовано обрано використання МСА, яка дозволяє забезпечити кращі показники якості у порівнянні з іншими еталонними системними архітектурами (ЕСА). Також, особливості МСА дозволяють якісніше реалізувати основні нефункціональні вимоги, які висуваються до систем класу e-Banking. Додатково було розглянуто основні архітектурні патерни, які мають забезпечити основні концепції МСА.

Отже, отримані результати підтверджують можливість використання знання-орієнтованих методів та інтелектуальних підходів до розробки та супроводу систем класу e-Banking, а застосування МСА в якості ЕСА дозволяє задовольняти сучасні нефункціональні вимоги до ПЗ систем e-Banking. Це є підґрунтям до розробки конкретних доменних моделей, алгоритмів та процедур бізнес-логіки, а також проєктування архітектури перспективної системи e-Banking, функціональність якої розширена модулем ІМП.

РОЗДІЛ 3
РОЗРОБКА МОДЕЛЕЙ, ПРОЦЕДУР БІЗНЕС-ЛОГІКИ ТА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ФУНКЦІОНУВАННЯ СИСТЕМ E-BANKING

3.1 Побудова доменної моделі для опрацювання експертних знань у процедурі функціонування системи e-Banking на прикладі задачі формування виписок за рахунками клієнтів

Однією з ключових функціональних складових систем e-Banking є підсистема формування виписок за рахунками клієнтів, яка забезпечує користувачам доступ до інформації про виконані фінансові операції за обраний період. З огляду на важливість цієї функції для кінцевого користувача, а також її значний вплив на навантаження системи, задача оптимізації процесу побудови виписок є актуальною як з практичної, так і з наукової точки зору.

У попередніх розділах було визначено, що перспективним напрямком розвитку систем e-Banking є використання знання-орієнтованих підходів у поєднанні з мікросервісною архітектурою, зокрема із застосуванням інтелектуального модуля прогнозування (ІМП), який дозволяє аналізувати поведінку користувачів та виконувати підготовку даних для формування так званих “прогнозованих виписок” заздалегідь. Реалізація такого підходу потребує формалізації предметної області, визначення її основних сутностей, зв’язків та правил взаємодії, що обумовлює необхідність побудови відповідної доменної моделі.

У межах даного дослідження розглядається домен “Побудова виписок за рахунками” (ПВР), який є складовою бізнес-логіки систем e-Banking. Цей домен охоплює процеси формування, обробки та надання користувачу виписок, включаючи отримання даних про транзакції, їх агрегацію, обробку та представлення у заданому форматі. Незважаючи на різноманітність реалізацій у різних банківських системах, з точки зору бізнес-логіки процес побудови виписки має типовий характер і включає однакові або подібні етапи виконання.

Для формалізації домену ПВР першочергово необхідно визначити основні інформаційні сутності, які беруть участь у відповідних бізнес-процесах. До таких сутностей, зокрема, належать: користувач (User), рахунок (Account), транзакція (Transaction), виписка за рахунком (Statement). Бізнес-правила взаємодії між цими сутностями визначають логіку формування виписки та можуть бути представлені на рисунку 3.1 у вигляді ER-діаграми, яка відображає структуру домену та основні залежності між його елементами [64].

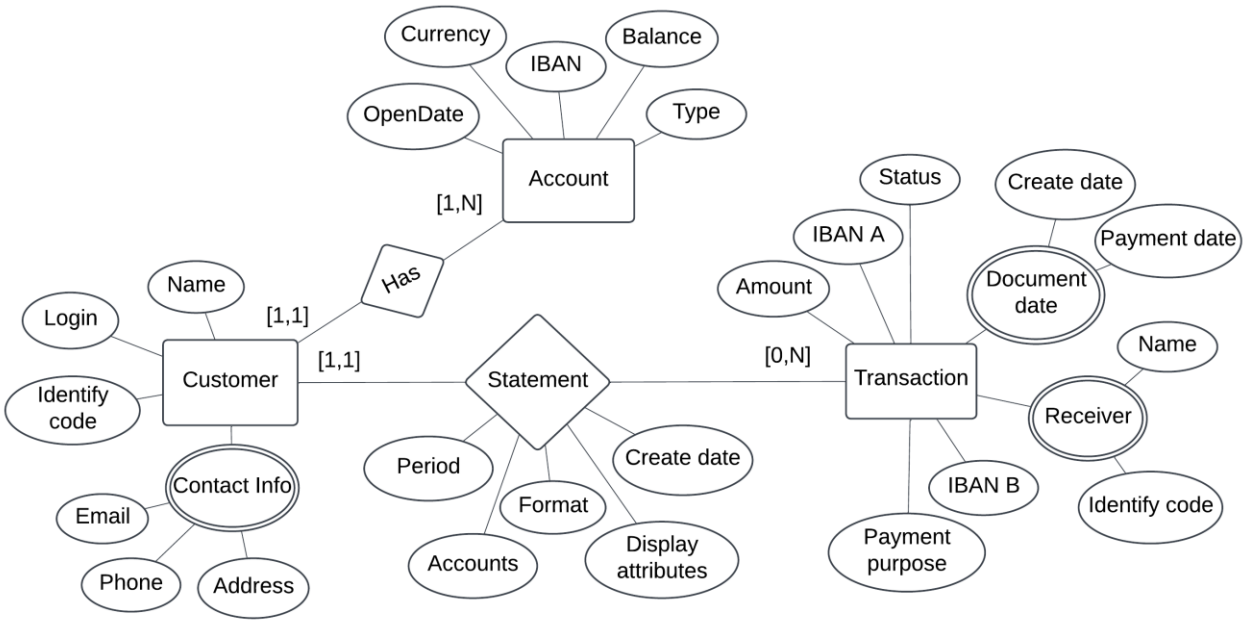

Рис. 3.1. ER-діаграма інформаційних сутностей у домені "виписка за рахунками"

Узагальнюючи підхід до формалізації предметної області, доменну модель систем класу e-Banking доцільно представити формулою 3.1 у вигляді кортежу:

$$DM = <C, R, A>, \qquad (3.1)$$

де C — множина сутностей домену, R — множина відношень між сутностями, A — множина правил та обмежень, що регламентують допустимі операції та взаємодії між елементами системи. Така формалізація дозволяє не лише описати структуру предметної області, але й визначити правила її

функціонування, що є важливим для подальшої розробки бізнес-логіки та інтелектуальних компонентів системи [46].

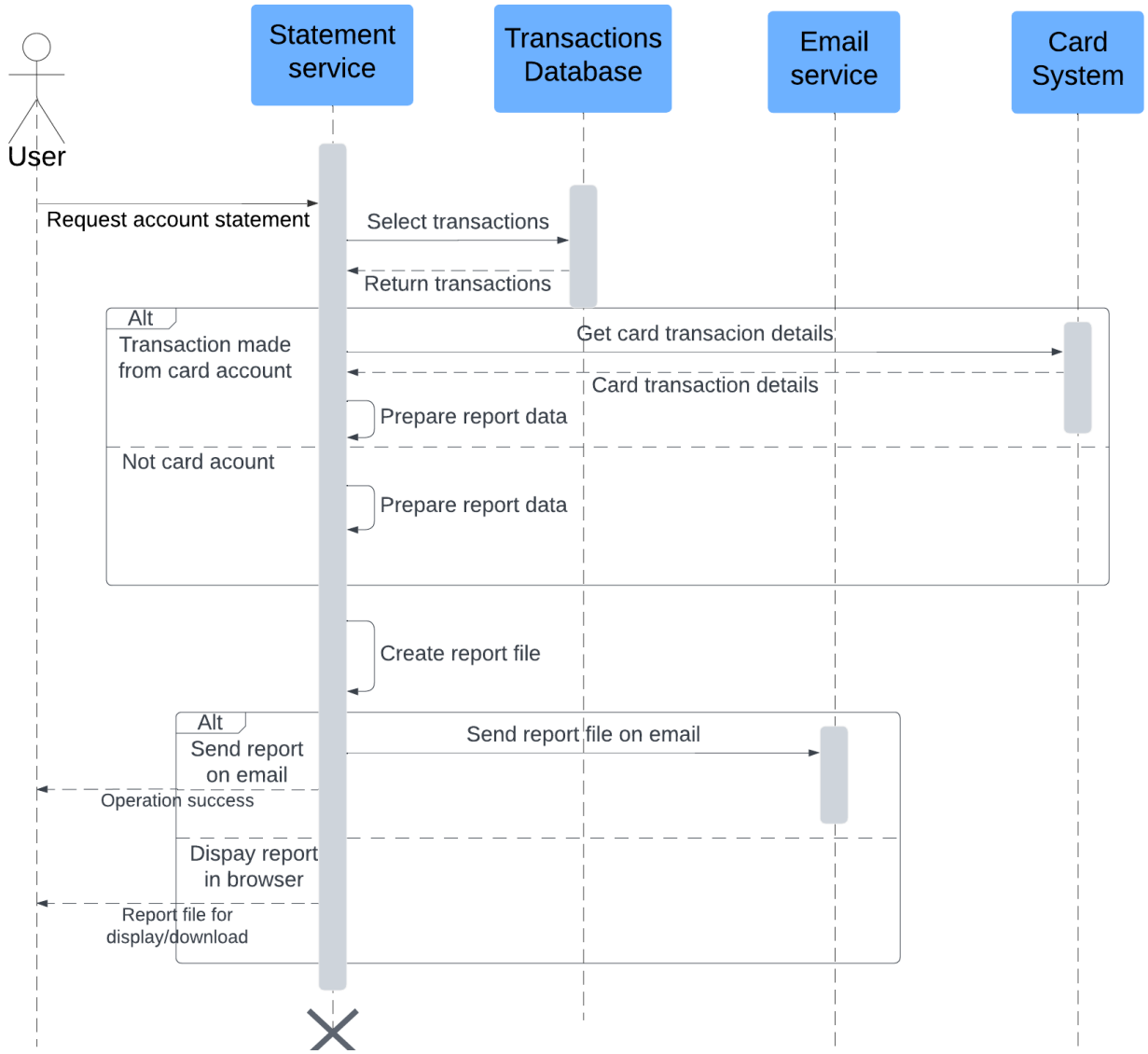

Рис. 3.2. Типовий алгоритм формування виписки за рахунком у вигляді UML-діаграми послідовності

На основі визначених сутностей та їх взаємозв'язків можливо перейти до формалізації поточного алгоритму побудови виписки за рахунком. У більшості сучасних систем e-Banking цей алгоритм реалізується як послідовність взаємопов'язаних операцій, що виконуються у відповідь на запит користувача. Зокрема, типовий процес включає отримання параметрів запиту (період, формат, спосіб отримання), вибірку відповідних транзакцій з бази даних,

обробку та агрегацію отриманих даних, формування вихідного документу у заданому форматі (наприклад, PDF або XLSX), а також передачу результату користувачу або його відправлення на електронну пошту. Суттєвою особливістю такого алгоритму є те, що всі зазначені операції, як правило, виконуються в межах єдиного процесу обробки запиту, що має характер “однієї транзакції”. При цьому окремі етапи, такі як підготовка даних або генерація файлу виписки, можуть бути ресурсоємними як з точки зору часу виконання, так і з точки зору використання обчислювальних ресурсів. Це призводить до того, що загальний час виконання запиту значною мірою залежить від обсягу оброблюваних даних та поточного стану системи. Додатковою складністю є необхідність отримання допоміжних даних із зовнішніх або суміжних підсистем банку. Наприклад, при формуванні розширених виписок може виникати потреба у зверненні до процесінгових систем для отримання інформації про деталі транзакцій, що збільшує кількість міжсистемних викликів і, відповідно, впливає на загальну продуктивність процесу [64, 65].

На рисунку 3.2 наведено UML-діаграму послідовності, яка відображає поточний типовий алгоритм формування виписки за рахунками [64, 66]. Проведений аналіз дозволяє виділити ряд суттєвих недоліків існуючого підходу до побудови виписок. Вони впливають на продуктивність роботи операцій побудови виписок за рахунками та системи e-Banking в цілому, а також, при певних умовах, ще й на надійність її функціонування.

- Виконання усіх складових алгоритму “в одну транзакцію”: алгоритм складається з декількох “важких” атомарних операцій, які не можна виконувати паралельно, але можливо розділити на окремі операції, кожна з яких виконується в окремому потоці. До таких операції відносяться підготовка даних та формування файлу виписки [9].
- Відсутність механізмів повторного використання вже сформованих даних: користувачеві досить часто може бути потрібна виписка, яку він вже раніше запитував, але система це не враховує.

- Відсутність механізмів прогнозування можливих дій користувача: система не виконує аналіз минулих операції побудови виписки користувача і не прогнозує параметри та можливий час наступного запиту від користувача, для того щоб “діяти на упередження”.
- Виконання операції в синхронному, по відношенню до користувача режимі: користувач очікує поки сервер виконає необхідні дії, що за певних умов може бути досить довго [9].
- Відсутність врахування поточної завантаженості системи іншими функціональними задачами: оскільки при побудові виписок необхідне виділення достатньої кількості системних ресурсів для функціонування фреймворків формування файлів, таких як наприклад Jasperreports [45], це може призвести до вичерпання виділених ресурсів при високій завантаженості системи іншими задачами [64].

Таким чином, можна зробити висновок, що існуючий підхід до побудови виписок за рахунками не забезпечує достатнього рівня ефективності функціонування системи e-Banking в умовах високих навантажень і потребує вдосконалення. Основним напрямком такого вдосконалення є перехід від монолітного, синхронного алгоритму до більш гнучкої моделі, яка передбачає декомпозицію процесу на окремі незалежні етапи, можливість їх асинхронного виконання, а також використання накопичених даних про поведінку користувачів. У цьому контексті доцільним є застосування інтелектуального модуля прогнозування, який дозволяє аналізувати історію запитів користувача та визначати найбільш ймовірні параметри майбутніх звернень до системи. Це, у свою чергу, створює можливість виконувати частину обчислень заздалегідь, у періоди з низьким навантаженням, що сприяє підвищенню загальної продуктивності системи. Крім того, до перспективних напрямків вдосконалення належить реалізація механізмів повторного використання вже сформованих виписок або їх частин, а також врахування поточного стану системних ресурсів при плануванні виконання відповідних задач. Це дозволяє

більш ефективно розподіляти навантаження та забезпечувати стабільну роботу системи навіть у пікові періоди.

Поряд із базовою доменною моделлю, представленою у вигляді множини сутностей, відношень та обмежень, доцільним є її подальший розвиток у напрямку знання-орієнтованого представлення предметної області. Такий підхід передбачає доповнення класичної структури доменної моделі елементами, що відображають поведінкові характеристики користувачів та накопичений експертний досвід функціонування системи. Запропонована доменна модель розширює типову доменну модель знання-орієнтованою складовою, яка дозволяє перейти від статичної моделі предметної області до моделі, що враховує знання про процеси взаємодії користувача із системою e-Banking. В межах такого підходу до складу доменної моделі вводиться онтологічний профіль користувача, який формалізує параметри його взаємодії із системою, а також система експертних доменних правил, що відображає накопичений досвід функціонування відповідних бізнес-процесів. Зазначені компоненти інтегруються у структуру моделі, розширюючи її семантичні можливості.

Онтологічний профіль користувача – це персоналізоване представлення знань про конкретного користувача, яке сформоване у відповідності до визначеної для систем e-Banking онтології, яка формально представлена у формулі 3.2 як кортеж із 4 підмножин [46]:

$$O = <C, R, I, A>, \qquad (3.2)$$

де C – множина сутностей, які можуть виконувати будь-які дії у системі, над якими виконуються дії або які є результатом виконання будь-якої дії, R – множена відношень між сутностями, I – множина індивідів(інстансів), A – множина усіх правил та обмежень, які накладаються на будь-які дії чи сутності у системі. Як видно з формули 4, онтологія пов’язана з доменною моделлю, а головною її відмінністю є те, що онтологія, на відміну від доменної моделі, оперує поняттям “індивід” [67].

Знання формуються на основі інформації про події та дії пов’язані з користувачем у системі, а також зв’язків користувача з іншими сутностями домену “виписок за рахунками”, і представлене у вигляді графа знань, який зберігається у БД знань, наприклад, Neo4J. Онтологічний профіль користувача (P_u) формально представлено у формулі 3.3 [46].

$$P_u \in O;\ u \in I, \qquad (3.3)$$

Експертні доменні правила (ЕДП) – це введені в систему додаткові критерії та/або умови, які дозволяють реагувати на різні зміни у домені виписок та корегувати прогноз відповідно до таких змін. Наприклад, “лише цього року, підприємства з обігом більше 100 мільйонів гривень повинні подати з 1 по 5 січня у податкову звіт про доходи за минуле півріччя”. Очевидно, що для такого звіту клієнт банку буде формувати виписку за усіма рахунками, і, з великою ймовірністю, у вказаних вище числах. Такого роду умови не можуть бути враховані при прогнозуванні за допомогою методів аналізу часових рядів, але дозволивши експертам зі сторони банку створювати та додавати в систему власні правила для уточнення прогнозу – ця задача також може бути вирішена з використанням знання-орієнтованого підходу.

Експертні доменні правила можуть бути представлені у вигляді формальної моделі (R), що є кортежом з 3 елементів [46]:

$$R = <C, A, P>, \qquad (3.4)$$

де C – умови застосування правила, A – дія, яку необхідно виконати, P – пріоритет правила. Самі правила створюються у нотації BPMN [31] та зберігаються у форматі XML, сумісному з використовуваним у системі BPM-engine. Одним з використовуваних програмних рішень є Flowable Engine [68]. На рисунку 3.3 наведено приклад збереженого у форматі XML, сумісному з нотацією BPMN, правила “якщо користувач є ФОП, то він має формувати виписку за попередній місяць на початку наступного місяця”. Введені в систему правила можуть зберігатися структуровано або безпосередньо на файловій системі серверів додатків, або у існуючій базі даних системи е-

Banking в окремих таблицях. Обидва варіанти дозволяють використовувати наявні ресурси без необхідності їх суттєвого розширення.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
             xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC"
             xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI"
             typeLanguage="http://www.w3.org/2001/XMLSchema"
             expressionLanguage="http://www.w3.org/1999/XPath"
             targetNamespace="http://bpmn.io/schema/bpmn">

  <process id="fop_statement_process" name="FOP Monthly Statement Generation" isExecutable="true">
    <startEvent id="StartEvent_Timer" name="Start Timer">
      <timerEventDefinition/>
    </startEvent>
    <task id="Task_CheckUserType" name="Check if user type == ФОП"/>
    <exclusiveGateway id="Gateway_IsFOP" name="Is user ФОП?"/>
    <task id="Task_GenerateStatement" name="Generate monthly statement"/>
    <endEvent id="EndEvent_Success" name="End (Success)"/>
    <endEvent id="EndEvent_NoAction" name="End (No Action)"/>

    <sequenceFlow id="flow1" sourceRef="StartEvent_Timer" targetRef="Task_CheckUserType"/>
    <sequenceFlow id="flow2" sourceRef="Task_CheckUserType" targetRef="Gateway_IsFOP"/>
    <sequenceFlow id="flow3" sourceRef="Gateway_IsFOP" targetRef="Task_GenerateStatement">
      <conditionExpression xsi:type="tFormalExpression"><![CDATA[${user.type == "ФОП"}]]></conditionExpression>
    </sequenceFlow>
    <sequenceFlow id="flow4" sourceRef="Gateway_IsFOP" targetRef="EndEvent_NoAction">
      <conditionExpression xsi:type="tFormalExpression"><![CDATA[${user.type != "ФОП"}]]></conditionExpression>
    </sequenceFlow>
    <sequenceFlow id="flow5" sourceRef="Task_GenerateStatement" targetRef="EndEvent_Success"/>
  </process>
</definitions>
```

Рис. 3.3. Приклад збереженого ЕДП у форматі XML

Після формального визначення зазначених компонентів доцільно відзначити, що онтологічний профіль користувача відображає не лише поточні параметри його запитів, але й історію взаємодії із системою, яка може бути представлена у вигляді впорядкованої послідовності подій. Така інтерпретація дозволяє розглядати відповідні дані як основу для подальшого застосування методів аналізу часових рядів у задачах прогнозування дій користувача. У свою чергу, експертні доменні правила, які є частиною множини A, забезпечують можливість формалізації як жорстких обмежень, пов’язаних із коректністю виконання операцій, так і знань, накопичених банком в процесі експлуатації системи. Це дозволяє враховувати не лише формальні аспекти функціонування системи, а й практичні особливості її функціонування в реальних умовах [46].

Отже, у даному підрозділі було запропоновано та побудовано знання-орієнтовану доменну модель для задачі формування виписок за рахунками клієнтів систем e-Banking. Розроблена знання-орієнтована доменна модель відрізняється від існуючих побудовою та застосуванням онтологічних профілів користувачів систем e-Banking, у поєднанні з експертними доменними правилами для обробки їх емпіричних знань. Це дозволяє розширити інформаційний базис і враховувати семантичний контекст вимог до функціоналу системи, що, в кінцевому рахунку, сприяє підвищенню якості обслуговування клієнтів відповідного банку. А також виконано формалізацію поточного алгоритму її реалізації, що дало змогу виявити його основні недоліки та обґрунтувати необхідність переходу до інтелектуальних підходів. Отримані результати створюють основу для подальшої розробки бізнес-логіки інтелектуального модуля прогнозування та вдосконалення процесу формування виписок, що розглядається у наступному підрозділі.

3.2 Розробка бізнес-логіки інтелектуального модуля прогнозування дій користувачів із використанням методів аналізу часових рядів

У попередньому підрозділі було побудовано знання-орієнтовану доменну модель для домену формування виписок за рахунками клієнтів систем e-Banking. Вона також в себе включає онтологічні профілі користувачів та експертні доменні правила. Наступним кроком є розробка бізнес-логіки ІМП, яка б забезпечувала практичне використання побудованої моделі у процесі функціонування системи. Основним призначенням ІМП є визначення найбільш ймовірного моменту часу формування виписки користувачем на основі аналізу історії взаємодії користувачів із системою. Таке прогнозування дозволяє заздалегідь підготувати дані для обробки запитів або ініціювати відповідні процеси, що сприяє зменшенню часу відгуку системи, зменшенню періодів пікових навантажень на систему та підвищенню якості обслуговування клієнтів в цілому.

Реалізація запропонованого підходу передбачає формалізацію процесу обробки вхідних даних, побудову послідовності етапів прийняття рішень та визначення механізмів інтеграції результатів прогнозування у загальний процес побудови виписок. При цьому в якості вхідної інформації використовуються дані, сформовані у межах знання-орієнтованої доменної моделі, що забезпечує врахування як історичних даних, так і експертних знань. Отримані результати були опубліковані в статті Дааса Т.І. та Ткачука М.В. «Застосування методів аналізу часових рядів та доменного моделювання при розробці інтелектуального модуля прогнозування в системі інтернет-банкінгу» у журналі «Системи обробки інформації». 2025. вип. 3(182), с.34–43.

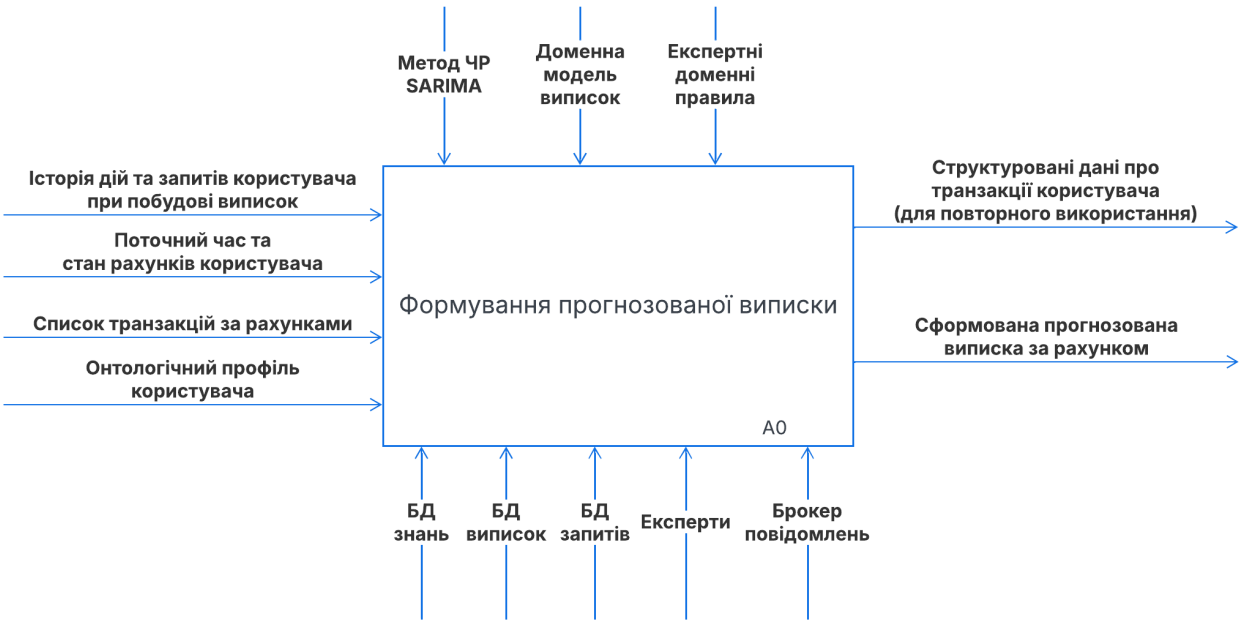

Рис. 3.4. Контекстна IDEF0-діаграма функціональності ІМП

Для зображення бізнес-логіки ІМП доцільно використати нотацію IDEF0, яка дозволяє формалізовано представити функціональну структуру процесу, визначити взаємозв’язки між його складовими та відобразити потоки даних. Використання даної нотації є доцільним у контексті дослідження, оскільки вона забезпечує наочне представлення алгоритму роботи ІМП з урахуванням вхідних даних, вихідних результатів, механізмів реалізації та обмежень, що

накладаються на процес. На рисунку 3.4 зображено контекстну IDEF0-діаграму, яка узагальнено відображає функціонал ІМП та визначає його місце у процесі формування виписок за рахунками клієнтів [46].

В якості вхідних даних для реалізації бізнес-логіки ІМП у складі домену “побудови виписок за рахунком” необхідно мати:

- Історію дій та запитів користувача при побудові виписок
- Поточний час та стан рахунків
- Список транзакцій за рахунками
- Онтологічний профіль користувача, який є структурованим представленням знань про поведінку користувача у системі

Вихідними даними є:

- Структуровані дані про транзакції користувача, які необхідні для можливості повторного використання підготовлених даних
- Сформована прогнозована виписка за рахунком: файл виписки

Складовими інтерфейсу управління є:

- Метод аналізу часових рядів SARIMA, доцільність та принципи застосування якого наведено у розділі 2.2
- Доменна модель
- Експертні доменні правила, які є статичним відображенням знань та поточних вимог до систем

Складовими механізмів – засобів, що необхідні для успішного функціонування бізнес-логіки, є:

- Бази даних: знань(онтологій), виписок, запитів(історій виписок)
- Експерти, які оцінюють і аналізують накопичені знання про поведінку клієнтів у системі та наявні “зовнішні” чинники впливу, з можливістю подальшого впливу на бізнес-логіку шляхом введення експертних доменних правил
- Брокер повідомлень, для можливості асинхронної взаємодії сервісів у пропонованій архітектурі.

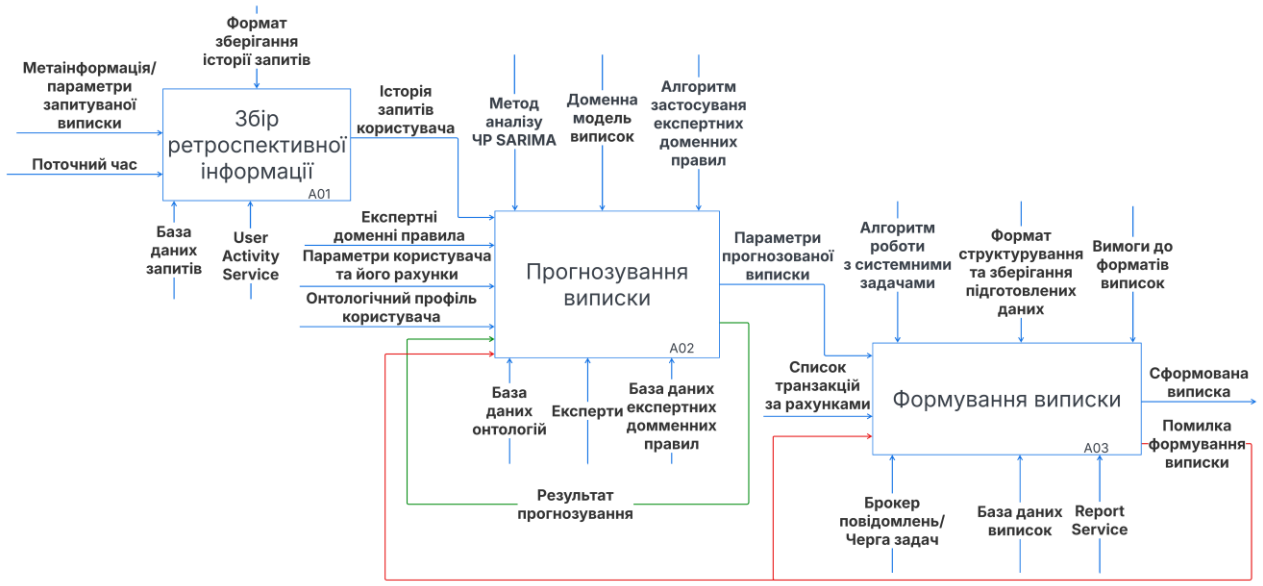

Рис. 3.5. Декомпозиція контекстної IDEF0-діаграми бізнес-логіки ІМП

На основі вище зазначених IDEF0-діаграм і наведених у [64] перспективних шляхів вдосконалення існуючих алгоритмів побудови виписок за рахунками користувачів може бути сформульовано загальний алгоритм побудови таких виписок з використанням ІМП, який має такі кроки:

1) Зберегти у БД запитів інформацію про параметри побудови виписки за запитом користувача: наприклад, дату запиту, список рахунків та формат виписки.
2) З певним періодом, який повинен конфігуруватися у системі, наприклад, 1 раз на добу, виконувати формування "прогнозів" таким чином:
    a) Виконати за допомогою методів MLE (метод максимальної правдоподібності) та/або ACF (графіків функції автокореляції) підрахунок коефіцієнтів p, q, d, P, Q, D, S, які використовуються у методі аналізу часових рядів (МАЧР) SARIMA [52, 53]
    b) Виконати прогнозування дати формування користувачем майбутньої виписки за допомогою МАЧР SARIMA
    c) Якщо у БД експертних доменних правил існує хоча б одне активне правило, то виконати уточнення прогнозу з урахуванням

введених у систему доменними експертами семантичних правил та побудованої доменної моделі виписок, інакше пропустити крок

d) Сформувати системну задачу на підготовку виписки та зберегти її у черзі для виконання.

3) У час найменшої завантаженості системи виконати побудову виписки за рахунком згідно створеної системної задачі:

a) Виконати пошук у БД виписок(так званому кеші) раніше сформованих(підготовлених) даних про транзакції, які підпадають під умови поточного запиту

b) Якщо дані знайдені, то доповнити їх даними про нові(порівняно з наявними у кеші) транзакції, інакше сформувати дані про транзакції, які підпадають під умови поточного запиту, з нуля

c) Підготувати дані про транзакції у форматі JSON та зберегти(закешувати) їх у БД виписок, в тому числі і для майбутнього "перевикористання"

d) Сформувати виписку на основі підготовлених даних та зберегти(закешувати) її у БД виписок.

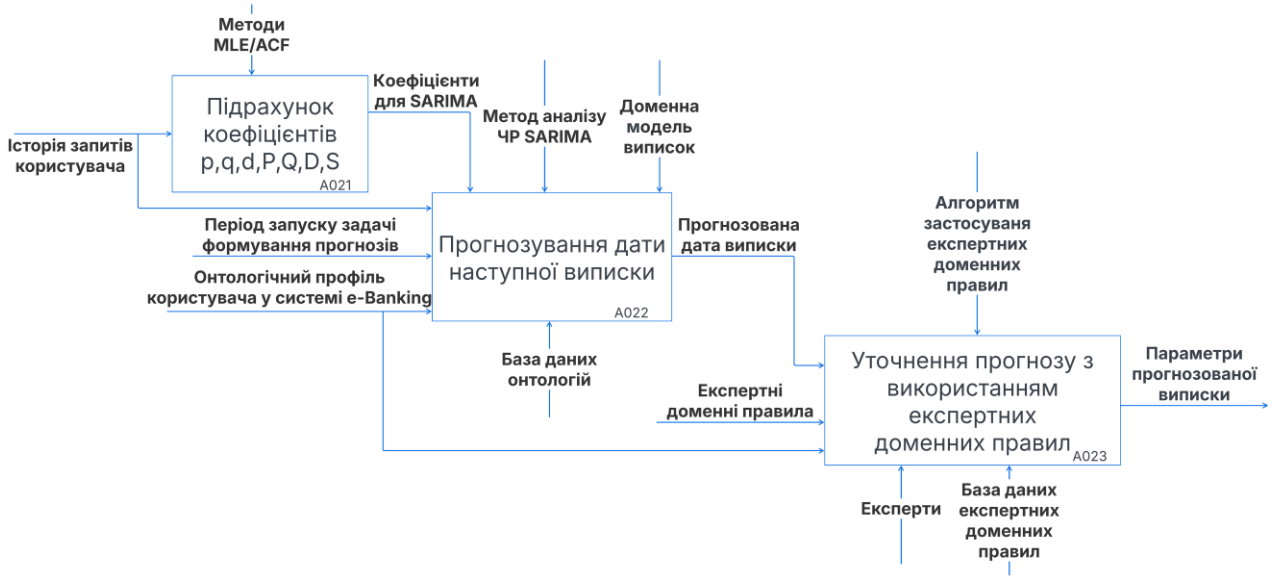

Рис. 3.6. Деталізація підзадачі прогнозування

Рис. 3.7. Деталізація підзадачі формування файлу виписки

На рисунку 3.6 виконано декомпозицію функціонального блоку A02, який відповідає 2-му кроку наведеного алгоритму бізнес-логіка, а рисунок 3.7 відображає декомпозицію функціонального блоку A03, який, в свою чергу, відповідає кроку 3 зазначеного алгоритму [46].

При безпосередньому запиті користувача на побудову виписки за рахунком система повинна виконати:

1) Пошук у БД виписок раніше сформованої за запитом користувача виписки. Якщо виписка знайдена і задовольняє параметрам(дати з/по, перелік рахунків, тощо) з запиту, то користувач отримує цей файл виписки і запит вважається обробленим, інакше – переходимо до наступного кроку.
2) Пошук сформованої "прогнозованої", за допомогою ІМП, виписки:
    a) Якщо виписка знайдена і задовольняє параметрам(дати з/по, перелік рахунків, тощо) з запиту, то користувач отримує цей файл виписки.
    b) Якщо виписка знайдена і не повністю задовольняє параметрам з запиту, то виконується доповнення раніше підготовлених даних новими транзакціями та формування нової виписки. При чому,

якщо формат виписки дозволяє додати нові транзакції до вже створеного файлу виписки, то дані додаються без повного переформування файлу виписки. Оновлені підготовлені дані та файл виписки кешуються. Інакше –переходимо до наступного кроку.

3) Якщо ж не виконано жодну з попередніх умов, то система виконує підготовку даних про транзакції за запитом користувача та формує виписку запитуваного формату. Підготовлені дані та файл виписки кешуються.

При цьому, обов’язковою вимогою є те, що запити користувачів на формування виписок завжди повинні оброблюватися в асинхронному режимі [46].

Розроблена бізнес-логіка інтелектуального модуля прогнозування базується на поєднанні методів аналізу часових рядів із знання-орієнтованими компонентами доменної моделі, що дозволяє забезпечити комплексний підхід до обробки інформації про поведінку користувачів. На відміну від підходів, що ґрунтуються виключно на статистичному аналізі, запропонована логіка передбачає використання додаткових джерел знань, зокрема онтологічних профілів та експертних доменних правил, що дозволяє враховувати не лише історичні закономірності, але й семантичні особливості предметної області. Це забезпечує більш точне визначення параметрів формування виписок та підвищує якість результатів прогнозування. Крім того, запропонована бізнес-логіка враховує необхідність функціонування в умовах змінного навантаження та динамічної поведінки користувачів систем e-Banking. Це досягається за рахунок можливості регулярного оновлення вхідних даних і повторного застосування процедур прогнозування, що дозволяє адаптувати результати до актуального стану системи. Такий підхід забезпечує не лише підвищення якості обслуговування клієнтів, але й створює передумови для оптимізації використання ресурсів системи, зокрема шляхом зменшення пікових навантажень, пов’язаних із формуванням виписок.

Отже, у даному підрозділі розроблено бізнес-логіку функціонування інтелектуального модуля прогнозування у складі системи e-Banking, яка базується на використанні знання-орієнтованої доменної моделі та методів аналізу часових рядів. Запропонований підхід забезпечує визначення параметрів формування виписок на основі аналізу історичних даних про поведінку користувачів із урахуванням експертних знань предметної області. Отримані результати створюють основу для подальшої розробки переліку критеріїв та кількісних метрик оцінки ефективності функціонування інтелектуального модуля прогнозування, що розглядається у наступному підрозділі.

3.3 Розробка критеріїв та кількісних метрик для оцінки ефективності функціонування інтелектуального модуля прогнозування у складі системи e-Banking

В процесі розробки та впровадження ІМП у складі системи e-Banking важливою є задача об’єктивної оцінки ефективності його функціонування. На відміну від традиційних підходів, де оцінювання обмежується окремими показниками продуктивності або часу відгуку, у даному випадку необхідним є комплексний підхід, що враховує як якість прогнозування дій користувачів, так і вплив відповідних механізмів на загальні характеристики системи. Це обумовлює необхідність формування системи критеріїв та кількісних метрик, які дозволяють всебічно оцінити як результати застосування ІМП у задачі формування виписок за рахунками клієнтів, так і використання МСА як еталонної системної архітектури для проєктування та розробки систем e-Banking. При цьому необхідно враховувати як суто технічні показники, так і переваги для кінцевих користувачів системи e-Banking та/або банку. Пропонується виділити 5 основних критеріїв, які доцільно застосовувати для оцінки ефективності алгоритму [46].

1. Точність прогнозу, яка розраховується як відношення кількості запитів користувачів на побудову виписки, які відбулися у прогнозований модулем ІМП час до усіх запитів користувачів на побудову виписок.

$$FA = Req_F / Req_A;\ \ FA \in [0, 1], \qquad (3.5)$$

де Req_F – кількість запитів користувачів на побудову виписки, які відбулися у прогнозований модулем ІМП час, Req_A – кількість усіх запитів користувачів на побудову виписок.

2. Відсоток перевикористання раніше підготовлених даних про транзакції або раніше сформованої виписки при обробці запитів користувачів на побудову виписок. Розраховується як відношення кількості запитів, для яких формування виписки завершується на 1 або 2 кроці алгоритму обробки запиту користувача на побудову виписки наведеному у попередньому розділі, до усіх запитів користувачів на побудову виписок.

$$DR = (Req_R / Req_A) * 100\%, \qquad (3.6)$$

де Req_R – кількість запитів користувачів на побудову виписки, для яких були використані раніше підготовлені дані про транзакції або раніше сформована виписка, Req_A – кількість усіх запитів користувачів на побудову виписок.

3. Відсоток часу, під час якого завантаженість центрального процесору (ЦП) серверів, що відповідають за домен виписок за рахунками, не перевищує допустимий рівень.

$$PL = (T_{CL} / T_A) * 100\%, \qquad (3.7)$$

де T_{CL} – час, у який завантаженість ЦП серверів, що відповідають за домен виписок, вище допустимого рівня встановленого банком, T_A – весь час роботи системи. Рівень допустимого навантаження на ЦП може бути визначений конкретним банком, але зазвичай, згідно в рекомендаціями, визначається на рівні 80% від доступного ресурсу ЦП. Такі рекомендації містяться, наприклад, у керівних документах для своїх систем від компанії Oracle, однієї з провідних світових ІТ-компаній [69]. Для моніторингу і збереження метрик може бути застосований будь-який вже використовуваний в банку програмний засіб,

призначений для цих операцій. Одним с популярних та широко використовуваних рішень є Prometheus та Grafana [70].

4. Середній час витрачений сумарно на виконання усіх кроків з побудови виписки.

$$T_{AVG} = (T_1 + ... + T_i + ... + T_n) / n, \quad (3.8)$$

де T_i – час (у мілісекундах) виконання і-го запиту на побудову виписки, n – загальна кількість операцій побудови виписок. При чому середній час повинен бути окремо розрахований для різного об'єму транзакцій у випсках та формату, оскільки, як було зазначено раніше, час формування виписки прямо залежить від її параметрів.

5. Відсоток клієнтських запитів, які завершилися помилкам, що пов'язані з очікуванням формування виписки за рахунком.

$$RE = (Req_E / Req_A) * 100\%, \quad (3.9)$$

де Req_E – кількість запитів користувачів на побудову виписки, які завершились помилкою, Req_A – кількість усіх запитів користувачів на побудову виписок. Прикладом однієї з найчастіше виникаючих помилок є автоматичне завершення системою сесії користувача по таймауту через дуже довге очікування відповіді на синхронний запит на побудову виписки.

Наведені критерії можуть бути застосовані без необхідності залучення суттєвих додаткових ресурсів для зняття та зберігання відповідних їх метрик в процесі експлуатації системи. Відповідні критерії є кількісними показниками для оцінки таких нефункціональних вимог (НФВ) до ПЗ, як продуктивність та надійність. Розроблені критерії та метрики будуть використані для проведення експериментального дослідження ефективності запропонованого підходу на прототипі фрагменту системи e-Banking, результати якого наведено у розділі 4.

3.4 Аналіз вимог та проєктування мікросервісної архітектури для реалізації функціоналу інтелектуального модуля прогнозування у складі системи e-Banking

Розробка бізнес-логіки ІМП, розглянута у попередньому підрозділі, потребує визначення відповідної програмної архітектури, яка забезпечить його ефективну інтеграцію у склад системи e-Banking. З урахуванням специфіки функціонування таких систем, що характеризуються високими вимогами до продуктивності, надійності та безперервності обслуговування, вибір архітектурного підходу є критично важливим етапом, що безпосередньо впливає на якість реалізації запропонованих моделей і алгоритмів. Сучасні системи e-Banking функціонують в умовах постійного зростання кількості користувачів, транзакцій та обсягів оброблюваних даних, що призводить до ускладнення їх внутрішньої структури. Традиційні монолітні архітектурні рішення дедалі частіше не забезпечують необхідного рівня масштабованості, продуктивності та гнучкості, що обумовлює необхідність переходу до більш сучасних підходів, зокрема МСА. У цьому контексті виникає задача аналізу вимог до архітектури системи та проєктування її структури з урахуванням інтеграції ІМП [32, 58].

Проєктування програмної архітектури системи e-Banking із інтегрованим ІМП повинно здійснюватися з урахуванням сукупності функціональних та нефункціональних вимог, що визначають якість функціонування системи в умовах реального використання. До функціональних вимог належить забезпечення коректної взаємодії ІМП з іншими компонентами системи, зокрема отримання історичних даних про дії користувачів, формування прогнозів та ініціювання відповідних процесів підготовки виписок. При цьому важливим є забезпечення прозорої інтеграції ІМП у бізнес-процеси системи без порушення існуючої логіки обробки запитів користувачів. Але, на рішення щодо вибору архітектури більше впливають саме нефункціональні вимоги, обумовлені специфікою домену e-Banking. Система повинна забезпечувати обробку великої кількості транзакцій за одиницю часу із гарантованою

коректністю та цілісністю даних. Високі вимоги до продуктивності передбачають мінімізацію часу відгуку при виконанні критичних операцій, таких як автентифікація користувачів, здійснення платежів чи побудова виписок. Не менш важливими є вимоги до масштабованості, що дозволяє адаптувати систему до змін навантаження, а також до надійності та відмовостійкості, які забезпечують безперервність обслуговування клієнтів навіть у випадку часткових збоїв окремих компонентів. Додатково слід враховувати необхідність поєднання синхронних та асинхронних протоколів взаємодії компонентів системи. У банківській сфері частина операцій (наприклад, автентифікація користувача чи проведення платежу) вимагає негайної відповіді, тоді як інші завдання (генерація виписок, підготовка аналітичних звітів) доцільно виконувати у “фоновому” режимі. Неправильний вибір механізмів комунікації може призвести до підвищення затримок, зниження продуктивності або перевантаження сервісів. Вирішення цих проблем вимагає формулювання спеціалізованих принципів проєктування MCA для систем e-Banking, які б враховували як технічні, так і доменні обмеження та особливості. Таким чином, сукупність наведених вимог визначає необхідність використання архітектурного підходу, який забезпечує декомпозицію системи на незалежні компоненти, підтримує їх автономне функціонування та дозволяє гнучко масштабувати окремі частини системи залежно від навантаження.

Разом із тим, проєктування MCA залишається складним процесом, що потребує глибокого доменного аналізу та врахування численних технічних обмежень [58]. У зв’язку з цим зростає інтерес до інтелектуальних підходів, що інтегрують методи штучного інтелекту (artificial intelligence – AI) у процеси розробки та супроводу MCA. Завдяки використанню алгоритмів машинного навчання, обробки природної мови, аналізу часових рядів і побудови онтологічних моделей з’являється можливість створювати архітектури, які не лише відтворюють логіку бізнес-домену, а й адаптуються до його змін [71]. Зокрема, для систем e-Banking, що мають відповідати вимогам високої продуктивності, безпеки надійності та супроводжуваності, такі AI-методи

відкривають перспективи переходу до більш гнучких рішень із підтримкою динамічного масштабування, адаптивної конфігурації і проактивного моніторингу. Окремо варто зазначити, що застосування цих підходів на таких етапах життєвого циклу (ЖЦ) продукту, як аналіз вимог та проєктування, дозволяє знизити витрати на початкову розробку програмної системи (ПС). Саме цей показник для МСА є таким, який має більші значення порівняно з іншими типами системних архітектур, в тому числі і МА. AI-підходи до розробки та супроводу МСА формують новий рівень автоматизації процесів ЖЦ ПС, тому що вони спрямовані на зменшення впливу людського фактору при прийнятті архітектурних рішень, підвищення узгодженості між бізнес-вимогами та технічною реалізацією, а також забезпечення ефективного конфігурування ПС. Узагальнюючи результати аналізу наукових джерел, усі інтелектуальні підходи до побудови та супроводу ПС з МСА можна поділити на три основні групи, кожна з яких охоплює певний етап ЖЦ цих системи, і ця класифікація наведена на рисунку 3.8 [72].

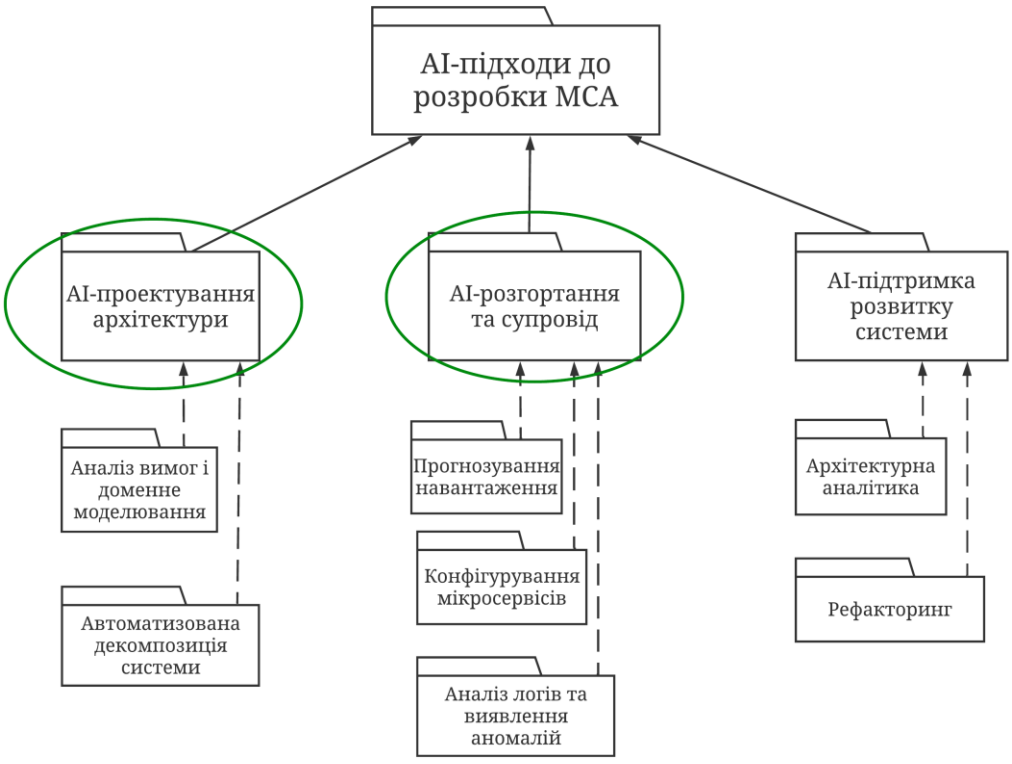

Рис. 3.8. Класифікація AI-підходів до розробки МСА

AI-проєктування архітектури охоплює етапи аналізу вимог і створення концептуальної структури системи. Його метою є автоматизація початкових етапів розробки — від аналізу текстових специфікацій до визначення меж майбутніх мікросервісів. Основними методами є:

- аналіз вимог і доменне моделювання з використанням методів обробки природної мови (NLP) для виділення бізнес-сутностей і побудови зв’язків між ними;
- автоматизована декомпозиція системи, що базується на алгоритмах кластеризації (clustering) та машинного навчання (machine learning) для виявлення потенційних груп функцій, які можуть бути реалізовані як окремі сервіси [71, 73].

AI-розгортання та супровід спрямоване на оптимізацію роботи вже розгорнутих систем і забезпечення їх стабільного функціонування. До цієї групи належать методи, що виконують:

- прогнозування навантаження, що використовують машинне навчання або аналіз часових рядів для передбачення пікових періодів активності користувачів;
- конфігурування мікросервісів, тобто автоматичне налаштування параметрів виконання, розподілу ресурсів і масштабування контейнерів залежно від поточного стану системи, використовуючи, наприклад, метод аналізу прецедентів (case-based reasoning – CBR) [74];
- аналіз логів та виявлення аномалій за допомогою алгоритмів anomaly detection, які дозволяють ідентифікувати відхилення поведінки сервісу і запобігати відмовам [75].

AI-підтримка розвитку системи передбачає використання методів аналітики для постійного вдосконалення архітектури та підвищення її стійкості. До основних методів цієї групи належать:

- архітектурна аналітика, що дозволяє виявляти надлишкові або надмірно пов’язані компоненти, оцінювати якість декомпозиції та визначати потребу в модифікаціях; для цього можуть бути застосовані графовий аналіз архітектури та кластеризація на основі метрик взаємодії сервісів в продуктивному середовищі;
- методи, що забезпечують автоматизований рефакторинг, за рахунок аналізу вихідного коду на предмет залежностей між сервісами та визначення оптимальних шляхів їх реструктуризації [76].

Попри очевидні переваги інтелектуальних підходів, їх упровадження в процеси розробки та супроводу МСА супроводжується низкою викликів. Найважливішим із них є обмежена якість та доступність даних, необхідних для навчання моделей. У системах e-Banking це питання особливо критичне, оскільки дані мають високий рівень чутливості, а їх використання для навчання моделей машинного навчання потребує дотримання нормативних вимог і політик безпеки. Відповідно, для побудови коректних моделей необхідне поєднання реальних і синтетичних даних, а також застосування механізмів анонімізації. Також впровадження нових інтелектуальних компонентів вимагає адаптації існуючих CI/CD-процесів, перегляду підходів до тестування й забезпечення сумісності між традиційними сервісами та AI-модулями, деякі з яких можуть вимагати значних ресурсів через високу обчислювальну складність реалізації і функціонування їх алгоритмів. Попри це, використання інтелектуальних підходів відкриває можливості для побудови систем, які здатні прогнозувати поведінку користувачів і навантаження, динамічно оптимізувати конфігурації сервісів і покращувати архітектурну структуру на основі накопичених знань [72]. Зокрема, застосування методів CBR для адаптивного управління конфігураціями програмних мікросервісів [74] дозволило приблизно на 15-20% скоротити час пошуку відповідних рішень при експериментальному дослідження тестових МСА застосунків за умов розміру бази даних прецедентів від 50 до 1000 записів.

Одним із перспективних напрямів розвитку інтелектуальних підходів у розробці МСА є знання-орієнтоване проєктування, яке передбачає використання формалізованих моделей знань для підтримки процесів декомпозиції, конфігурування та супроводу систем. На відміну від суто алгоритмічних рішень, цей підхід базується на накопиченні, структуризації та повторному використанні знань, отриманих у попередніх проєктах або під час функціонування системи. У контексті e-Banking це дозволяє враховувати як типові сценарії користувацької поведінки, так і особливості архітектурних патернів, перевірених у галузі. Ключовим елементом знання-орієнтованого проєктування є побудова онтологічних моделей домену, які описують сутності, зв’язки між ними та правила взаємодії. Такі моделі забезпечують семантичну узгодженість між бізнес-рівнем і технічною реалізацією системи, слугуючи основою для автоматизованого генерування архітектурних шаблонів. Використання онтологій також сприяє підвищенню прозорості прийняття архітектурних рішень і полегшує інтеграцію з інтелектуальними модулями, що здійснюють аналіз і прогнозування. Також одним із дієвих механізмів реалізації знання-орієнтованого підходу є CBR-метод, який дозволяє приймати рішення на основі схожих раніше розв’язаних ситуацій. У процесі проєктування МСА це означає можливість автоматичного вибору оптимальних архітектурних патернів, параметрів конфігурації або стратегій масштабування, виходячи з історії успішних розгортань і поведінки системи в аналогічних умовах [77].

В цілому, проєктування МСА для систем e-Banking ґрунтується на певній множині методів та підходів. Вони дозволяють виконати декомпозицію предметної області, спроєктувати якісну взаємодію між виділеними компонентами та забезпечують можливість інтеграції інтелектуальних механізмів подальшої підтримки пропонованих архітектурних рішень. Базовим принципом є декомпозиція системи на сервіси з чітко визначеними межами відповідальності (bounded context). Етап проєктування системи є одним із перших згідно типового життєвого циклу програмного забезпечення. Застосування такого підходу на етапі проєктування дозволяє мінімізувати

залежності між сервісами, підвищити їх автономність функціонування та спростити майбутнє масштабування системи. Для домену e-Banking це дає змогу забезпечити відокремлення функціональності, пов’язаної з обробкою транзакцій, управлінням рахунками, веденням профілів клієнтів, формуванням виписок та іншими критично важливими процесами. Що у свою чергу забезпечує одну з важливих вимог до систем цього класу: відсутність впливу нових модулів на якість та логіку роботи вже впровадженого функціоналу. Провідну роль для цього у проєктуванні МСА відіграє доменне моделювання (DDD), яке використовується для визначення меж сервісів та формалізації бізнес-логіки предметної області [32]. В розрізі систем e-Banking це означає побудову моделей, що відображають ключові інформаційні сутності домену. DDD допомагає забезпечити відповідність пропонованих архітектурних рішень заданій предметній області та зменшити ризик дублювання функціональності між різними сервісами [78]. Другим важливим аспектом є вибір способу та протоколів комунікації між сервісами. У банківських системах неможливо обмежитися лише синхронними викликами через REST (Representational State Transfer) чи gRPC (Google Remote Procedure Calls), через те, що вони створюють додаткові затримки при високих навантаженнях, а також потребують більше ресурсів [79]. Використання ж event-орієнтованих підходів із застосуванням брокерів повідомлень має розвантажити систему, а саме передавати події асинхронно та забезпечити кращу стійкість до пікових навантажень.

Типові принципи проєктування МСА полягають у чіткому розділенні доменів, відокремленому зберіганні даних, мінімальних сферах відповідальності сервісів та незалежних життєвих циклах компонентів. Виконаний аналіз існуючих підходів до декомпозиції ІС показав, що типові принципи проєктування МСА добре себе показують у системах, до яких не висуваються високі вимоги до швидкодії та узгодженості транзакцій. Декомпозиція на занадто дрібні сервіси призводить до того, що виконання значної частини ключових бізнес-операцій вимагає в такому випадку значну

міжсервісну взаємодію. Це стосується таких операцій як формування різних видів платежів, перевірка коректності реквізитів, обробка транзакцій або формування виписки. Такі виклики потребують додаткових накладних витрат, як ресурсних, так і часових, хоча по суті виконують мінімальну бізнес-логіку. В якості прикладів можна привести такі операції, які необхідні для формування і проведення платежу: пошук залишку по рахунку, отримання реквізитів контрагента, отримання користувацьких налаштувань. Формально, вони мають бути реалізовані в окремих мікросервісах, оскільки ці ж функції необхідні і для багатьох інших функціональних частин системи e-Banking. В той же час подібна реалізація негативно впливатиме на показники швидкодії. Особливо це стосується таких операцій, як масовий імпорт платежів, де здебільшого неможливо зробити єдиний виклик, а відповідно накладні витрати зростають в арифметичній прогресії. На противагу, можна навести приклади подібних операцій, але складність і супроводження яких занадто високі, щоб не виносити їх в окремі мікросервіси. Це може бути мікросервіс, що виконує криптографічні операції, роботу з OTP-кодами (One-Time Password) чи авторизацію [32].

На основі наведеного вище можна формально, у вигляді формули, визначити основний принцип декомпозиції системи на сервіси, який може бути застосований при проєктуванні систем e-Banking:

$$DD = f(TRC, NC, OLC) = a \times OLC - b \times TRC - c \times NC, \qquad (3.10)$$

де *DD* – глибина декомпозиції, *TRC* – складність узгодження транзакцій (з точки зору баз даних), виконуваних у домені, *NC* – кількість мережевих міжсервісних запитів, *OLC* – складність підтримки бізнес-логіки «спільних операцій», *a, b, c* – вагові коефіцієнти, які знаходяться в проміжку [0,1], при цьому $a+b+c=1$. В результаті можна сказати, що особливості предметної області e-Banking, призводять до необхідності адаптувати підходи до декомпозиції у МСА і обов’язково враховувати ще й негативний вплив на показники якості системи. Якщо негативний вплив на швидкодію виконання певних операцій значно більший ніж вигода, яка може бути отримана від

зниження дублювання коду, то вищенаведені операції мають бути реалізовані декілька разів. Тобто в рамках кожного сервісу, де вони потрібні [32].

Нижче наведено перелік визначених концептуальних принципів і особливості їх адаптації до систем класу e-Banking:

1. Архітектурний стиль DDD. Принцип розділення складної системи контексти чітко поєднується з концепцією MCA і дає кращі результати при залученні великої кількості стейкхолдерів, які можуть надати якісну інформацію про систему з різних точок зору. За допомогою DDD у MCA досить легко реалізувати функціонал, який буде імплементувати знання-орієнтовані методи, які можна винести у окремий мікросервіс, який буде надавати API усім іншим сервісам системи. Але обов'язковим є врахування обмежень застосування DDD, які наведені раніше.

2. Протокол комунікації між сервісами, на які поділена система за допомогою DDD. При застосуванні у системах e-Banking мають поєднуватися як синхронні виклики, так і асинхронні. Синхронні виклики мають бути застосовані до операцій, які обов'язково мають відразу повернути результат виконання, наприклад, авторизація в системі. Асинхронні виклики мають бути застосовані, якщо немає нагальної потреби в негайному поверненні результату або ж отримання результату може бути реалізовано за допомогою адаптації функціональних вимог, наприклад, push-сповіщення або формування великих виписок, що описано у [46]. Для асинхронного обміну повідомленнями обов'язковим є використання брокерів повідомлень [62].

3. Використання архітектурних патернів, які дозволять забезпечити відповідність вимогам MCA і визначеним показникам якості ПЗ для систем e-Banking. Патерн API Gateway доцільно застосовувати для реалізації механізмів контролю доступу та засобів автентифікації при будь-якій міжсервісній взаємодії у системі. Патерн Database per Service має бути застосований обмежено через високий ступінь взаємозалежності збережених даних у системах e-Banking, які можуть бути пов'язані з декількома різними інформаційними сутностями. А високі вимоги до консистентності даних

унеможливлюють дублювання їх у різних базах даних (БД). Saga Pattern було визначено як один з обов’язкових механізмів побудови МСА у системах e-Banking. До систем цього класу висуваються дуже високі вимоги до управління транзакціями, а особливо складні вони в частині обробки збоїв з подальшим відкатом усього ланцюжка операцій. Багато операцій потребує або «відкат усієї транзакції» за можливістю, або ж проєктування системи таким чином, щоб існували компенсаційні механізми у випадку неможливості повного фізичного відкату транзакції на БД. Реалізація патерну Circuit Breaker має бути виконана за допомогою інструментів системи Kubernates, яка має вбудовані механізми моніторингу працездатності кожного запущеного мікросервісу. Патерн Event Sourcing має бути реалізований впровадженням журналів подій, а з точки зору архітектури і програмних засобів – інструментами, що застосовуються для трейсингу, такі як Zipkin та OpenTelemerty [32].

4. Імплементація інтелектуальних підходів у процеси проєктування та супроводу систем e-Banking на основі МСА. Для більш якісної обробки значного об’єму доменних знань мають бути застосовані інструменти AI-проєктування архітектури [80], а саме метод NLP (Natural Language Processing) [72, 81]. Він має автоматизувати процес виявлення інформаційних сутностей та їх зв’язків, що буде в подальшому корисно для доменного моделювання. Також вже існують різноманітні засоби автоматизованої побудови основних видів діаграм [82]. Такі засоби як ChatUML, PlantUML чи Eraser дозволяють досить зручно будувати різни типи UML-діаграм на основі введеного користувачем тексту. Іншим напрямком імплементації інтелектуальних підходів є AI-розгортання та супровід компонентів МСА, що спрямовані на адаптацію роботи вже розгорнутих систем і забезпечення їх стабільного функціонування. Одним з методів цього класу є метод CBR, доцільність використання якого для адаптивного управління конфігураціями програмних мікросервісів була доведена у роботі [74].

У підсумку, з урахуванням визначених вимог, принципів та підходів в процесі дослідження було розроблено архітектуру та програмно реалізовано

прототип фрагменту системи e-Banking, який обмежений доменом «формування виписок за рахунком». При цьому запропонована архітектура ґрунтується на принципах декомпозиції системи за бізнес-доменами з виділенням окремих сервісів, що мають чітко визначені межі відповідальності та власні життєві цикли функціонування. Функціональні вимоги для цього домену були визначені у розділі 3.2, у якому запропоновано вдосконалений алгоритм побудови виписок за рахунками користувачів. Передбачається використання модуля ІМП, який реалізує застосування знання-орієнтованих методів, для забезпечення належного рівня для показників якості систем e-Banking, шляхом покращення двох атрибутів якості: продуктивності і надійності. Основні компоненти задіяні у роботі ІМП у складі системи e-Banking показані на рисунку 3.9 у вигляді UML-діаграми пакетів [32, 46].

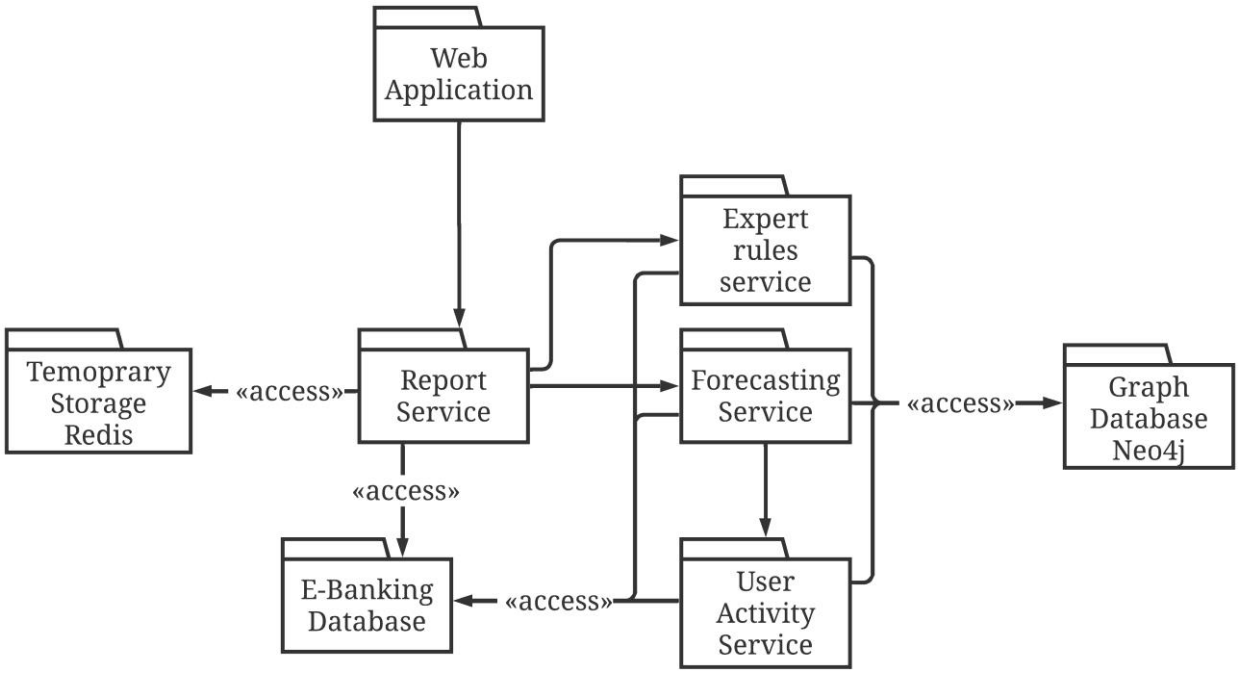

Рис. 3.9. Діаграма пакетів складових сервісів ІМП у складі e-Banking

Пропоноване рішення складається з таких мікросервісів, які задіяні у пропонованій бізнес-логіці ІМП:

1. Report Service – мікросервіс відповідає за задачу підготовки даних для виписки та безпосередньо формування файлу виписки, а також роботу з кешуванням та перевикористанням даних.
2. Forecasting Service – мікросервіс відповідає за безпосередньо прогнозування параметрів виписки з використанням знання-орієнтованого підходу.
3. Expert Rules Service – мікросервіс відповідає за уточнення сформованих прогнозів за допомогою експертних доменних правил, а також за введення, обробку і зберігання списку правил.
4. User Activity Service – відповідає за зберігання та обробку історії запитів користувача під час формування виписок за рахунками.

На рисунку 3.10 показано розроблену UML-діаграму розгортання [30] пропонованої МСА, яка відображає фізичну архітектуру вказаного фрагменту системи e-Banking. Такий підхід до структуризації архітектури дозволяє забезпечити ізоляцію функціональних компонентів, підвищити їх автономність та створити умови для незалежного масштабування окремих сервісів залежно від навантаження. На діаграмі показано поділ системи на мікросервіси. Представлено як мікросервіси безпосередньо для домену «формування виписок за рахунками», так і деякі службові сервіси, які так чи інакше задіяні у процесах домену. Також діаграма містить мікросервіси, які призначені для імплементації визначених вище архітектурних патернів. Для кожного мікросервіса визначено протокол комунікації, на основі визначених раніше принципів міжсервісної взаємодії [32].

У межах діаграми виділено логічну Secure Zone, що визначає довірене середовище виконання, у якому розгортаються основні сервіси системи e-Banking. Такий підхід дозволяє ізолювати критично важливі компоненти, що працюють із фінансовими даними, та забезпечити необхідний рівень безпеки і контролю доступу. Основними елементами фізичної архітектури є контейнери Docker, які виступають як середовище розгортання окремих мікросервісів. Використання контейнеризації забезпечує ізоляцію компонентів, спрощує

процес розгортання та дозволяє реалізувати незалежне масштабування кожного сервісу. У межах кожного контейнера функціонують відповідні програмні компоненти, реалізовані із використанням фреймворків сімейства Spring Cloud, що забезпечують підтримку міжсервісної взаємодії, конфігурації та управління сервісами.

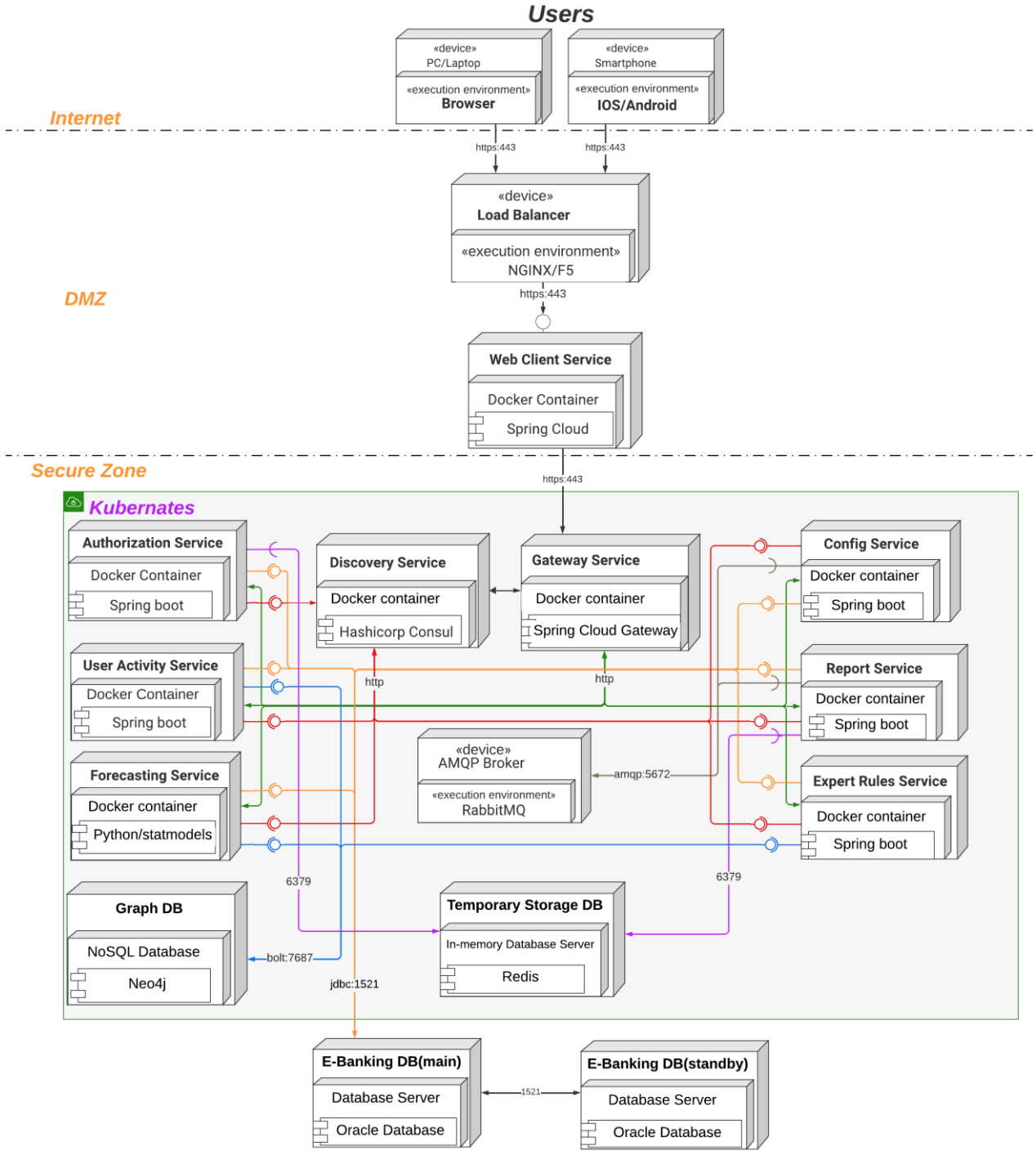

Рис. 3.10. Діаграма розгортання системи e-Banking на основі MCA

Пропоновані сервіси/компоненти мають таке призначення:

1. Load Balancer. Балансувальник навантаження забезпечує розподіл вхідних HTTPS-запитів між доступними екземплярами сервісу веб-клієнта. Він виступає першим рівнем обробки запитів, підвищуючи доступність системи та забезпечуючи відмовостійкість. Також він виконує функції термінації TLS-з'єднань.
2. Web Client Service. Даний сервіс реалізує клієнтську частину веб-застосунку та забезпечує взаємодію користувача із системою через браузер/мобільний додаток. Він виступає проміжною ланкою між фронтендом і серверною частиною МСА. Усі запити користувача передаються далі до внутрішніх сервісів через Gateway.
3. Gateway Service. Є центральною точкою входу до внутрішніх мікросервісів у межах Secure Zone. Він відповідає за маршрутизацію запитів, їх агрегацію та передачу до відповідних сервісів. Це дозволяє приховати внутрішню структуру системи та централізувати обробку запитів.
4. Discovery Service. Виконує роль динамічного реєстру, який дозволяє сервісам автоматично знаходити адреси один одного в середовищі Kubernetes, що є критично важливим для масштабованості системи.
5. Config Service. Забезпечує централізоване управління конфігураціями мікросервісів. Він дозволяє змінювати параметри роботи сервісів без їх повторного розгортання. Це спрощує супровід системи та забезпечує узгодженість налаштувань.
6. Authorization Service. Відповідає за безпеку та контроль доступу, здійснюючи аутентифікацію користувачів і видачу токенів для перевірки прав доступу при кожній взаємодії з бізнес-логікою.
7. AMQP Broker. Брокер повідомлень забезпечує асинхронну взаємодію між мікросервісами. Він обміну подіями та нотифікаціями між сервісами.

8. Graph DB. Графова база даних використовується для зберігання знання-орієнтованих структур, зокрема онтологічних моделей. Вона дозволяє ефективно працювати зі зв’язками між сутностями. Це є важливим для реалізації знання-орієнтованого підходу в системі.
9. Temporary Storage DB. Тимчасове сховище використовується для кешування даних і зберігання проміжних результатів обробки. Це дозволяє зменшити затримки при доступі до часто використовуваної інформації. Також використовується для обміну даними між сервісами.
10. E-Banking DB. Основна база даних забезпечує зберігання критично важливої фінансової інформації системи. Використання primary/standby конфігурації дозволяє забезпечити відмовостійкість і резервування даних. Це є необхідною умовою для систем класу e-Banking.

Призначення сервісів Report Service, Forecasting Service, Expert Rules Service та User Activity Service було описано раніше.

Отже, у даному підрозділі було виконано аналіз вимог до програмної архітектури системи e-Banking із урахуванням інтеграції ІМП, обґрунтовано доцільність використання MCA як базового архітектурного підходу та сформульовано принципи її побудови. На основі цього було розроблено архітектуру відповідного фрагменту системи, що охоплює домен «формування виписок за рахунками», а також визначено склад і взаємодію її основних компонентів. Запропонована архітектура забезпечує необхідний рівень масштабованості, надійності та продуктивності, а також створює основу для подальшої програмної реалізації та експериментального дослідження ефективності ІМП у складі системи e-Banking, що розглядається у наступному розділі.

3.5 Висновки до розділу 3

У даному розділі було розроблено моделі, бізнес-логіку та архітектуру перспективної системи e-Banking, яка розширена шляхом інтеграції ІМП, що має на меті підвищити ефективність функціонування системи в цілому.

У підрозділі 3.1 було побудовано доменну модель системи e-Banking для домену “побудова виписок за рахунками”. Визначено основні інформаційні сутності з їх атрибутами та зв’язки між сутностями. Запропонована доменна модель є знання-орієнтованою та відрізняється від існуючих побудовою та застосуванням онтологічних профілів користувачів систем e-Banking, у поєднанні з експертними доменними правилами для обробки їх емпіричних знань, що дозволяє розширити інформаційний базис і враховувати семантичний контекст вимог до функціоналу таких систем.

У підрозділі 3.2 розроблено бізнес-логіку ІМП. Основна ідея полягає в тому, щоб система прогнозувала майбутні запити користувачів з побудови виписок за рахунками та заздалегідь розраховувала і підготовлювала дані, що можуть потребувати значні системні обчислювальні ресурси. В якості основного математичного методу для прогнозування використано метод аналізу часових рядів SARIMA. Бізнес-логіка ІМП представлена у вигляді набору пов’язаних IDEF0-діаграм.

У підрозділі 3.3 було визначено систему критеріїв та метрик для оцінки якості функціонування ІМП та пропонованого підходу в цілому. Пропоновані критерії враховують як суто технічні показники, так і переваги для кінцевих користувачів системи e-Banking та/або банку. Визначені критерії та метрики будуть використані в процесі експериментального дослідження та аналізу результатів якості роботи прототипу системи, що буде виконано у розділі 4.

У підрозділі 3.4 було виконано проєктування МСА для реалізації пропонованого підходу у системі e-Banking, зокрема імплементацію ІМП. Архітектура наведена у вигляді UML-діаграми розгортання компонентів. Провідну роль у проєктуванні МСА відіграє доменне моделювання, яке використовується для визначення меж сервісів та формалізації бізнес-логіки предметної області. Було визначено сервіси, на які має бути поділена система, а також протоколи для їх взаємодії. Для імплементації пропонованої МСА було розроблено концептуальні принципи її проєктування та визначено перелік

архітектурних патернів, придатних для інтеграції у системи e-Banking та які мають дозволити досягти кращих показників якості ПЗ.

РОЗДІЛ 4
ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ ЗАСТОСУВАННЯ ЗАПРОПОНОВАНОГО ПІДХОДУ

4.1 Мотивація вибору стеку технологій для реалізації мікросервісної архітектури у системі e-Banking

Реалізація запропонованої у попередньому розділі МСА з інтегрованим ІМП потребує обґрунтованого вибору стеку технологій, який забезпечить можливість ефективного впровадження розроблених моделей і алгоритмів у програмний продукт. Враховуючи специфіку систем e-Banking, що характеризуються високими вимогами до продуктивності, надійності, масштабованості та безпеки, обрані технологічні рішення повинні не лише відповідати сучасним підходам до побудови розподілених систем, але й забезпечувати підтримку як синхронної, так і асинхронної взаємодії між компонентами. Окрему увагу при цьому приділено можливості інтеграції ІМП, який базується на методах аналізу часових рядів та знання-орієнтованих підходах, що зумовлює необхідність використання технологічного стеку, що поєднує різні платформи та мови програмування. Вибір конкретних інструментів та платформ визначався вимогами до архітектури системи, особливостями реалізації ІМП, а також необхідністю забезпечення ефективної взаємодії між її компонентами в умовах реального функціонування.

Вибір технологічного стеку для реалізації МСА з інтегрованим у неї ІМП ґрунтується на множині принципів, що визначають якість функціонування системи в умовах реального навантаження у продуктовому середовищі банку. В першу чергу, обрані технології повинні забезпечувати модульність і слабку зв’язаність компонентів, що є концептуальною вимогою МСА та дозволяє забезпечити незалежний життєвий цикл для сервісів. Важливим критерієм також є підтримка як синхронних, так і асинхронних протоколів взаємодії, що забезпечує можливість оптимального розподілу навантаження та ресурсів між

критичними та фоновими процесами [83]. Окрему увагу приділено здатності технологій ефективно працювати з різними типами даних, зокрема платіжними транзакціями, накопиченими часовими рядами та знання-орієнтованими структурами даних. Має також бути враховано можливість інтеграції компонентів, реалізованих на різних платформах, а також підтримку масштабування і відмовостійкості системи без суттєвого ускладнення її архітектури.

Основу реалізації мікросервісної частини системи становить використання мови програмування Java та екосистеми фреймворків Spring [84], зокрема Spring Boot і Spring Cloud. Було використано JDK 21 та Spring 7, що є останніми стабільними версіями з довготривалою підтримкою від вендора (LTS version). Вибір цієї технологічної платформи обумовлений її зрілістю, широким поширенням у промисловій розробці та наявністю розвиненого інструментарію для побудови розподілених систем [85]. Застосування Spring Boot дозволяє значно спростити процес створення окремих мікросервісів за рахунок використання принципу “convention over configuration”, що забезпечує швидке розгортання застосунків із мінімальними витратами на початкову конфігурацію [86, 87]. Крім того, Spring Boot надає вбудовану підтримку створення REST-сервісів, роботи з базами даних, обробки транзакцій до БД та інтеграції з іншими програмними компонентами системи, що є критично важливим для систем класу e-Banking. Використання Spring Cloud, у свою чергу, забезпечує реалізацію ключових інфраструктурних механізмів та патернів MCA за рахунок інтеграції з низкою спеціалізованих компонентів. Зокрема, для реалізації задачі пошуку мікросервісів пропонується модуль Spring Cloud Service Discovery, який виконує функції реєстрації мікросервісів та їх динамічного виявлення іншими компонентами системи. Це дозволяє сервісам взаємодіяти між собою без жорстко заданих адрес, що є критично важливим у середовищах із динамічним масштабуванням. При чому, підтримується як робота с власним сервісом Netflix Eureka, так і інтеграція з зовнішніми системами, такими як HashiCorp Consul або Apache Zookeeper. Для використання в цьому дослідженні

було обрано HashiCorp Consul наявність безпекових механізмів контролю доступу, як наприклад, ACL-токени, налаштування яких є відносно легкою задачею. Для централізованого управління конфігураціями застосовується модуль Config Server, який забезпечує зберігання параметрів налаштування у зовнішніх репозиторіях у різних форматах та їх динамічне завантаження мікросервісами під час роботи. Це спрощує супровід системи та дозволяє змінювати конфігурацію без повторного розгортання сервісів. Маршрутизація запитів і організація єдиної точки входу до системи реалізується за допомогою модулю Spring Cloud Gateway, який повністю реалізує патерн API Gateway. Він забезпечує прийом зовнішніх запитів, їх маршрутизацію до відповідних мікросервісів, а також може виконувати додаткові функції, такі як фільтрація, логування та базовий контроль доступу до ресурсів та endpoint-ів конкретних мікросервісів. Балансування навантаження між екземплярами мікросервісів забезпечується за рахунок вбудованих механізмів клієнтського балансування, що дозволяє рівномірно розподіляти запити та підвищувати відмовостійкість системи [88, 89].

Для реалізації клієнтської частини системи використано фреймворк Angular 20 (LTS версія), який забезпечує побудову сучасного веб-інтерфейсу та ефективну взаємодію користувача із системою через браузер. Використання Angular дозволяє реалізувати компонентний підхід до розробки інтерфейсу, що прямо відповідає модульній концепції МСА. Взаємодія між клієнтською та серверною частинами здійснюється із використанням REST-протоколу з передачею даних у форматі JSON, що є стандартним підходом для побудови розподілених веб-систем.

Окремим компонентом розробленої МСА є мікросервіс прогнозування (Forecasting Service), який реалізує функціонал ІМП та побудований із використанням мови програмування Python. Вибір Python обумовлений наявністю розвинених бібліотек для аналізу часових рядів, зокрема бібліотеки statsmodels [91], яка використовується в системі для роботи з моделями згідно методу SARIMA. Застосування цього інструментарію дозволяє виконувати

обробку історичних даних про дії користувачів, будувати знання-орієнтовані моделі та визначати найбільш ймовірні моменти формування виписок. Винесення ІМП в окремий мікросервіс забезпечує його незалежність від інших компонентів системи, що дозволяє масштабувати обчислювальні ресурси окремо від основної бізнес-логіки та спрощує модифікацію або заміну алгоритмів прогнозування без впливу на інші сервіси. Для зберігання та обробки знання-орієнтованих структур у складі ІМП використовується графова база даних Neo4j [92], яка забезпечує ефективне представлення онтологічних моделей домену. Використання графової моделі даних дозволяє явно задавати зв’язки між сутностями, що є необхідним для реалізації онтологічних профілів користувачів та врахування контексту їх поведінки. Завдяки використанню протоколу Bolt забезпечується ефективна взаємодія між мікросервісом прогнозування та графовою базою даних, що дозволяє з мінімальними витратами отримувати та обробляти знання, необхідні для прогнозування. Додатково, у складі системи передбачено мікросервіс експертних доменних правил (Expert Rules Service), який реалізує обробку доменних знань у вигляді формалізованих правил. Для цього використовується фреймворк Flowable, який надає можливість моделювання та виконання бізнес-процесів і правил за допомогою нотації BPMN та механізмів її виконання. Застосування Flowable дозволяє реалізувати гнучкий механізм управління правилами без необхідності жорсткого кодування бізнес-логіки, що спрощує їх модифікацію та розширення. Мікросервіс експертних правил взаємодіє з ІМП, доповнюючи результати прогнозування з урахуванням доменних обмежень і експертних знань, що в сукупності забезпечує підвищення якості прийняття рішень у процесі формування виписок.

Взаємодія між мікросервісами у складі системи реалізується із використанням поєднання синхронних та асинхронних протоколів комунікації, що відповідає вимогам до різних типів операцій у системах e-Banking. Для реалізації синхронної взаємодії використовується REST-протокол із передачею даних у форматі JSON, що забезпечує простоту інтеграції між сервісами та

швидке отримання результату для критичних операцій. Такий підхід застосовується, зокрема, для запитів, які потребують негайної відповіді, включаючи отримання даних користувача або ініціювання побудови виписки. Для реалізації асинхронної взаємодії використовується протокол AMQP із застосуванням брокера повідомлень RabbitMQ [92], що дозволяє організувати обмін повідомленнями між мікросервісами у фоновому режимі. Це особливо доцільно для задач, пов’язаних із обробкою великих обсягів даних або виконанням тривалих операцій, таких як попередня підготовка даних для виписки чи обробка результатів прогнозування. Поєднання синхронних і асинхронних механізмів дозволяє оптимізувати навантаження на систему, зменшити затримки та підвищити надійність системи.

Для забезпечення зберігання та обробки даних у системі використовується поєднання різних типів баз даних, що ефективно зберігати різнорідні дані. Основною реляційною базою даних є Oracle, яка використовується для зберігання транзакційних даних, інформації про рахунки користувачів та інших критично важливих сутностей. Використання Oracle обумовлене її високою надійністю, підтримкою транзакційності та забезпеченням цілісності даних, що є ключовими вимогами для систем e-Banking. Доступ до реляційної бази даних реалізується через протокол JDBC, що забезпечує стандартний механізм взаємодії з даними у середовищі Java. Окрім цього, у системі використовується in-memory база даних Redis [93], яка призначена для тимчасового зберігання проміжних даних, кешування результатів обчислень та зменшення навантаження на основну базу даних. Застосування Redis дозволяє суттєво підвищити продуктивність системи за рахунок швидкого доступу до часто використовуваних даних [94]. Разом із графовою базою даних Neo4j, що використовується у складі ІМП, така комбінація сховищ забезпечує гнучке та ефективне управління як транзакційними, так і знання-орієнтованими даними у системі.

Розгортання мікросервісів у складі системи реалізується із використанням технологій контейнеризації та оркестрації, зокрема Docker і Kubernetes. Кожен

мікросервіс упаковується у вигляді окремого Docker-контейнера, що забезпечує ізоляцію середовища виконання, незалежність від конкретної інфраструктури та відтворюваність процесу розгортання. Такий підхід дозволяє уникнути проблем сумісності між різними середовищами та забезпечує уніфікований механізм доставки програмних компонентів. Оркестрація контейнерів з мікросервісами здійснюється за допомогою системи Kubernetes, який забезпечує автоматичне розгортання, масштабування та управління життєвим циклом мікросервісів у кластері. Зокрема, Kubernetes дозволяє реалізувати механізми автоматичного масштабування залежно від навантаження, балансування запитів між екземплярами сервісів, а також відновлення їх працездатності у випадку збоїв. Використання цих технологій забезпечує високу гнучкість інфраструктури, ефективне використання обчислювальних ресурсів та підвищення показників надійності, продуктивності та масшабованості системи, що є особливо важливим для систем e-Banking, які характеризуються нерівномірним та динамічним навантаженням [95].

Важливою вимогою до сучасних складних систем є забезпечення спостережуваності (observability) та контролю за станом системи в процесі її функціонування в режимі реального часу. Для цього у розробленій системі використовується поєднання інструментів Prometheus, Grafana та Zipkin, які забезпечують збір, зберігання та візуалізацію метрик, а також трасування запитів між мікросервісами. Prometheus виконує функцію збору та агрегації метрик із різних компонентів системи, тоді як Grafana забезпечує їх візуалізацію у вигляді дашбордів, що дозволяє оперативно аналізувати стан системи та виявляти потенційні проблеми. Для відстеження проходження запитів через різні мікросервіси використовується Zipkin, який реалізує механізми розподіленого трасування. Додатково, екосистема Spring надає вбудовану підтримку інтеграції з інструментами трасування, що дозволяє автоматично додавати ідентифікатори (trace-id) до запитів та відслідковувати їх проходження через усі компоненти системи. Це забезпечує можливість якісного аналізу продуктивності, виявлення вузьких місць та швидкого реагування на

збої, що є необхідною умовою для забезпечення надійності та безперервності функціонування системи [70].

4.2 Особливості програмної реалізації прототипу системи

У даному підрозділі розглянуто особливості програмної реалізації запропонованого підходу. У межах цього дослідження було обрано фрагмент предметної області, який обмежений доменом формування виписок за рахунками клієнтів. Такий підхід дозволяє, з одного боку, зосередитися на детальному дослідженні запропонованих методів і моделей, а з іншого – забезпечити актуальність та повноту отриманих результатів експериментального дослідження.

Реалізація запропонованої у розділі 3.4 МСА передбачає розподіл функціональності між окремими мікросервісами, кожен з яких відповідає за виконання певної частини бізнес-логіки. Відповідно, у системі було реалізовано набір сервісів, що забезпечують обробку запитів користувачів, формування виписок, виконання прогнозування та обробку експертних правил. Взаємодія між мікросервісами організована з використанням як синхронних, так і асинхронних механізмів, що дозволяє ефективно обробляти запити різного типу та забезпечувати необхідний рівень продуктивності. Для синхронної взаємодії було використано можливості модулю Spring Cloud OpenFeign, який дозволяє з описаного інтерфейсу виконати автогенерацію коду для викликів між сервісами. Нижче наведено приклад вихідного коду для з описом інтерфейсу для збереження виписки мікросервісом Report Service.

```
@FeignClient(value="gateway-service", contextId="reportServiceId,
          configuration = ReportServiceFeignConfiguration.class)
public interface ReportClient {
    @PostMapping(value = "/report-service/api/reports/v1.0/save",
                 consumes = MediaType.APPLICATION_JSON_VALUE,
                 produces = MediaType.APPLICATION_JSON_VALUE)
    ReportResponseDto saveReport(SaveReportRequest request);
```

```
}
```

Для асинхронної взаємодії між сервісами, яка згідно бізнес-логіки передбачається при виконанні кроків алгоритму, які передбачають створення системних задач, було використано брокер повідомлень RabbitMQ, який розгорнуто також у Docker-контейнері під управлінням Kubernates. При чому, розгортання сервісу не потребувало додаткових кастомних налаштувань окрім обов’язкових для запуску. Для взаємодії мікросервісів з брокером повідомлень було використано можливості модулів Spring AMQP, Spring Cloud Stream та Spring Cloud Functions, які дозволяють засобами функціонального програмування реалізувати механізми запису та вичитування записів у підтримувані брокери повідомлень. Конфігурація взаємодії із брокером здійснюється у файлах налаштувань мікросервісів (application.yml), де визначаються параметри підключення, імена черг, exchange та правила маршрутизації повідомлень. Нижче наведено приклад конфігурації відповідних каналів у брокері повідомлень RabbitMQ для задачі попередньої підготовки даних для запитуваної виписки:

```
spring:
  rabbitmq:
    host: ${app.rabbit.host:localhost}
    port: ${app.rabbit.port:5672}
    username: ${app.rabbit.username:guest}
    password: ${app.rabbit.password:guest}
  cloud:
    stream:
      bindings:
        input_prepare_data_action_channel:
          destination: prepare_data_action_channel
          group: groupConsumers
          consumer:
            concurrency: ${prepare.data.concurrency:1}
            autoBindDlq: true
```

```
        content-type: application/json
      output_prepare_data_action_channel:
        destination: prepare_data_action_channel
        group: groupConsumers
        consumer:
          autoBindDlq: true
        content-type: application/json
```

Одним з ключових елементів реалізації МСА є мікросервіс Gateway Service, який виконує функцію єдиної точки входу до системи та забезпечує маршрутизацію запитів до відповідних мікросервісів. У розробленому прототипі цей функціонал реалізовано із використанням Spring Cloud Gateway. Приклад конфігурації маршрутизації наведено у вигляді фрагмента коду, який демонструє визначення маршрутів на основі URI запитів та їх перенаправлення до відповідних сервісів.

```
spring:
   cloud:
     gateway:
       routes:
        - id: auth-service
          uri: lb://auth-service/
          predicates:
            - Path=/auth-service/**
          filters:
            - StripPrefix=1

        - id: report-service
          uri: lb://report-service/
          predicates:
            - Path=/report-service/**
          filters:
            - StripPrefix=1
```

```
- id: forecasting-service
  uri: lb://forecasting-service/
  predicates:
    - Path=/forecasting-service/**
  filters:
    - StripPrefix=1
```

Розгортання мікросервісів системи здійснюється у кластерному середовищі під управлінням Kubernetes, що забезпечує централізоване управління контейнерами, контроль їх стану та автоматизацію процесів масштабування. Кожен мікросервіс розгортається у вигляді окремого Docker-контейнера, який інкапсулює в себе всі необхідні залежності середи його виконання. Важливою особливістю конфігурації є обов’язкове забезпечення мінімального рівня відмовостійкості, що досягається шляхом підтримки не менше двох одночасно працюючих екземплярів (нод) для кожного критично важливого сервісу. Такий підхід дозволяє гарантувати безперервність обслуговування запитів користувачів навіть у випадку відмови одного з екземплярів, а також забезпечує можливість балансування навантаження між ними. Додатково, для контролю працездатності мікросервісів використовуються механізми readiness та liveness probes, які дозволяють Kubernetes автоматично визначати стан сервісів та виконувати їх перезапуск у разі виявлення збоїв або некоректної роботи. При чому, реалізоване рішення дозволяє банку додаткову відмовостійкість шляхом розділення кластеру між різними датацентрами (ЦОД) з використанням додаткового апаратного балансування. Моніторинг стану системи та навантаження на окремі мікросервіси реалізується за рахунок вбудованих механізмів Kubernetes у поєднанні з інструментами збору метрик: Prometheus та Grafana. На основі сукупності зібраних даних реалізується механізм динамічного масштабування, а саме горизонтального масштабування, який дозволяє автоматично змінювати кількість екземплярів мікросервісів залежно від поточного рівня навантаження,

наприклад, використання ЦП або кількості оброблюваних запитів. При цьому важливою особливістю є можливість гнучкого налаштування критеріїв масштабування відповідно до внутрішніх вимог та стандартів конкретного банку. Зокрема, банки можуть самостійно визначати порогові значення навантаження, політики масштабування та обмеження на кількість екземплярів сервісів, що дозволяє адаптувати поведінку системи до специфіки бізнес-процесів та вимог до продуктивності. У періоди пікового навантаження, що характерні для систем e-Banking, кількість екземплярів окремих сервісів може бути збільшена, тоді як у періоди зниження активності – зменшена для оптимізації використання обчислювальних ресурсів. Це дає можливість системі адаптуватись до змін навантаження, підвищує її якість функціонування та дозволяє досягти необхідного рівня продуктивності без надлишкового використання обчислювальних ресурсів.

Таким чином, у даному підрозділі було розглянуто особливості імплементації запропонованого підходу у складі реальної системи e-Banking, що функціонує на основі МСА. Реалізований прототип охоплює ключові компоненти, необхідні для забезпечення взаємодії мікросервісів, обробки запитів користувачів, виконання прогнозування за допомогою ІМП та застосування експертних доменних правил. Використання сучасного технологічного стеку, механізмів контейнеризації та оркестрації, а також інструментів асинхронної взаємодії дозволило створити гнучке та масштабоване середовище виконання, яке відповідає вимогам до продуктивності, надійності та масштабованості систем e-Banking. Отриманий програмний прототип дає можливість провести експериментальне дослідження із використанням визначених у попередніх підрозділах критеріїв та метрик. Проведення експериментальних досліджень дозволяє перейти до кількісної оцінки показників ефективності запропонованих у дослідженні рішень, результати якої наведено у наступному підрозділі.

4.3 Отримані результати імплементації запропонованої моделі та архітектури

В даному підрозділі наведено результати експериментального дослідження реалізованого прототипу у складі МСА системи e-Banking. Основна увага зосереджена на кількісній оцінці показників якості функціонування системи, отриманих у процесі її тестування за різних сценаріїв навантаження та режимів роботи. Представлені результати дозволяють проаналізувати ефективність запропонованих рішень, а також визначити їх вплив на ключові характеристики системи, зокрема продуктивність, надійність та ефективність використання ресурсів. Невід'ємною частиною розробки є верифікація створюваних прототипів. Для експериментального дослідження та оцінки їх якості існують різні методи тестування ПЗ. Серед них можна виділити такі основні:

- верифікація працездатності прототипу за допомогою unit- та модульного тестування з використанням бібліотек JUnit та Mockito [96];
- навантажувальне тестування: яка кількість ресурсів задіяна при різних профілях навантаження, який час відгуку системи, який відсоток помилок, який відсоток критичного завантаження ЦП;
- тестування на відмовостійкість: середній час відновлення після збоїв, середній час між відмовами.

Метою експерименту було оцінити вплив запропонованої архітектури на ключові атрибути якості ПЗ, такі як продуктивність, масштабованість та надійність, і порівняти їх з аналогічними показниками для цього ж функціоналу у монолітній архітектурі. Тестування виконувалось в межах функціоналу, реалізованого у пропонованому прототипі. Для проведення експериментів було використано тестове середовище української IT-компанії CS [44]. Для збору основних метрик було виконано 20 тестових запусків з емуляцією клієнтського навантаження на систему у 10 паралельних запитів в секунду протягом 2х годин. Для емуляції клієнтського навантаження було використано ПЗ JMeter [97]. При цьому виконувалось формування виписок різного об'єму. Вимірювались такі показники:

1) час виконання запиту;
2) пропускна здатність;
3) завантаження ЦП Report Service
4) відсоток запитів, які завершились помилкою;
5) Відсоток перевикористання раніше підготовлених даних

У табл. 4.1 наведено порівняння середнього часу формування виписки (T_{AVG}) для поточного типового алгоритму та запропонованого мікросервісного рішення. Поточний алгоритм є синхронним і реалізований у системі e-Banking, яка побудована на основі МА. У пропонованому мікросервісному рішенні передбачено повністю асинхронний режим формування виписок, який додатково ще розділений на підзадачі. Загальний час формування виписки для запропонованого рішення рахувався як сума часу виконання кожної з підзадач.

Таблиця 4.1. Час формування виписки під навантаженням

Кількість сторінок	Час для поточного алгоритму (мс)	Час для пропонованого алгоритму (мс)	
		Без урахування очікування у черзі задач	З урахуванням очікування у черзі задач
1	212	134	169
5	790	255	570
10	1410	400	995
20	3750	640	2880
50	8100	1520	5100
100	16150	3100	7160
150	23800	4650	11900
500	80500	18950	41700
850	121500	41500	76100

Розрахунок показника пропускної здатності за одиницю часу виконується як відношення кількості паралельних сесій до часу відгуку. У табл. 4.2 показано приріст відповідного показника. Слід також зазначити, що при зниженні

кількості паралельних запитів до системи до 1 було помічено, що пропускна здатність майже не відрізняється.

Таблиця 4.2 Пропускна здатність формування виписки під навантаженням

Кількість сторінок	Пропускна здатність для рішення на основі моноліту (request/second)	Пропускна здатність для рішення на основі мікросервісів (request/second)	Приріст пропускної здатності рішення на основі мікросервісів
1	47.17	59.17	25.4%
5	12.66	17.54	38.5%
10	7.09	10.05	41.7%
20	2.67	3.47	30.0%
50	1.23	1.96	59.3%
100	0.62	1.40	125.0%
150	0.42	0.84	100.0%
500	0.12	0.24	100.0%
850	0.08	0.13	62.5%

Тестування на відмовостійкість показало, що загальний час, який потрібен на відновлення працездатності сервісу, який працює під управлінням Kubernates, не перевищує 2х хвилин. При чому, понад 1 хвилину займали внутрішні задачі Java та Spring, які виконуються для ініціалізації JRE та контексту веб-застосунку. За умови якщо для вирішення збою не потрібна участь людини, то сервіс відновлює працездатність в межах цього часу. У табл. 4.3 показано відсоток запитів, які завершились помилкою (RE).

Таблиця 4.3. Відсоток запитів, що завершились помилкою

Кількість сторінок	Відсоток помилок для рішення на основі моноліту	Відсоток помилок для рішення на основі мікросервісів
1	0.01%	0.01%

5	0.02%	0.02%
10	0.08%	0.03%
20	0.15%	0.04%
50	0.3%	0.07%
100	0.5%	0.11%
150	0.8%	0.17%
500	2%	0.39%
850	5%	0.83%

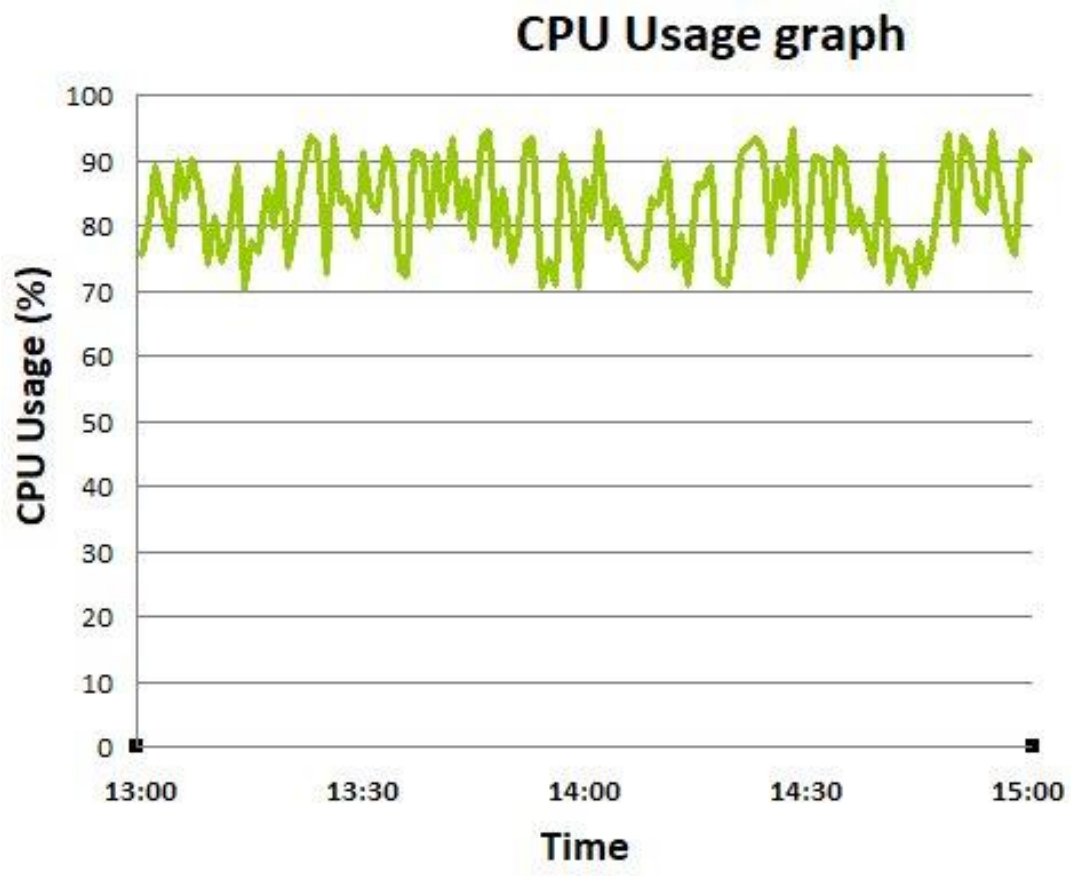

Рис. 4.1 Графік використання ЦП для поточного алгоритму з монолітною архітектурою

Під час проведення експерименту з навантажувальним тестуванням було також отримано та зібрано метрики завантаження ЦП (CPU – Central processing unit), що зображено на рис. 4.1 та 4.2. Для отримання метрик було використано систему Prometheus, а для збору, зберігання та візуалізації – систему Grafana.

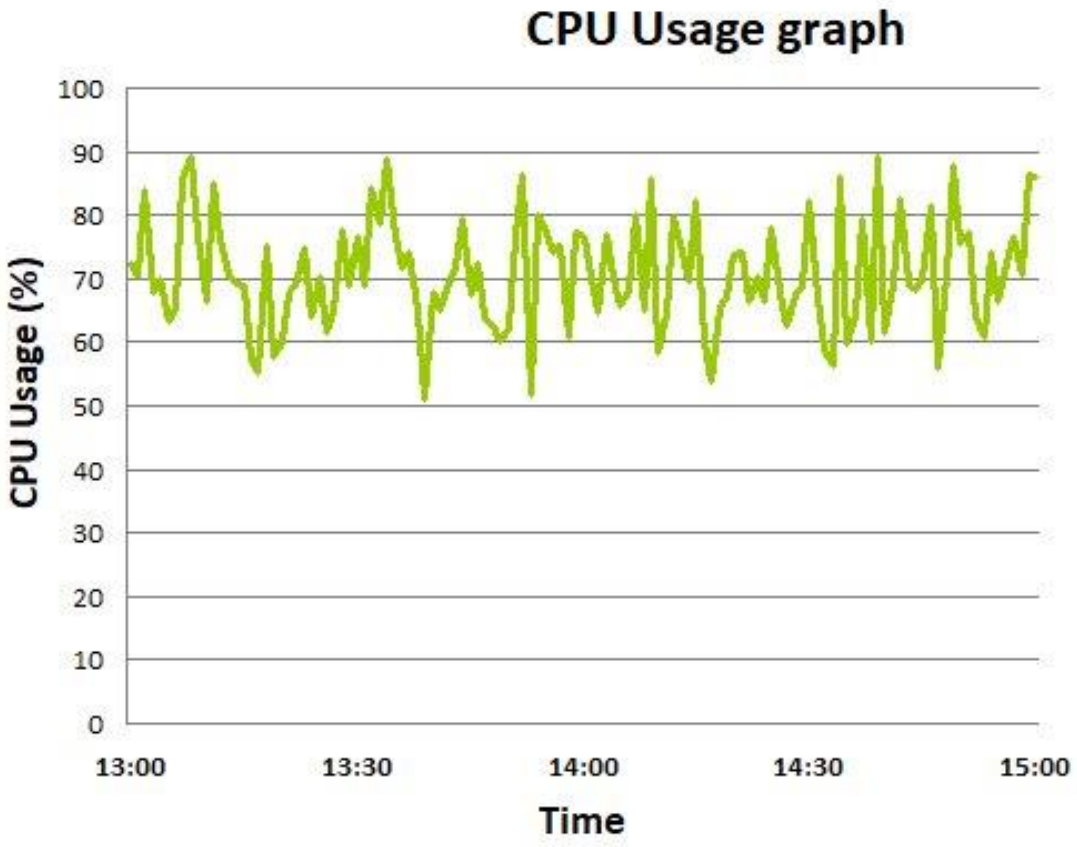

Рис. 4.2 Графік використання ЦП для пропонованого алгоритму з мікросервісною архітектурою

У табл. 4.4 наведено результати оцінки ефективності етапу попередньої підготовки даних для виписок. Показники перевикористання раніше підготовлених даних (DR) було отримано на основі аналізу зібраної статистики побудови виписок за період з квітня по грудень 2025 року у реальному банківському середовищі, зокрема щодо часу формування та параметрів запитів користувачів. На основі цих знеособлених даних було виконано емуляцію у тестовому середовищі сценарію попереднього збереження підготовлених результатів з оцінкою їх можливого перевикористання.

Таблиця 4.4. Оцінка ефективності попередньої підготовки даних

Об'єм перевикористаних даних	Відсоток перевикористання
Повне перевикористання даних (100% даних)	3-4%

Часткове перевикористання більшості даних (60-90% даних)	15-20%
Часткове перевикористання даних (30-50% даних)	35-40%

Окремо варто зазначити, що було проведено оцінку ефективності запропонованого варіанту декомпозиції системи на мікросервіси, результати якої наведені у табл. 4.5. Для цього було запропоновано різні варіанти поділу розроблюваного прототипу системи, межі домену якого було окреслені у попередніх розділах. Серед запропонованих варіантів було 3 альтернативи, які передбачали поділ на такі мікросервіси:

1. 2 мікросервіса:
    a. authgateway-service
    b. statement-service
2. 7 мікросервісів:
    a. report-service
    b. forecasting-service
    c. expert-rules-service
    d. user-activity-service
    e. auth-service
    f. config-service
    g. apigateway-service
3. 10 мікросервісів:
    a. report-service
    b. payments-service
    c. timer-task-service
    d. forecasting-service
    e. sarima-service
    f. expert-rules-service
    g. user-activity-service

h. auth-service
i. config-service
j. apigateway-service

Таблиця 4.5. Оцінка ефективності декомпозиції на мікросервіси

Пропонований поділ на мікросервіси	OLC	TRC	NC	DD
Варіант №1: 2 мікросервіса	9	2	1	3.7 (недостатня)
Варіант №2: 7 мікросервісів	4	3	3	0.5 (оптимальна)
Варіант №3: 10 мікросервісів	1	6	5	-2.3 (надмірна)

Для аналізу були вибрані такі вагові коефіцієнти: a = 0.5, b = 0.3, c = 0.2, що означає, що для даного класу систем найважливішим критерієм є складність підтримки спільної бізнес-логіки. Значення для показників OLC та TRC виставлялись на основі об'єму та "складності" (необхідної кваліфікації розробника, часу на реалізацію, тощо) реалізації необхідного програмного коду, а для показника NC на основі кількості потенційних мережевих запитів, необхідних при такій реалізації.

Отже, в даному розділі наведено результати експериментального дослідження розробленого прототипу системи e-Banking для домену “побудови виписок за рахунками”. Під час збору показників функціонування було також використано систему з переліком критеріїв та метрик, яка запропонована у розділі 3.3 для оцінки якості функціонування ПЗ. В наступному розділі буде виконано оцінку і аналіз отриманих показників та зроблено висновки щодо доцільності використання запропонованого у роботі підходу.

4.4 Оцінка і аналіз отриманих показників якості продукту при використанні мікросервісної архітектури та інтелектуального модуля прогнозування

У попередньому підрозділі було наведено результати експериментального дослідження прототипу ІМП у складі МСА системи e-Banking, а також визначено значення відповідних метрик якості функціонування системи. У даному підрозділі здійснюється їх узагальнення та інтерпретація з метою оцінки ефективності запропонованих архітектурних та алгоритмічних рішень. Особливу увагу приділено аналізу впливу застосування ІМП та МСА на показники продуктивності, надійності, ефективності використання ресурсів, а також обґрунтуванню доцільності обраної декомпозиції системи та механізмів попередньої підготовки даних.

Як видно з рис. 4.1 та 4.2 в середньому досягнуто зменшення навантаження на ЦП на 12-15%, а також суттєво зменшено час критичної завантаженості ЦП [69]. Це мінімізує ризик збоїв у роботі серверів та підвищує загальну надійність систем e-Banking в цілому. В свою чергу, дані наведені у табл. 4.1 та 4.2 показують позитивний вплив запропонованого МСА підходу на показники продуктивності та масштабованості системи e-Banking, а дані з табл. 4.3 разом із графіками на рис. 4.1 та 4.2, підтверджують підвищення показників надійності її функціонування.

Окрему увагу слід приділити результатам оцінки відсотка перевикористання попередньо підготовлених даних. Підхід, використаний для збору відповідної метрики, дозволив визначити, у якій частині випадків запити повністю або частково співпадають із прогнозованими. Отримані результати підтверджують наявність повторюваних поведінкових шаблонів серед клієнтів сегменту ФОП та юридичних осіб, що доводить доцільність використання механізму попередньої підготовки даних. Варто додатково зазначити, що при застосуванні для передпідготовки даних так званих Spot Instatnces, ресурсів які надають хмарні провайдери (AWS, Azure, тощо) за суттєво зниженою ціною у

періоди "простою" датацентрів, банки можуть отримувати й економічну вигоду від запропонованого підходу за рахунок зниження інфраструктурних витрат.

Аналіз результатів оцінки якості декомпозиції системи на мікросервіси, який наведено у табл. 4.5. показав, що для домену e-Banking критичним балансування між складністю підтримки бізнес-логіки та супутніми витратами на узгодження транзакцій і зайву мережеву взаємодію, що робить "наносервісний" підхід неприпустимим, як і концентрацію значної бізнес-логіки в одному сервісі. Було визначено, що надлишкова декомпозиція доменної області призводить до суттєвого зростання міжсервісної комунікації, що негативно впливає на продуктивність та надійність системи. В результаті проведеного дослідження було встановлено, що обраний поділ на 7 сервісів є математично обґрунтованим.

Створений прототип фрагменту e-Banking системи, архітектура якого наведена на рис. 3.9 і 3.10, а також результати його експериментальних досліджень демонструють, що пропонований підхід забезпечує суттєве покращення визначених цільових показників якості системи. Підхід полягає в застосуванні МСА в поєднанні з архітектурними патернами та інтелектуальними методами. Було зафіксовано приріст показника продуктивності (табл. 4.1), зниження затримки виконання операцій (табл. 4.2) та підвищення показника надійності (табл. 4.3, рис. 4.1 і 4.2). Це було досягнуто завдяки розподіленій обробці та імплементації асинхронного виконання частини операцій. Отримані результати підтверджують, що поєднання МСА та знання-орієнтованих методів є дієвим та якісним шляхом подолання раніше не досліджених проблем. Запропонований модельно-технологічний інструментарій (рис. 3.10), порівняно, наприклад, з запропонованим в [59], адаптовано під застосування у системі знання-орієнтованих моделей та інтелектуальних методів. Це дозволило якісніше імплементувати розроблену МСА і забезпечити відповідні показники якості ПЗ систем e-Banking. Крім того, отримали подальший розвиток принципи проєктування МСА для систем e-Banking шляхом формування колекції проектних патернів, які забезпечують

можливість забезпечення більш високих показників якості функціонування таких систем, із використанням множини кількісних метрик для їх оцінки.

4.5 Практичні рекомендації щодо використання запропонованого підходу

Застосування ІМП у складі МСА систем e-Banking є доцільним насамперед у тих сценаріях, що характеризуються наявністю повторюваних поведінкових шаблонів користувачів та регулярністю виконання операцій. До таких задач, зокрема, належить формування виписок за рахунками, де запити користувачів часто мають періодичний характер. У подібних умовах використання ІМП дозволяє здійснювати попередню підготовку даних та зменшувати затримки обробки запитів. Водночас ефективність запропонованого підходу суттєво залежить від наявності достатнього обсягу історичних даних, що дозволяють виявляти закономірності у поведінці користувачів, тому його застосування є менш доцільним для сценаріїв із нерегулярною або слабо прогнозованою активністю.

Важливим аспектом практичного впровадження є організація інфраструктури розгортання та забезпечення ефективного управління обчислювальними ресурсами. Використання контейнеризації та оркестрації за допомогою Kubernetes дозволяє реалізувати динамічне масштабування мікросервісів залежно від поточного навантаження. При цьому доцільним є налаштування мінімальної кількості вузлів кластера на рівні не менше двох, що забезпечує базову відмовостійкість системи. Банківські установи, з урахуванням власних політик та вимог до якості обслуговування, можуть визначати індивідуальні критерії масштабування, зокрема порогові значення навантаження на ЦП, пам’ять або інші показники функціонування системи, при досягненні яких ініціюється додавання нових вузлів або екземплярів сервісів. Додатково, показники, отримані для визначених у роботі критеріїв та метрик якості функціонування системи, доцільно інтегрувати у механізми прийняття рішень щодо масштабування у Kubernetes. Зокрема, використання таких

метрик, як рівень завантаження ЦП, час відгуку або відсоток помилок, дозволяє формувати більш адаптивні конфігурації та політики автоскейлінгу. Це забезпечує не лише реакцію на поточне навантаження, але й врахування прогнозованих змін у поведінці користувачів, що особливо важливо при використанні ІМП.

Застосування знання-орієнтованого підходу в поєднанні з ІМП є доцільним у системах, де доступні структуровані історичні дані та існують сталі сценарії взаємодії користувачів із системою. В таких умовах використання онтологічних профілів та експертних доменних правил дозволяє підвищити точність прогнозування та забезпечити більш глибоке врахування семантичного контексту запитів. Це, в свою чергу, сприяє підвищенню ефективності функціонування МСА систем e-Banking та створює передумови для подальшої автоматизації процесів прийняття рішень у межах програмної системи.

4.6 Висновки до розділу 4

У даному розділі було виконано програмну реалізацію прототипу системи e-Banking, який обмежений доменом побудови виписок за рахунками, і мотивовано обрано стек технологій для реалізації пропонованої мікросервісної архітектури. Реалізований прототип було експериментально досліджено, а отримані показники оцінено та проаналізовано.

У підрозділі 4.1 було обґрунтовано вибір технологічного стеку та описано принципи його формування з урахуванням вимог до продуктивності, масштабованості та надійності систем e-Banking. Особливістю обраного підходу є використання технологічного стеку, що поєднує різні платформи та мови програмування, що дозволяє ефективно реалізувати як обчислювально складні задачі прогнозування, так і високонавантажені сервіси обробки запитів. Крім того, використання екосистеми Spring забезпечує стандартизовані механізми побудови та інтеграції мікросервісів, що спрощує їх масштабування та супровід.

У підрозділі 4.2 було розглянуто особливості програмної реалізації прототипу системи, який обмежений доменом побудови виписок за рахунками, а також наведено ключові аспекти організації міжсервісної взаємодії та конфігурації системи. Ключовою особливістю реалізації є поєднання синхронних і асинхронних механізмів взаємодії мікросервісів, що дозволяє ефективно розподіляти навантаження між компонентами системи. Додатково, використання контейнеризації та оркестрації забезпечує можливість динамічного масштабування та адаптації системи до змін умов її функціонування.

У підрозділі 4.3 було наведено результати експериментального дослідження прототипу, визначено параметри проведення експериментів та отримано значення метрик, що характеризують продуктивність, надійність і ефективність використання ресурсів системи. Особливу увагу приділено метрикам, що відображають вплив ІМП на процеси попередньої підготовки даних, зокрема відсотку їх перевикористання. Це дозволило кількісно оцінити ефективність прогнозування поведінки користувачів та обґрунтувати доцільність застосування відповідних алгоритмів у системах e-Banking.

У підрозділі 4.4 було виконано аналіз отриманих результатів, встановлено вплив застосування ІМП у поєднанні з МСА на ключові показники якості системи, а також обґрунтовано доцільність використання запропонованого підходу для систем e-Banking. Це дозволило підтвердити ефективність використання знання-орієнтованих методів у поєднанні з МСА для вирішення задач оптимізації роботи систем e-Banking.

У підрозділі 4.5 було сформульовано практичні рекомендації щодо використання запропонованого підходу в системах e-Banking, зокрема визначено умови доцільності застосування ІМП та особливості його інтеграції у МСА. Окрему увагу приділено рекомендаціям щодо організації інфраструктури розгортання та використання метрик якості для адаптивного управління масштабуванням системи.

Отже, результати проведеного дослідження підтверджують ефективність застосування ІМП у складі МСА для підвищення продуктивності, надійності та масштабованості систем e-Banking. Запропонований підхід дозволяє враховувати поведінкові особливості користувачів, оптимізувати використання обчислювальних ресурсів та забезпечувати більш стабільне функціонування системи в умовах змінного навантаження. Отримані результати створюють основу для практичного впровадження запропонованих рішень, а також їх подальшого розвитку для інших доменів системи e-Banking.

ВИСНОВКИ

За результатами роботи було досягнуто основну мету дослідження: розроблено знання-орієнтовану модель та запропоновано принципи проєктування МСА для систем e-Banking, що забезпечують підвищення показників продуктивності та надійності їх функціонування. Також було успішно виконано усі поставлені задачі дослідження. Нижче наведено отримані результати, що підтверджують актуальність роботи, а також її наукову та практичну значимість:

1. Виконано аналіз існуючих підходів до побудови систем класу e-Banking. Визначено основні проблеми розробки і супроводу цих систем. Встановлено, що ключовими проблемами є складність масштабування, зростання навантаження на систему при пікових запитах, а також обмежена гнучкість традиційних архітектурних рішень. Окрему увагу приділено аналізу недоліків монолітних підходів та обґрунтуванню доцільності переходу до МСА.
2. Визначено основні функціональні та нефункціональні вимоги до систем класу e-Banking. Виконано порівняльних аналіз сучасних систем.
3. Вперше розроблено знання-орієнтовану доменну модель на основі онтологічних профілів користувачів та експертних правил, що забезпечило формалізацію складної бізнес-логіки та врахування семантичного контексту.
4. Запропоновано інтелектуальний модуль прогнозування на базі методу аналізу часових рядів SARIMA, який дозволяє адаптивно керувати обчислювальними ресурсами шляхом передбачення дій користувачів. Реалізований підхід забезпечує можливість визначати найбільш ймовірні моменти формування запитів від користувачів на побудову виписки. Це, в свою чергу, дозволяє заздалегідь підготовлювати необхідні дані та зменшувати навантаження на систему у пікові періоди.

5. Удосконалено процедуру формування виписок шляхом комбінованого застосування знання-орієнтованої доменної моделі та методу аналізу часових рядів SARIMA. Такий підхід забезпечує узгоджене використання історичних даних і експертних знань, що підвищує ефективність обробки запитів. У результаті досягається скорочення часу формування виписок та підвищення стабільності роботи системи.
6. Визначено проблемно-орієнтовані архітектурні принципи та патерни MCA, адаптовані для банківського домену, а також запропоновано формулу розрахунку глибини декомпозиції сервісів. Запропонований підхід дозволяє обґрунтовано визначати рівень декомпозиції системи з урахуванням балансу між складністю бізнес-логіки та витратами на міжсервісну взаємодію. Це сприяє підвищенню ефективності проєктування архітектури систем e-Banking.
7. Розроблено компонентну архітектуру для ПЗ систем e-Banking на основі MCA. Архітектура передбачає виділення окремих мікросервісів для реалізації ключових функціональних компонентів, включаючи ІМП та обробку експертних правил. Це забезпечило незалежність компонентів, гнучкість масштабування та спрощує супровід системи.
8. Розроблено систему критеріїв та кількісних метрик оцінки якості. Запропоновані метрики дозволяють оцінити продуктивність, надійність та можливості до масштабування системи. Їх використання дозволяє об’єктивно проаналізувати результати експериментального дослідження та виконати порівняння з результатами для різних архітектурних рішень.
9. Запропоновано стек технологій для програмної реалізації системи та розроблено прототип фрагменту системи e-Banking, який обмежений доменом побудови виписок за рахунками користувачів. Обраний стек поєднує різні мови програмування та фреймворки/бібліотеки, що дозволяє ефективно реалізувати як обчислювальні задачі прогнозування, так і високонавантажені сервіси.

10. Експериментально перевірено прототип фрагменту системи e-Banking. Отримані результати демонструють стабільне покращення ключових показників якості системи при використанні ІМП у складі МСА. Проведене дослідження підтверджує ефективність запропонованого підходу в умовах змінного навантаження та різних сценаріїв використання системи.
    a. В 1.5–2 рази зменшено час формування виписок за рахунками користувачів
    b. Підвищено пропускну зданість системи на 50-60%
    c. Зменшено навантаження на ЦП у середньому на 12–15%
    d. Суттєво скорочено час критичної (>80%) завантаженості ЦП
    e. В середньому в 4 рази зменшено відсоток помилок при формуванні виписок

Виконано усі поставлені задачі дослідження. Отримані результати відповідають очікуваній науковій та практичній новизні.

Основні результати роботи були використані у звітах на кафедрі інтелектуальних програмних систем і технологій (ІПСіТ) Харківського національного університету імені В.Н. Каразіна при виконанні НДР за темою «Концептуальні моделі, методи та технології створення адаптивних інформаційних систем на основі знання-орієнтованих підходів та засобів розробки програмного забезпечення» (МОН України № ДР: 0121U110310) у період з 2022 по 2024 роки, див. додаток Б. А також використані розробки навчально-методичного забезпечення кафедри ІПСіТ: розробки та оновлення робочих програм та завдань для лекцій, лабораторних та практичних робіт студентів за ОП F3 (бакалаври) – «Комп’ютерні науки» у дисциплінах «Основи бізнес-аналізу та інженерії вимог до програмного забезпечення» та «Методи та засоби доменного моделювання варіабельних програмних систем», див. додаток В. Крім того, результати дослідження були апробовані у реальних ІТ-проєктах української компанії CS, що показано у додатку Г.

Виконана робота вирішує актуальну науково-практичну задачу застосування знання-орієнтованих моделей та інтелектуальних методів у поєднанні з перевагами мікросервісної архітектури для розробки систем e-Banking. Метою застосування цього підходу є підвищення якості процесів проєктування та розробки цих систем для забезпечення вимог до відповідного рівня показників якості їх функціонування. Отримані результати підтверджують досягнення поставленої мети дисертаційного дослідження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Даас Т.І., Ткачук М.В. Про один підхід до розробки банківських інформаційних систем: знання-орієнтовані моделі та мікросервісні архітектури // Збірник наукових праць міжнародної науково-технічної конференції «Комп'ютерне моделювання в наукоємних технологіях» (КМНТ-2022). м. Харків, 23-25 листопада 2022 р. – С. 54 – 57
2. Tan T.H., Tan T.H. E-Banking SAF: A TOGAF-NIST Aligned Security Architecture Framework for E-Banking Systems. The 7th International Conference on Information and Computer Technologies. 2024. Honolulu, Hawaii, USA. P. 1–6
3. Коваль В. Правове визначення інформаційної безпеки електронного банкінгу. Молодий вчений. 2023. №1 (113), С. 121–125. https://doi.org/10.32839/2304-5809/2023-1-113-25
4. Newman S. Building Microservices, 2nd Edition. New York. USA: O`Reilly Media, 2021. 616 p.
5. Фаулер М. Архітектура корпоративних програмних застосунків – М., Видавничий дім Вільямс, 2006. – 544 с.
6. Koneru N. Containerization Best Practices-Using Docker and Kubernetes for Enterprise Applications. Journal of Information Systems Engineering and Management. Vol. 10, no. 45s. 2025. p.724-746. https://doi.org/10.52783/jisem.v10i45s.8905
7. Зінов'єв Д.В., Ткачук М.В.. Тріщенко І.В. Моделі та технології забезпечення якості сервіс-орієнтованих програмних систем: сучасний стан та перспективні напрямки досліджень // Матеріали міжн. науков-техн. конференції КМНТ-2021, (м. Харків, 23-25 квітня 2021 року) – Х.: ХНУ імені В.Н. Каразіна, 2021. – С. 166-169.
8. Cojocaru M., Uta A., Oprescu A.M. MicroValid: A Validation Framework for Automatically Decomposed Microservices. IEEE International

Conference on Cloud Computing Technology and Science (CloudCom - 2019), 2019. pp. 78 – 86. https://doi.org/10.1109/CloudCom.2019.00023

9. Даас Т.І., Ткачук М.В. Аналіз сучасного стану і перспектив розвитку у галузі розробки та супроводу систем інтернет-банкінгу. Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління. 2024. вип. 62. С.6-18. https://doi.org/10.26565/2304-6201-2024-62-01
10. IFOBS.Corporate сайт. [Електронний ресурс] URL: https://csltd.com.ua/products/corporate_online_banking/ (дата звернення: 29.01.2026)
11. IFOBS.Private сайт. [Електронний ресурс] URL: https://csltd.com.ua/products/private_online_banking/ (дата звернення: 29.01.2026)
12. iBank2.Corporate сайт. [Електронний ресурс] URL: https://dbosoft.com.ua/#/products/business/web_banking_business/ (дата звернення: 29.01.2026)
13. iBank2.Private сайт. [Електронний ресурс] URL: https://dbosoft.com.ua/#/products/private/web_banking_private/ (дата звернення: 29.01.2026)
14. IBank2, криптобіблиіотека “Гепард 2.0” [Електронний ресурс] URL: https://dbosoft.com.ua/assets/about/licenses/eo_2099.pdf (дата звернення: 29.01.2026)
15. IBank2, центр сертифікації ключів “Integra CA” [Електронний ресурс] URL: https://dbosoft.com.ua/assets/about/licenses/eo_1541.pdf (дата звернення: 29.01.2026)
16. Privat24 сайт. [Електронний ресурс] URL: https://privatbank.ua/udalenniy-banking/privat24 (дата звернення: 29.01.2026)

17. Monobank сайт. [Електронний ресурс] URL: https://www.monobank.ua/ (дата звернення: 29.01.2026)
18. Ткачук М. В., Сокол В. Є. Деякі проблеми управління IT-інфраструктурою підприємства: сучасний стан та перспективи розвитку. Східно-Європейський журнал передових технологій. 2010. № 6/2 (48). С. 68–72. https://doi.org/10.15587/1729-4061.2010.3234
19. OWASP Top 10 сайт. [Електронний ресурс] URL: https://owasp.org/www-project-top-ten (дата звернення: 29.01.2026)
20. OWASP Application Security Verification Standard v5.0.0. Available at: https://github.com/OWASP/ASVS/raw/v5.0.0/5.0/OWASP_Application_Security_Verification_Standard_5.0.0_en.pdf
21. Про затвердження Положення про організацію заходів із забезпечення інформаційної безпеки в банківській системі України: Постанова НБУ №95 від 28.09.2017р.
22. Офіційний сайт Grafana. [Електронний ресурс] URL: https://grafana.com (дата звернення: 29.01.2026)
23. Офіційний сайт Prometheus. [Електронний ресурс] URL: https://prometheus.io (дата звернення: 29.01.2026)
24. Офіційний сайт Zipkin. [Електронний ресурс] URL: https://zipkin.io (дата звернення: 29.01.2026)
25. Офіційний сайт Jaeger. [Електронний ресурс] URL: https://www.jaegertracing.io (дата звернення: 29.01.2026)
26. V. Vernon, Implementing Domain-Driven Design. Boston, USA: Addison-Wesley, 2013
27. Daas T.I. Towards domain modeling approach to software development for bank information systems // Матеріали XXIII Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів «Стан, досягнення та перспективи інформаційних систем і технологій». м. Одеса, 20 – 21 квітня 2023 р. – Одеса, Видавництво ОНТУ, 2023 р. – С. 183 – 185

28. V. Nair, Practical Domain-Driven Design in Enterprise Java. New York, USA: Apress, 2019
29. S. Millett, Patterns, Principles and Practices of Domain-Driven Design. Hoboken, USA: John Wiley & Sons, 2015
30. OMG Unified Modeling Language Specification. Version 2.5 (2015). Available at: https://www.omg.org/spec/UML/2.5/PDF
31. OMG Business Process Model and Notation (BPMN). Version 2.0.2 (2014). Available at: https://www.omg.org/spec/BPMN/2.0.2/PDF
32. Daas, T., Tkachuk, M. (2026). Development of intelligent model-technological tools for e-Banking systems based on microservices. Eastern-European Journal of Enterprise Technologies, 1 (2 (139)), 48-57. https://doi.org/10.15587/1729-4061.2026.351604
33. Тищенко О. І. Огляд сучасних тенденцій на ринку онлайн-банкінгу в Україні. Електронний журнал "Економіка і суспільство". 2017. №13. С. 1237–1243. URL: https://economyandsociety.in.ua/journals/13_ukr/206.pdf (дата звернення: 11.02.2026)
34. Brosens J., Kruger R. M., Smuts H. Guidelines for Designing e-Statements for e-Banking. The Second African Conference for Human Computer Interaction: Thriving Communities. December 2018. p. 1–6
35. Shah D., Thaker M. A Review of Time Series Forecasting Methods. International journal of research and analytical reviews. April 2024. Vol. 11, No. 2. p. 749–755
36. Buchatskaya V., Buchatsky P., Teploukhov S. Forecasting Methods Classification and its Applicability. Indian Journal of Science and Technology. November 2015. Vol. 8(30). p. 1-8. https://dx.doi.org/10.17485/ijst/2015/v8i30/84224
37. Liu Z., Zhu Z., Gao J., Xu C. Forecast methods for time series data: A survey. IEEE Access. 2021, Vol. 9, p. 91896–91912. https://doi.org/10.1109/ACCESS.2021.3091162

38. Gupta V. Modeling Time-Series and Spatial Data for Recommendations and Other Applications: PhD dissertation / Indian Institute of Technology Delhi. 2022. 178 p.

39. Joorabloo N., Jalili M., Ren Y. A new temporal recommendation system based on users` similarity prediction. 11th International Conference on Knowledge Discovery and Information Retrieval. 2019. p. 555 – 560. https://doi.org/10.5220/0008377205550560

40. Gomez-Losada A., Duch N. Time Series Forecasting by Recommendation: An Empirical Analysis on Amazon Marketplace. International Conference on Business Information Systems. 2019. Vol. 1, p. 45 – 54. https://doi.org/10.1007/978-3-030-20485-3_4

41. Newman S. Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith. New York. USA: O'Reilly, 2019. 256 p.

42. Hamzehloui M., Sahibuddin S. and Ashabi, A. A Study on the Most Prominent Areas of Research in Microservices. International Journal of Machine Learning and Computing. 2019. Vol. 9, No. 2. p. 242–247. https://doi.org/10.18178/ijmlc.2019.9.2.793

43. Mendonca N., Jamshidi P., Garlan D., Developing Self-Adaptive Microservice Systems: Challenges and Directions. IEEE Software. 2019. Vol. 38 (Issue 2). p. 70–79. https://doi.org/10.1109/MS.2019.2955937

44. Офіційний сайт компанії CS. Available at: https://csltd.com.ua/

45. Офіційний сайт Jaspersoft. [Електронний ресурс] URL: https://community.jaspersoft.com/ (дата звернення: 21.02.2026)

46. Даас Т.І., Ткачук М.В. Застосування методів аналізу часових рядів та доменного моделювання при розробці інтелектуального модуля прогнозування в системі інтернет-банкінгу. Системи обробки інформації. 2025. вип. 3 (182). С.34-43. https://doi.org/10.30748/soi.2025.182.04

47. Даас Т. І., Ткачук М. В. Застосування методів аналізу часових рядів для розробки інтелектуального модуля прогнозування у системах e-Banking.

Матеріали XVII Міжнародної науково-практичної конференції «Інформаційні технології і автоматизація 2024». м. Одеса, 31 жовтня – 1 листопада 2024 р. – Одеса, Видавництво ОНТУ, 2024 р. – С. 442– 445

48. G.E. Box, G.M. Jenkins, G.C. Reinsel, Time Series Analysis: Forecasting and Control, 5th Edition, New York, USA: John Wiley & Sons, August 2015.
49. S. Dhawani, T. Manishkumar. A Review of Time Series Forecasting Methods. International journal of research and analytical reviews, vol. 11, no. 2. p. 749–755, April 2024.
50. C. Peng, N. Aichen, L. Duanyang, J. Wei, M. Bin. Time Series Forecasting of Temperatures using SARIMA: An Example from Nanjing. IOP Conf. Series: Materials Science and Engineering 394 052024, p. 1-7, 2018. https://doi.org/10.1088/1757-899X/394/5/052024
51. Z. Liu, Z. Zhu, J. Gao, and C. Xu. Forecast methods for time series data: A survey. IEEE Access, 2021, vol. 9, p. 91896-91912. https://doi.org/10.1109/ACCESS.2021.3091162
52. U.M. Sirisha, M.C. Belavagi, G. Attigeri. Profit Prediction Using ARIMA, SARIMA and LSTM Models in Time Series Forecasting: A Comparison. IEEE Access, 2022, vol 10. pp. 124715-124727. https://doi.org/10.1109/ACCESS.2022.3224938
53. Kemalbay G., Berak Korkmazoglu O. Sarima-arch versus genetic programming in stock price prediction. Sigma Journal of Engineering and Natural Sciences. June 2021, vol.36(2), pp.110-122. https://doi.org/10.14744/sigma.2021.00001
54. Farsi M., Hosahalli D., Manjunatha B.R. Parallel genetic algorithms for optimizing the SARIMA model for better forecasting of the NCDC weather data. Alexandria Engineering Journal. February 2021, vol.60(1). pp. 1299-1316. https://doi.org/10.1016/j.aej.2020.10.052
55. Patil R., Nagaraj D. M., Polisgowdar B.S. and Rathod S. Forecasting potential evapotranspiration for Raichur district using seasonal ARIMA

model. MAUSAM Journal. 2022. vol. 73. No. 2. p.433-440. https://doi.org/10.54302/mausam.v73i2.5488

56. Hyndman R.J. Forecasting: Principles and Practice (3rd edition). Australia: OTexts, 2021.
57. Roid I., Panasenko Y., Huba S. Digital banking architecture: things to consider when building banking software. URL: https://yalantis.com/blog/technical-side-of-digital-banking-software (дата звернення: 21.03.2026)
58. Daas T.I. Design principles and tools to develop of microservice architecture for e-banking systems // Матеріали XVI Міжнародної науково-практичної конференції "Пріоритетні шляхи розвитку науки і освіти". Львів. 29-30 вересня 2025 р. – С. 51 – 55.
59. Deshpande R. Application Of Spring Boot Microservice Architecture for Scaling Banking Applications. The American Journal of Engineering and Technology. 2025 Vol. 7(9), p.152–158. https://doi.org/10.37547/tajet/Volume07Issue09-09
60. Tomic M., Dimitrieski V., Vjestica M. Towards Applying API Gateway to Support Microservice Architectures for Embedded Systems. 12th International Conference on Information Society and Technology (ICIST 2022). March 2022. Kopaonik, Serbia. p. 86–91. Available at: https://www.researchgate.net/publication/361952256_Towards_Applying_API_Gateway_to_Support_Microservice_Architectures_for_Embedded_Systems
61. Daraghmi E., Zhang C., Yuan S. Enhancing Saga Pattern for Distributed Transactions within a Microservices Architecture. Applied Sciences. 2022. Vol. 12, No. 6242. https://doi.org/10.3390/app12126242
62. Ćatović A., Buzađija N., and Lemes S. Microservice development using RabbitMQ message broker. Science, Engineering and Technology. April 2022. Vol. 2, no. 1, pp. 30–37. https://doi.org/10.54327/set2022/v2.i1.19

63. Elrashidy M., Mansour H. Microservices Architecture in Fintech: A Case Study on Scalable Loan Processing with Classical Design Patterns. Intelligent Methods, Systems, and Applications (IMSA). July 2025. p. 382-387. https://doi.org/10.1109/IMSA65733.2025.11167186

64. Даас Т.І., Ткачук М.В. До питання вдосконалення алгоритмів побудови виписок за рахунками клієнтів систем e-Banking // Збірник наукових праць міжнародної науково-технічної конференції «Комп'ютерне моделювання в наукоємних технологіях» (КМНТ-2024). м. Харків, 27-29 листопада 2024 р. – С. 73 – 76

65. Rosano R. Exploring Local E-Statement Banking Solutions. March 2024. URL: https://fintelite.ai/exploring-local-e-statement-banking-solutions/ (дата звернення 14.03.2026)

66. John R. Complete Guide to Automated Bank Statement Processing: Benefits, Challenges, Method. October 2024. URL: https://www.docsumo.com/blogs/bank-statement-extraction/what-is (дата звернення 09.02.2026)

67. Dubielewicz I., Hnatkowska B., Huzar Z., Tuzinkiewicz L. Domain modeling in the context of ontology. Foundations of Computing and Decision Sciences. Mar. 2015. vol. 40(1). p. 3–15. https://doi.org/10.1515/fcds-2015-0001

68. Офіційний сайт продукту Flowable. URL: https://documentation.flowable.com/latest/ (дата звернення: 19.03.2026)

69. Oracle Cloud Infrastructure Documentation. Monitoring. URL: https://docs.oracle.com/en-us/iaas/Content/Monitoring/Concepts/alarmsbestpractices.htm (дата звернення: 22.03.2026)

70. Pai K., Srinivas, B. J. Enhanced Visibility for Real-Time Monitoring and Alerting in Kubernetes by Integrating Prometheus, Grafana, Loki, and Alerta. International Journal of Scientific Research in Engineering Management. June 2024, Vol. 8(6). https://doi.org/10.55041/IJSREM35639

71. Gundla S.R. AI-Assisted Legacy Modernization: Automating Monolith-to-Microservice Decomposition. International Journal of Networks and Security. 2025. Vol. 5(1). P.147-173. https://doi.org/10.55640/ijns-05-01-09

72. Даас Т.І., Ткачук М.В. Інтелектуальні підходи до розробки та супроводу мікросервісних архітектур: класифікація, основні виклики та досвід застосування // Збірник наукових праць міжнародної науково-технічної конференції «Інтелектуальні технології у міждисциплінарних дослідженнях» (ІТМД-2025). м. Харків, 12-14 листопада 2025 р. – С. 101 – 104.

73. Navarez D., Battaglia N., Fernandez A. Designing Microservices Using AI: A Systematic Literature Review. Software. 2025. Vol. 4(1). https://doi.org/10.3390/software4010006

74. Ткачук М. В., Зінов'єв Д.В. Розробка та дослідження алгоритмічної моделі для адаптивного управління конфігураціями програмних мікросервісів. Системи обробки інформації. 2024. № 2(177). - С.116–120. https://doi.org/10.30748/soi.2024.177.12

75. Moreschini S., Pour S., Lanese I. AI Techniques in the Microservices Life-Cycle: A Systematic Mapping Study. Computing. 2025. Vol. 107. https://doi.org/10.1007/s00607-025-01432-z

76. Namanyay G., Arpit J. AI-Driven Code Optimization and Refactoring for Large-Scale Software Development. Journal of Quantum Science and Technology. 2025. Vol.2(2). p. 270 – 282. https://doi.org/10.63345/jqst.v2i2.282

77. Avornicului M.C., Bresfelean Vasile P. Model driven development of online banking systems. Annals of the University of Oradea, Economic Science Series, 2011, Vol 20, Issue 1, p. 795– 800

78. Nguyen V.H. An Architectural View Model for Designing and Implementing Microservices-based Systems: Use Case in FinTech. Procedia Computer Science. 2024. Vol. 237. p. 667–674. https://doi.org/10.1016/j.procs.2024.05.152

79. Максименко В.О., Фролов О.В. Порівняльний аналіз продуктивності gRPC та RESTful API мікросервісів з використанням MongoDB та MS SQL Server. Наукові праці Донецького національного технічного університету. Серія: “Інформатика, кібернетика та обчислювальна техніка”: Всеукр. наук. зб. – Дрогобич : ДонНТУ, 2025. – №1(40). с.51-59. https://doi.org/10.31474/1996-1588-2025-1-40-51-59

80. Moreschini S., Pour S., Lanese I. AI Techniques in the Microservices Life-Cycle: A Systematic Mapping Study. Computing. 2025. Vol. 107 (100). https://doi.org/10.1007/s00607-025-01432-z

81. Prasanth Y., Nakul Sh. Integrating Natural Language Processing and Software Engineering. International Journal of Software Engineering and Its Applications. 2015. Vol. 9, No. 11. p. 127–136. Available at: https://www.researchgate.net/publication/292299148_Integrating_Natural_Language_Processing_and_Software_Engineering

82. Abdelnabi E. A., Maatuk A. M., Abdelaziz T. M., Elakeili S. M. Generating UML Class Diagram using NLP Techniques and Heuristic Rules. 20th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA). Monastir, Tunisia, 2020, pp. 277–282. https://doi.org/10.1109/STA50679.2020.9329301

83. Pillutla M.K.V.S.P.S (2025) Enterprise-Scale Microservices Architecture: Domain-Driven Design and Cloud-Native Patterns Using the Spring Ecosystem, European Journal of Computer Science and Information Technology, 13 (45), 1-10. https://doi.org/10.37745/ejcsit.2013/vol13n45110

84. Spring Framework official documentation. Available at: https://spring.io/

85. Jeleń, M., & Dzieńkowski, M. (2021). The comparative analysis of Java frameworks: Spring Boot, Micronaut and Quarkus. Journal of Computer Sciences Institute, 21, 287–294. https://doi.org/10.35784/jcsi.2724

86. Szymanek, G., & Smołka, J. (2025). Comparative performance analysis of Spring Boot and Quarkus frameworks in Java applications. Journal of Computer Sciences Institute, 37, 484–491. https://doi.org/10.35784/jcsi.8047

87. Krekora, K. (2024). Performance analysis of .Net and Spring Boot microservices on Microsoft Azure. Journal of Computer Sciences Institute, 33, 258–263. https://doi.org/10.35784/jcsi.6268

88. Sohith Sri Ammineedu Yalamati. Resilient microservice patterns using Java 17 and Spring Boot 3.2 in Cloud-Native Systems. International Journal of Science and Research Archive, 2025, 16(03), 431–447. https://doi.org/10.30574/ijsra.2025.16.3.2559

89. Mamidi S. (2026) Microservices with Spring Boot: Patterns and Anti-Patterns. International Journal For Multidisciplinary Research, 8(1). https://doi.org/10.36948/ijfmr.2026.v08i01.67514

90. Офіційний сайт бібліотеки Statsmodels. URL: https://www.statsmodels.org/stable/index.html (дата звернення: 21.01.2026)

91. Офіційний сайт продукту Neo4j. URL: https://neo4j.com/ (дата звернення: 22.01.2026)

92. Офіційний сайт продукту RabbitMQ. URL: https://www.rabbitmq.com/ (дата звернення: 02.02.2026)

93. Офіційний сайт продукту Redis. URL: https://redis.io/ (дата звернення: 03.02.2026)

94. Joshi, P. K. (2023). Redis Cache Optimization for Payment Gateways in the Cloud. International Journal of Emerging Trends in Computer Science and Information Technology, 4(2), 28-36. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I2P104

95. S. Patil, N. D G and A. Bhat, "Dynamic Autoscaling and Scheduling in Kubernetes Clusters with LSTM and ILP," in Journal of Communications Software and Systems, vol. 21, no. 4, pp. 465-476, November 2025. https://doi.org/10.24138/jcomss-2023-0147

96. Офіційний сайт бібліотеки JUnit6. URL: https://docs.junit.org/6.0.3/overview.html (дата звернення: 23.01.2026)

97. Indrianto, I. (2023). PERFORMANCE TESTING ON WEB INFORMATION SYSTEM USING APACHE JMETER AND

BLAZEMETER. Jurnal Ilmiah Ilmu Terapan Universitas Jambi, 7(2), 138–149. https://doi.org/10.22437/jiituj.v7i2.28440

ДОДАТКИ

Додаток А.

Список публікацій здобувача за темою дисертації

Статті у наукових фахових виданнях, що входять до міжнародних наукометричних баз Scopus та Web of Science:

1. Daas, T., Tkachuk, M. (2026). Development of intelligent model-technological tools for e-Banking systems based on microservices. Eastern-European Journal of Enterprise Technologies, 1 (2 (139)), 48-57. https://doi.org/10.15587/1729-4061.2026.351604

   (Особистий внесок здобувача: розробка архітектури, розробка програмного забезпечення, експериментальне дослідження і аналіз результатів, написання тексту роботи. Особистий внесок Миколи Ткачука: концептуалізація, перевірка тексту роботи, перевірка наукової достовірності отримуваних результатів, редагування)

Статті у наукових фахових виданнях України:

2. Даас Т.І., Ткачук М.В. Аналіз сучасного стану і перспектив розвитку у галузі розробки та супроводу систем інтернет-банкінгу. Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління. 2024. вип. 62. С.6– 18. https://doi.org/10.26565/2304-6201-2024-62-01

   (Особистий внесок здобувача: проведення аналізу систем інтернет-банкінгу, визначення перспектив розвитку. Особистий внесок Миколи Ткачука: концептуалізація, перевірка тексту роботи, редагування)

3. Даас Т.І., Ткачук М.В. Застосування методів аналізу часових рядів та доменного моделювання при розробці інтелектуального модуля прогнозування в системі інтернет-банкінгу. Системи обробки

інформації. 2025. вип. 3 (182). С.34-43. https://doi.org/10.30748/soi.2025.182.04

(Особистий внесок здобувача: побудова доменної моделі, розробка алгоритму роботи інтелектуальноо модуля прогнозування, верифікація прототипу. Особистий внесок Миколи Ткачука: концептуалізація, перевірка тексту роботи, перевірка наукової достовірності отримуваних результатів, редагування)

Наукові праці, які засвідчують апробацію матеріалів дисертації:

4. Даас Т.І., Ткачук М.В. Про один підхід до розробки банківських інформаційних систем: знання-орієнтовані моделі та мікросервісні архітектури // Збірник наукових праць міжнародної науково-технічної конференції «Комп'ютерне моделювання в наукоємних технологіях» (КМНТ-2022). м. Харків, 23-25 листопада 2022 р. – С. 54 – 57.
5. Daas T.I. Towards domain modeling approach to software development for bank information systems // Матеріали XXIII Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів «Стан, досягнення та перспективи інформаційних систем і технологій». м. Одеса, 20 – 21 квітня 2023 р. – Одеса, Видавництво ОНТУ, 2023 р. – С. 183 – 185.
6. Даас Т.І., Ткачук М.В. Застосування методів аналізу часових рядів для розробки інтелектуального модуля прогнозування у системах e-Banking. Матеріали XVII Міжнародної науково-практичної конференції «Інформаційні технології і автоматизація 2024». м. Одеса, 31 жовтня – 1 листопада 2024 р. – Одеса, Видавництво ОНТУ, 2024 р. – С. 442– 445.
7. Даас Т.І., Ткачук М.В. До питання вдосконалення алгоритмів побудови виписок за рахунками клієнтів систем e-Banking // Збірник наукових праць міжнародної науково-технічної конференції «Комп'ютерне моделювання в наукоємних технологіях» (КМНТ-2024). м. Харків, 27-29 листопада 2024 р. – С. 73 – 76.

8. Daas T.I. Design principles and tools to develop of microservice architecture for e-banking systems // Матеріали XVI Міжнародної науково-практичної конференції "Пріоритетні шляхи розвитку науки і освіти". Львів. 29-30 вересня 2025 р. – С. 51 – 55.
9. Даас Т.І., Ткачук М.В. Інтелектуальні підходи до розробки та супроводу мікросервісних архітектур: класифікація, основні виклики та досвід застосування // Збірник наукових праць міжнародної науково-технічної конференції «Інтелектуальні технології у міждисциплінарних дослідженнях» (ІТМД-2025). м. Харків, 12-14 листопада 2025 р.

Додаток Б.

ЗАТВЕРДЖУЮ
Проректор з наукової та науково-педагогічної роботи
Харківського національного університету
імені [illegible]

Борис САМОЙЛОВ
"__" _____ 2026 р.

АКТ
про використання результатів дисертаційної роботи
ДААС Тимур Імад Ахмад
«ЗНАННЯ-ОРІЄНТОВАНІ МОДЕЛІ ТА МІКРОСЕРВІСНІ АРХІТЕКТУРИ ДЛЯ РОЗРОБКИ СИСТЕМ ІНТЕРНЕТ-БАНКІНГУ»,
що подається на здобуття наукового ступеня доктора філософії (PhD)
з галузі знань12 – інформаційні технології,
за спеціальністю 122 – комп'ютерні науки

Ми, що нижче підписалися, директор ННІ Комп'ютерних наук та штучного інтелекту к.т.н., доцент Д.Ю. Узлов, завідувач кафедри інтелектуальних програмних систем і технологій к.т.н. О.І. Олешко, професор кафедри інтелектуальних програмних систем і технологій д.т.н., професор М.В. Ткачук, склали цей акт у тому, що результати дисертаційної роботи Дааса Т.І. впроваджені в навчальний процес на кафедрі інтелектуальних програмних систем і технологій ННІ комп'ютерних наук та штучного інтелекту.

Запропоновані у роботі моделі та процедури для знання-орієнтованого проектування функціональних модулів систем e-Banking, а також програмні засоби, які розроблені із застосуванням переваг мікросервісної архітектури, були використані в лекційному матеріалі та в лабораторних практикумах до дисциплін «Основи бізнес-аналізу та інженерії вимог до програмного забезпечення» і «Методи та засоби доменного моделювання варіабельних програмних систем».

Директор ННІ комп'ютерних наук
та штучного інтелекту — Дмитро УЗЛОВ

Завідувач кафедри інтелектуальних
програмних систем і технологій — Олег ОЛЕШКО

Професор кафедри інтелектуальних
програмних систем і технологій — Микола ТКАЧУК

Додаток В.

ЗАТВЕРДЖУЮ
Проректор з наукової та науково-педагогічної роботи
Харківського національного університету
імені В.Н. Каразіна
Борис САМОРОДОВ
"__" _____ 2026 р.

АКТ
про використання результатів дисертаційної роботи
ДААС Тимур Імад Ахмад
«ЗНАННЯ-ОРІЄНТОВАНІ МОДЕЛІ ТА МІКРОСЕРВІСНІ АРХІТЕКТУРИ ДЛЯ РОЗРОБКИ СИСТЕМ ІНТЕРНЕТ-БАНКІНГУ»,
що подається на здобуття наукового ступеня доктора філософії (PhD)
з галузі знань12 – інформаційні технології,
за спеціальністю 122 – комп'ютерні науки

Ми, що нижче підписалися, директор ННІ Комп'ютерних наук та штучного інтелекту к.т.н., доцент Д.Ю. Узлов, завідувач кафедри інтелектуальних програмних систем і технологій к.т.н. О.І. Олешко, професор кафедри інтелектуальних програмних систем і технологій д.т.н., професор М.В. Ткачук, склали цей акт у тому, що результати дисертаційної роботи Дааса Т.І. впроваджені в навчальний процес на кафедрі інтелектуальних програмних систем і технологій ННІ комп'ютерних наук та штучного інтелекту.

Запропоновані у роботі моделі та процедури для знання-орієнтованого проектування функціональних модулів систем e-Banking, а також програмні засоби, які розроблені із застосуванням переваг мікросервісної архітектури, були використані в лекційному матеріалі та в лабораторних практикумах до дисциплін «Основи бізнес-аналізу та інженерії вимог до програмного забезпечення» і «Методи та засоби доменного моделювання варіабельних програмних систем».

Директор ННІ комп'ютерних наук
та штучного інтелекту Дмитро УЗЛОВ

Завідувач кафедри інтелектуальних
програмних систем і технологій Олег ОЛЕШКО

Професор кафедри інтелектуальних
програмних систем і технологій Микола ТКАЧУК

Додаток Г.

Акти про практичне застосування отриманих результатів.

ТОВ «СІЕС КОНСАЛТІНГ»
Україна, 61072, м. Харків
Вул. Вартових Неба, 42А
Tel/Fax: +38 (067) 162 59 52
www.csltd.com.ua, csltd@csltd.com.ua

CS
INSPIRE CONFIDENCE

06.04.2026 №2

АКТ

щодо впровадження результатів дисертаційної роботи на здобуття ступеня доктора філософії за спеціальністю 122 - Комп'ютерні науки (галузь знань 12 - Технічні науки)
Дааса Тимура Імада Ахмада

ТОВ «СІЕС КОНСАЛТІНГ» було впроваджено запропонований у роботі знання-орієнтований алгоритм побудови виписок за рахунками користувачів у системах інтернет-банкінгу у своїх продуктах. Також було застосовано архітектурні принципи проєктування систем класу інтернет-банкінгу при розробці продуктів компанії. Застосування пропонованого у роботі підходу дозволило покращити показники продуктивності, надійності та масштабованості системи інтернет-банкінгу.

Результати дисертаційного дослідження представляють інтерес для нашої компанії.

Акт виданий для пред'явлення за місцем захисту дисертації та не є підставою для фінансових розрахунків.

З повагою,

Директор ТОВ «СІЕС КОНСАЛТІНГ» Ігор БАБЧЕНКО

УКРАЇНА Товариство з обмеженою відповідальністю «СіЕс Консалтінг» №31636043 місто Харків

Онлайн сервіс створення та перевірки кваліфікованого та удосконаленого електронного підпису

ПРОТОКОЛ
створення та перевірки кваліфікованого та удосконаленого електронного підпису

Дата та час: 18:17:38 04.05.2026

Назва файлу з підписом: Daas_T.I._Diss.pdf.p7s
Розмір файлу з підписом: 3.6 МБ

Перевірені файли:
Назва файлу без підпису: Daas_T.I._Diss.pdf
Розмір файлу без підпису: 3.5 МБ

Результат перевірки підпису: Підпис створено та перевірено успішно. Цілісність даних підтверджено

Підписувач: ДААС ТИМУР ІМАД АХМАД
П.І.Б.: ДААС ТИМУР ІМАД АХМАД
Країна: Україна
РНОКПП: 3528101439
Організація (установа): ФІЗИЧНА ОСОБА
Час підпису (підтверджено кваліфікованою позначкою часу для підпису від Надавача): 18:17:37 04.05.2026
Сертифікат виданий: КНЕДП АЦСК АТ КБ "ПРИВАТБАНК"
Серійний номер: 5E984D526F82F38F0400000063785C01BFD36F07
Алгоритм підпису: ДСТУ 4145
Тип підпису: Удосконалений
Тип контейнера: Підпис та дані в одному файлі (CAdES enveloped)
Формат підпису: З повними даними ЦСК для перевірки (CAdES-X Long)
Сертифікат: Кваліфікований

Версія від: 2026.04.06 13:00