Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

# МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна система централізованої авторизації для освітнього веб-середовища навчального закладу»

Виконав: студент 2-го курсу, групи 1ПІ-22м
спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Малініч П. П.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ:
Коваленко О. О.
(прізвище та ініціали)
«15» грудня 2023 р.

Опонент: к.т.н., проф. каф. ЗІ:
Кондратенко Н. Р.
(прізвище та ініціали)
«15» грудня 2023 р.

**Допущено до захисту**
Завідувач кафедри ПЗ
д.т.н, проф. Романюк О.Н.
(прізвище та ініціали)
«15» грудня 2023 р.

ВНТУ – 2023

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

| | |
|---|---|
| УЗГОДЖЕНО | ЗАТВЕРДЖУЮ |
| Директор центру дистанційної освіти ВНТУ | Завідувач кафедри ПЗ |
| к.т.н., професор Паламарчук Є.А. | д.т.н., професор Романюк О.Н. |
| «19» вересня 2023 року | «19» вересня 2023 р. |

**З А В Д А Н Н Я**
**НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙУ РОБОТУ СТУДЕНТУ**

Малінічу Павлу Павловичу

1. Тема роботи – Інформаційна система централізованої авторизації для освітнього веб-середовища навчального закладу

Керівник роботи: Коваленко Олена Олексіївна, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від «18» вересня 2023 року № 247.

2. Строк подання студентом роботи

5 грудня 2023 року

3. Вихідні дані до роботи: середовище розробки Visual Studio Code, мови розробки – PHP, JavaScript та Python, операційна система – GNU/Linux, ідентифкація, автентифікація та авторизація, технології централізованого входу в систему, інформаційні системи SSO.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз сучасного стану використання технологій централізованого входу в систему у сфері освіти; проектування архітектури, методів, засобів для централізованого входу в

систему; розробка компонентів системи централізованої авторизації для освітнього веб-середовища навчального закладу; експериментальні дослідження кіберзахищеності та працездатності системи централізованої авторизації; економічна частина; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: архітектура та дизайн системи централізованого входу; розроблені методи та моделі; блок-схеми алгоритмів роботи додатку; тестування додатку.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1-4 | Коваленко О.О., к.т.н., доцент. кафедри ПЗ | 19.09.23 | 05.12.23 |
| 5 | Причепа І. В. к.е.н., доцент кафедри ЕПВМ | 25.11.23 | 05.12.23 |

7. Дата видачі завдання 19 вересня 2023 р.

## КАЛЕНДАРНИЙ ПЛАН

| з/п | Назва етапів магістерської кваліфікаційної Роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз сучасного стану використання технологій централізованого входу в систему у сфері освіти | 20.09.2023 – 09.10.2023 | Вик. |
| 2 | Проектування архітектури, методів, засобів для централізованого входу в систему | 10.10.2023 – 20.10.2023 | Вик. |
| 3 | Розробка компонентів системи централізованої авторизації для освітнього веб-середовища навчального закладу | 21.10.2023 – 14.11.2023 | Вик. |
| 4 | Експериментальні дослідження кіберзахищеності та працездатності системи централізованої авторизації | 15.11.2023 – 24.11.2023 | Вик. |
| 5 | Економічна частина | 25.11.2023 – 05.12.2023 | Вик. |

Студент _____ (підпис) Малініч П.П. (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ (підпис) Коваленко О.О. (прізвище та ініціали)

# АНОТАЦІЯ


УДК 004.056.523

Малініч П. П. Інформаційна система централізованої авторизації для освітнього веб-середовища навчального закладу. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2023. 174 с.

На укр. мові. Бібліогр.: 42 назви; рисунків: 47; таблиць 14.

У магістерській кваліфікаційній роботі подано результати дослідження технологій єдиної ідентифікації, аутентифікації та авторизації. Обґрунтовано доцільність удосконалення методів та засобів центрального входу у систему.

Магістерська робота містить аналіз сучасного стану технологій централізованого входу в систему, розробку методів для вдосконалення функціонування централізованого входу у сфері освіти, модель гейміфікації для публічного тесту Тюринга, а також опис архітектури та програмної структури інформаційної системи централізованої авторизації для освітнього веб-середовища навчального закладу.

У роботі виконано проектування архітектури SSO-системи та її компоненти гейміфікації. Для удосконалення процесу централізованого входу користувачів для Вінницького національного технічного університету виконано аналіз роботи попередньої версії SSO-системи для JetIQ, її документації, а також процеси комунікації між співробітниками та підрозділами, що залучені у наданні доступу користувачам. Запропоновано напрями удосконалення системи JetIQ та її IT-екосистеми.

Наукова новизна досліджень полягає в удосконаленні методу проектування REST API-сервісів, методу визначення довіри до клієнтського пристрою, моделі гейміфікації.



Ключові слова: SSO, IdP, ідентифікація, автентифікація, авторизація, система централізованого входу.

# ABSTRACT

UDC 004.056.523

Malinich P. Information system of centralized authorization for the educational web environment of an educational institution. Master's thesis in the specialty 121 – Software Engineering, educational program – Software Engineering. Vinnytsia: VNTU, 2023. 174 p.

In Ukrainian. Bibliography: 42 items; figures: 47; tables: 14.

The master's qualification work presents the results of research on technologies of unified identification, authentication, and authorization. The expediency of improving methods and means of central access to the system has been substantiated.

The master's thesis includes an analysis of the current state of centralized login technologies, the development of methods to improve the functioning of centralized access in educational sphere, a gamification model for the public Turing test, as well as a description of the architecture and software structure of the information system of centralized authorization for the educational web environment of the educational institution.

The work includes the design of the architecture of the SSO system and its gamification components. To improve the process of centralized user login for Vinnytsia National Technical University, an analysis of the operation of the previous version of the SSO system for JetIQ, its documentation, and the communication processes between employees and departments involved in providing user access was carried out. Directions for improving the JetIQ system and its IT ecosystem are proposed.

The scientific novelty of the research lies in improving the method of designing REST API services, the method of determining trust in the client device, and the gamification model.

Keywords: SSO, IdP, identification, authentication, authorization, centralized login system.

ЗМІСТ

# ВСТУП

**Обґрунтування вибору теми дослідження**. Використання та впровадження систем SSO (єдиного входу) у нові сфери є актуальним, оскільки це дозволяє користувачам більш зручно входити до різних систем та додатків одноразово з використанням одного облікового запису, а також організаціям це рішення дозволяє уніфікувати управління доступом користувачів та підвищити захищеність їх доступу.

При використанні систем SSO користувачам не потрібно пам'ятати велику кількість паролів для різних систем і додатків. Це допомагає зменшити навантаження на пам'ять та покращує безпеку. Адміністратори можуть централізовано управляти доступом користувачів до різних систем та ресурсів, що спрощує процеси адміністрування та забезпечує більшу консистентність [1]. Коли користувачі мають одну точку входу, ризик помилкового надання доступу зменшується. Це спрощує управління правами доступу та допомагає відстежувати і аналізувати активність користувачів, що дозволяє вчасно реагувати на загрози та порушення безпеки. Також SSO системи можуть включати механізми багатофакторної аутентифікації і інші заходи безпеки, які зменшують ризик несанкціонованого доступу [2]. Також ці системи можуть інтегруватися з різними додатками і системами, навіть якщо вони розташовані в різних місцях. Це робить їх гнучкими та універсальними. Централізовані системи SSO допомагають зменшити витрати на підтримку інфраструктури, оскільки вона спрощує управління автентифікацією і авторизацією [3].

Зокрема при впровадженні систем централізованої ідентифікації, автентифікації та авторизації користувачів, заклади вищої можуть зіткнутися із недостатком функціоналу існуючих SSO-систем, складність міграції та адаптації до потреб освітнього процесу, а також недостаток кваліфікованого персоналу для впровадження подібних систем. Найкращою опцією є користування послуг компаній, що займаються системною інтеграцією у сфері IT, однак це не завжди можливо для бюджетних установ [4].

Отже, розвиток нових методів і засобів централізованої автентифікації, авторизації та ідентифікації є актуальною темою і потребує нових досліджень в напрямку моделювання, створення нових програмних модулів підвищення захисту користувачів та інтеграції з освітніми веб-середовищами.

**Зв'язок роботи з науковими програмами, планами, темами.** Робота виконана згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

**Мета та завдання дослідження.** Метою роботи є підвищення рівня інтегрованості освітнього веб-середовища навчального закладу; удосконалення методів та засобів створення програмних модулів для централізованої автентифікації, авторизації, ідентифікації та захисту даних користувачів.

Основними **завданнями** дослідження є:

- провести аналіз існуючих методів і засобів централізованої автентифікації, авторизації та ідентифікації;
- виконати аналіз існуючих інформаційних систем централізованого входу в систему, а також ефективність їх використання у сфері освіти;
- запропонувати покращення для програмних компонентів існуючих систем, які можна удосконалити та використати у новій розробці;
- розробити нові методи для вдосконалення функціонування централізованої автентифікації, авторизації та ідентифікації у сфері освіти;
- розробити модель гейміфікації для публічного тесту Тюринга;
- розробити програмну архітектуру та компоненти системи централізованої авторизації для освітнього веб-середовища навчального закладу;
- провести експериментальні дослідження кіберзахищеності та працездатності розробленої системи.

**Об'єкт дослідження** – процеси централізованої автентифікації, авторизації, ідентифікації та їх програмна реалізація.

**Предмет дослідження** – методи та засоби централізованої автентифікації, авторизації та ідентифікації.

**Методи дослідження.** У процесі досліджень використовувались:

- методи аналізу та синтезу – для аналізу існуючих систем, проектування та побудови архітектури систем централізованої автентифікації, авторизації та ідентифікації;
- моделювання та теорії алгоритмів для удосконалення методів та засобів захисту та реалізації розробки програмного забезпечення централізованого входу в систему (SSO); комп'ютерне моделювання процесів тесту Тюринга у клієнт-серверному тривимірному середовищі.

**Наукова новизна отриманих** результатів:

1. Подальшого розвитку отримав метод проектування API-сервісів систем за технологією централізованої автентифікації, авторизації та ідентифікації, який, на відміну від існуючих, використовує систему розподілення приватних та публічних API-ендпойнтів, що дозволяє підвищити якість проектування програмних API-компонентів, рівень захищеності даних, які передаються між різними мікросервісами, а також сприяє зменшенню ризику експуатації вразливостей, що можуть бути виявлені у тих чи інших API-ендпойнтах.
2. Подальшого розвитку отримав метод визначення довіри до клієнтського пристрою, який, на відміну від існуючих, використовує концепцію Zero Trust, що дозволяє підвищити рівень довіри до клієнтського пристрою та рівень його захищеності.
3. Удосконалено модель гейміфікації для публічного тесту Тюринга, яка, на відміну існуючої, забезпечує можливість створювати більш складні тести "я не робот" з ігровою складовою у тривимірному середовищі, що дозволяє підвищити рівень захищеності системи автонтифікації, авторизації та ідентифікації.

**Практична цінність отриманих результатів.** Практична цінність результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано моделі систем

технології централізованої автентифікації, авторизації та ідентифікації у сфері освіти, модель гейміфікації для публічного тесту Тюринга, на основі яких, розроблені та запроваджені програмні компоненти для централізованого входу в систему для освітнього веб-середовища навчального закладу.

**Впровадження.** Впровадження результатів досліджень підтверджуються актом впровадження Центру дистанційної освіти Вінницького національного технічного університету (ВНТУ) – вдосконалення системи єдиного входу системи JetIQ MY з використанням методів та коду програмних компонентів, реалізованих у цій роботі.

*Особистий внесок здобувача***.** Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: модель гейміфікації для публічного тесту Тюринга; метод моделювання систем технології централізованої автентифікації, авторизації та ідентифікації у сфері освіти; вдосконалені бекенд та фронтенд компоненти системи централізованої авторизації для освітнього веб-середовища навчального закладу.

**Апробація матеріалів магістерської кваліфікаційної роботи.** Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на Міжнародній науково-практичній конференції «Інформаційні технології і автоматизація – 2023» [4], на Міжнародній науково-технічній конференції «Сучасні тенденції розвитку техніки та технологій - 2023» [5] та Міжнародній науково-практичній інтернет-конференції «Розвиток науки та техніки України під час воєнного стану» (2023) [6].

**Публікації.** Основні результати досліджень опубліковано в 3 наукових працях – у матеріалах конференцій.

**Структура та обсяг роботи.** Магістерська кваліфікаційна робота складається зі вступу, п’яти розділів, висновків, списку літератури, що містить 42 найменування, 5 додатків. Робота містить 47 ілюстрації, 14 таблиць.

# 1. АНАЛІЗ СУЧАСНОГО СТАНУ ВИКОРИСТАННЯ ТЕХНОЛОГІЙ ЦЕНТРАЛІЗОВАНОГО ВХОДУ В СИСТЕМУ У СФЕРІ ОСВІТИ

## 1.1 Огляд технологій централізованого входу в систему

Наявність єдиного входу на всі сайти організації сьогодні є не лише питанням комфорту користувачів, але й підвищення кіберзахисту. Вельми велика кількість ЗВО мають крім основного сайту певну кількість додаткових підсайтів. Це можуть бути сайти конференцій, підрозділів навчального закладу, форуми, електронна система навчання (LMS) тощо. Відсутність системи єдиного входу призводить до необхідності мати на кожен підсайт окремий логін та пароль. Як наслідок, користувачі можуть постійно плутати дані входу чи реєструватись на підсайтах під різними електронними поштами, що в у свою чергу призводить до складності управління користувачами та відновлення їх доступу. Використання системи єдиного входу (SSO), що включає в себе централізований сервіс ідентифікації (IdP) здатне вирішити подібні проблеми.

З точки зору звичайного користувача система SSO виконує переважно роль форми входу. Заходячи на сайт, який працює з такою системою, користувач клікає на кнопку "ввійти в систему", яка переадресовує його на форму (рис. 1.1).

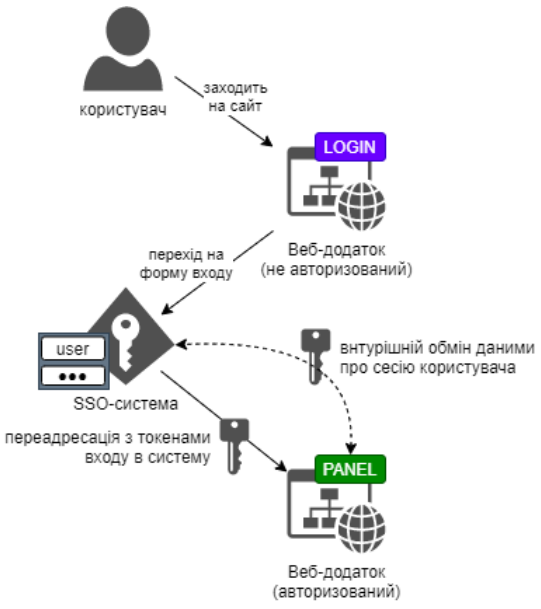


Рисунок 1.1 – Принцип роботи системи SSO для користувача

Система SSO запам’ятовує з якого сайту (чи веб-додатку) і сторінки зайшов користувач. Коли користувач вводить логін та пароль, система SSO перевіряє правильність їх комбінації у базі IdP (сервісу ідентифікації користувачів). Якщо пароль правильний, тоді система перевіряє чи налаштована у користувача мультифакторна аутентифікація (Multi-factor authentication, скор. MFA). У випадку якщо налаштована – система SSO просить ввести додаткові дані MFA. При успішному завершенні процесу автентифікації система SSO генерує одноразові токени доступу (User-Grant) та робить редірект з ними на спеціальну URL-адресу веб-додатку, на який початково заходив користувач. Під час відкриття цієї URL-адреси веб-додаток отримує токену юзер-гранту та перевіряє їх правильність на сервері SSO. В результаті веб-додаток отримує дані про авторизацію користувача та створює для нього відповідну сесію. У різних реалізаціях та протоколах даний процес може в незначній мірі відрізнятися через особливості протоколу чи додатковим процедурам кіберзахисту.

Перший етап входу називається автентифікацією. Цей процес дозволяє системі впевнитися, що за поточним запитом на автентифікацію є саме користувач, за кого він себе видає. Існує три категорії автентифікації: "що я знаю" – пароль, ключова фраза тощо; "що я маю" – фізичний ключ (смарт-карта), мікрочіп, смартфон тощо; "ким я є" – відбиток пальця, сітківки ока тощо. Зазвичай використовується, проте більш надійним способом є багатофакторна автентифікація (MFA). Автентифікацію можна назвати багатофакторною при використанні двох або більше видів підтвердження автентичності користувача [1].

За автентифікацію у системах SSO відповідає IdP. Являє собою комплекс засобів та методів, що забезпечують створення, збереження ідентифікації та керування нею, автентифікації довіреним програмам у межах SSO. Для забезпечення безпеки користувач не має доступу безпосередньо до IdP. Ідентифікація та автентифікація забезпечується через додатковий прошарковий сервіс та за допомогою протоколу передачі даних. Такий сервіс зазвичай

називають сервіс-провайдером аутентифікації та авторизації (англ. *Authentication and Authorization Service Provider,* A&A SP) [2].

Другим етапом є авторизація. Відповідно до автентифікації система перевіряє які права доступні користувачу. У разі позитивного результату користувач отримує права доступу до переліку ресурсів. За авторизацію у системах SSO відповідає сервіс-провайдер автентифікації та авторизації (A&A SP). У системах SSO він зазвичай містить саму логіку процесу авторизації, зокрема керуючись правами доступу до ресурсів. Зазвичай користувач безпосередньо має доступ до A&A SP, який у свою чергу комунікує з IdP, для забезпечення безпечної авторизації [3].

Для функціонування системи SSO необхідно забезпечити обмін даних автентифікації та авторизації між модулями. Це забезпечується за допомогою різноманітних протоколів передачі даних. Зокрема найпопулярнішими є:

- OAuth;
- SAML;
- LDAP;
- Kerberos;
- RADIUS;
- SMB Auth;
- OpenID Connect.

OAuth являє собою протокол передачі даних, що розроблявся початково для компанії Twitter, щоб забезпечити надання обмеженого доступу до облікових даних користувача стороннім додаткам. Хоч цей протокол не розроблявся початково для систем SSO, проте він є ефективним механізмом забезпечення роботи SSO у сполученні з іншими протоколами. Для забезпечення роботи SSO існують такі компоненти та концепції: токен, клієнт, сервер авторизації, власник ресурсів. При успішній автентифікації на сервері авторизації створюється токен, який отримує клієнт [7]. При запиті на доступ до даних клієнт використовує цей токен. Головними перевагами OAuth є надійність зв’язку та безпекова підтримка

з постійними оновленнями безпеки. Працює на основі XML [4]. Використовується у частині API таких відомих компаній як Facebook, Google та Microsoft.

SAML – відкритий стандарт передачі даних для автентифікації та авторизації, що спеціалізується здебільшого для забезпечення комунікації між IdP та SP. SAML як і OAuth базується на XML [8]. Значна частина використання протоколу SAML є системи SSO для веб-застосунків. Хоч у одному домені й легко забезпечити взаємодію модулів SSO, проте при використанні різних доменів це викликає безліч проблем сумісності, для вирішення яких було створено SAML. Загалом SAML розрізняє три ролі: об’єкт (зазвичай людина), IdP та A&A SP. Інструменти SAML не спеціалізована на одному виду автентифікації та здатна підтримувати багатофакторну автентифікацію. Протокол SAML може працювати у таких режимах:

- режим запиту на підтвердження;
- режим запиту автентифікації;
- режим усунення артефактів;
- режим керування ідентифікатором імен;
- режим централізованого виходу з системи;
- режим відображення ідентифікатора назви.

Як правило для передачі протокол SAML інкапсулюється у SOAP обгортку, яка у свою чергу інкапсулюється у HTTP обгортку. Для такого виду передачі рекомендується використання TLS (Transport Layer Security – захист на транспортному рівні передачі даних) та XML-підпис [8].

LDAP (Lightweight Directory Access Protocol) – відкритий стандартний протокол для отримання доступу, взаємодії та передачі даних через мережу. Не розроблявся безпосередньо для систем SSO, проте його складові доволі часто використовуються для ідентифікації та автентифікації. Має широкий функціонал для взаємодії з каталогами. При роботі із системами SSO використовується центральне управління користувачів, їх груп та правами доступу. Також

доступний функціонал синхронізації облікових записів між різними системами. Один з основних переваг протоколу є висока сумісність із іншими найпопулярнішими протоколами. Зазвичай захист реалізується за допомогою TLS [9].

Kerberos – мережевий протокол для забезпечення безпеки при передачі через незахищені й відкриті мережі та розроблявся безпосередньо для реалізації єдиного входу для SSO систем. Основа реалізація єдиного входу реалізується через одноразову аунтифікацію, у результаті якої сервер автентифікації надає запит серверу надання токенів, який у свою чергу надає токен користувачу. У результаті користувач із токеном у наступні рази надає токен для ідентифікації без потреби автентифікації для отримання доступу до різних ресурсів під час одного сеансу [10]. Кожен токен має обмежений термін дії, що покращує безпеку. Протокол Kerberos базується на шифруванні з використанням синхронного ключа та підтвердження третьою стороною. Зазвичай передається по UDP. Підтримується багатьма UNIX-подібними системами та ОС Windows.

RADIUS (Remote Authentication Dial-In User Service) – мережевий клієнт-серверний протокол для централізованої автентифікації, авторизації, управління обліковими записами користувачів та їх синхронізації. Використовується у парі із іншими протоколами для доповнення можливостей, зазвичай OAuth або SAML. RADIUS має засоби для обліку використання мережевих ресурсів. Працює на прикладному рівні через TCP та UDP [11]. Як правило реалізується у вигляді системної служби у Windows та Unix-подібних операційних системах, а також підтримується цілим рядом мережевого обладнання [12].

SMB (Server Message Block) – протокол комунікації, здебільшого використовується на ОС Windows для передачі даних та комунікації. Крім передачі файлів цей протокол також використовується для автентифікації та авторизації [13]. Має можливість передаватися на транспортному рівні через TCP/IP, NetBT та QUIC. За свою історію протокол мав досить багато виявлених

вразливостей. Це стало потужним рушієм для покращення нових версій протоколу з багатшим додатковим функціоналом по сьогодні.

OIDC (OpenID Connect) – відкритий протокол автентифікації та авторизації, базується на основі OAuth версії 2.0. Основне призначення полягає у забезпеченні високого рівня безпеки із стандартизацією механізму обміну інформації про ідентичність користувачів між IdP та користувачами. Для реалізації дані упаковуються у JSON токен, шифрується та підписується [14]. Інформація коли і як запакований токен має бути присутня у відповіді.

Отож для закладів ЗВО системи SSO мають надважливу роль як для забезпечення безпеки, так і зручності освітнього процесу. Для цього необхідно правильно підібрати всі частини, які забезпечують роботу SSO, а саме: сервіс-провайдер A&A, IdP та протоколи. Саме вдале поєднання цих частин дозволяє забезпечити достатній рівень гнучкості, безпеки та зручності.

## 1.2 Аналіз аналогів інформаційної системи централізованої авторизації для освітнього веб-середовища навчального закладу

Існує два варіанти розгортання SSO-систем: розміщення на власному сервері (self-hosted) та хмарне (cloud-hosted). Для розгортання self-hosted системи, яка могла б поєднати у собі функціонал як IdP-систем так і A&A SP-систем необхідно будувати та налаштовувати досить складну інфраструктуру з різних програмних рішень, перед чим необхідно виконати достатньо ретельний аналіз існуючої IT-інфраструктури та розробку архітектури. Для супроводу подібних систем потрібно мати штат висококваліфікованих DevOps-інженерів та системних адміністраторів, що далеко не завжди можливо для IT-відділів навчальних закладів, зокрема і ЗВО України технічного спрямування [4]. На відміну від self-hosted програмних рішень, хмарні SSO поєднують у собі функціонал IdP та SP. Перевагами хмарних рішень є те, що їх значно простіше налаштовувати, у них є досить потужна технічна підтримка. Крім того, компанії, що надають хмарний сервіс SSO постійно слідкують за кібербезпекою як свого

програмного забезпечення у хмарі, в чому значно виграють у self-hosted рішень. Серед self-hosted IdP систем можна виділити наступні: Apache Directory, FreeIPA, Microsoft Active Directory та OpenLDAP. Серед self-hosted сервіс-провайдерів A&A можна виділити наступні: SimpleSAMLphp, OpenAM та Shibboleth.

Apache Directory – безкоштовний кросплатформне рішення написане на Java, що дозволяє легко інтегрувати його із багатьма існуючими рішеннями, а також доповнювати власноруч. Окрім підтримки на таких операційних системах як Linux, Mac OS X та Windows, доступна більшість з популярних форматів для кожної з операційних вказаних систем. Реалізовано підтримку протоколів LDAP (як основний) та Kerberos (як додатковий, опціональний), а також використовується протокол захищеної зміни паролів. Для конфігурування використовуються файли LDIF, що є популярним рішенням для рішень з підтримкою протоколу LDAP [15]. Таке рішення допомагає легшому конфігуруванню людей, що раніше працювали з рішеннями, що підтримують LDAP. Також існує Apache Directory Studio, що базується на Eclipse, та працює практично з усіма LDAP-серверами. Він дозволяє переглядати, модифікувати, створювати та видаляти записи у зручному середовищі, з підсвічуванням синтаксису та інформативними підказками згідно контексту, працювати із схемами формату OpenLDAP. Також має редактор контролю доступу, що дозволяє керувати правами доступу графічно чи вручну.

FreeIPA – це безкоштовна система керування ідентифікацією з відкритим кодом, що забезпечує централізовану систему ідентифікації, політики безпеки та аудиту. Написана на мовах C та Python для операційних систем на базі Linux/UNIX, проте новіші версії дозволяють використання для Windows і Mac OS X, та інтеграцію із Microsoft Active Directory за допомогою Cross Forest Trust. Використовує протоколи Kerberos або SAML для автентифікації користувачів і безпеки взаємодії між різними компонентами, LDAP для зберігання інформації у директорії. Також FreeIPA має власний протокол для управління

сертифікатами безпеки [16]. Керування доступом та адміністрування мережі здійснюється за допомогою веб-інтерфейсу або командного рядка.

Microsoft Active Directory – служба каталогів розроблена компанією Microsoft для операційних систем Windows Server, постачається як базова функція ліцензії Windows Server у вигляді набору процесів та служб. Хоч початково Active Directory не розроблявся для цілей SSO у якості IdP, проте з часом Active Directory оновлювався та став незамінною складовою для багатьох SSO на базі Windows Server, зокрема і для інших продуктів Microsoft. Використовує протоколи LDAP та Kerberos. Підтримує багатофакторну аутентифікацію для забезпечення як двофакторної автентифікації, так і для інших методів автентифікації. Має безліч функцій, таких як служба реплікації для розподілу даних каталогу у мережі, механізми записів та індексації, набір схем та правил для визначення об’єктів та атрибутів у каталогах, централізоване управління обліковими записами користувачів та групами [17]. Всі ці функції забезпечують зручність у налаштуванні та підтримці.

OpenLDAP – базова відкрита реалізація IdP на основі протоколу LDAP, елементи та стандарти якої використовуються й іншими IdP. Хоч це і базова реалізація, яка не може похвастатися багатьма додатковими функціями та зручностями, проте забезпечення підтримка на практично всіх більш чи менш популярних операційних системах, таких як Linux, Unix (HP-UX, AIX), Android, macOS, OpenVMS, Solaris, Microsoft Windows та z/OS. Ліцензія OpenLDAP дозволяє як безкоштовне використання, так можливість зміни та розповсюдження. Такий підхід дозволив забезпечити широке використання та популяризацію даної IdP. Для доповнення можливостей використовуються додаткові протоколи, такі як SAML, OAuth або OpenID Connect. OpenLDAP був розроблений для високої продуктивності та ефективної маштабованості, підтримує різні види автентифікацій та має завдяки можливості модифікації має безліч клієнтських бібліотек для реалізації додаткових можливостей [18].

SimpleSAMLphp – відкритий та безкоштовний PHP-додаток, ліцензія якого дозволяє використовувати, модифікувати та розповсюджувати. За удосконалення беруться ентузіасти, які розвивають додаткові можливості. SimpleSAMLphp підтримує протоколи SAML, LDAP та Kerberos. Така шикора підтримка дозволяє має широкий вибір IdP для побудови SSO. Також є підтримка метаданих, інструменти для забезпечення захисту інформації, введення аудиту подій для відстеження дій користувачів та системних подій. Основною перевагою SimpleSAMLphp є відносна простота інсталяції, налаштування та підтримки, що при наявності кваліфікованих фахівців дозволить інтегрувати його із різними додатками та сервісами [19].

OpenAM – відкритий Java-додаток, ліцензія якого дозволяє використовувати, модифікувати та розповсюджувати, використовується у якості SP для систем SSO. OpenAM підтримує протоколи SAML, OAuth, Kerberos та декілька менш відомих, що дозволяє бути універсальним до поєднання з різноманітними IdP. Також має гнучку архітектуру, що дозволяє легко розширити можливості при підключенні потрібних плагінів чи розширень. Перевагами OpenAM є наявність інтеграції з різноманітними сервісами та додатками, зручних засобів управління користувачами та керування доступом до ресурсів, збору аналітичних даних й генерації звітів [20].

Shibboleth – відкритий Java-додаток на базі протоколу SAML який використовується як SP для систем SSO. Також компанією був розроблений протокол Shibboleth, який у поєднанні із SAML забезпечує роботу систем SSO. Також є опціональна підтримка Kerberos. Із переваг данна A&A SP дозволяє передавати атрибути користувачів у обхід контенту, має розширені можливості управління доступом до ресурсів, легко піддаються масштабованості при збереженні високої продуктивності. Варто відмітити, що Shibboleth добре підходить для об’єднання існуючих SSO у так звану федерацію, проте це реалізувати складніше у порівнянні з аналогами [21].

Реалізації сервіс-провайдерів A&A, найбільш близький по функціоналу до системи, що має бути розроблена зведено у таблиці 1.1.

Таблиця 1.1 – Порівняння аналогів

| **Найменування** | **SimpleSAMLphp** | **OpenAM** | **Shibboleth** |
|---|---|---|---|
| Мова | PHP | Java | Java |
| Складність впровадження | висока | висока | висока |
| Підтримка протоколів | SAML,<br>LDAP,<br>Kerberos | SAML,<br>OAuth,<br>Kerberos | SAML, Shibboleth,<br>Kerberos |
| Підтримка багатофакторної автентифікації | у залежності від конфігурації | у залежності від конфігурації | у залежності від конфігурації |
| Масштабованість | у залежності від конфігурації та вимог може варіюватися від середньої до високої, підтримка кластерів | висока, підтримка кластерів | висока, підтримка кластерів |
| Складність об'єднання SSO у федерації | висока | у залежності від конфігурації | висока |

До аналогів можна також віднести попередню версію системи SSO для JetIQ та її IdP API-модулі. На даний момент система лише частково реалізована, має підтримку протоколів SAML та OAuth. Технічне завдання (додаток А) до даної роботи передбачає її удосконалення якщо це можливо або повне переписання її коду. Крім функціоналу self-hosted систем, система повинна мати також інтеграцію з різними хмарними сервісами, подібно до того як це реалізовано у марних SSO.

Варто зазначити і присутність хмарних рішень для SSO. Зазвичай вони поставляються як комплект готового IdP, SP та протоколів їх взаємодії, рідше з можливістю вибору компонентів. У загальному основний плюс таких рішень є забезпечення стабільності роботи, покращення легкості масштабованості та

забезпечення високого рівня безпеки. Також такі рішення мають і свої недоліки, серед яких найбільш популярними є непроста інтеграція self-hosted ресурсів, доволі висока вартість та неможливість швидкого відновлення доступу при некоректному налаштуванні правил доступу [4].

Okta – одне із найпопулярніших рішень для хмарної реалізації SSO, що має 2 вектори розвитку: для робочих середовищ IT-компаній задля забезпечення автентифікації та авторизації співробітників та для компаній для автентифікації користувачів. Okta має широкий вибір для інтеграції із існуючими робочими середовищами, зокрема Slack, Workday, Zendesk та інші. З переваг має широку підтримку різних видів багатофакторної автентифікації. є об’ємні можливості управління доступом до ресурсів, використання API для забезпечення легкої інтеграції безпечної авторизації, автоматизована система управління та реєстрації користувачів, зручний моніторинг активності та потенційних загроз. Okta використовує такі протоколи як SAML, OpenID Connect, RADIUS та LDAP. Працює система SSO на основі підписки на кожного користувача. Має лише два види платних підписок: стандартна та розширена. Розширена підписка відрізняється моніторингом багатьох аспектів за допомогою штучного інтелекту при відстеженні та аналізі контекстів локації, даних пристрою, даних мережі та багато іншого [22].

Microsoft Entra ID – система ідентифікації та управління хмарної екосистеми компанії Microsoft, що використовується і у якості SSO. Ця система здебільшого орієнтується на Microsoft 365. Є підтримка протоколів SAML, OAuth, Kerberos, LDAP та OpenID Connect. Entra ID має 3 види підписок та 1 розширення для керування ідентичностями. Існує базова, Premium P1 та Premium P2 підписки. Premium P2 відрізняється від Premium P1 захистом ідентичностей та розширеними можливостями самообслуговування кінцевих користувачів. Premium P1 відрізняється від базової розширеними можливостями доступу та керування програмами, адміністрування, самообслуговування кінцевих користувачів, функціями умовного доступу, керування ідентичностями та

покращених звітів. Маючи досить плюсів Entra ID має досить багато мінусів, зокрема найбільшим є обмежена підтримка інтеграції з іншими хмарними застосунками, що не входять у екосистему Microsoft. Хоч базові функції Entra ID є безкоштовними для освітніх підписок Office 365 [23], проте ці функції є досить обмеженими. Активно впроваджена дана система і у деяких українських ЗВО [24].

OneLogin – бізнес-орієнтоване хмарне рішення SSO, яке легко інтегрується із більшістю існуючих ресурсів та середовищ, які використовуються у бізнесі різного масштабу. Також підтримує багатофакторну автентифікацію. Для забезпечення роботи SSO використовуються протоколи SAML, OpenID Connect та OAuth. Із переваг одночасно підтримує автентифікацію декількох облікових записів для застосунків, зокрема наявна можливість надати доступ через свій ідентифікатор іншій людині на певний час, для користувачів є підтримка автентифікації через Google аккаунт та найпопулярніші соціальні мережі, підтримка понад двадцяти мов інтерфейсу.

Для закладів ЗВО хмарні рішення мають досить багато переваг, проте не є доступними через дороговизну підписки на кожного студента чи викладача. Також важливим є їх недоступність для бюджетних ЗВО, так як вони не продають свої послуги на майданчиках держзакупівель України. Єдиним хмарним варіантом для бюджетних ЗВО є Microsoft Entra ID, що має безкоштовну базову підписку за умови, що заклад вже використовує хмарні продукти Microsoft. Проте можливостей у цієї підписки недостатньо для повноцінної реалізації SSO для освітніх потреб, щоб забезпечити важливі функції як умовний доступ тощо. Крім того, якщо навіть навчальний заклад і використовує цей сервіс, необхідно правильно розрахувати безкоштовний ліміт запитів Graph API щоб забезпечити всі потреби сучасного ЗВО.

Отож Self-hosted системи мають високий рівень функціональності та гнучкості, проте є досить складними для реалізації, оскільки потрібно підбирати та поєднувати SP, IdP та протоколи для забезпечення їх роботи та взаємодії.

Хмарні рішення є більш доступними у плані простоти розгортання та експлуатації, мають зручні готові інструменти для аналізу подій та потенційних небезпек, проте ці рішення не є доступними для бюджетних ЗВО.

### 1.3 Обґрунтування доцільності удосконалення методів та засобів централізованої автентифікації, авторизації та ідентифікації у сфері освіти

Доцільність використання та удосконалення методів та засобів централізованої автентифікації, авторизації та ідентифікації у сфері освіти обумовлені активним розвитком інформаційних технологій, а також великим розривом у якості та захищеності хмарних та self-hosted SSO-сервісів. Успіхи хмарних провайдерів в удосконаленні своїх послуг надання сервісу SSO є безумовно значними. Їх послуги є затребувані компаніями та організаціями різного масштабу, і як наслідок їх продукти повністю відповідають стандартам IT-індустрії. Однак для self-hosted рішень на даний момент подібна гнучкість не може бути забезпечена без кваліфікованого конфігурування, розгортання та супроводу подібних систем, а також без використання спеціальних плагінів. Для досягнення надійності та захищеності у self-hosted рішень до рівня хмарних аналогів потрібно постійно займатись удосконаленням методів та засобів централізованого входу у систему.

Однією з головних проблем всіх SSO сервісів є захист від кібератак. Оскільки використання систем SSO передбачає централізацію входу у всі цифрові сервіси організації, то всі проблеми з безпекою даного процесу теж зосереджуються у одному місці. З однієї сторони це навіть добре, оскільки потрібно менше зусиль для кіберзахисту IT-інфраструктури [5], проте з іншої сторони у такого підходу є і мінуси:

- необхідно мати більш кваліфікованих співробітників з IT-безпеки;
- будучи центральною точкою входу в систему, SSO стає єдиною точкою відмови (англ. *Single point of failure, SPOF*). Навіть при достатній кількості серверних чи хмарних потужностей для гарячого резервування при

настанні кіберінциденту робота системи SSO паралізується, і як наслідок це призводить до зупинки роботи всіх сервісів, які від неї залежать.

Найбільш критично важливими компонентами SSO-системи є база даних, API-ядро та сервіс ідентифікації користувачів (IdP). При відмові однієї з цих компонент робота всієї системи та всіх залежних від неї сервісів повністю паралізується. Якщо для баз даних та сервісів IdP ще вироблені загальні принципи кіберзахисту та оптимальних підходів до архітектури, то з REST API-інтерфейсами не існує чіткої стандартизації цих процесів. Для REST API-інтерфейсів існує лише загальний набір загальноприйнятих правил з програмного дизайну REST API [25]:

- запит та відповідь API мають бути у форматі JSON;
- мають бути використані іменники замість дієслів у шляхах ендпоїнтів;
- збірки з іменниками мають бути названі у множині;
- реалізація вкладених ресурсів для ієрархічних об’єктів;
- ретельне забезпечення обробки помилок та стандартні HTTP-коди помилок;
- фільтрація, сортування та поділ на сторінки;
- дотримання належних практик кібербезпеки;
- кешування дані для підвищення продуктивності;
- версіонування API (версія API).

Найкращим втіленням цих рекомендації серед публічних API можна вважати Microsoft Graph API [26]. Він є достатньо якісно задокументованим, до нього легко приєднувати нові сервіси, а також з ним легко вміють працювати інтелектуальні боти, що здатні створювати програмний код [27]. Хоча методики захисту цього сервісу від кіберзагроз ніде публічно не задокументовані, відомо лише що всі подібні API-сервіси використовують архітектурний принцип "security-first" [28]. Він полягає у тому що стратегії кіберзахисту програмного забезпечення закладені у самі парадигми архітектури програмного забезпечення. Згідно їм, дані мають бути захищені під час обробки, зберігання та передачі.

Оскільки не існує чітких стандартизованих "security-first" методів із проектування API-інтерфейсів, а вже існуючі не повністю розкривають їх застосування у SSO-системах поза межами функціоналу стандартних мережевих протоколів, тому потребують розвитку методи проектування API-сервісів систем технології централізованої автентифікації, авторизації та ідентифікації.

Іншою безпековою проблемою SSO-систем є високе навантаження від шкідливих ботів, які можуть як здійснювати перебір паролів, так і перевантажувати систему важкими запитами, що насправді є одним із видів DDoS-атак. До прикладу пілотна версія SSO-системи для JetIQ має функцію автодоповнення логіну. Однак її дуже легко перевантажити важкими запитами, що може призвести до припинення можливості входу користувачів у систему. Саме для цього потрібно розвинути концепцію Zero Trust, розробивши на її основі метод визначення довіри до клієнтського пристрою в залежності від його поведінки.

Zero Trust (укр. *Нульова довіра*) – це концепція в інформаційній безпеці, яка передбачає, що не слід довіряти жодному користувачеві чи пристрою в мережі, навіть якщо вони знаходяться у межах власної корпоративної мережі. Замість того, щоб виходити з припущення, що все, що перебуває в корпоративній мережі, є довіреним, концепція Zero Trust вимагає постійного перевірки і підтвердження ідентифікації та дозволів для кожного користувача та пристрою, навіть тих, які вже перебувають в мережі. Впровадження Zero Trust - це неперервний процес, і важливо регулярно переглядати та оновлювати стратегію безпеки, враховуючи нові загрози та технології. Нині пропонується використовувати хмарні рішення для впровадження його використання на веб-сайтах (такі як CloudFlare WAF чи Cisco Zero Trust Security) [29], однак вони залежать від SSO-рішень, розробленими іншими компаніями та створюють залежність від одного постачальника, тому розробка нового методу є виправданою.

Іншою проблемою SSO-систем є необхідність захисту від спроб увійти у систему ботам. Захист від перебору паролів ботами є безумовно необхідним, але він сам по собі не є достатнім, якщо пропускатиме ботів, які вже знають правильні дані для входу. У подібному випадку виникає як мінімум два досить неприйнятні для будь-якого веб-додатку ризики кібербезпеки:

- маючи правильні дані для входу та успішно скориставшись ними їх у системі SSO, шкодоносний бот зможе збільшити свої можливості для виявлення слабких місць додатків, які знаходяться за SSO;
- у подібній ситуації бот, який дає своїм вигодоотримувачам масовий доступ до різних акаунтів може порушити прийнятні умови використання веб-додатку, не наносячи серйозної шкоди самому додатку. Однак це може відкрити для його вигодоотримувачів не передбачені правилами сервісу переваги, які наприклад у освітньому процесі можуть бути недопустимими (наприклад розгадка відповідей тестів, чи отримання доступу до матеріалів, які викладач ще не опублікував).

Лише ці два кейси вимагають значних зусиль для їх запобігання. Найбільш доцільно зробити це спочатку на стороні SSO, оскільки саме він є першим рубежем, на якому необхідно присікати активності ботів. З іншої сторони, у адміністратора веб-додатків завжди залишиться правильний шлях для налагодження взаємодії з правильними ботами – завдяки правильно спроектованому з точки зору кібербезпеки API. А для подолання активності потенційно небажаних ботів необхідно застосовувати такі підходи як JavaScript Challenge разом із тестом Тюрінга квестового характеру [29]. Однак нині деякі тести Тюрінга нині успішно долаються засобами комп’ютерного зору та загального штучного інтелекту (англ. *Artificial general intelligence*, AGI). Концепція боротьби ключа та відмички передбачає що засоби кіберзахисту як і засоби його обходу будуть постійно вдосконалюватись, конкуруючи між собою. Саме тому є сенс у вдосконаленні не лише методів JavaScript Challenge, а й доповнюючі їх тести Тюрінга квестового характеру, і як наслідок є сенс у

розробці нової моделі гейміфікації публічного тесту Тюринга, яка зможе поліпшити засоби захисту від ботів.

Одним із важливих питань реалізації вищезгаданих методів та моделей є вибір між реалізацією повністю нової SSO-системи з ними, чи удосконалення вже існуючої. Технічне завдання даної роботи (додаток А) передбачає розгляд можливостей існуючих систем щодо їх вдосконалення.

Насамперед важливо розуміти що створити нову SSO-систему повністю з нуля надзвичайно складно і є непосильним завданням не лише для одного розробника, а й для цілої команди. Тому важливо максимально використати найбільше можливостей вже існуючих програмних бібліотек, у тій мірі наскільки це дозволятимуть їх ліцензії (наприклад GNU LGPL та MIT). Наприклад, ліцензія MIT надає широкі права щодо використання, модифікації та поширення програмного забезпечення, яке вона охоплює. Це сприяє швидкому розповсюдженню та використанню коду в різних проектах. Більше того окремі модулі розглянутих у попередньому розділі програмні аналоги SSO-систем виділені у окремі бібліотеки, наприклад: OAuth2 від PHPLeague, SAML PHP Toolkit від OneLogin та ShibAuthBundle від SimpleSAMLphp. Це дозволяє не обмежуватись архітектурою вже наявних рішень, а інтегрувати їх ключовий функціонал у свою.

Хоча пілотна версія 0.2 SSO-системи для JetIQ не відповідає вимогам до її програмної архітектури, однак її IdP-модулі APIv1 та APIv2 цілком можуть бути вдосконалені та використані для створення нової SSO-системи, цілком відкинувши необхідність у створенні нового IdP-модуля для неї. У той самий час можливостей готових A&A SP-систем, таких як SimpleSAMLphp, OpenAM та Shibboleth не достатньо для задоволення вимог SSO-системи для JetIQ, оскільки для їх реалізації знадобиться суттєво змінювати програмну архітектуру цих систем, що не є виправданим у порівнянні із використанням їх відокремлених бібліотек у новій програмній архітектурі. До переваг створення нового A&A SP-

рішення можна віднести можливість впровадження більш сучасних технологій та запропонованих методів та моделі.

Отже на базі IdP-компонентів SSO-системи для JetIQ та базових бібліотек деяких аналогів з відкритим поточним кодом цілком можливо створити нову SSO-систему, у якій будуть реалізовані удосконалені методи і моделі, що розробляються у цій роботі. Обрано найбільш оптимальний підхід, який дозволяє не переписувати велику кількість з нуля і при тому залишити гнучкість у створенні нової архітектури.

### 1.4 Постановка завдання дослідження та удосконалення методів та засобів централізованої автентифікації, авторизації та ідентифікації

Завдяки виконанню аналізу предметної області, основних понять автентифікації, авторизації та ідентифікації, аналізу аналогів та модулів центральної авторизації, що використовуються у ВНТУ стало можливим формування основних завдань для дослідження. Серед них ключовими є наступні:

- провести аналіз існуючих методів і засобів централізованої автентифікації, авторизації та ідентифікації;
- виконати аналіз існуючих інформаційних систем централізованого входу в систему, а також ефективність їх використання у сфері освіти;
- запропонувати покращення для програмних компонентів існуючих систем, які можна удосконалити та використати у новій розробці;
- розробити нові методи для вдосконалення функціонування централізованої автентифікації, авторизації та ідентифікації у сфері освіти;
- розробити модель гейміфікації для публічного тесту Тюринга;
- розробити програмну архітектуру та компоненти системи централізованої авторизації для освітнього веб-середовища навчального закладу;
- провести експериментальні дослідження кіберзахищеності та працездатності розробленої системи.

## 1.5 Висновки

У першому розділі виконано аналіз предметної області, розглянуто існуючі аналоги, обґрунтовано доцільність розробки та поставлено завдання дослідження. В ході аналізу можливостей програмних бібліотек, які використовують існуючі аналоги встановлено що існує можливість використання їх переваг разом із вдосконаленими архітектурними підходами, запропонованими методами та моделлю.

Згідно технічному завданню до даної роботи (додаток А) наступним чином необхідно проаналізувати можливості вдосконалення існуючої системи SSO для JetIQ, або якщо це неможливо спроектувати повністю сумісну з нею нову SSO-систему з реалізацією у ній притаманним існуючим аналогам переваг та нових методів, а також адаптувати цю систему до роботи з системою управління навчальним процесом JetIQ.

# 2. ПРОЕКТУВАННЯ АРХІТЕКТУРИ, МЕТОДІВ, ЗАСОБІВ ДЛЯ ЦЕНТРАЛІЗОВАНОГО ВХОДУ В СИСТЕМУ

## 2.1 Вибір підходів та засобів для створення архітектури веб-додатків

З огляду на розглянуті технології централізованого входу у першому розділі (пункт 1.1), їх аналогів (пункт 1.2) та їх можливостей їх компонентів (пункт 1.3) можна зрозуміти основні вимоги до програмної архітектури. Ці вимоги передбачають що існує два обов'язкових компоненти у такій системі. Типова архітектура SSO-систем складається з двох компонентів: сервіс-провайдеру A&A та сервісу IdP (рис. 2.1).

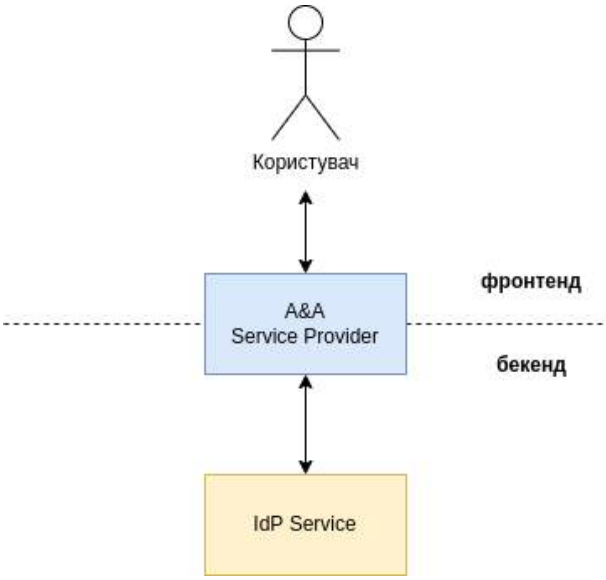


Рисунок 2.1 – Типова структура SSO-системи

Користувач має прямий доступ до сервіс-провайдеру A&A, який виступає в ролі посередника до сервісу IdP. Для цього сервіс-провайдер поділений на бекенд- та фронтенд-частини. Щоб зрозуміти як правильно реалізувати та поєднати ці компоненти необхідно розглянути які архітектури існують для веб-додатків, і які з них найбільше підходять для проектування SSO-систем.

Проектування інформаційних систем поділяється на дві області: розробка програмної архітектури всієї системи та дизайн програмних компонентів (під

цим терміном мається на увазі їх проектування). У той час як архітектура займається загальною структурою системи, зв’язками її структурних елементів та якісними характеристиками системи, до як дизайн зосереджується на структурній організації кожного структурного елементу, його програмній реалізації [30]. Більш детально їх спільні та відмінні риси наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння підходів для програмної архітектури та дизайну

| Аспекти | Патерни програмної архітектури | Патерни програмного дизайну (проектування) |
|---|---|---|
| **Область застосування** | Високорівнева структура всієї системи | Вузькопрофільні рішення у вигляді модулів та компонентів |
| **Рівень абстракції** | Макрорівень загального плану | Мікрорівень дизайнерських рішень |
| **Призначення** | Визначення структури системи та її компонентів | Створення рішень поточних питань проектування |
| **Вплив** | Загальна системна масштабованість та швидкодія | Реюзабельність та відновлюваність окремих модулів та компонентів |
| **Деталізація** | Взаємодія між компонентами/модулями на рівні системи | Рішення проектування окремих компонентів/модулів |
| **Використання** | Спрямовує побудову всієї системи | Детальне проектування кожного компоненту |
| **Приклади** | Мікросервіси, шарова та клієнт-серверна архітектура | Шаблони ООП і модульного дизайну |

Серед архітектурних стилів, які застосовуються для створення веб-додатків та SSO-систем можна виділити наступні: багатошаровий (layered), клієнт-серверний (client-server), подіє-орієнтований (event-driven), мікроядро (microkernel), мікросервіси (microservices), брокер (broker), шина подій (event-bus), класна дошка (blackboard), компонентно-орієнтований (component-based) та черги повідомлень (message queue).

Для сервісів SSO застосовується декілька архітектурних стилів. Перш за все клієнт-серверний, оскільки інші додатки підключаються до IdP-серверу, та

до сервіс-провайдеру A&A у ролі клієнтів. Оскільки систему SSO сама по собі поділяється як мінімум на дві складові з чітким підпорядкуванням – IdP та A&A SP, то найбільш доцільно буде використати для цього багатошарову (layered) архітектуру. Крім двох шарів у системі можуть бути компоненти, створені на різних платформах чи мовах програмування, а також зовнішні компоненти, такі як бази даних, які при розгортанні будуть зібрані у вигляді окремих контейнерах. При розгортанні у вигляді контейнерів необхідно застосовувати мікросервісну архітектуру (microservices), що має бути передбачити на етапі проектування. Для обміну інформацією між мікросервісами в деяких сценаріях може застосовуватись підхід черги повідомлень (message queue).

Згідно технічного завдання при розробці необхідно використати раніше розроблений мікрофреймворк IQCoreApp, а також деякі напрацювання попередньої версії системи SSO для JetIQ, які як і мікрофремворк, написані мовою програмування PHP, що означає що вона і надалі залишатиметься в проєкті основною мовою програмування. Іншим аргументом на користь PHP є необхідність використання написаних на ньому бібліотек OAuth2 від PHPLeague, SAML PHP Toolkit від OneLogin та ShibAuthBundle від SimpleSAMLphp. Передбачається використання середовища розробки Visual Studio Code.

Крім PHP у попередній реалізації існують окремі компоненти, написані мовами програмування Python та Java. Можливо їх доведеться і надалі залишити в проєкті. Для реалізації фронтенд-частини передбачено використання мови програмування JavaScript, яка є невід’ємною частиною фронтенду. Для роботи над фронтенд частиною передбачається використання Bootstrap 5 та JavaScript-бібліотеки jQuery 3.5.

### 2.2 Проектування бізнес-процесів системи централізованої авторизації

У проектуванні систем SSO важливо дослідити не лише процес входу користувачів у систему, а й процеси комунікації між різними групами осіб, які призводять до отримання та/або створення даних для входу (англ. *login credentials*) користувачів у систему. На основі цих комунікацій відбуваються регулярно повторюємі та чітко систематизовані дії, що звуться бізнес-процесами.

Для кращого розуміння бізнес-процесів спершу потрібно розуміти які особи або групи осіб у них залучені, оскільки всі бізнес-процеси обертаються навколо людей та/чи їх груп. Згідно дослідженням проведених в системі управління навчальним процесом JetIQ, є чотири основних груп осіб (т.з. акторів), від яких залежать процеси, що впливають на вхід у систему: кінцеві користувачі (студенти та викладачі), співробітники деканатів та відділу кадрів, а також адміністратор JetIQ (рис. 2.2). Подібна комунікація має дві сторони у передачі даних для входу: надавачі даних (співробітники) та реципієнти даних (кінцеві користувачі).

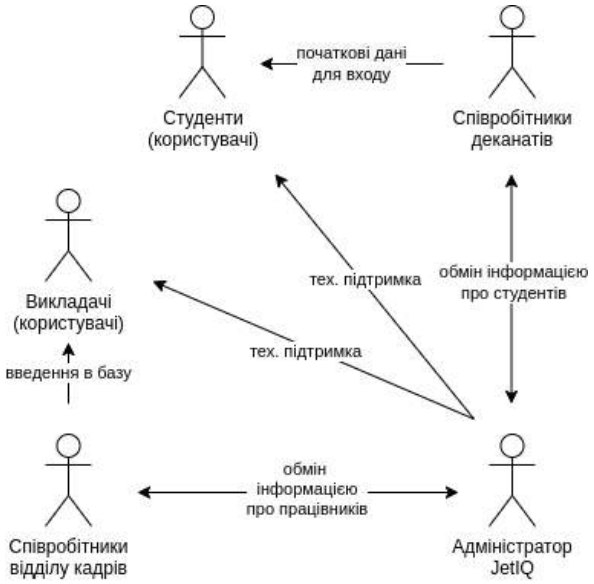


Рисунок 2.2 – Схема комунікації між групами осіб,
задіяних у наданні доступу кінцевим користувачам

Згідно проведених досліджень у системі управління навчальним процесом JetIQ існує два канали отримання даних для входу у систему:

- співробітники деканату та/або відділу кадрів. Співробітниками деканату імпортуються у систему електронного деканату дані студентів із загальнонаціональної освітньої системи ЄДЕБО (яка у свою чергу бере їх із системи Дія). В ході цього процесу для кожного користувача має автоматизовано створюватись новий обліковий запис у сервісі IdP. Для співробітників цей процес починається з відділу кадрів: вони додають нового співробітника у системі відділу кадрів, яка у свою чергу передає дані у сервіс IdP (рис. 2.3) і надсилає їх на електронну пошту.
- технічна підтримка JetIQ. Користувачі через неї можуть отримати нові дані для входу (якщо вони не були отримані раніше) чи збити пароль.

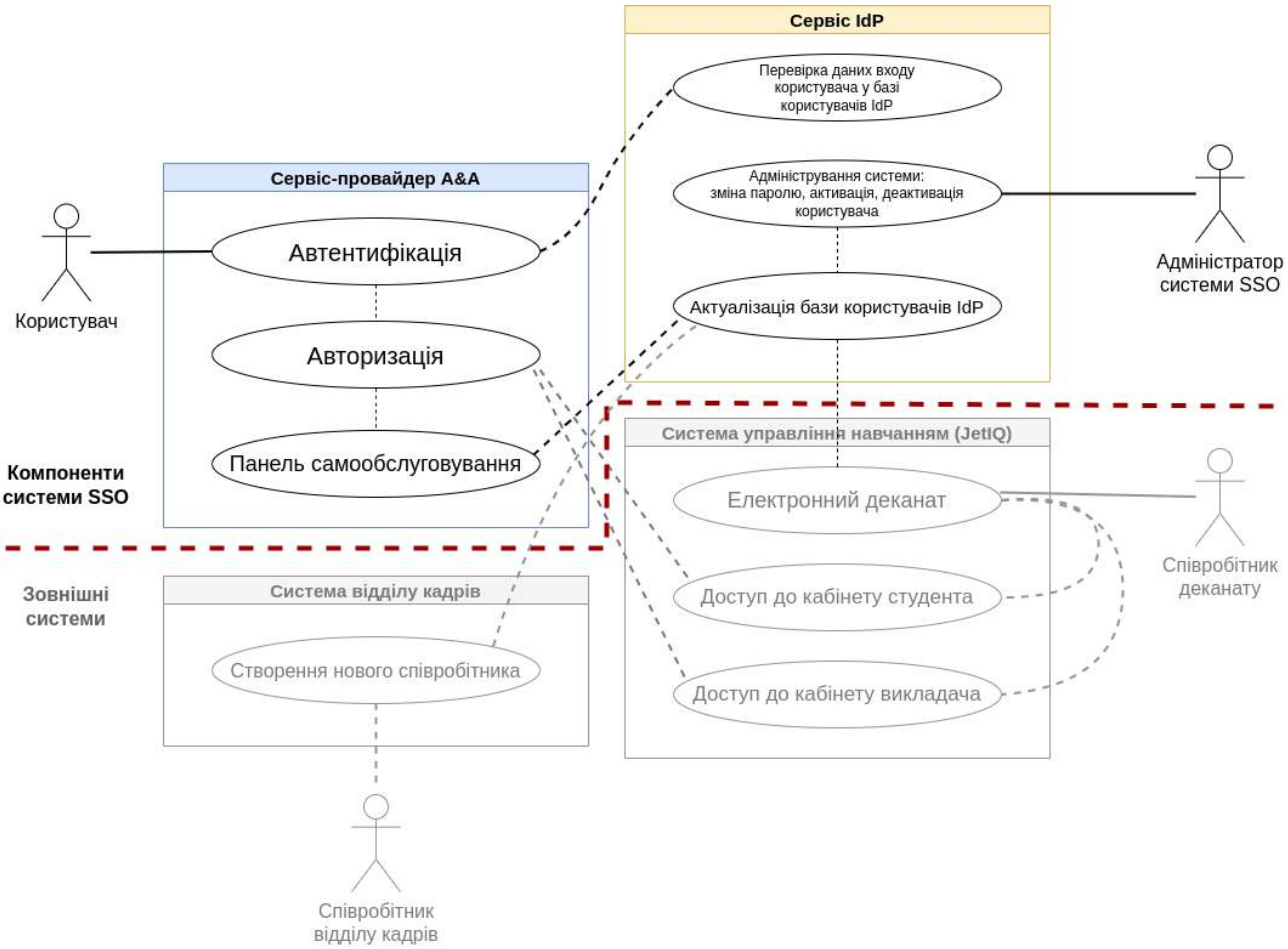


Рисунок 2.3 – Use Case Діаграма

На рисунку 2.3 показано дві ролі які безпосередньо взаємодіють із системою SSO: кінцеві користувачі (студенти та викладачі), а також адміністратор системи SSO. Кінцеві користувачі лише взаємодіють із фасадом системи SSO – сервіс-провайдером A&A, в той час як адміністратор має безпосередній доступ до бази користувачів IdP. Інші дві ролі – співробітники деканату та відділу кадрів хоча і не мають безпосереднього зв’язку з системою SSO, однак мають вплив на неї за посередництвом систем, з якими вони працюють. Однак при вході у систему SSO всі чотири ролі виступають у ролі кінцевого користувача і лише в ході процесу авторизації набувають всіх необхідних їм прав для того щоб виступати у властивій для них ролі.

Процес входу користувача. Першим чином користувач входить у систему проходячи процес автентифікації. У даному процесі задіяні обидві складові системи SSO – як сервіс-провайдер A&A, так і сервіс IdP. Згідно Вимог до програмної архітектури SSO-системи для JetIQ процес введення облікових даних користувача розділено на три стадії: проходження теста Тюринга (за необхідності), введення логіну (мається на увазі *username*, рис. 2.4) та введення паролю.


Рисунок 2.4 – Процес верифікації логіну (мається на увазі *username*)

На рисунку 2.4 зображено схему першої стадії автентифікації, де відбувається верифікація логіну користувача. Вимоги до даної системи передбачають введення три види "логінів":

- числовий ідентифікатор співробітника (з автодоповненням);
- код залікової книжки студента (з автодоповненням);
- повне прізвище, ім’я та по-батькові викладача (з автодоповненням);
- нікнейм студента (набір латинських букв, чисел, дефісу та підкреслення).

Наступною стадією входу у систему є перевірку паролю (рис. 2.5). Вона розпочинається за умови якщо користувач вже успішно пройшов перевірку логіну.

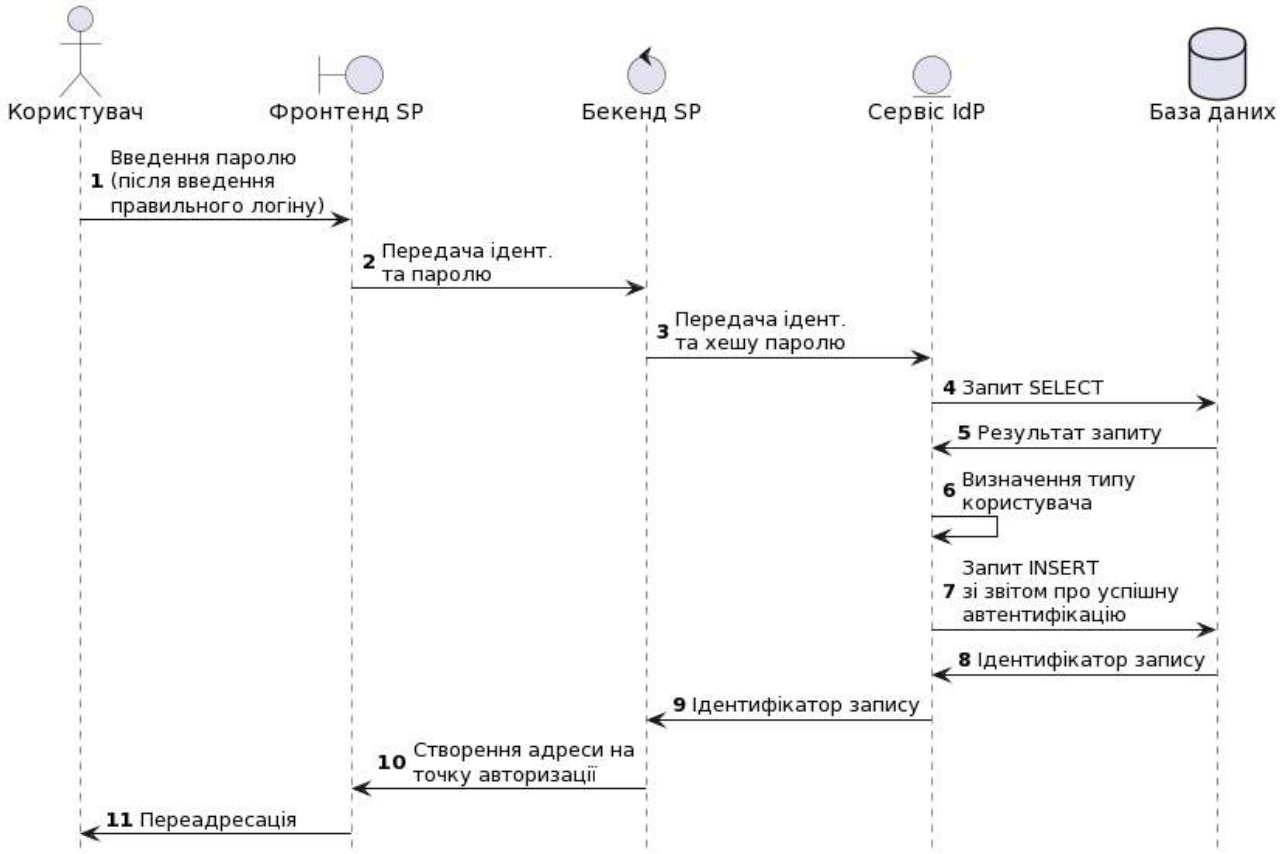


Рисунок 2.5 – Процес верифікації паролю та реєстрації даної події у БД

Ідентифікатор користувача та захешований пароль на стороні бекенду мають передаватись на сторону сервісу IdP. Сервіс IdP у свою чергу має виявити чи потрібний ідентифікатор користувача та хеш паролю наявні у базі. У разі знайдення у базі даних відповідного співпадіння сервіс IdP реєструє успіх

автентифікації та повертає бекенд-частині код автентифікації, який у подальшому використовується у процесі авторизації.

Альтернативна автентифікація. Для зручності існує необхідність у альтернативних способах автентифікації: за допомогою Google та Microsoft акаунтів. Такий підхід вимагає високого рівня довіри до даних провайдерів ідентифікаційних даних. Для того, щоб забезпечити їх використання, необхідно налаштувати з ними OAuth2-зв’язок як з провайдерами ідентифікаційних даних, а також забезпечити кінцевим користувачам можливість додавати подібні варіанти входу в систему.

Перехід з SSO у сервіси. Перехід у навчальні сервіси та LMS передбачається за трьома протоколами: OAuth2, SAML та JetIQ SID. Перші два протоколи є найбільш універсальними та їх реалізація наявна у багатьох сучасних CMS та LMS. Однак більша частина автономних модулів JetIQ використовує власний нестандартний протокол "SID", що виник як модифікація протоколу OAuth1. Згідно проведеним у даній роботі дослідженням, для цих модулів доволі важко додати підтримку стандартизованих протоколів авторизації не руйнуючи зв’язків між ними. Однак незважаючи на свою нестандартність протокол JetIQ SID має й переваги, наприклад можливість швидко автентифікувати користувача при переході між окремими автономними модулями JetIQ без переадресації на сторінку SSO-системи. Більш детальна увага даному протоколу приділяється у розділі 3, пункті 3.1.

Відновлення паролю. Процес відновлення паролю користувача є типовим для більшості існуючих SSO-систем. Типову схему активності у даному зображено на рисунку 2.6. На даній схемі у незначній мірі удосконалено процес верифікації користувача з використанням сервісу CAPTCHA. Електронна пошта на даний момент є найбільш довіреним каналом відновлення паролю, оскільки найбільш вживані поштові сервіси як Gmail, Outlook, Yahoo, iCloud Mail та AOL вживають достатньо надійні технології захисту акаунту коритсувача.

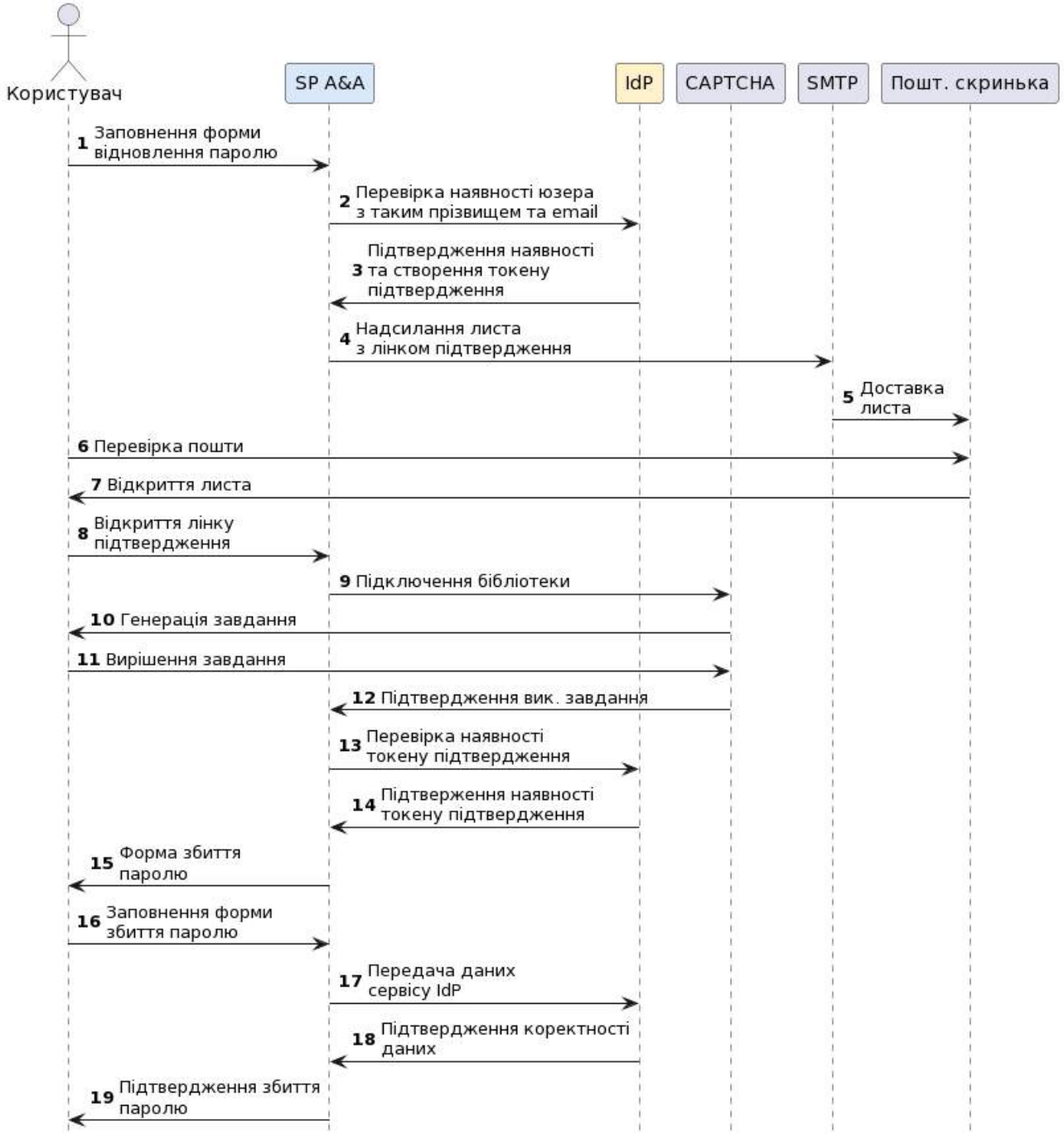


Рисунок 2.6 – Процес скидання паролю у системі SSO

Раніше існував такий спосіб взлому акаунтів як повторне створення поштової адреси, що було обумовлено що деякі поштові сервіси видаляли поштові акаунти по проходженню деякого часу відсутності активності в них [31]. Особливо часто таке траплялось у поштових сервісів, що не слідкували за безпекою користувачів. Це давало зловмисникам можливість повторно створити поштову скриньку з таким же ім’ям. Пізніше були розроблені рекомендації [32]

для поштових сервісів та адміністраторів e-mail серверів, які передбачають блокування акаунту та зниження його вмісту, проте залишають поштову адресу недоступною.

Створення та імпорт користувачів. Аналізуючи схему комунікації між акторами на рисунках 2.2 та 2.3 можна зрозуміти, що дані про співробітників на даний момент вводяться співробітниками відділу кадрів або вручну, або імпортуються з таких систем як Дія та доповнюються ними. Однак процес створення студентів у інформаційних системах ЗВО відбувається за дещо іншою схемою. За результатами проведеного дослідження потоків даних у системі JetIQ встановлено схему потоку даних (рис. 2.7), в ході якого створюються початкові відомості про студента. У цьому процесі одночасно задіяно цілий ряд інформаційних систем.

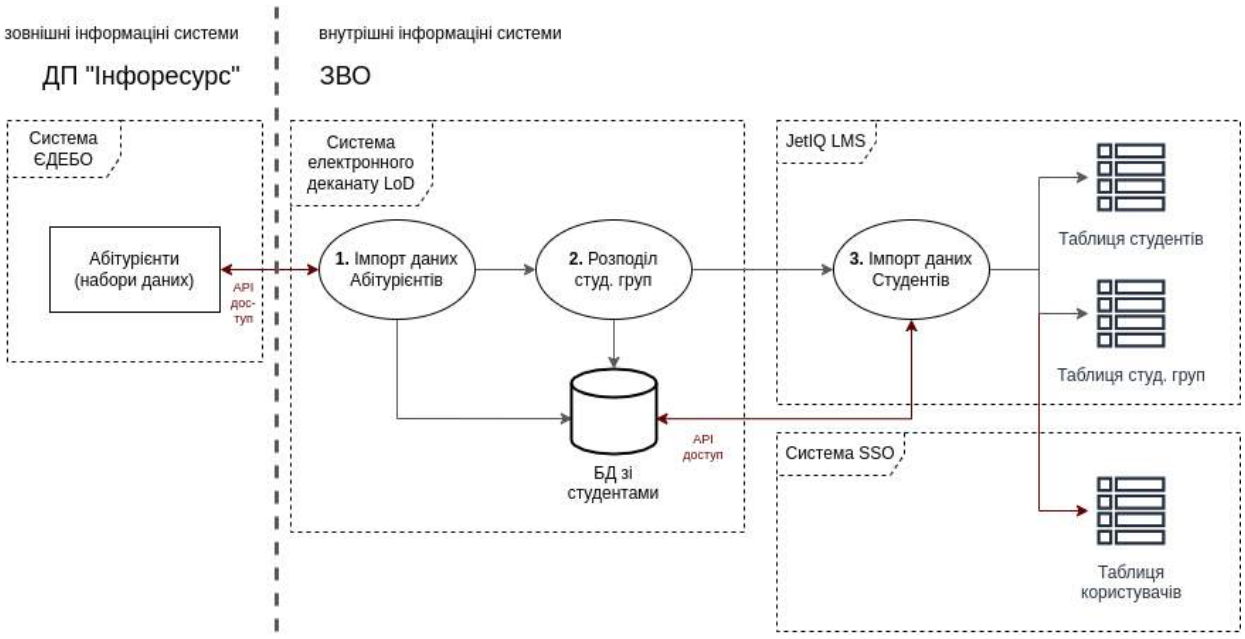


Рисунок 2.7 – Схема потоку даних створення відомостей про студента

Як видно з рисунку 2.5, першим чином дані про студента з’являються у системі ЄДЕБО на етапі коли він стає абітурієнтом на вступ у ЗВО. Розпорядником бази ЄДЕБО є державне підприємство "Інфоресурс", яке знаходиться у безпосередньому підпорядкуванні Міністерства освіти і науки

України. Дані в базу абітурієнтів ЄДЕБО потрапляють з загальнонаціональної системи Дія. Коли абітурієнта вже зараховано у ЗВО, дані про них імпортуються у систему електронного деканату LoD (1), де опрацьовуються співробітниками деканату та розбиваються на академічні групи (2). Завершальна стадія підготовки початкових відомостей про студента (3) здійснюється також співробітниками деканату, під час якого проводиться підготовка особистого кабінету студента у системі JetIQ, а також генерується одноразовий пароль і разом із даними користувача прописується у базі даних системи SSO.

Впорядкування бази даних. В ході дослідження структури бази даних JetIQ виявлено що студенти та співробітники знаходяться у різних, майже ніяк не пов’язаних між собою таблицях реляційної бази даних. У той самий час одній таблиці з викладачами і іншими працівниками знаходяться також деперсоніфіковані ролі служб. Подібний стан справ значно ускладнював роботу макетної версії системи SSO для JetIQ, яка через це не могла повноцінно працювати у автономному режимі без основної системи JetIQ. При подальшому проектуванні архітектури важливо з’ясувати як у подальшому можливо їх розбити та/або об’єднати, щоб вони забезпечували найбільш стабільні і автономну роботу, а також забезпечували надійний захист від несанкціонованого доступу.

В ході вивчення структури бази даних виявлено таблицю ідентифікаторів учасників чату AvatarID, у якій у hex-форматі містились ідентифікатори студентів та викладачів. Розглянуто таблицю ідентифікаторів AvatarID як основу для подальшої уніфікації всіх персоніфікованих користувачів у єдину базу для централізованого входу у систему.

Отож, перед впровадженням нової версії системи SSO для JetIQ необхідним кроком є впорядкування бази даних систем JetIQ та SSO. План розміщення різних користувачів на даний момент у базах даних різних систем наведено у таблиці 2.2.

Таблиця 2.2 – Типи користувачів та їх розміщення у БД різних систем

| Тип користувачів | Значення TOU (Type of User) | Таблиця у БД JetIQ | Таблиця у БД SSO |
|---|---|---|---|
| Студенти | `0` | `students` | `pwd` |
| Викладачі та співробітники | `1` | `teachers` | `pwd` |
| Неперсоніфіковані ролі підрозділів для доступу до служб | `2-255` | `teachers` | `pwd` |

В ході дослідження встановлено, що присутні у таблицях співробітників БД JetIQ неперсоніфіковані записи службових користувачів не використовуються самою системою JetIQ, а лише системою SSO, що дозволяє перерозбити цю таблицю на етапі впровадження рішення, що розробляється у цій роботі. Найбільш оптимальна схема перерозподілу окремих типів доступу зображена на рисунку 2.8. Подібний розподіл дозволить у майбутньому впровадити підхід делегованого доступу до службових акаунтів.

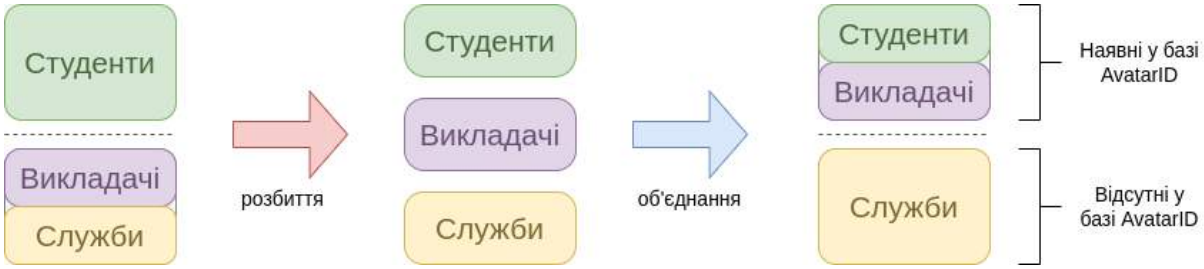


Рисунок 2.8 – Перерозподіл користувачів за ознакою персоніфікованості

Делегований доступ до службових ролей. На даним момент за допомогою єдиного логіну та паролю неперсоніфікованого службового доступу входять у окремі модулі та автономні підсистеми JetIQ цілі відділи, що протирічить прийнятим у теперішній час стандартам з кібербезпеки [33]. Одним із прикладів проблем, які можуть виникнути при використанні одного на всіх працівників підрозділу логіну та паролю можна вважати ситуацію, коли один/одна із

працівників звільняється та переходить працювати у конкуруючу компанію. У такому випадку у такого працівника залишається спільний для його/її минулих колег логін та пароль. Новий працедавець може за додаткову винагороду чи за допомогою шантажу змусити його/її видати дані для колективного неперсоніфікованого службового доступу та саботувати роботу організації, з якої прийшов(ла) працівник. Іншою проблемою є можливість викрадення даних для входу зловмисниками за допомогою шкодоносного ПЗ на пристрою або за допомогою фішингу [34]. Найгіршим у цій проблемі є те, що крім зривання роботи всього відділу, буде дуже важко встановити у кого саме вкрали логін та пароль щоб запобігти повторення проблеми. Отже спільного користування одними і тими самими даними для входу взагалі не повинно існувати.

Найбільш ефективним вирішенням такої проблеми є підхід делегованого доступу. В освітньому середовищі його найбільш доцільно забезпечувати для (рис. 2.9).

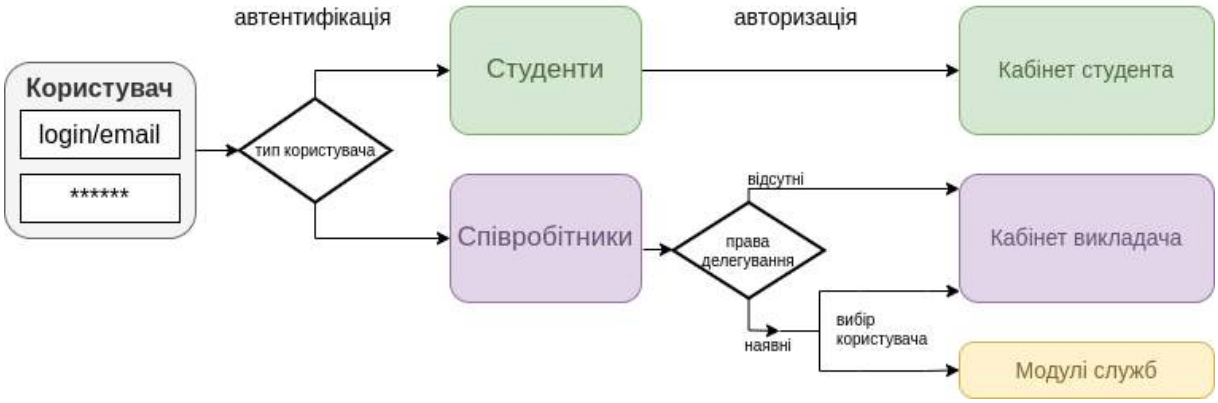


Рисунок 2.9 – Алгоритм роботи делегованого доступу

В ході впровадження делегованого доступу необхідно визначити таблицю зіставлення (N:1) між індивідуальними обліковими записами співробітників та службових ролей, забезпечуючи такий самий доступ, які раніше надавались персоніфікованими ролями підрозділів для доступу до служб, але на цей раз з повним контролем індивідуальних співробітників, що його використовують. Якщо користувач звільняється або переводиться у інший підрозділ – його прив’язка до службової ролі має втрачатись.

Однак з інституціональної точки зору подібний перехідний період неможливий без поетапного перехідного періоду з наступними кроками:

1. перерозподіл таблиць у базі даних із збереженням окремої форми входу для неперсоніфікованих служб у новій версії системи SSO для JetIQ;
2. створення у базі даних системи SSO таблиці зіставлення з відношенням N:1 співробітників (N) до службових ролей (1);
3. запровадження подвійного режиму входу до службових ролей: за делегованим доступом, а також старим способом – за логіном та паролем;
4. закриття доступу за неперсоніфікованим логіном та паролем;
5. запровадження MFA для захисту привілейованого доступу (рис. 2.10) із заохоченням співробітників, що використовують привілейований доступ до використання MFA;
6. закриття доступу до привілейованих служб без MFA.

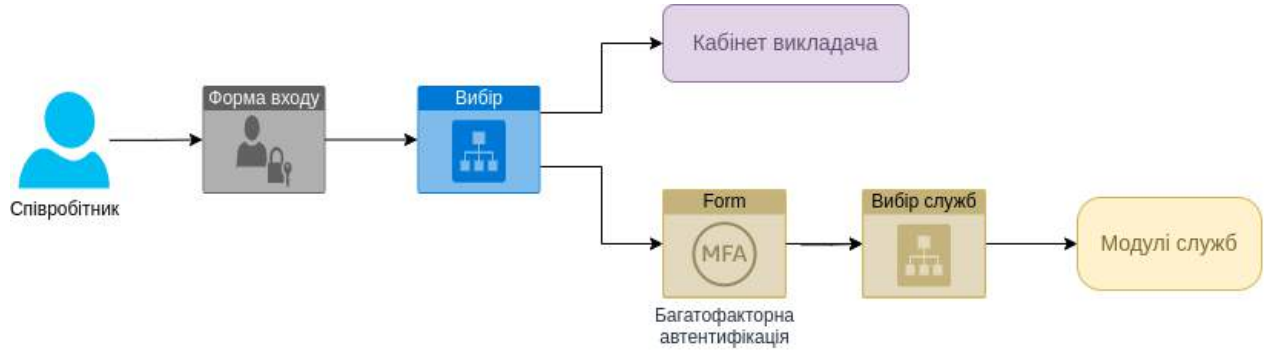


Рисунок 2.10 – Захист привілейованого доступу до службових ролей за допомогою двофакторної автентифікації

При впровадженні делегованого доступу користувачі, що заходитимуть за старим способом повинні будуть отримати повідомлення з попередженням, що старий спосіб входу невдовзі припинить своє існування щоб у них було достатньо часу для відповідних дій.

Отже для проектування програмної архітектури систем SSO важливо спочатку визначити які бізнес-процеси відбуваються у вже існуючих інформаційних системах та у забезпеченні освітнього процесу. В ході розробки

бізнес-процесів визначено основні функції всіх учасників, задіяних у роботі інформаційної системи централізованої авторизації для освітнього веб-середовища навчального закладу; проаналізовано та удосконалено процеси верифікації логіну, паролю та його відновлення; проаналізовано потоки даних створення відомостей про студента; визначено які зміни у базі даних та програмній архітектурі необхідні щоб забезпечити максимально ефективну роботу задіяних інформаційних систем та їх захист від несанкціонованого доступу.

### 2.3 Проектування програмної архітектури системи централізованої авторизації та її інтеграції з іншими системами

Згідно технічного завдання (додаток А), архітектура програмного рішення, що розробляється повинна відповідати Вимогам до програмної архітектури SSO-системи JetIQ MY версії 1.0 (далі – Документація). Всі вимоги не розміщені у відкритому доступі, однак найбільш узагальнені з них можуть бути процитовані:

- нова версія системи матиме нову назву "JetIQ MY";
- існують окремі вимоги до сервісу IdP та сервіс-провайдеру A&A;
- передбачено застосування мікросервісів, однак немає чітких вимог до стилю програмного архітектури;
- описано спосіб обміну інформацією з JetIQ та протокол авторизації у його сервісах SID;
- коротко описано процеси аутентифікації та авторизації користувачів різних типів: звичайних користувачів (студентів і співробітників) та службових ролей;
- опис типів логінів: циферні (ідентифікатор співробітника), ПІБ (співробітники), номер залікової книжки (студенти), кастомний юзернейм (т.з. "нік" студента), а також вхід за email-адресами як основний і універсальний спосіб входу для всіх користувачів;

- опис автодоповнення логінів (для тих користувачів, у кого це увімкнено у налаштуваннях приватності) – для студентів та викладачів;
- крім безпосередньо реалізації сервіс-провайдеру A&A серед веб-додатків має бути реалізована панель самообслуговування службовими акаунтами, а також адмін-панель;
- у панелі самообслуговування користувач повинен мати можливість змінювати свій пароль, способи відновлення паролю, прив'язку до способів мультифакторної автентифікації, параметри конфіденційності, а також централізоване управління акаунтами у всіх корпоративних сервісах та сайтах;
- для всіх веб-додатків системи JetIQ MY має бути використаний фреймворк Jet-сайтів другого покоління (PHP);
- для всіх веб-додатків системи JetIQ MY має бути забезпечено підтримку підходу blue-green deployment;
- для звичайних користувачів (студентів і співробітників) та службових ролей мають існувати два окремі IdP-сервіси з різними вимогами до їх API.

Для зручності випуску нових версій документація передбачає застосування підходу "Blue-green deployment". Він дозволяє оновлювати або розгортати нову версію програми без переривань у роботі. Цей підхід дозволяє мінімізувати вплив на користувачів та забезпечує зручність відкату до попередньої версії у разі виявлення проблем. Основна ідея полягає у наявності двох окремих середовищ, які називаються синє (blue) та зелене (green). В один момент часу лише одна версія програми активна для користувачів наприклад синя, тоді як інша версія – зелена може бути розгорнута або оновлена.

Однією з переваг цього підходу є можливість повернутися до попередньої версії швидко, якщо виявиться, що нова версія має непередбачені проблеми. Це забезпечує менший ризик та мінімальний час простою системи під час релізу. Blue-green deployment також полегшує тестування нової версії в реальних умовах перед тим, як вона потрапить до виробництва.

Зазвичай синя версія є активною за вмовчанням для користувачів, і всі запити від користувачів йдуть до неї. Нова версія зазвичай розгортається у зеленому середовищі, яке є ізольованим. Для того щоб переконатись, що нова версія працює коректно виконуються тести та спеціальні перевірки. У DevOps-інженерів та системних адміністраторів є можливість перенаправити частину трафіку на зелену версію, щоб спостерігати як вона працює під навантаженням справжніх користувачів. Для відстеження будь-яких проблем у зеленому середовищі має вестись моніторинг під час роботи з реальними користувачами. У результаті зелена версія, яка пройшла всі фінальні підготування і витримала випробування справжнім трафіком випускається у реліз, займаючи місце синьої.

Для можливості регулювання розподілу трафіку між синіми та зеленими версіями варто застосовувати можливості балануальника HTTP-трафіку, таких як nginx. Саме цю реалізацію веб-серверу найбільш доцільно використати, оскільки його функціонал може бути розширений завдяки мові Lua. Система Blue-green deployment згідно документації має працювати виключно на рівні веб-додатків (B). Перемикання версій системи IdP можливе лише завдяки символічним посиланням та сокетам UNIX, у чому неможливо застосувати згаданий підхід.

На основі наведених вимог або їх опису зрозуміло що система повинна мати гарну підтримку модульності та можливості для розширення функціоналу. Але при цьому згідно вимог має зберігатись чіткий поділ на IdP-сервіси та сервіс-провайдери A&A. Найкраще для такого завдання підходить саме багатошаровий (layered) стиль архітектури у поєднанні з мікросервісним підходом. Багатошарова архітектура, де на кожному рівні розміщені веб-додатки та мікросервіси найбільше задовольняють всі архітектурні вимоги документації. Крім того, багатошарова архітектура сприяє відповідності принципам SOLID (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion), що полегшує розширення та модифікацію коду. Розробка кожного рівня може відбуватися незалежно від інших рівнів, що полегшує

розподілений розвиток. Різні команди можуть працювати над різними рівнями, не впливаючи одна на одну.

В ході дослідження виявлено системи, з якими має взаємодіяти система SSO JetIQ MY (рис. 2.11). У таких зв'язках ця система переважно виступає як провайдер даних про користувача і його сесію. Інші сервіси виступають як репецієнти цих даних отримуючи дані за токенами протоколу SID або зворотнім відкликом OAuth2 чи SAML.

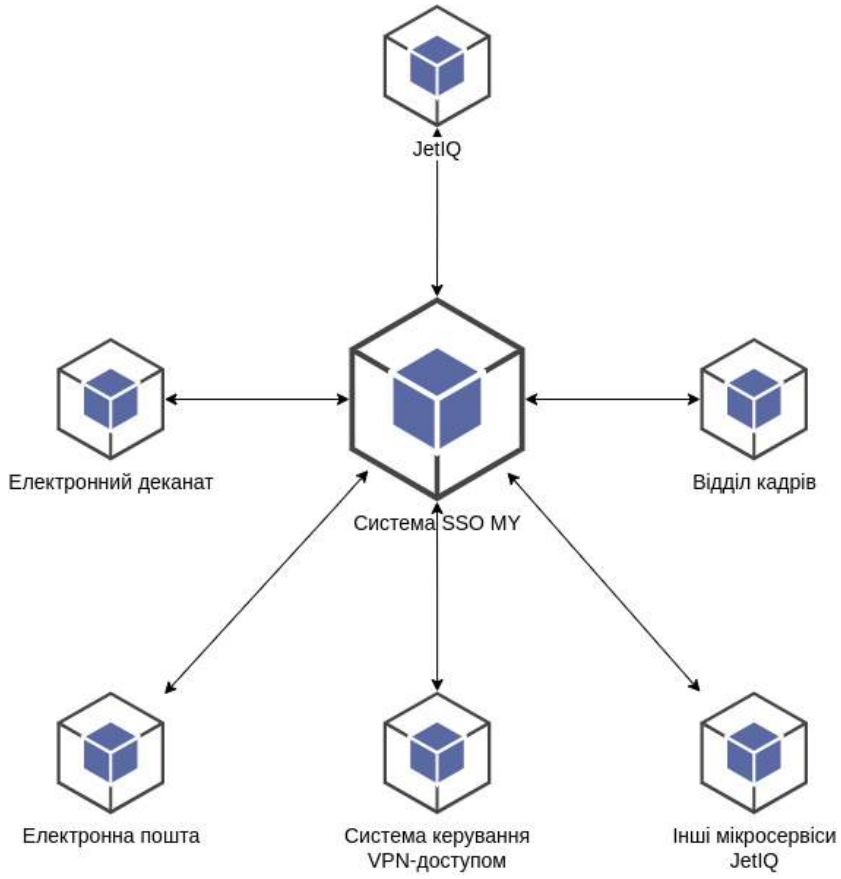


Рисунок 2.11 – Взаємодія системи SSO MY з іншими системами

Однак такі системи як JetIQ, Електронний деканат, ERP-система відділу кадрів виступають також у ролі провайдерів актуальних даних, з яких система SSO отримує відомості про необхідність та характер змін даних про користувача.

Розбиття на шари. При розподілі на шари необхідно враховувати типову архітектуру SSO-систем, яка сама по собі є дворівневою (рис. 2.1). Згідно концепції SSO-систем шари більш доцільно називати рівнями. Рівень сервіс-провайдерів A&A приймає участь у взаємодії з користувачем, і таким чином є вищим за рівень сервісів IdP. Кожен із цих двох рівнів можна розбити на два підрівні, включивши характерні для веб-сервісів компоненти. Рівень сервісів IdP складатиме ядро SSO-системи яке можна розбити на умовні рівень баз даних (найнижчий рівень) та API-рівень, на якому реалізована вся програмна логіка IdP-сервісів та їх протоколів. Рівень сервіс-провайдерів A&A, на якому відбувається взаємодія з користувачем можна розбити на характерні складові веб-додатків: бекенд та фронтенд (рис. 2.12).

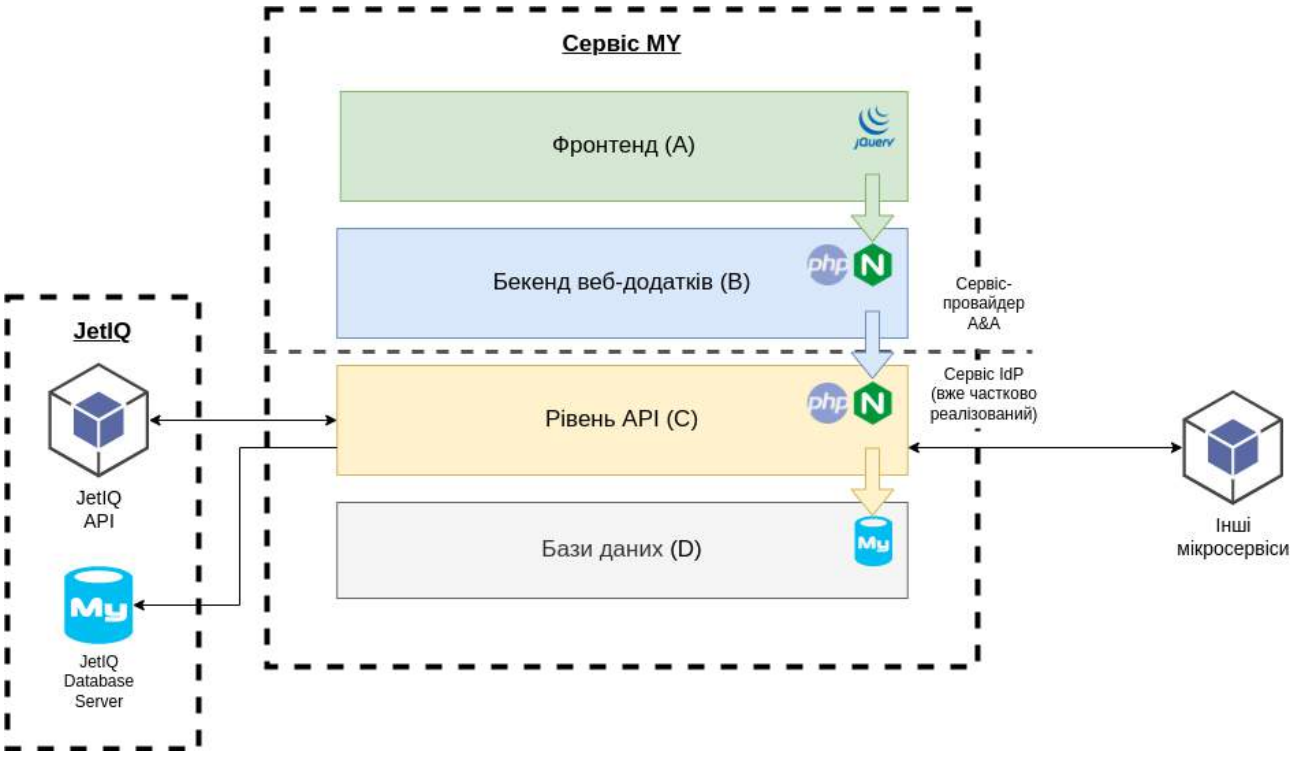


Рисунок 2.12 – Структурна схема нової системи, в архітектурі якої реалізуються багатошаровий стиль та мікросервіси

Система SSO виступає як повноцінний веб-сервіс має автономність від інших систем, таких як JetIQ, Електронний деканат, Відділ кадрів. Вона з однієї точки зору частково залежить від них, оскільки збирає з них дані про користувачів. З іншої сторони інші системи, інші системи також частково залежать від системи SSO, оскільки вона є єдиним вікном входу користувачів у систему.

Рівень API. Документація передбачає існування на рівні API ряд автономних модулів, які працюють через інтерфейс REST API, деякі з модулів мають інтерфейси LDAP, OAuth2 та SAML. У таблиці 2.3 наведені вже існуючі модулі, які з незначними змінами можна використати. Зірочкою * позначено назви модулів, розробка яких передбачена даною роботою (у розділі 3.1). Двома зірочками ** позначено модулі, початково написані на Java (APIv1), але на даний момент через проблеми із сумісністю та їх невдалим програмним дизайном переписується командою розробників на мову програмування PHP (APIv2).

Таблиця 2.3 – Модулі API-рівня SSO-системи JetIQ MY

| **Модуль** | **Кодова назва** | **Опис** |
|---|---|---|
| Анонімний доступ | `anonymous**` | Модуль, який надає користувачам можливості автодоповнення логіну та його початкову верифікацію |
| Виявлення та виправлення помилок | `audit` | Дозволяє оперативно виявляти та усувати помилки у базі даних, програмній конфігурації та журналі подій. Використовується CLI-інструментами `db-tools`, (зокрема у режимі crontab-завдань), доповнюючи у цьому функціонал інструментів адміністрування користувачами модуля `iq-sync`.<br><br>Залежить від модулів `community` та `enterprise`. |

Продовження таблиці 2.4

| | | |
|---|---|---|
| IdP-ядро для користувачів | `community**` | Надає методи для отримання доступу до відомостей про звичайних (персоніфікованих) користувачів, пов'язаних з ними сервісів, параметри їх сесій, а також тимчасові токени OAuth2, SAML та SID. Використовується модулями `services` та `ldap-proxy` для аутентифікації та авторизації користувачів у локальних сервісах. |
| IdP-ядро для неперсоніфікованих ролей | `enterprise**` | Надає методи для отримання доступу до відомостей про неперсоніфікованих користувачів (службові доступи), пов'язаних з ними сервісів, параметри їх сесій, а також тимчасові токени JetIQ SID. |
| Обмін даними користувачів з JetIQ (у вигляді двох окремих модулів) | `iq-retro*` `iq-sync*` | Має передбачати надання інтерфейсу адміністрування (`iq-sync`) користувачами для Електронного деканату JetIQ і ERP відділу кадрів, а також передбачати імпорт даних користувачів (`iq-retro`) з бази даних JetIQ. |
| Управління БД | `db-tools**` | Python скрипти апгрейду та міграції БД. Залежать від модулів `audit` та `iq-sync`. |
| LDAP та RADIUS інтеграція | `ldap-proxy**` `radiusd*` | LDAP та RADIUS-обгортки над модулем `community`, що портуються на Python з бібліотеками `ldapserver` та `pyrad` |
| OAuth2, SAML та SID інтеграція | `services` | Надає інтерфейси цих протоколів для користування довіреним веб-додаткам та `api-proxy` (рівень веб-додатків). Працює на базі модуля `community`. |
| Зв'язок з акаунтами інших сервісів (розшир. ф-ціоналу модулю `services`) | `services*` | Має дозволяти користувачам підключати сторонні сервіси, що передбачені системою SSO. Має працювати на базі модуля `community`. |
| Журналювання | `logger` | Записує всі події у системі у базу даних журналу. Використовується всіма іншими модулями. |
| Розмежування API-доступу | `web-root*` | Має підвищувати захист API-модулів від несанкціонованого доступу та бути інтерфейсом для всіх інших модулів. |

Для того, щоб зрозуміти як пов’язані API-модулі між собою, на основі вимог документації створено схему залежностей між ними (рис. 2.13). У загальному рівень API складається з 11-ти API-модулів. Найбільш базовими з них є два IdP-модулі, які є ядром IdP-системи. Для автентифікації та авторизації зовнішні сервіси підключаються до спеціалізованих A&A "мікросервіс-провайдерів" на рівні API, розроблених під відповідні протоколи: OAuth2, SAML, LDAP та RADIUS.

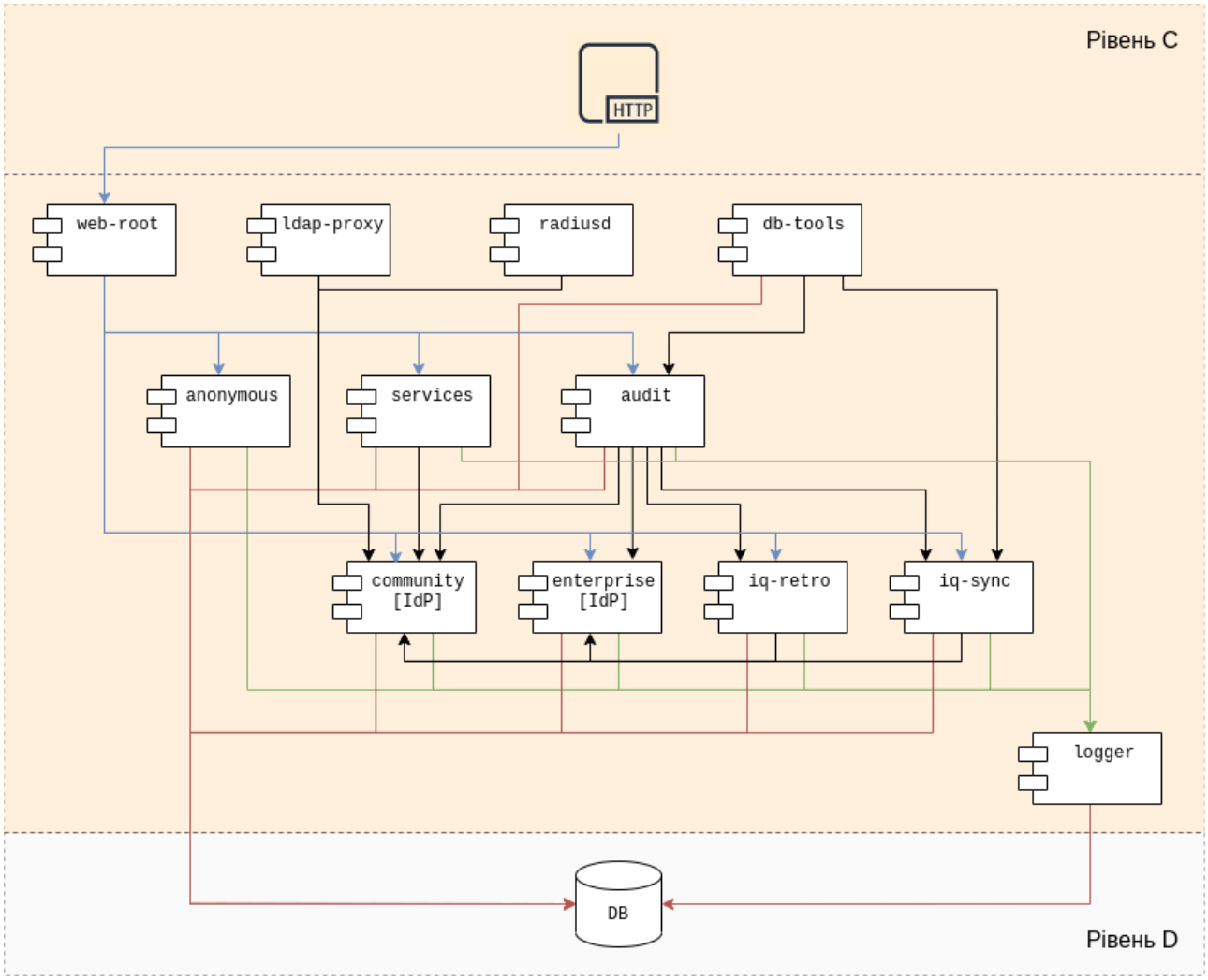


Рисунок 2.13 – Діаграма залежностей між модулями API

Червоним на рисунку 2.13 позначено підключення до БД MySQL (рівень D). Зеленим позначено асинхронні виклики журналювання (logging). Блакитним позначено HTTP/HTTPS трафік, який пропускає через себе модуль `web-root`.

Чорним позначено програмні зв’язки між модулями. Як у першій версії API-ядра (APIv1), так і для зворотної сумісності в другій (APIv2) у модулях написаних на PHP використовується переважно процедурна парадигма програмування. Об’єктно-орієнтована парадигма використовується лише у API-модулях, написаних на Python та Java.

Рівень бекенд веб-додатків. Документація передбачає наступний (після API з IdP) рівень для роботи сервіс-провайдеру A&A (рівень веб-додатків B). в ході вивчення документації виділено 7 окремих компонентів (таблиця 2.4), які мають забезпечувати функції сервіс-провайдеру A&A та адміністрування облікових записів користувачів.

Таблиця 2.4 – Компоненти рівня веб-додатків SSO-системи JetIQ MY

| **Модуль** | **Кодова назва** | **Шлях URI** | **Опис** |
|---|---|---|---|
| Проксі-доступ до публічних API-методів | `APIv1Proxy` | `/api/` | Цей компонент має забезпечувати доступ до API-рівня додаючи захист від ботів (метод описано у розділі 2.5) |
| Панель самообслугову-вання користувача | `AccountPanel` | `/account/` | Панель в якій користувач має змінювати свій логін, пароль, електронну пошту та змінювати параметри конфіденційності. Також у ній буде можливість керувати зовнішніми акаунтами в які надається доступ системою SSO (наприклад електронною поштою) |
| Панель управління | `AdminPanel` | `/admin/` | Має забезпечувати керування користувацькими акаунтами, процесом їх синхронізації з системою LMS (JetIQ) та іншими сервісами |
| Переадресація на сервіси з використанням процедури входу OAuth2 | `GoRedirector` | `/go/` | У цьому компоненті буде відбуватись переадресація із сторонніх сервісів для авторизації та повернення у ці сервіси назад |

Продовження таблиці 2.5

| | | | |
|---|---|---|---|
| Мікрофреймворк Jet-сайтів другого покоління | `IQCoreApp` | - | Удосконалена версія мікрофреймворку Jet-сайтів другого покоління із підтримкою об'єктних-сесій та покращеним захистом від DDoS-атак |
| Сервіс-провайдер для доступу служб (неперсоніфіковані) | `StaffAuth` | `/staff/` | Забезпечує реалізацію бізнес-логіки входу у систему неперсоніфікованих користувачів (доступу служб), а також їх подальшу авторизацію у системі JetIQ та сторонніх сервісах |
| Сервіс-провайдер для користувачів (персоніфіковані) | `UserAuth` | `/user/` | Забезпечує реалізацію бізнес-логіки автентифікації та авторизації користувачів у системі SSO, а також їх подальшу авторизацію у системі JetIQ та сторонніх сервісах |

Проаналізовано як компоненти рівня веб-додатків взаємодіють із API-модулями та розроблено схему їх взаємодії (рис. 2.14). Шарова архітектура передбачає залежність верхнього шару від нижнього, тому всі стрілки направлені вниз.

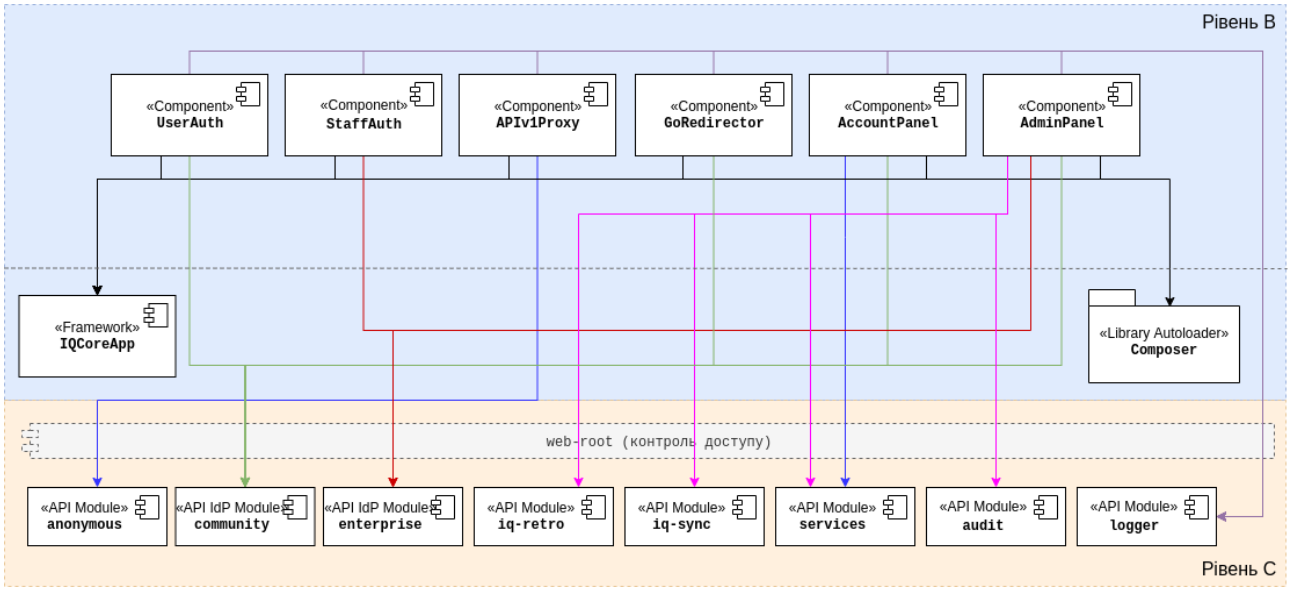


Рисунок 2.14 – Діаграма зв'язків між рівнями B та C

Чорним на рисунку позначено внутрішні програмні зв’язки (безпосереднє лінкування). Кольорові стрілки позначають зв’язки REST API між рівнями. Червоними стрілками показано використання IdP для службового доступу, зеленими – IdP для персоніфікованих користувачів (студенти та співробітники). Рожевим позначено адміністративний доступ, фіолетовим позначено виклики логування, синім – решту зв’язків.

Розгортання. Зв’язки між рівнями у вигляді REST API дозволяють зробити окремі рівні більш незалежними, що є добре коли необхідно забезпечити масштабування та/або відмовостійкий гарячий резерв. Найбільш зручним розгортанням для розробки та тестування є контейнери у середовищі Docker (рис. 2.15).

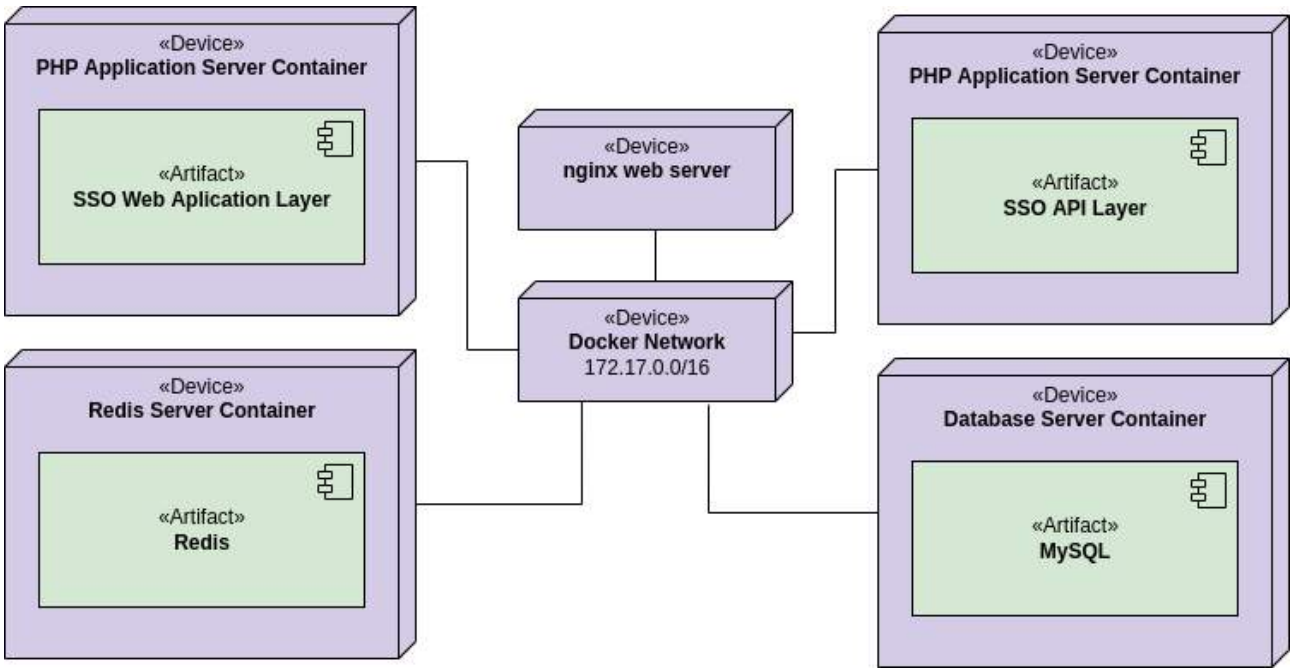


Рисунок 2.15 – Діаграма розгортання системи у середовищі Docker

При розгортанні у вигляді контейнерів необхідно використовувати середовище для збереження даних сесій. Найбільш краще для даного завдання підходить база "ключ-значення" Redis. Розгортання без контейнерів є також можливим, тоді можна працювати без окремої бази "ключ-значення". Однак наприклад коли безконтейнерне розгортання відбувається у хмарному

середовищі, тоді її все таки доведеться додавати. До прикладу у хмарі Amazon Web Services Redis можна замінити таким не менш продуктивним рішенням від Amazon як DynamoDB. Це стає можливим при використанні AWS SDK для PHP. На рисунку 2.16 зображено типову схему хмарного розгортання у середовищі Amazon Web Services. У хмарах з подібним набором технологій (наприклад Azure та GCP) схема буде такою ж, або подібною, в залежності від реалізації доступних там баз "ключ-значення".

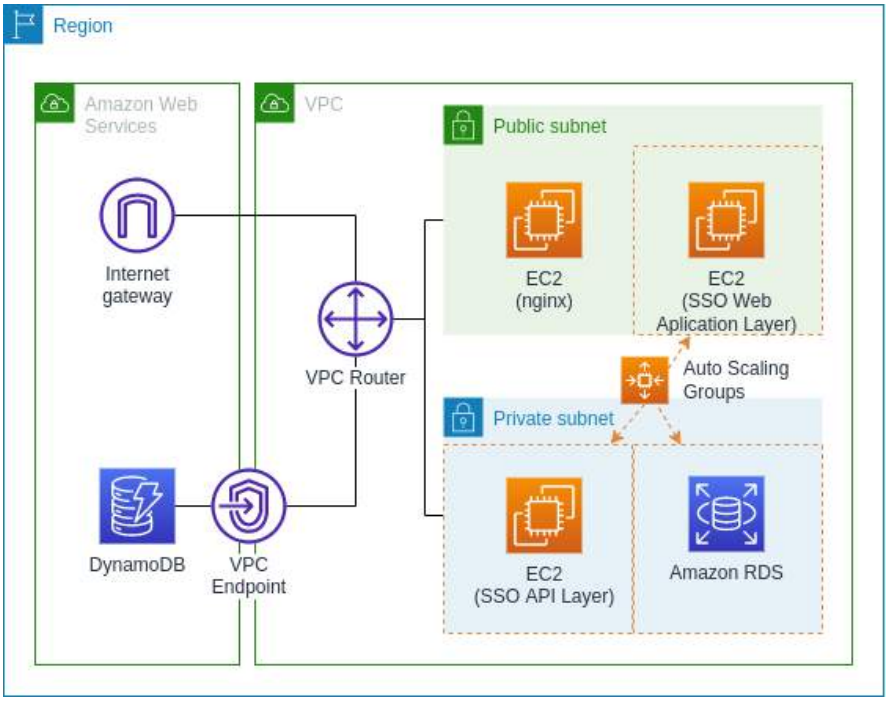


Рисунок 2.16 – Схема розгортання системи у хмарному середовищі AWS

На рисунку 2.16 зображено найбільш простий приклад розгортання для високонавантажених систем з використанням груп динамічного розширення потужності (у термінології AWS це називається Auto Scaling Groups). Однак за необхідності подібну схему можна вдосконалити завдяки використанню контейнерів ECS у хмарі, що дасть можливість більш ефективно використовувати ресурси.

## 2.4 Розробка методу проектування REST API-сервісів систем технології централізованої автентифікації, авторизації та ідентифікації

Проблеми кібербезпеки у REST API-сервісах. Інтерфейс REST API незважаючи на те, що завдяки його простоті здається що його кіберзахист є простим завданням, часто це зовсім не так. Існує цілий спектр кібератак, направлений на REST API інтерфейси, серед яких можна виділити API-ін'єкцію, різні типи DDoS-атак, підвищення повноважень, порушення процесів автентифікації та авторизації тощо [35]. Необхідно розробити метод, який міг би зменшити ризик експлуатації найбільш небезпечних типів атак, які можуть виникнути у будь-який момент життєвого циклу ПЗ, серед яких можна виділити наступні:

- Атака на периметр. Родібна атака може включати спроби проникнення в систему, мережу чи додаток через слабкості або вразливості, які знаходяться на зовнішній межі цієї системи. Атаки на периметр можуть бути спрямовані на отримання несанкціонованого доступу, крадіжку конфіденційної інформації, завдання шкоди системі або використання ресурсів зловмисниками.
- Атака ідентичності (також відома як атака на автентифікацію) – це тип кібератак, спрямований на злам або обійдення механізмів визначення особистості користувача, системи або сервісу з метою отримання несанкціонованого доступу або інших видів зловживань. Ці атаки можуть бути спрямовані на елементи ідентифікації, такі як паролі, токени, сесійні ключі, біометричні дані тощо.
- MITM-атаки (Man in the middle) – це атаки, при яких виникає проміжний хост, що виконує роль проксі, який перехоплює дані для автентифікації, або дані користувача. Часто спрямовані на користувачів мобільних додатків та IoT-системи, хоча зазвичай є малоефективним у взаємодії між публічними сервісами [36].

- API-ін’єкція – атака, що передбачає розширений несанкціонований доступ до API через фасад будь-якого додатку непередбаченим розробником такого додатку. Найбільш часто таке відбувається при некоректному екрануванні GET- та POST-параметрів HTTP-запитів, наприклад коли розробник додатку-фасаду формує ці запити вручну, підставляючи змінні у рядок параметрів наприклад таким чином (PHP):

  ```
  "?parameter1=$var1&parameter2=$var2"
  ```

  Зловмисник може зробити API-ін’єкцію, додавши у значення змінної `$var2`, що приходить з фасаду наприклад текст `"&parameter1=NEWVAL"`, і таким чином параметр API-запиту "`parameter1`" набуде значення `"NEWVAL"`, яке може допомогти зловмиснику для у продовженні взлому.
- Атака довільного доступу до даних (Arbitrary Data Access Attack) - це вид кібератак, під час якого зловмисник намагається отримати доступ до конкретних даних в інформаційній системі чи додатку. Ця атака може мати різні форми залежно від контексту та вразливостей в системі. Основною метою зловмисника в такій атаки є незаконне отримання, зміна або видалення конфіденційної інформації. У API-додатках дозволяє отримати доступ до непередбачених API-інтерфейсом даних, наприклад з бази даних чи навіть оперативної пам’яті. Виникає при недостатній валідації та очищенні вхідних даних. Досить часто виникає у поєднанні з API-ін’єкцією.
- Атака довільного виконання коду (Remote Code Execution або RCE) є видом кібератаки, під час якої зловмисник намагається виконати свій власний, зазвичай зловмисний, код на вразливій системі чи сервері. Досить часто виникає у поєднанні з API-ін’єкцією. Ця атака може мати серйозні наслідки, оскільки вона може дозволити атакуючому отримати контроль над системою, виконувати команди від імені адміністратора та виконувати різноманітні дії в залежності від вразливості. Ця атака є найбільш небезпечною серед всіх інших атак.

Реалізація методу. Для мінімізації перелічених проблем даний метод першим чином передбачає класифікацію всіх REST API-ендпоїнтів за класами критичності. Найбільш часто REST API-ендпоїнти зібрані у деревовидну структуру, для полегшення документування та подальшого використання. Перший етап передбачає перебір всіх ендпоїнтів та визначення їх у окремі класи. На рисунку 2.17 зображено найбільш просту класифікацію за рівнем доступу з поділом на три класи.

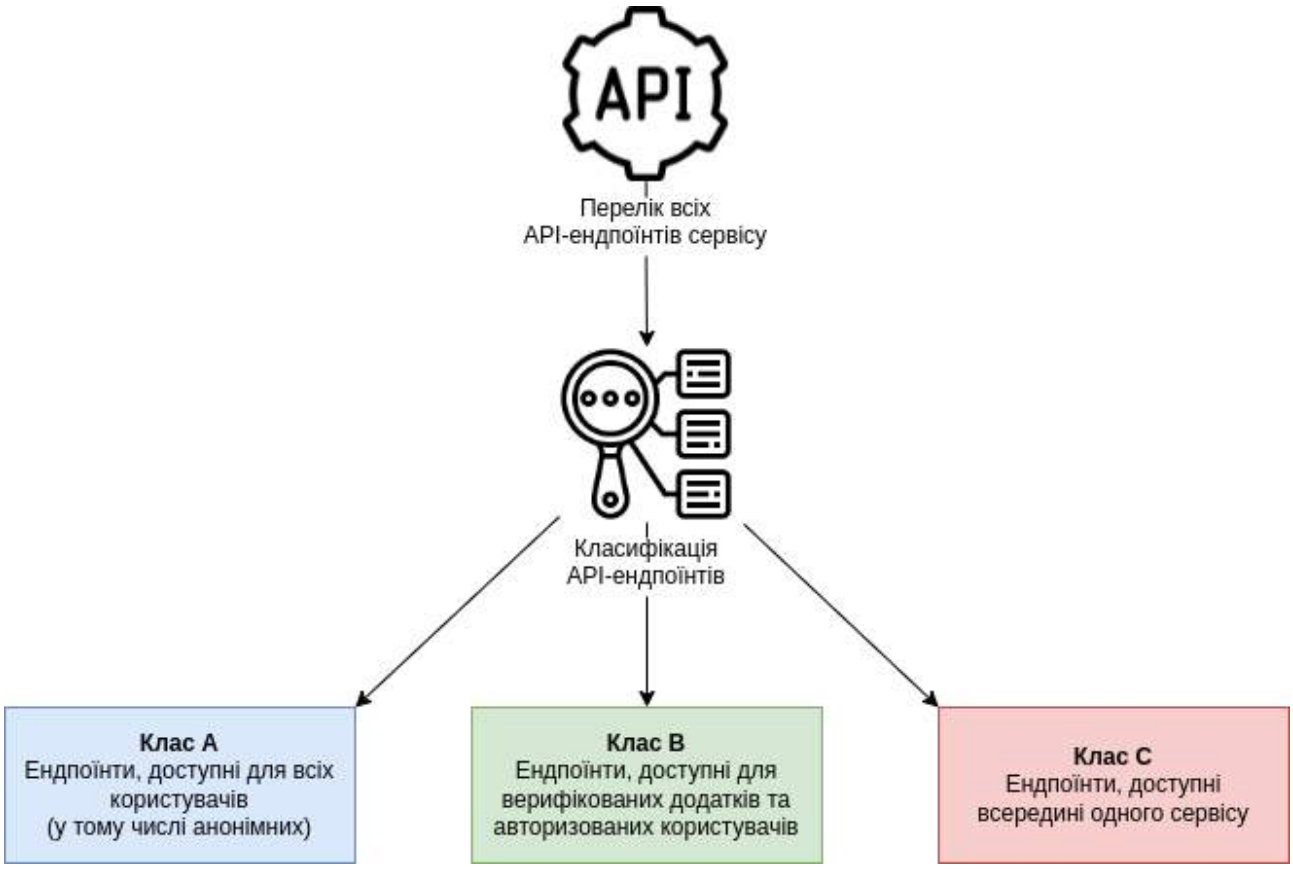


Рисунок 2.17 – Процес класифікації ендпоїнтів

Найбільш привілейовані API-методи потрапляють у найбільш захищений клас С. Вони залишаються доступними лише між внутрішніми контейнерами сервісу. Наступний клас В дозволяє доступ до своїх API-ендпоїнтів переважно перевіреним та авторизованим додаткам. Останній клас А формує так званий публічний API-інтерфейс, який доступний для всіх користувачів, у тому числі як

для анонімних так для авторизованих, а також для додатків з низьким рівнем довіри, таких як мобільні додатки.

Кінцеве застосування методу полягає у тому що для кожного класу існуватимуть точки доступу (рис. 2.18), кожна з яких буде надавати також доступ і до менш захищених методів, подібно до того, як це передбачено у концепціях обмеження доступу до класів ООП [37].

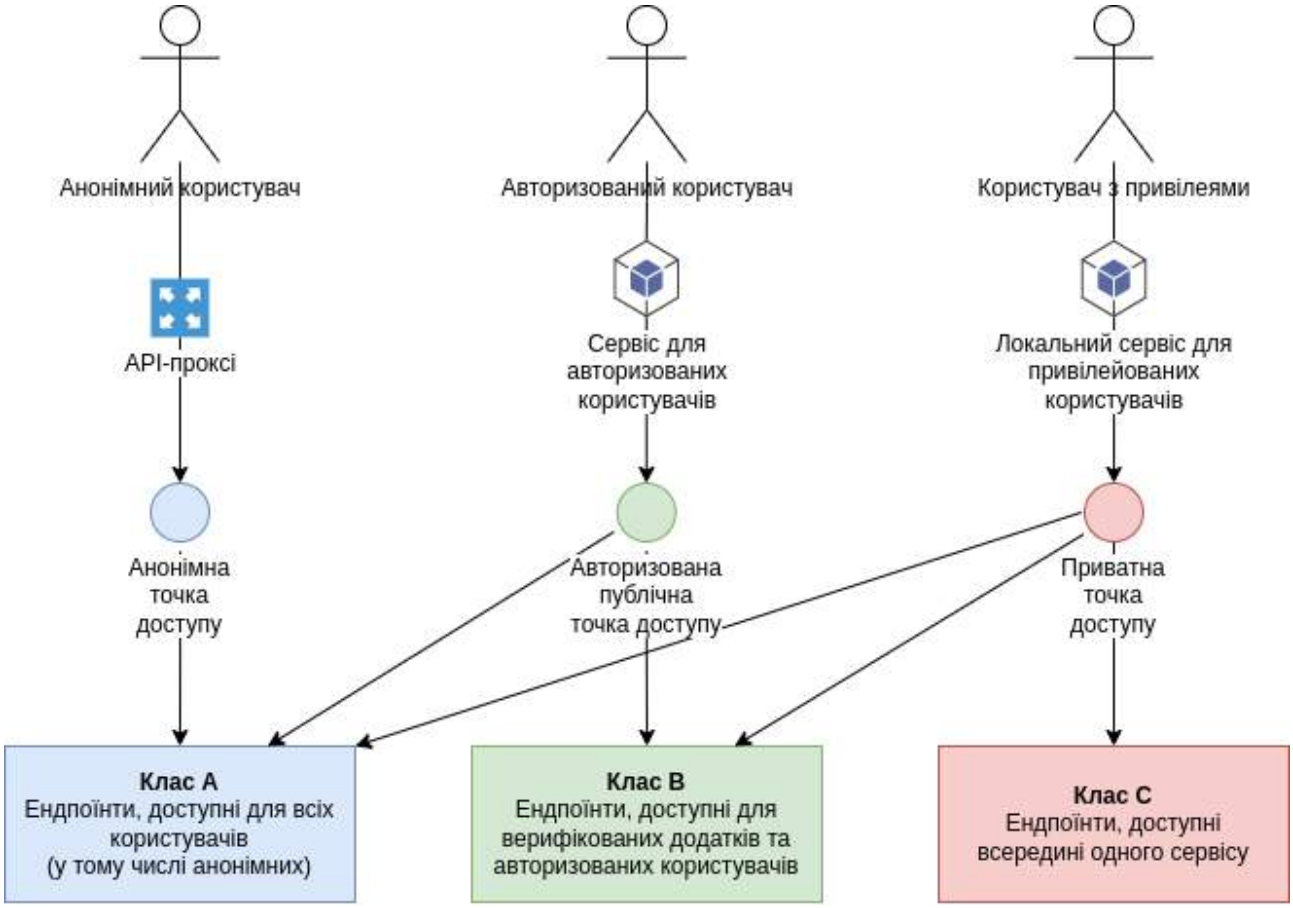


Рисунок 2.18 – Розподіл видимості класів за точками доступу

Для ендпоїнтів класів B та C також передбачається створення підкласів з наборами правил мережевого екрану WAF. У напівавтоматизованій конфігурації інженер з безпеки зможе обрати або існуючий підклас безпеки, або створити новий з визначенням нових правил ACL для такого мережевого екрану (рис. 2.19).

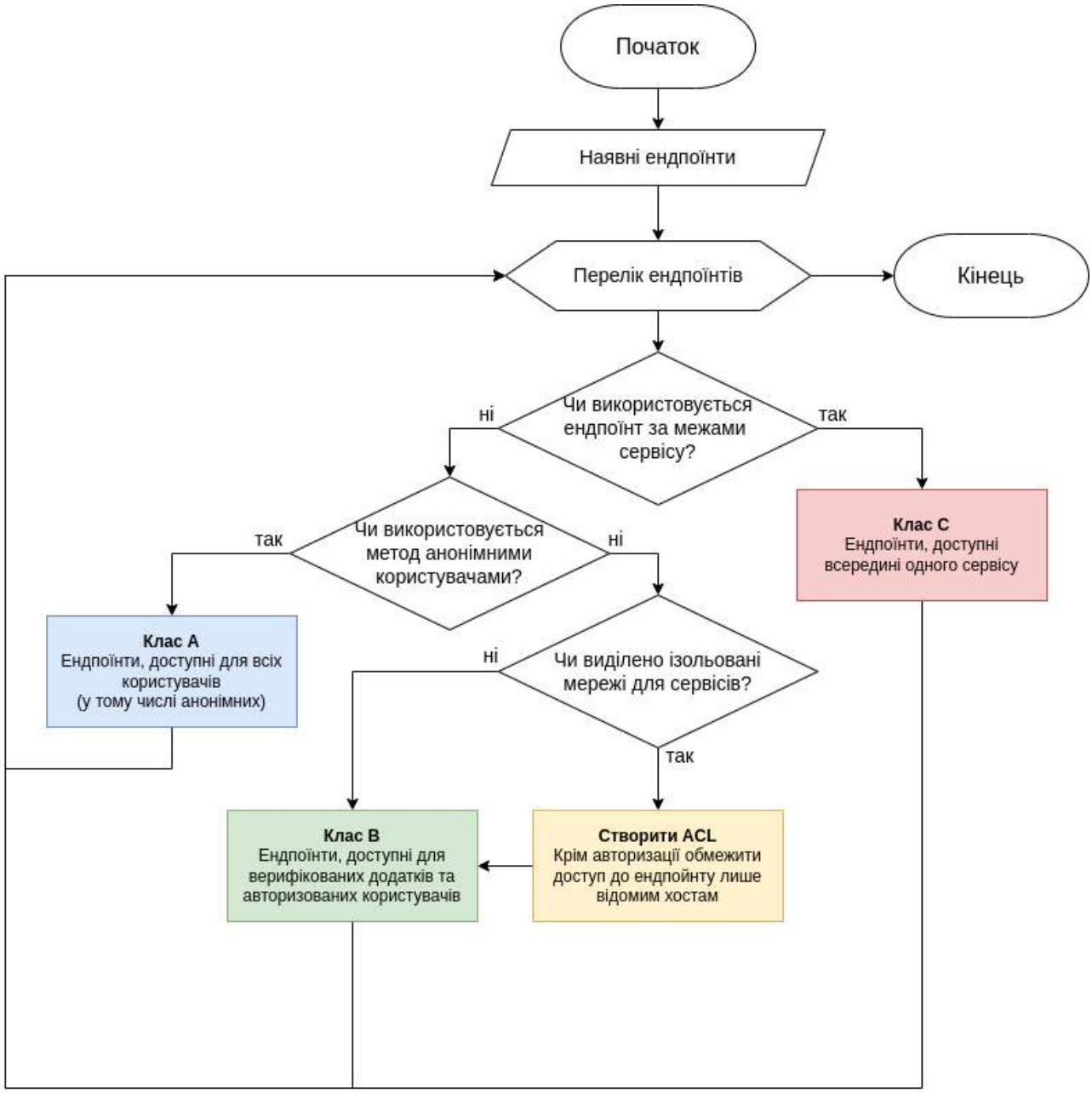


Рисунок 2.19 – Процес розбиття ендпоїнтів на класи та підкласи

Розбивка API-ендпоїнтів по окремим класам та підкласам зберігається у конфігураційних файлах формату JSON що полегшує їх повторне редагування у редакторі конфігурації, а також створює можливість генерувати їх початкову конфігурацію за допомогою AI-інструментів, таких як GitHub Copilot. Крім того інженер з безпеки зможе брати набір правил з готових шаблонів.

Розбиття на окремі HTTP/HTTPS точки доступу дозволить крім методів, передбачених у внутрішньому WAF екрані комбінувати їх із зовнішніми

мережевими ACL та Security Group'ами хмари, де такий додаток буде розгорнуто. Це збільшує можливості для відраження DDoS-атак та запобіганню несанкціонованому доступу.

Тобто крім запропонованого у даному методі захисту за допомогою HTTP/HTTPS WAF з'являється можливість для більш провинутого захисту на 3-му та 4-му рівні моделі OSI. Також трафік окремих класів REST API можна загорнути у додатковий рівень захисту, наприклад SSL-автентифікацію чи VPN-канал.

У результаті застосування даного методу у середовищі API системи JetIQ MY з'явилась можливість більш гнучко налаштовувати захист, моніторинг та журналювання використання REST API ендпоїнтів (рис. 2.20).

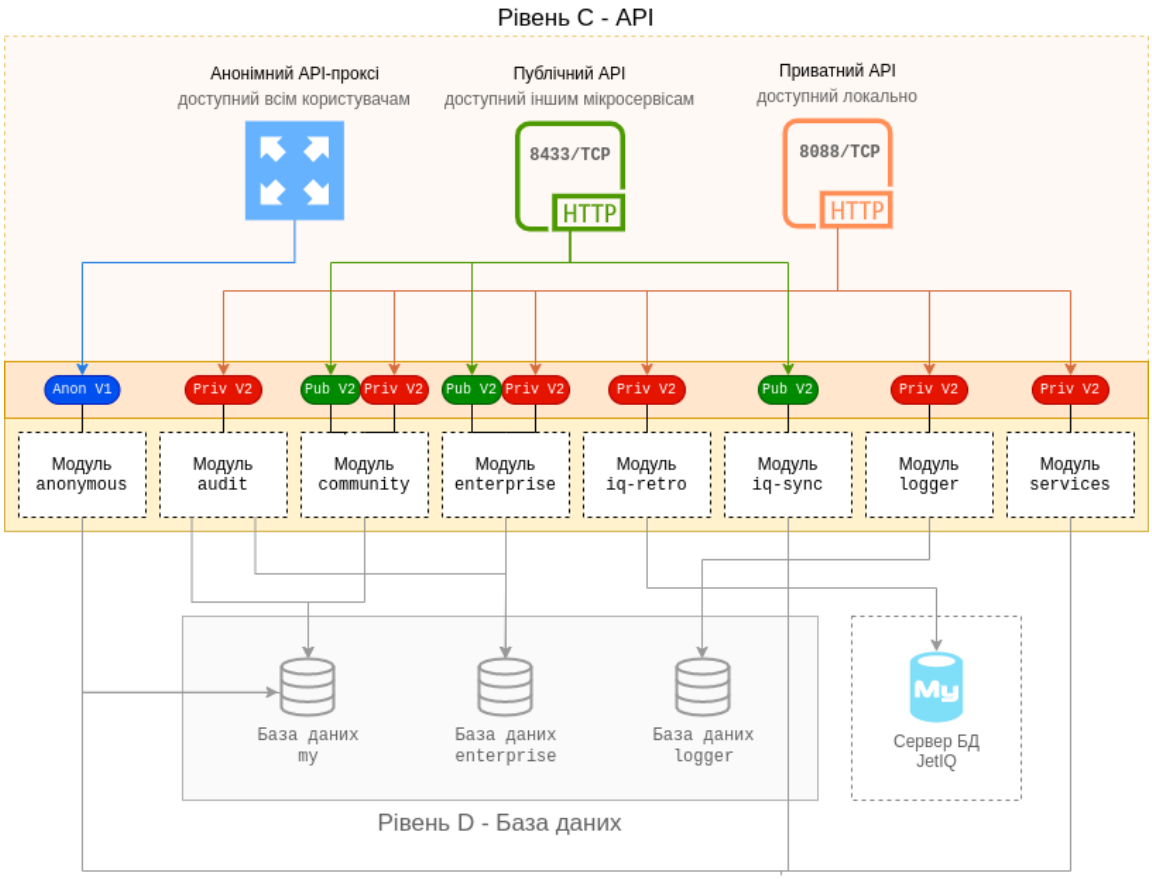


Рисунок 2.20 – Результат застосування методу у системі JetIQ MY

Це досягається завдяки інтеграції згенерованих JSON-правил для кожного класу у процес генерації URI-маршрутів API-ендпоїнтів для кожної HTTP/HTTPS точки доступу REST API.

### 2.5 Розробка методу визначення довіри до клієнтського пристрою

Під час роботи систем ідентифікації, автентифікації та авторизації відбувається чимало процесів, що спричиняються зловмисниками та/або різними ботами, які не мають бути залучені у роботи цих систем. Цьому запобігають різні системи, такі як фаєрвол веб-додатків (WAF), а також тестування Тюринга (наприклад CAPTCHA).

Однак у деяких веб-додатках один раз правильно пройшовши тест CAPTCHA стає можливим перебір паролів, що було виявлено в ході експерименту над JetIQ з попереднім модулем входу (рис. 2.21). В ході експерименту було перебрано паролі після успішного проходження такого тесту.

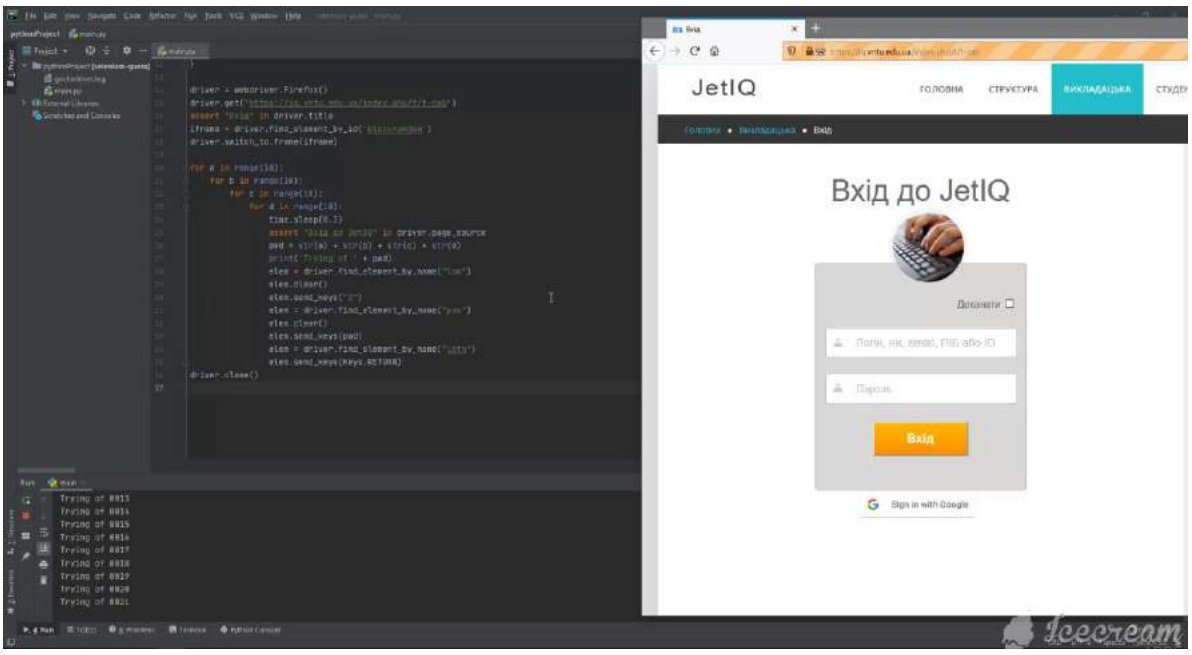


Рисунок 2.21 – Перебір паролів у середовищі Selenium

В кінці експерименту робота скрипту Selenium призвела до зависання та недоступності сторінки входу для всіх користувачів. Подібний експеримент у

середовищі Selenium проведено з публічним API-ендпоїнтом, що дозволяє новим користувачам autocomplete для більш зручного входу у кабінет користувача перший раз за прізвищем (рис 2.22), але на цей раз із виснаженням запитами.

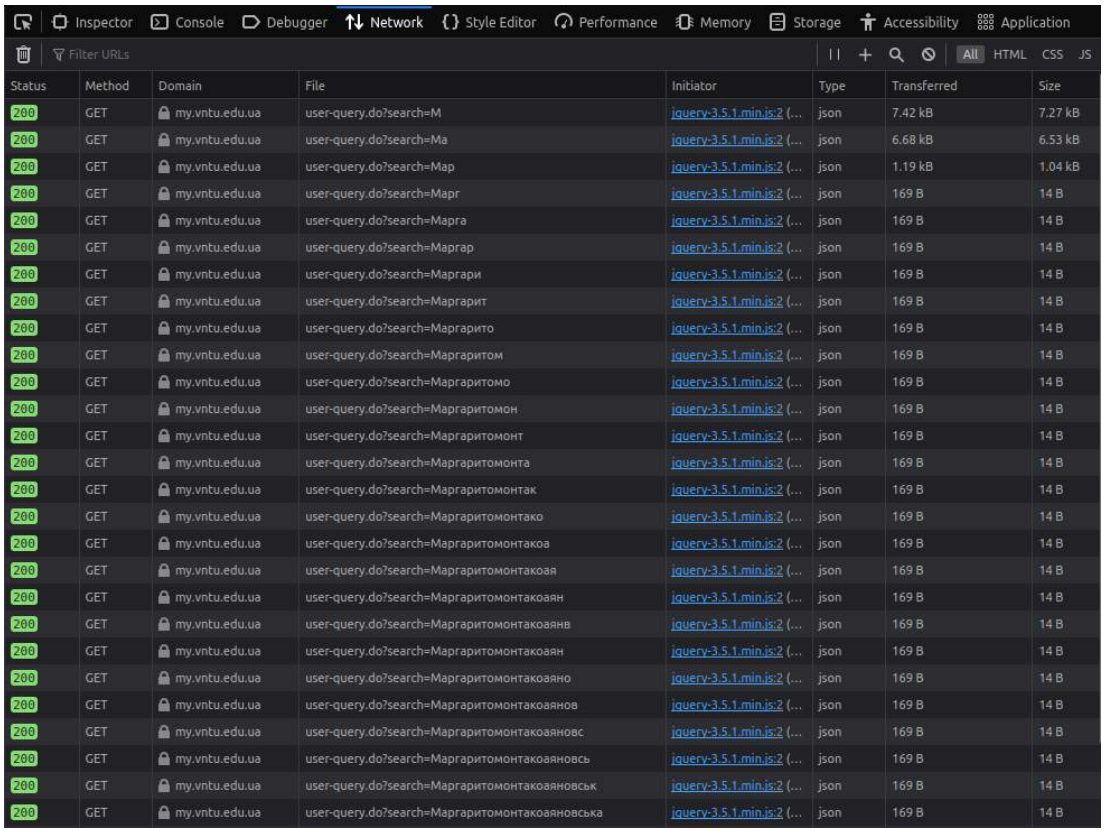


Рисунок 2.22 – Типовий сценарій роботи публічного API-ендпоїнту, за яким шкодоносний бот може спричинити виснаження запитами

В результаті виконання другого експерименту (рис 2.17) встановлено що публічний REST API-сервіс легко піддається виснаженню. Навіть з використанням кешу запитів на рівні компоненту REST API-проксі виснаження сервісу запитами всерівно залишається можливим. REST API-проксі – це проміжний шар або сервер, який дозволяє взаємодіяти з RESTful API, виконуючи функції проксі-сервера. REST API-проксі дозволяє клієнту взаємодіяти з віддаленим API через інтерфейс проксі-сервера, що може приховувати реальні характеристики та імплементацію віддаленого API.

Виходячи з існування цих двох проблем у попередньої версії системи SSO для JetIQ, а також ймовірності їх появи у новій версії системи розроблено метод визначення довіри до клієнтського пристрою. Його суть полягає у визначення довіри по чотирьом параметрам: ідентифікатор сесії користувача, IP-адресі, номеру автономної системи з якої походить IP-адреса, а також країна їх походження. Метод передбачає встановлення ієрархічної довіри до кінцевого пристрою. Новим клієнтам за відсутності інформації про рейтинг довіри до вищих за них по ієрархії мережевих об’єктів присвоюється максимальний кредит довіри. У іншому випадку кредит довіри спадкується від вищого по ієрархії об’єкта (рис 2.23). Стрілками на рисунку показаний напрям спадкування рівня довіри до клієнта. Між об’єктами у ієрархії існують зв’язки 1:N (один до багатьох).

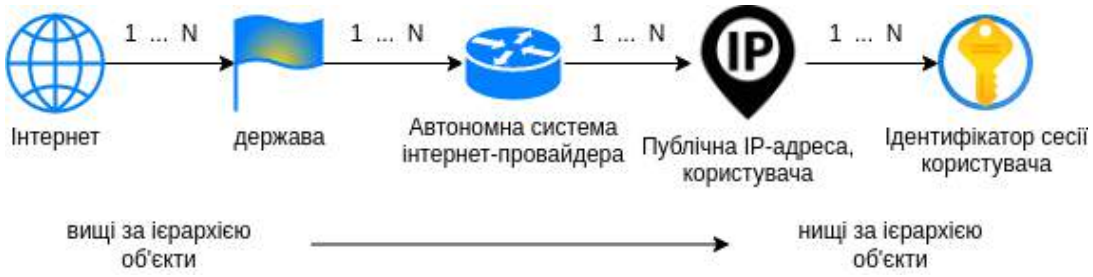


Рисунок 2.23 – Ієрархія об’єктів, що використовуються у методі визначення довіри до клієнтського пристрою

Для збільшення або зменшення довіри до того чи іншого об'єкта використовується принцип лічильника. На рівні держав застосовується множник, що відповідає рівню розповсюдженню інтернет-пристроїв відносно населення країни. Для ідентифікації браузеру використано вже наявний у документації ідентифікатор браузеру UAToken. Він власне і є ключем, що ідентифікує клієнт браузера. Загальний алгоритм методу зображено на рисунку 2.24.

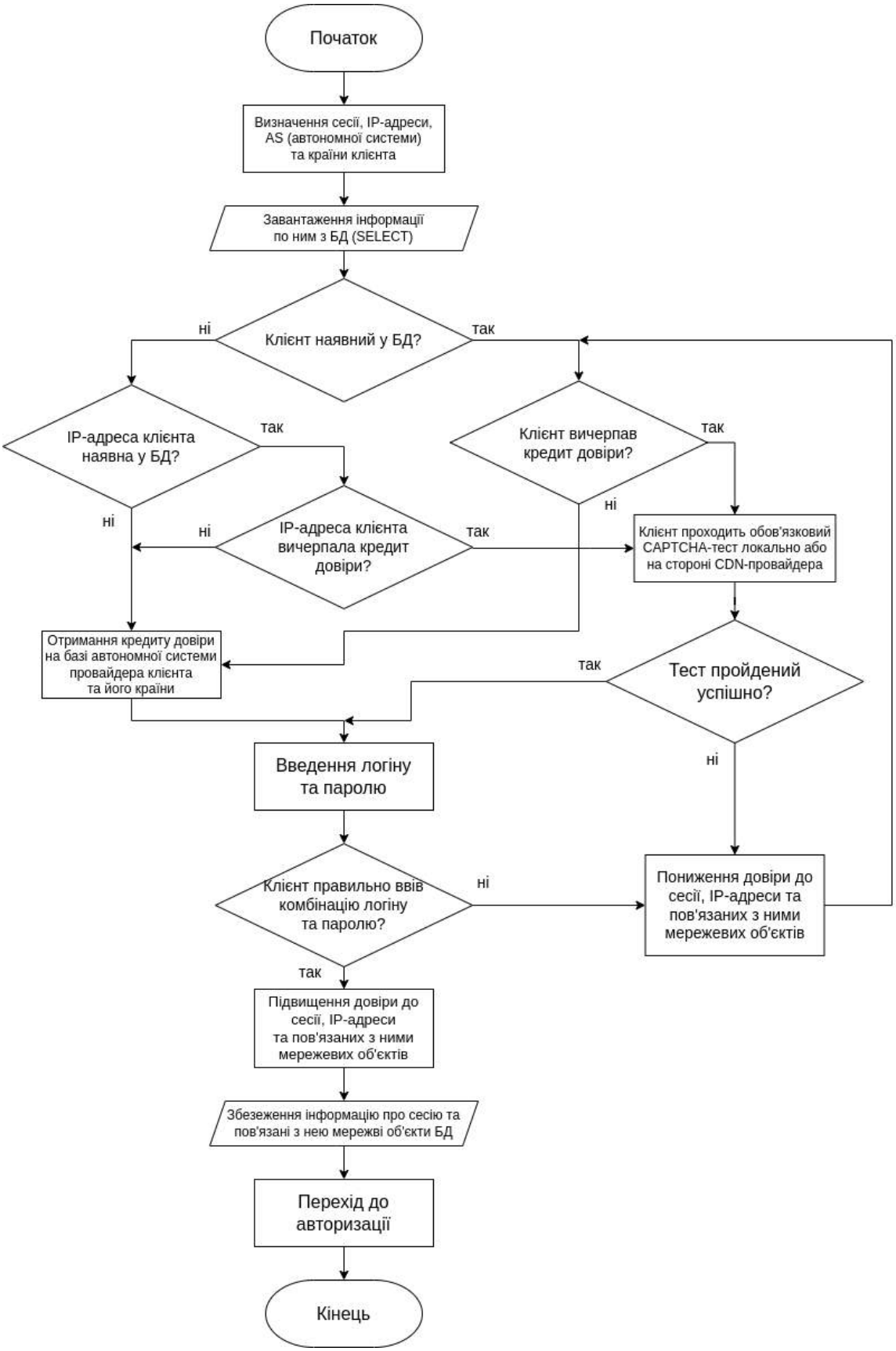


Рисунок 2.24 – Алгоритм роботи методу

Для роботи даного методу SSO-система має мати певні елементи архітектури фаєрволу веб-додатків (WAF). Найбільш базові елементи, а також їх зв’язки, необхідні для роботи методу зображені на рисунку 2.25. Фаєрвол веб-додатків CDN-сервісу на даній схемі є опціональним.

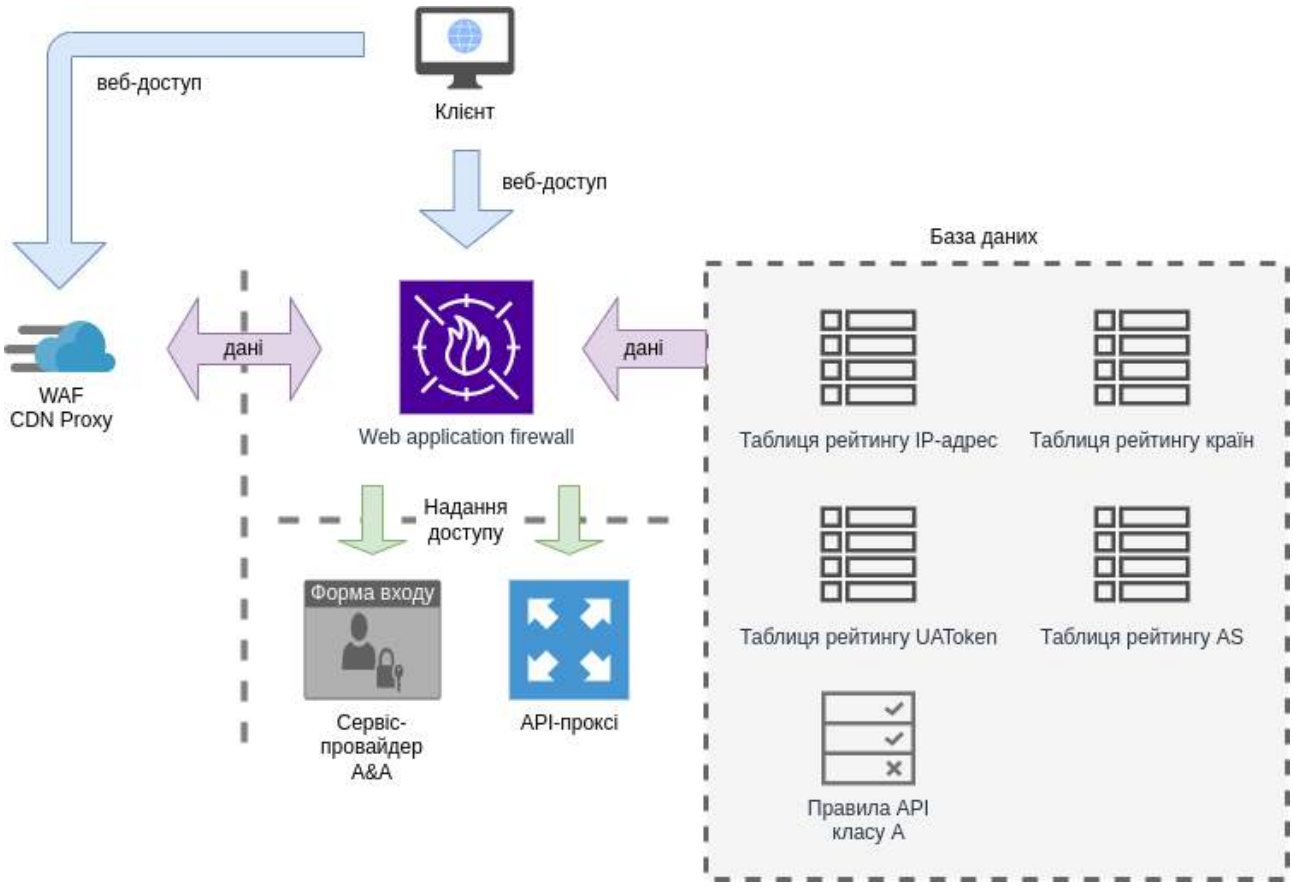


Рисунок 2.25 – Структура Web application firewall для реалізації методу визначення довіри до клієнтського пристрою

На рейтинг довіри до об’єктів також негативно впливають порушення ACL-правил API-проксі, ознаки DDoS-атак, а також виявлені AI-аналізатором лог-файлів веб-серверу аномалії, характерні для інших шкодоносних атак.

Отже, метод визначення довіри до клієнтського пристрою базується на існуючих методах роботи активного фаєрволу веб-додатків (WAF). Створений на базі них метод покращує їх застосування у SSO-системах.

### 2.6 Розробка моделі гейміфікації публічного тесту Тюринга

На момент написання даної роботи системи комп'ютерного зору дедалі частіше показує успіх у вирішенні більш простих тестів CAPTCHA. Концепція змагання замка та відмички передбачає що тести Тюрінга, такі як CAPTCHA, як і комп'ютерні програми що здатні їх вирішити будуть ставати дедалі складнішими. Зараз існує доволі палка дискусія між вченими, а також між розробниками програмного забезпечення стосовно того, що ще доведеться робити для того, щоб відокремити справжніх людей від надрозумного штучного інтелекту. Справа у тому що рівень штучного інтелекту невпинно зростає і ніхто нічого не може з цим подіяти. Що стосується тесту Тюринга, то існує проблема що з часом він може стати настільки складним, що людям і самим стане непросто його проходити.

На порятунок приходять дві технології: JavaScript Challenge та гейміфікація. Перша є доволі дієвою, проте її чекають дуже серйозні виклики у майбутньому змаганні ключа та відмички. Цілком ймовірно що з часом її ефективність буде хвилеподібним чином зростати, а потім спадати, щоб потім знову зрости, зрештою знов послідує чергове спадання. Тому цілком можливо що для того щоб пістрахувати її у більш вразливі моменти буде використовуватись інтерактивний тест Тюринга. Зараз для цього став більш поширений методі гейміфікації публічного тесту Тюринга. Досить часто нині трапляються гейміфікації цього тесту, наприклад з пазлами, фігурками, які потрібно обернути за напрямком руки, або наприклад в ролі собаки-вівчарки змусити вівцю зайти у правильні ворота.

Останній приклад передбачає управління персонажем у тривимірному просторі. Це дуже гарний підхід, адже AI на перших порах буде важко ним керувати. Основною проблемою такого підходу є можливість застосування скриптів швидкого проходження (т.з. Speedrun script), які можливо запускати зокрема у середовищі Selenium. Комп'ютерне бачення без проблем може ідентифікувати рівень гри гейміфікації та запустити відповідний скрипт

проходження. Іншою проблемою є те що на даний момент з’являються speedrun-боти які можуть керувати примітивними персонажами у гарно розрізненому просторі. Такі боти використовуються гравцями у онлайн-іграх щоб патрулювати територію, збирати ресурси, купувати та продавати ігрові товари без участі гравця.

Саме тому необхідно створити модель гейміфікації, яка б могла створювати рівні з випадковим просторовим плануванням, а також могла б запропонувати UI/UX-рішення, яке було б зрозумілим для людей, але водночас була б складною для автоматизації. В даній моделі гейміфікація передбачається керування антропоморфним персонажем, який проходитиме через складний ландшафт, де можна послизнутись або впасти, оскільки людина швидше навчається орієнтуванню у просторі [38] та має більший досвід ходьби і уникненню падінню.

Нову модель гейміфікації створено на базі наступних моделей: ReLU, Сигмоїду та Гауса (рис. 2.26). Таким чином вона їх поєднує та розширює. Перші дві моделі часто використовуються як основа для генеративних змагальних мереж (*англ.* Generative adversarial network, GAN). За допомогою використаних моделей відбувається навчання, генерація та валідація планів поверхів. Найбільш оптимальним підходом до побудови моделі є можливість виконання паралельних процесів. В рамках моделі передбачено 5 окремих паралельних процесів: навчання, генерації і валідації планів поверхів, а також побудова рівнів і власне геймплей. Перші чотири процеси ініціюються планувальником завдань. Це є найбільш оптимальним підходом, оскільки нині обчислювальний ресурс GPU, який потрібен для двох перших процесів, є значно дорожчим у хмарному середовищі, тому його більш оптимально використовувати по спотовій моделі обчислень. Подібна модель організовує розпродаж невикористовуваних ресурсів хмари, за рахунок чого можна серйозно заощадити на хмарних обчисленнях правильно підлаштувавши під цю модель процеси у власній системі.

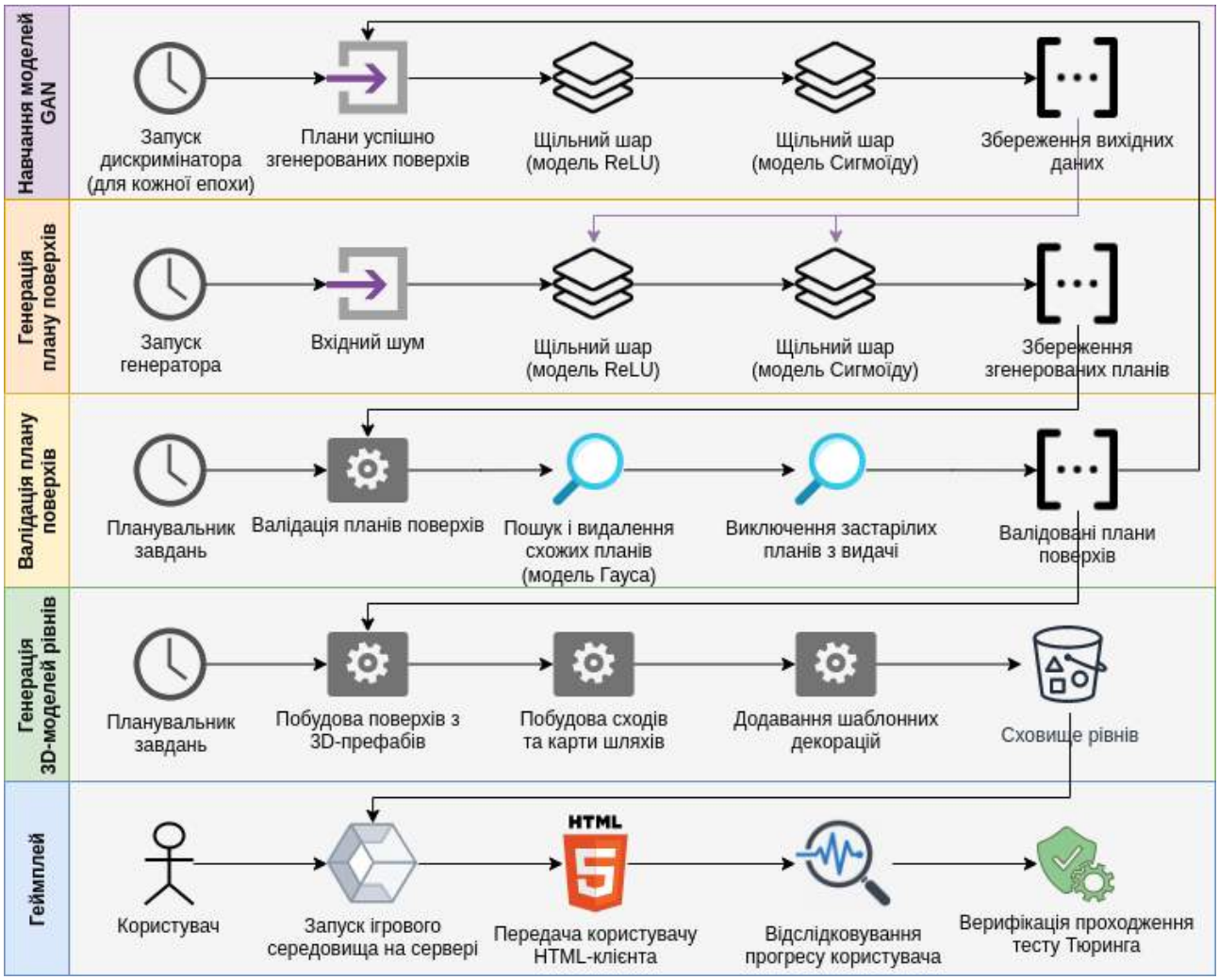


Рисунок 2.26 – Вдосконалена модель гейміфікації публічного тесту Тюринга

Перші три рівні моделі генерують плани поверхів – bitmap-зображення, в яких кожен піксель означає секцію коридору чи сходів. Плани рівнів мають лише три кольори: чорний (коридори) та червоний (клітка сходів), а також фон (білий). У першому рівні розробленої моделі відбувається навчання моделі на базі вже створених планів поверхів. Другий рівень відповідає за генерацію нових планів поверхів. В результаті можуть бути далеко не ідеальні карти поверхів (рис. 2.27). Їх верифікує третій рівень, усуваючи неможливі у реалізації рівнів варіанти. Після відкоригованого навчання зростає кількість планів поверхів, які відповідають вимогам фільтру (рис. 2.28), і як наслідок модель стає дедалі більш ефективною.

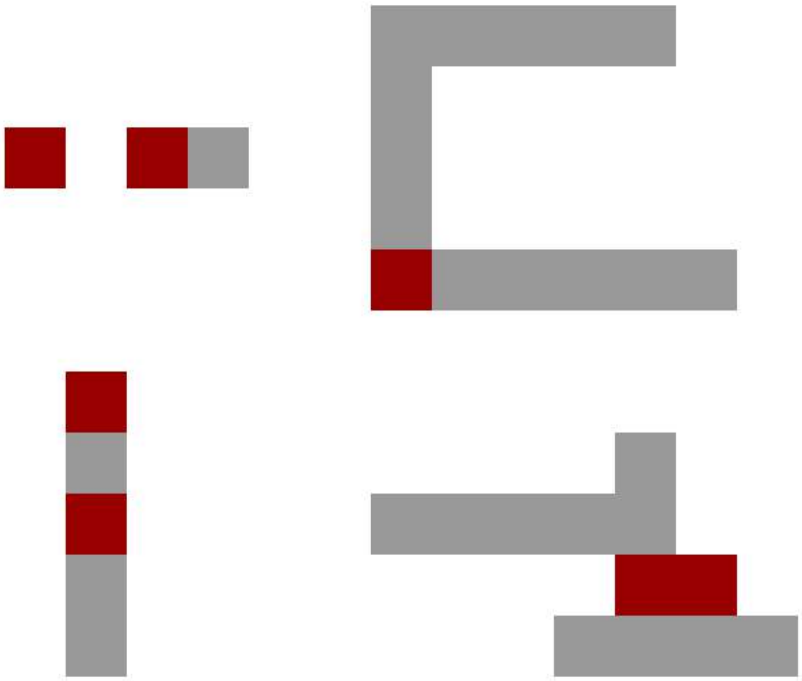

Рисунок 2.27 – Приклад неправильно згенерованих планів поверхів

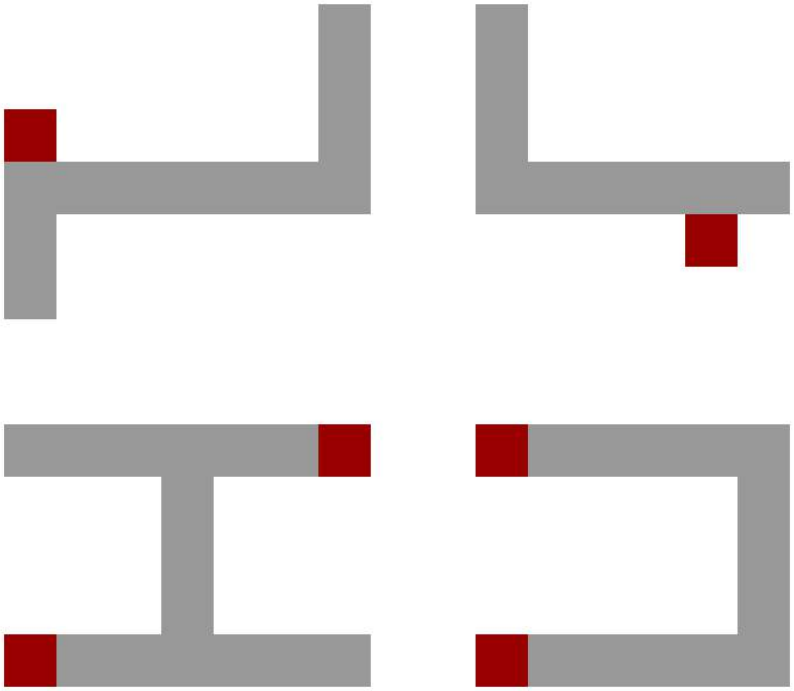

Рисунок 2.28 – Приклад правильно згенерованих планів поверхів

На четвертому рівні моделі відбувається збірка рівнів згідно планів поверхів. На місці кожного пікселю плану у тривимірному просторі спавниться (динамічно створюється) відповідний префаб – фрагмент наперед створеного поверху. У кінці поверх дублюється потрібну кількість раз, що відповідає кількості поверхів. На п’ятому рівні відбувається сам ігровий процес. Два останні рівні реалізуються у третьому розділі даної роботи. Програмна компонента гейміфікації публічного тесту Тюринга реалізується у вигляді окремого автономного сервісу, який є вибірковим (систему можна буде встановити як з ним, так і без нього).

### 2.7 Висновки

В другому розділі магістерської роботи представлено аналіз існуючих підходів та засобів для розробки SSO-систем. Досліджено не покритий документацією функціонал компоненти API системи SSO для JetIQ. Розроблено бізнес-процеси, програмну архітектуру, а також методи та модель гейміфікації для інформаційної системи централізованої авторизації для освітнього веб-середовища навчального закладу.

Отже, згідно технічного завдання та розробленої програмної архітектури SSO-система має бути реалізована у вигляді двох підсистем: REST API IdP-сервісу та веб-додатку сервіс-провайдеру A&A. У них мають бути реалізовані розроблені методи та модель гейміфікації.

# 3. РОЗРОБКА КОМПОНЕНТІВ СИСТЕМИ ЦЕНТРАЛІЗОВАНОЇ АВТОРИЗАЦІЇ ДЛЯ ОСВІТНЬОГО ВЕБ-СЕРЕДОВИЩА НАВЧАЛЬНОГО ЗАКЛАДУ

## 3.1 Вдосконалення структури та коду REST API-сервісів

У другому розділі досліджено не покритий документацією функціонал компоненти REST API системи SSO для JetIQ, а також запропоновано ряд покращень його роботи. У даному підрозділі пропонується реалізація всіх необхідних покращень для даної компоненти. Ця компонента існує у двох версіях: APIv1 та APIv2 (перебуває у стані активної розробки). Перша версія реалізована одразу на двох мовах програмування складається з двох модулів: модуль взаємодії із сервісами (написаний на мові програмування PHP) та модуль користувацьких сервісів (написаний на мові програмування Java на базі фреймворку Apache Struts). Документація передбачає усунення двомовності та уніфікацію цих двох модулів у одному API-компоненті APIv2. Також у документації передбачено подальше повна заміна API першої версії другою, але з забезпеченням максимально можливої сумісності ендпоїнтів другої з першою. Документація передбачає використання у другій версії URI-маршрутизації подібної до Apache Struts.

Серед вдосконалень сервісу REST API другої версії, які виконано у даній можна виділити наступні:

- реалізовано метод розподілу на окремі точки доступу (описаний у підрозділі 2.4); для цього створено API-проксі;
- підвищено захищеність процесу авторизації у протоколі JetIQ SID завдяки зіставленню IP-адреси та User-Agent браузера чи моб. додатку (рис. 3.1);
- запропоновано підвищення захищеності процесу ланцюгової авторизації у протоколі JetIQ SID завдяки перевірці UAToken клієнта (рис. 3.2);
- розроблено модулі `iq-retro`, `iq-sync` та `db-tools` для синхронізації даних користувачів із LMS-системою JetIQ (рис. 3.3).

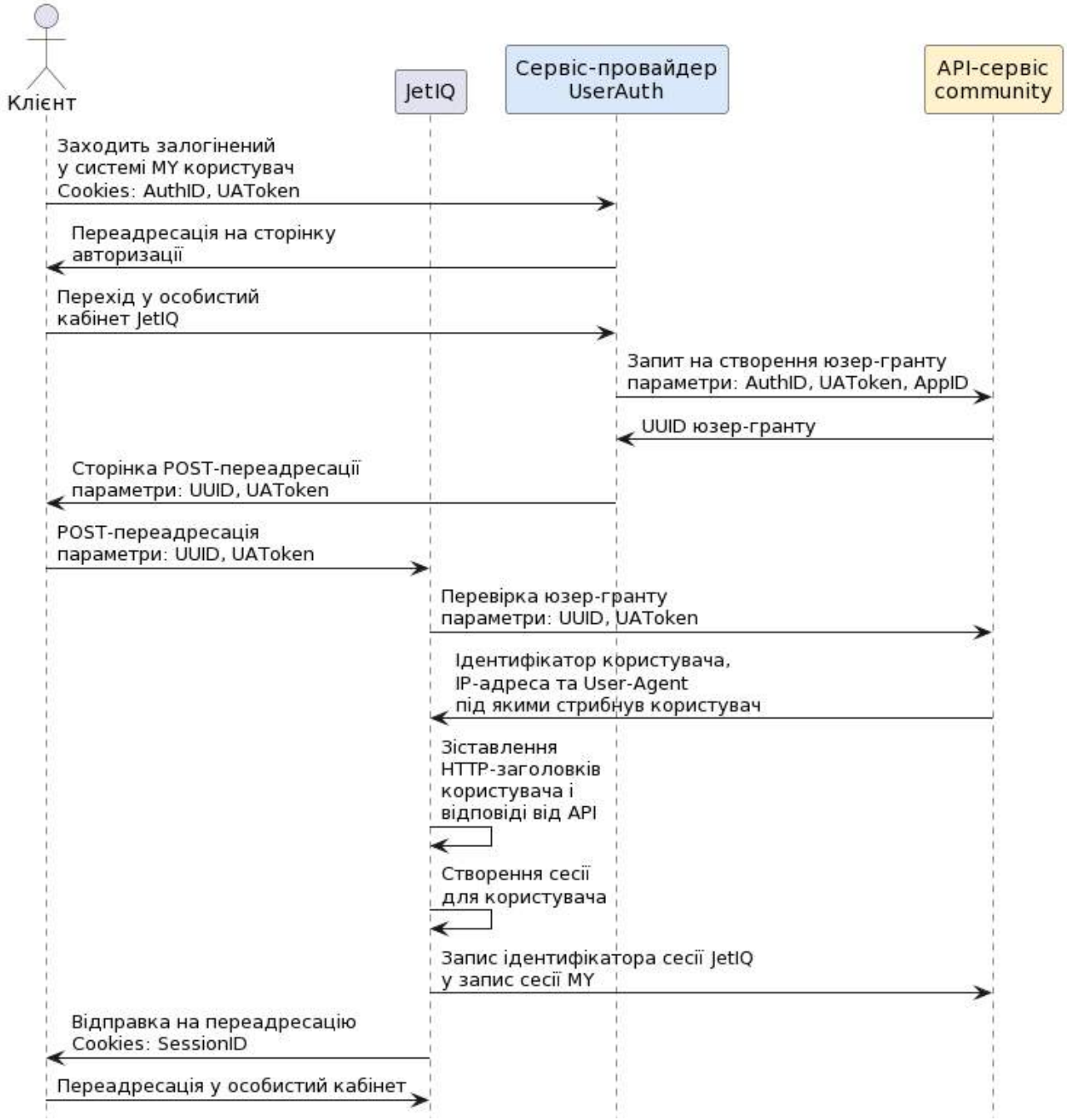


Рисунок 3.1 – Процес авторизації у протоколі JetIQ SID

На рисунку 3.1 зображено процес авторизації у протоколі JetIQ SID із запропонованими удосконаленнями. У цьому протоколі даний процес запозичений у протоколу OAuth1 але із деякими відмінностями, такими як передача ідентифікатору браузеру користувача UAToken та запис у системі MY ідентифікатора сесії JetIQ для подальшої ланцюгової авторизації.

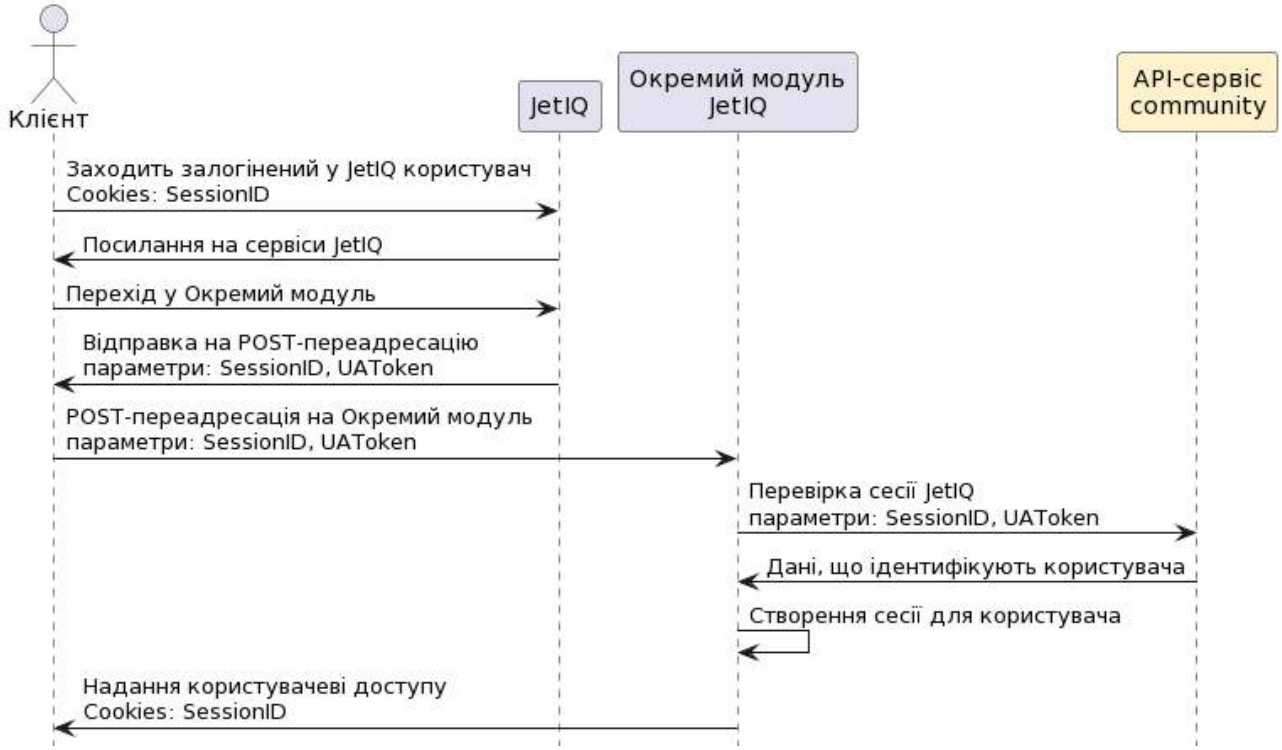


Рисунок 3.2 – Процес ланцюгової авторизації у протоколі JetIQ SID

На рисунку 3.2 зображений процес ланцюгової авторизації, передбачений у протоколі JetIQ SID. Необхідність у цьому процесі заважає замінити протокол JetIQ SID у цій системі управління навчанням на більш стандартизований OAuth2. Даний процес передбачає що авторизований користувач на сервісі JetIQ зможе переходити на окремі модулі без створення окремого юзер-гранту. Найкращим способом підвищення безпеки ланцюгової авторизації є застосування технології JSON Web Tokens (JWT). Ця технологія була розроблена спеціально для випадків, де немає можливості виконувати часті запити до REST API. Однак поки що її важко реалізувати у legacy-додатках, написаних на більш старих версіях PHP, наприклад 5.6, оскільки її бібліотеки для цієї мови програмування вимагають як мінімум версію 7.4. Тому для таких випадків, де немає можливості використовувати JWT тимчасово запропоновано перевіряти ідентифікатор браузеру клієнта (UAToken) при виконанні стрибків ланцюгової авторизації.

Ідентифікатор UAToken має з’являтись у системі SSO разом із сесію

користувача коли користувач успішно проходить процес автентифікації Цей ідентифікатор передається при переході із системи SSO на різні сайти, у тому числі при використанні ланцюгової авторизації. Це суттєво поліпшує процес відстеження помилок та інших аномалій у системі журналювання (logging). Наприклад можуть виникати випадки коли ланцюгова авторизація відбувається безуспішно, і тоді завдяки ідентифікатору UAToken можна з'ясувати у якого користувача і за яких обставин це відбулось. Цей ідентифікатор використовується як додатковий ключ у багатьох API-запитах, пов'язаних із авторизацією. Він є опціональним полем при використанні протоколу OAuth2, який сервіс може отримати про користувача.

У першій версії системи для опрацювання користувацьких REST API-запитів використовувася створений на базі фреймворка Apache Struts модуль "`anonymous`" для опрацювання запитів, що надходять від фронтенд-частини SSO-системи. Модуль мав здатність кешувати найбільш часто використовувані запити, що з однієї сторони розвантажувало SSO-систему, від надлишкових запитів, але в той же самий час не захищало від виснаження API-запитами. Для вирішення цієї проблеми та забезпечення подальшої сумісності вже розроблених додатків, які використовували цей модуль розроблено фасад у вигляді API-проксі, в якому можливо виконувати як кешування, так і застосовувати розроблені методи (див. підрозділи 2.4 та 2.5). Більш детально яким чином він працює описано у підрозділі 3.2.

Модулі `iq-retro`, `iq-sync` та `db-tools` є складовими синхронізатора користувачів. Синхронізатор підтягує актуальні дані із системи JetIQ у систему SSO. У даній конфігурації система SSO самостійно отримує початкові дані лише при появі студента або співробітника у системі. Всі інші актуальні дані про студентів та співробітників надходять із JetIQ через синхронізатор (рис. 3.3).

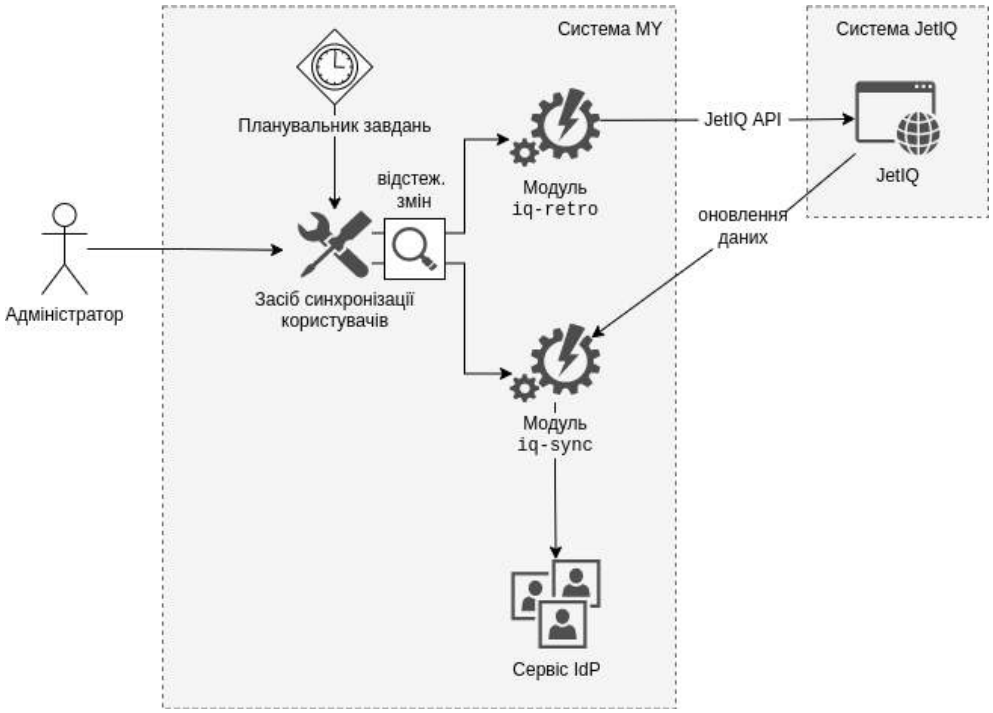


Рисунок 3.3 – Принцип роботи синхронізатора користувачів

БД JetIQ містить найбільш актуальну інформацію про користувачів, оскільки підсистеми відділу кадрів та електронного деканату оновлюють у першу чергу її. Відмінність двох розроблених модулів полягає у тому, що модуль `iq-retro` видобуває актуальні дані із JetIQ API, а модуль `iq-sync` надає інтерфейс для оповіщення про зміни.

Завдяки поліпшенням та створенням нових модулів для рівня API для системи SSO для JetIQ стає можливим подальше створення користувацького фасаду у вигляді сервіс-провайдеру автентифікації та авторизації, а також панелі самообслуговування. Лістинг файлів, створених при роботі над API наведено у додатку Г.

### 3.2 Розробка бекенд-частини системи централізованої авторизації

На бекенд-частину покладено безліч функцій по взаємодії системи SSO з користувачем: вхід у систему, вхід за допомогою сторонніх сервісів, відновлення паролю та інші (рис. 3.4). Бекенд-частина розподілена на веб-додатки. Основні веб-додатки визначено у процесі створення архітектури у другому розділі даної

роботи, їх основний функціонал наведено у таблиці 2.5. Їх взаємозв’язки з рівнем API показані на рисунку 2.12. В якості фреймворка для створення бекенд частини обрано раніше створений мікрофреймворк IQCoreApp. Він написаний на мові програмування PHP, розповсюджується за вільною ліцензією MIT та на його базі може бути легко створена система, якій потрібна шаблонізація, багатомовність та підтримка об’єктно-орієнтованих сесій з підтримкою Redis та Amazon DynamoDB.

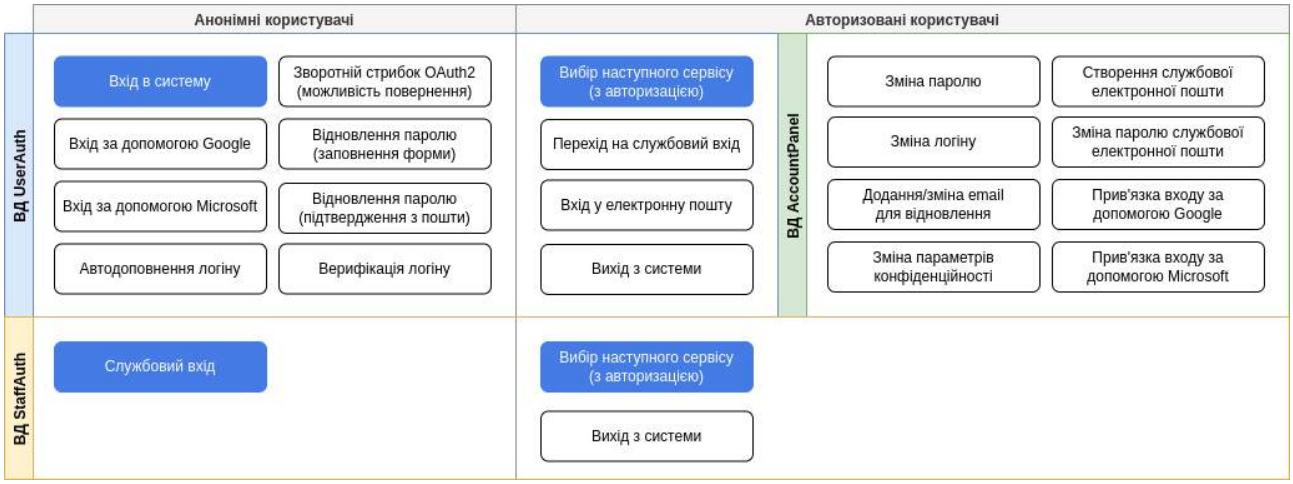


Рисунок 3.4 – Основний функціонал, що реалізований у бекенд-частині

У цій роботі створено удосконалену версію цього фреймворка із вбудованим Web Application Firewall, в якому реалізований метод визначення довіри до клієнтського пристрою, що був розроблений у підрозділі 2.5. Реалізація WAF включає у себе API-зв’язок з хмарними WAF, такими як AWS WAF та CloudFlare для більш гнучкого захисту від DDoS-атак.

Кожен веб-додаток, попри те, що запускається у окремому контейнері, має спільну кодову базу з іншими, що забезпечується міжкласовими зв’язками (рис. 3.5). Існує чотири базових класи, які надаються мікрофремворком, які у свою чергу використовуються всіма веб-додатками.

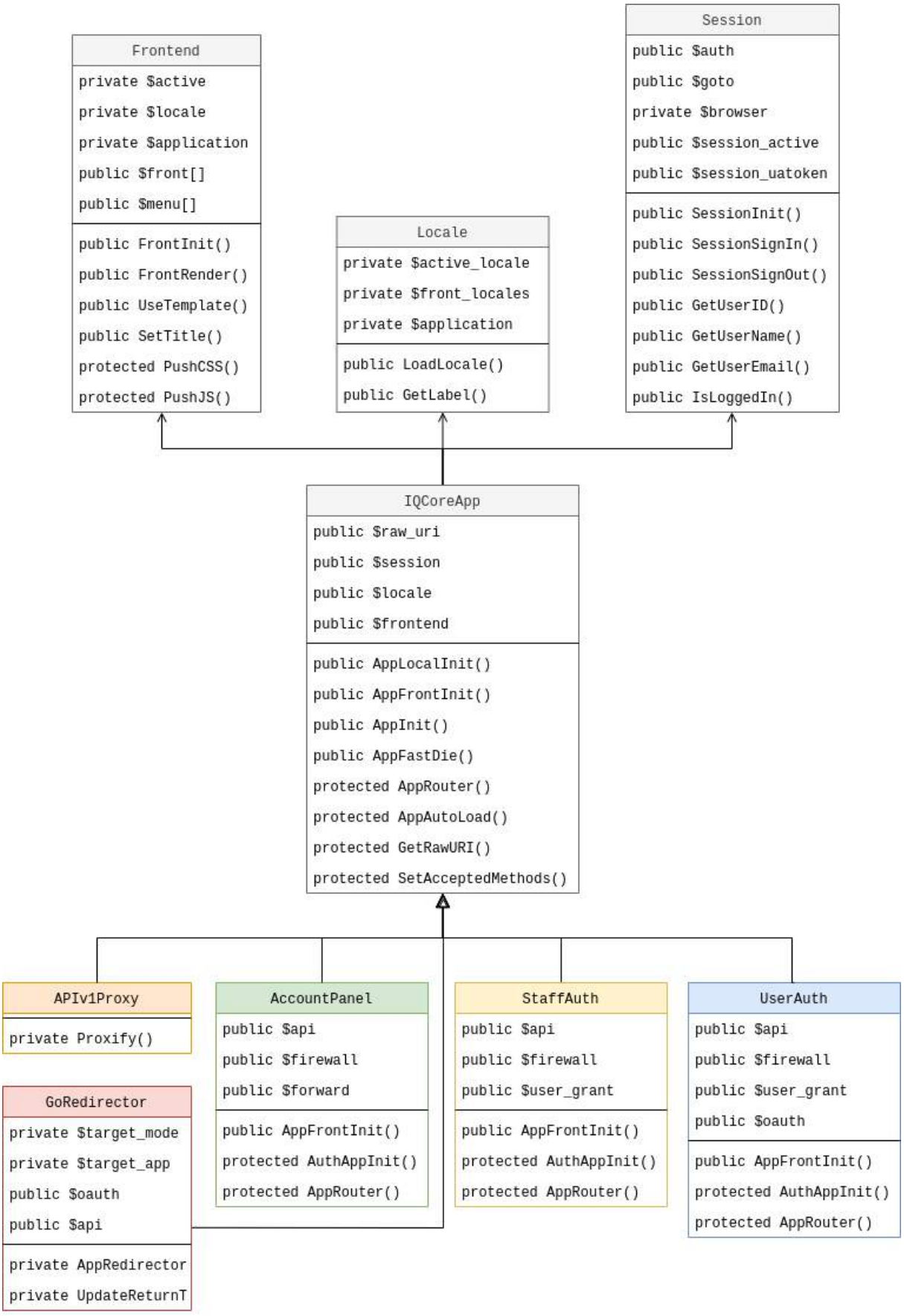


Рисунок 3.5 – Діаграма класів сервіс-провайдеру A&A

Створений API-проксі для підвищення сумісності із попередніми версіями API також інтегровано у рівень веб-додатків. Завдяки інтеграції з вбудованим у мікрофреймворк WAF від дозволяє більш потужно захищати внутрішній сервіс API від перевиснаження запитами, спричиненими цільовими DDoS-атаками.

Він надає доступ до наступних основних API-методів (YAML-код їх визначення розміщується у додатку Г):

- `user-query`. У цьому методі відбувається пошук за логіном чи прізвищем. Це дасть користувачам, які не підв'язали пошту входити за допомогою логіну чи ПІБ. Для зручності для початкових входів у систему при вводі перших цифр або букв має виводитись список автодоповнення. Відповідь надходить у форматі JSON (рис. 3.5). Це дасть користувачеві обрати себе у списку.
- `user-verify`. Цей метод дозволяє верифікувати логін чи прізвище за яким намагається увійти користувач. Як результат успішної верифікації він повертає ідентифікатор AvatarID, який у подальшому використовується у процесі автентифікації.

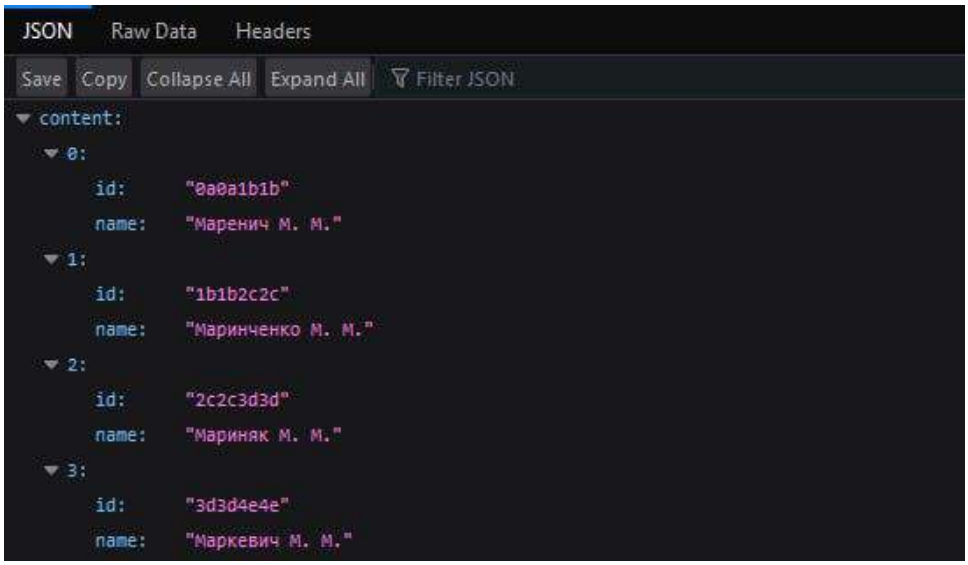


Рисунок 3.6 – Приклад виконання запиту "Map" у ендпоїнті `user-query`

- `user-nick`. Цей метод доступний лише авторизованим користувачам та дозволяє студентам реєстрації у JetIQ перевірити чи занятий їх нік, чи ні.

- `waf-status`. Дозволяє фронтенд-частині виявити чи не виникло обмежень у його браузеру, пов’язаних із частими невдалими спробами входу.

Коли користувач підключається до будь-якого з веб-додатків вперше, то WAF присвоює його браузеру токен UAToken, який зберігається за ним весь час користування сервісом SSO. Кожен персоніфікований користувач має унікальний ідентифікатор AvatarID, який являє собою 8-ми значне hex-число. Крім використання у системі SSO рекомендується його впровадження у інших сервісах ЗВО, що пов’язані з JetIQ. Також цей ідентифікатор використовується у протоколах OAuth2 та SAML для ідентифікації користувача у зовнішніх сервісах.

### 3.3 Розробка фронтенд-частини системи централізованої авторизації

Фронтенд-частина, як видима для користувача частина будь-якого веб-додатку виконує роль графічного інтерфейсу. У SSO-системі важливий user-friendly підхід, щоб користувачу було найбільш легко ним користуватись. Також важливо щоб веб-додаток сервіс провайдера A&A швидко завантажувався і мав адаптивний інтерфейс. Саме тому для нього обрано найбільш легкий фронтенд-фреймворк Bootstrap 5 та JavaScript-бібліотеку jQuery 3.5.

Перед розробкою фронтенд-частини проведено розробку UI/UX всіх діалогів і форм з урахуванням побажань експертів. Для нових користувачів університету важливо швидко потрапляти у особистий кабінет. Саме тому для них реалізовано автодоповнення логінів (рис. 3.7). Для того щоб їм не доводилось заново вводити логін додано функцію запам’ятовування логіну при вводі паролю (рис. 3.8). Всі імена на скріншотах вигадані і не мають стосунку до справжніх людей.


Рисунок 3.7 – Автодоповнення для нових користувачів

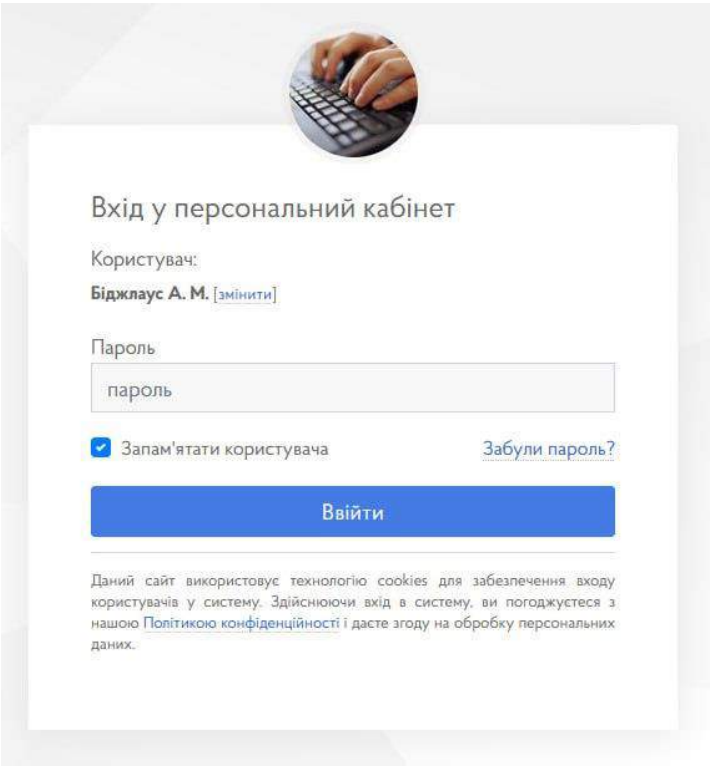


Рисунок 3.8 – Форма введення паролю

Коли користувач обрав опцію запам’ятовування логіну після виходу із системи він/вона побачить пропозицію скористатись запам’ятованим логіном (рис. 3.9). Якщо одним браузером користується кілька користувачів системи – тоді система пропонуватиме вибір між різними обліковими записами.

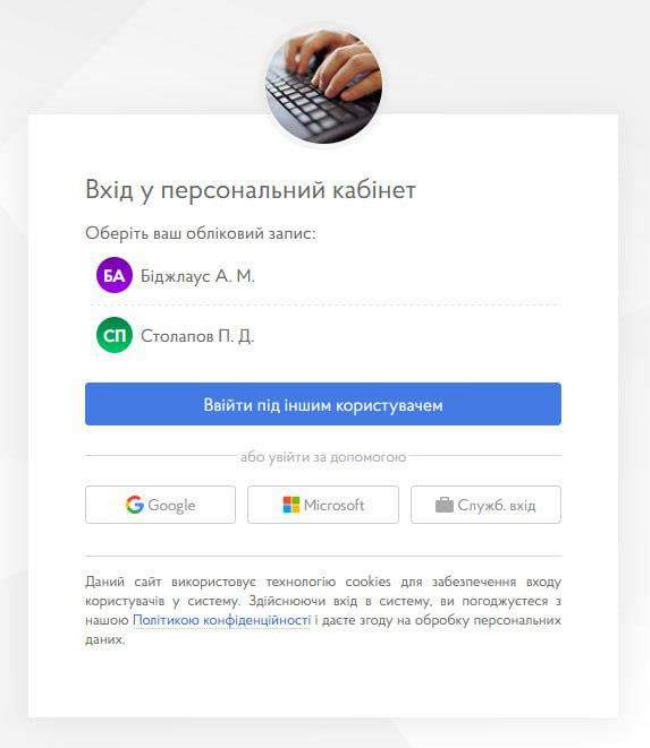


Рисунок 3.9 – Вибір запам’ятованих користувачів

Функція автодоповнення логіну в першу чергу розрахована на нових користувачів та користувачів з обмеженими можливостями. Для викладачів та співробітників вона пропонує вибір серед прізвищ та ініціалів, а для студентів – автодоповнює номери залікових книжок, що використовуються для входу. Оскільки такий спосіб входу не є повністю безпечним, користувачам пропонується замінити автодоповнюємий логін на вхід за електронною поштою. При наявності пошти Google чи Microsoft користувачі зможуть заходити у систему не менш комфортним чином (рис. 3.9) – завдяки кнопкам миттєвого входу.

### 3.4 Розробка панелі самообслуговування користувача

Відключити своє з’явлення у випадаючих списках та додати акаунт Google чи Microsoft користувач може у панелі самообслуговування користувача (рис. 3.10). Така панель дозволяє користувачам організації керувати всіма акаунтами на сайтах організації з одного місця. При розробці панелі самообслуговування користувача застосовано підхід MVC.

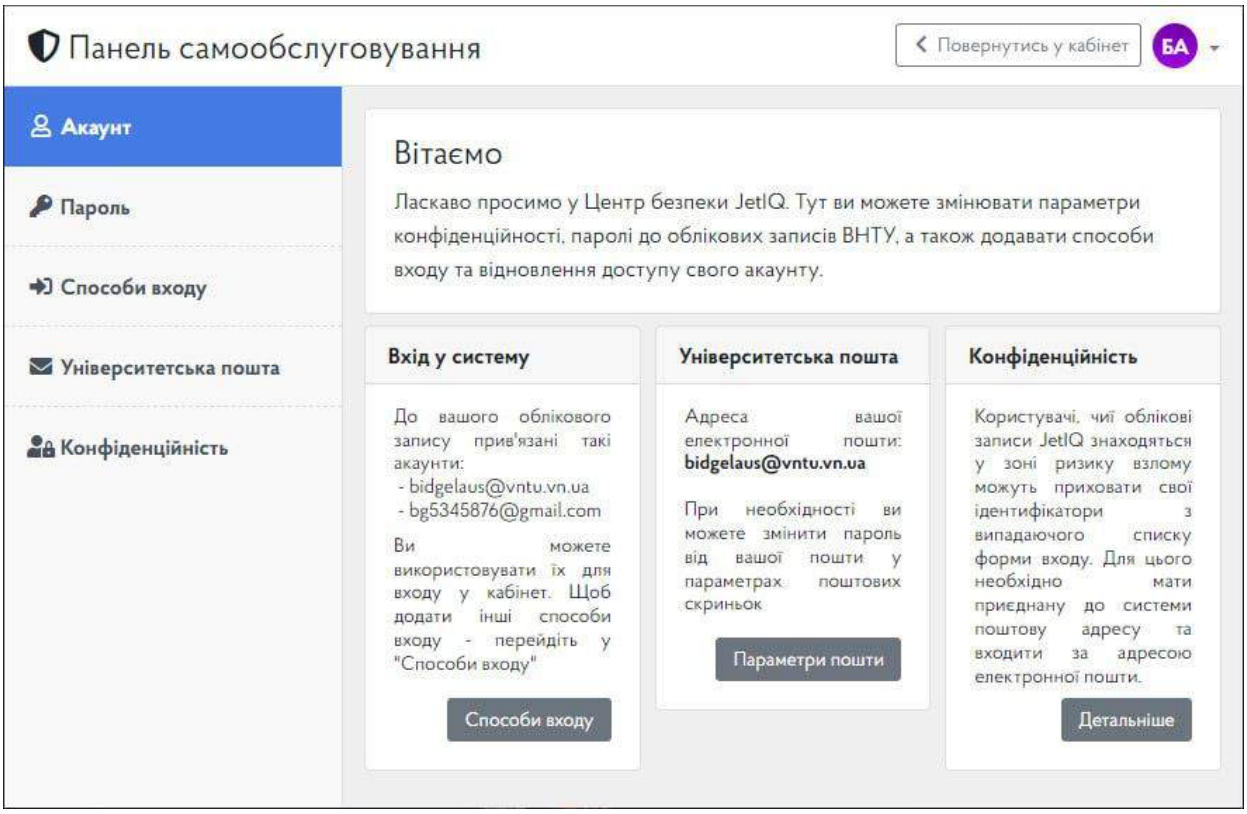


Рисунок 3.10 – Панель самообслуговування користувача

До панелі можна підключити API управління корпоративною поштою, такі як Doveadm API (поштовий сервер на основі Dovecot) або хмарні сервіси, такі як Google Workspace (у минулому G Suite) та Microsoft 365 (у минулому Office 365). Користувачі зможуть самостійно створювати собі поштові скриньки на цих сервісах, а також змінювати паролі до них.

### 3.5 Розробка модулів взаємодії із зовнішніми сервісами

Для підключення нових додатків рекомендується використовувати протокол OAuth2. Додатки додаються через зміну конфігураційних JSON-файлів. При необхідності адміністратор може налаштувати швидкий вхід на сервіси Google Workspace та Microsoft 365 за допомогою протоколу SAML. Як опцію для подальшого розвитку делегованого службового доступу додано можливість для користувачів додавання MFA від Google Authenticator та Microsoft Authenticator (при наявності зв’язку із Azure).

При використанні JetIQ в якості системи-компаньона система MY дозволяє налаштувати інтеграцію з мобільними додатками JetIQ для викладача та студента. Оскільки ці додатки не передбачають підтримки протоколу OAuth2, автентифікація у них відбувається через протокол JetIQ SID. На даний момент до впровадження у них використання протоколу OAuth2 ці додатки працюють у режимі менш безпечних додатків, тобто пропускають логін та пароль через себе.

Серед інших менш безпечних додатків можна виділити VPN-доступ. На даний момент для кожного користувача генерується окремий пароль для VPN-доступу [39]. Це реалізовано за допомогою протоколу RADIUS. На даний момент ця опція недоступна для всіх користувачів, але лише для тих, кому адміністратор надав такий доступ.

Для адміністрування користувачів створено адмін-панель. Вона максимально подібна до панелі самообслуговування. Функціонал адмін-панелі здебільшого тотожний панелі самообслуговування, але адміністратор може робити це масово.

Адмін-панель потрібна для перших етапів впровадження системи та міграції даних, але використовуватись вона буде лише службою IT-підтримки, оскільки її функції частково можна перекласти на системи відділу кадрів та електронного деканату.

### 3.6 Розробка програмної компоненти гейміфікації публічного тесту Тюринга

Для того щоб переконатись чи є людина роботом пропонується гейміфікований тест Тюринга. Користувач може як один із варіантів обрати або більш складний інтерактивний тест, або гейміфікацію.

Гра, що закладена у гейміфікацію, поялгає у тому, що головний персонаж, який є кур'єром, має знайти потрібну квартиру (рис. 3.11) у багатоквартирному будинку або гуртожитку. Але є кілька незвичайних обставин: відомо лише слово, що написане на дверях, а поверх можна з'ясувати лише по наявності відповідного слова на поштових скриньках. Крім того підлога коридорів має помітні тріщини, які потрібно перестрибувати, а сходи теж зазнали руйнувань, спричинених війною: деякі сходинки надбиті, а деякі прогинаються, через що можна втратити баланс та розгубити піцу, що доставляється. Світло час від часу блимає через проводку, яка іскрить. Звучить як справжній жах не лише для людини, а й для інтелектуальних ботів, які у майбутньому зберуться доводити що вони є людиною.

Рисунок 3.11 – Викривлені написи на дверях будинку

Модель будівлі створюється за допомогою моделі (рис. 2.21), створеної у підрозділі 2.6. Модель спочатку генерує план поверхів, після чого згідно створеного плану будується модель кожного поверху (рис. 3.12), після чого вони зшиваються у ігровий рівень у вигляді єдиного будинку.

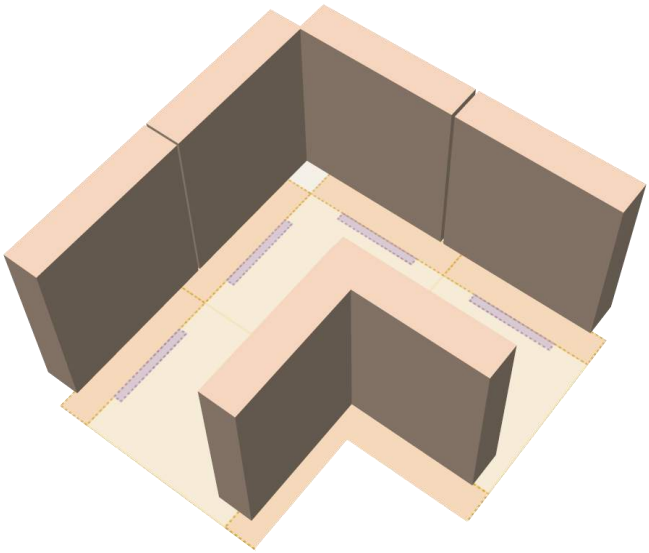

Рисунок 3.12 – Побудова поверху з готових блоків

Використано безкоштовні декорації та базову модель людини з Unreal Engine Marketplace. Розроблено дизайн головного персонажу гри (рис. 3.13) завдяки модифікації обличчя, одягу та волосся.

Рисунок 3.13 – Головний персонаж-кур’єр

Гра працює на основі двигуна Unreal Engine 5. Гейміфікація працює на клієнт-серверній архітектурі, яка дає користувачеві підключатись до ігрового серверу прямо із браузеру (рис. 3.14). Завдяки підходу Offscreen rendering, вся ігрова логіка та рендеринг працює на стороні серверу, а користувач лише керує із веб-інтерфейсу. Контейнери із сервером спавняться із стандартного образу Docker. Їх запускає сервер контролю та верифікації тесту (2), коли клієнт вперше до нього підключається (1). Коли ігровий сервер готовий до роботи, сервер контролю та верифікації робить редірект на нього. Під час гри та по її завершенні ігровий сервер через REST API надсилає проміжні та кінцеві результати тестування.

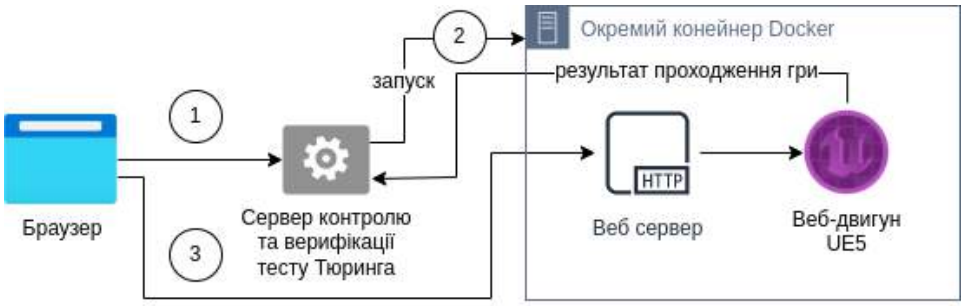


Рисунок 3.14 – Запуск ігрового серверу у контейнері Docker

Цілком можливо що в процесі змагання ключа та відмички інтелектуальні боти почнуть проходити гру за допомогою speedrun-скриптів. Ці скрипти автоматизують швидке проходження конкретної гри. Вони зазвичай існують у вигляді програмного коду або набір інструкцій, які використовуються гравцем для автоматизації певних завдань або дій в грі.

Однак це не обмежує можливості даної моделі гейміфікації до подолання активності ботів: на стороні гри цілком можливо запровадити античіт-захист, а також за допомогою засобів штучного інтелекту можливо проводити аналіз ігрової поведінки гравця щоб визначити чи є він ботом чи ні.

Гейміфікований публічний тест Тюринга не є обов'язковим компонентом розробленої SSO-системи і може використовуватись окремо.

### 3.7 Висновки

У третьому розділі виконано вдосконалення структури та коду існуючих REST API та IdP сервісів системи SSO для JetIQ, розроблено бекенд-частину та фронтенд-частину сервіс-провайдеру автентифікації та авторизації, розроблено панель самообслуговування, модулі взаємодії з іншими сервісами, а також програмну компоненту. Використано найбільш сучасні фреймворки та програмні бібліотеки. Для розробки бекенд-частини використано власну розробку – мікрофреймворк IQCoreApp, а також бібліотеки OAuth2 від PHPLeague, SAML PHP Toolkit від OneLogin та ShibAuthBundle від SimpleSAMLphp. Для розробки фронтенд-частини використано фронтенд-фреймворк Bootstrap 4. Розроблені програмні компоненти відповідають вимогам технічного завдання.

# 4. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ КІБЕРЗАХИЩЕНОСТІ ТА ПРАЦЕЗДАТНОСТІ СИСТЕМИ ЦЕНТРАЛІЗОВАНОЇ АВТОРИЗАЦІЇ

## 4.1 Опис сценаріїв тестування кіберзахищеності та працездатності системи централізованої авторизації

Тестування кіберзахисту та працездатності веб-додатків – це процес, спрямований на виявлення та виправлення потенційних слабких місць у системі з метою підвищення її безпеки та стійкості. Цей процес може включати різні види тестів, зокрема: Тестування на захист від взлому та проникнення (Penetration Testing), тестування на безпеку застосунків (Application Security Testing), а також тестування на навантаження та стійкість (Load and Stress Testing).

## 4.2 Тестування кіберзахищеності системи централізованої авторизації

Експерти по тестуванню на взлом та проникнення намагаються активно проникнути в систему, використовуючи ті ж методи, що й потенційні зловмисники. Це включає в себе спроби зламати систему, використовуючи різні підходи, такі як SQL-ін'єкції, перехоплення сесій, переповнення буфера та інші. Такі типи тестування включають в себе перевірку застосунків на наявність можливих вразливостей. Вони можуть включати статичний аналіз коду, динамічні аналізи застосунків та сканування вразливостей.

Тестування кіберзахисту та працездатності веб-додатків - це комплексний процес, що включає в себе різні види тестів для забезпечення повноцінного покриття та виявлення можливих проблем. Також важливо слідкувати за безпековими оновленнями всіх використаних бібліотек та вживати рекомендованими їх розробниками відповідні заходи безпеки для захисту системи від нових загроз.

Захист від API-ін’єкцій, XSS та CSRF-атак реалізовано на рівні модуля мережевого екрану веб додатків (WAF) для фреймворків Jet-сайтів другого

покоління. Однак завжди потрібно перевіряти чи коркетно він працює у різних веб-додатках, їх модулях та окремих формах. Для тестування цих вразливостей використано такі інструменти як Selenium, Metasploit та OWASP ZAP. Результати їх роботи наведено у таблиці 4.1. За результатами їх роботи внесено зміни до програмного коду (підпис "усунуто"). Проблеми у модулях рівня API інших розробників передано їх авторам. Відповідні пункти компонент, в яких не виявлено проблем стоїть прочерк "–".

Таблиця 4.1 – Результати тестування кіберзахищеності окремих компонент

| Назва компоненти | API-ін’єкції | SQL-ін’єкції | XSS-атаки | CSRF-атаки |
|---|---|---|---|---|
| Рівень API | - | усунуто | - | - |
| API-проксі | усунуто | - | - | - |
| Інтерфейси OAuth2, SAML | - | - | - | - |
| Інтерфейси LDAP та RADIUS | усунуто | - | - | - |
| Модуль переадресації | усунуто | - | - | - |
| Панель самообслуговування | - | - | - | - |
| Мікрофремворк IQCoreApp | - | - | - | - |
| Сервіс-провайдер для служб | - | - | - | - |
| Сервіс-провайдер для користувачів | - | - | - | - |
| Модулі інтеграції з JetIQ та їх панель управління | усунуто | - | - | - |
| Модулі інтеграції зі сторонніми сервісами | - | - | - | - |

### 4.3 Тестування працездатності системи централізованої авторизації

Для визначення надійності додатку необхідно виконати тест на навантаження та стійкість, а також тест на відмову компонентів. Спершу виконано тестування на навантаження та стійкість. Мета цього виду тестування – визначити, наскільки добре веб-додаток витримує велику кількість користувачів або великі обсяги даних. Також перевіряється стійкість системи до DDoS-атак та збоїв.

Проведено тестування найбільш вразливої до атак публічного APIv1 завдяки виснаженням API-запитами (рис. 4.1). Тестування виконано за допомогою loader.io при ресурсах 2 vCPU та 1 GiB оперативної пам’яті (EC2-інстанс типу t3.micro у хмарі AWS). 40% запитів містили нові вхідні дані, відповіді на які не потрапляли у кеш.

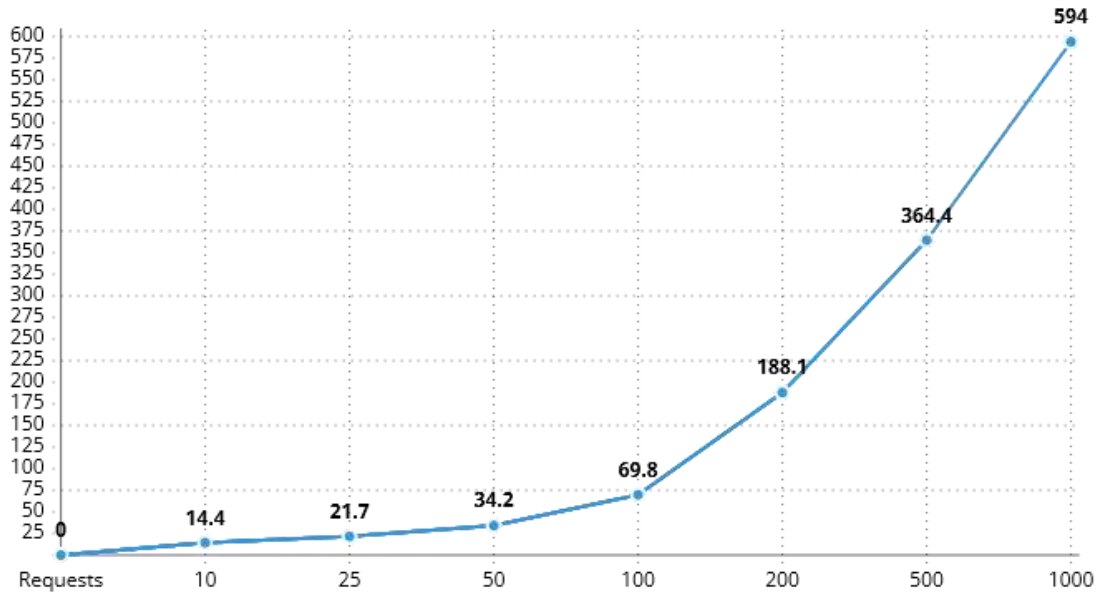


Рисунок 4.1 – Час відповіді під час випробування запитами (мсек.)

Тестування на відмову компонентів виконувалось шляхом почергового відключення таких компонентів як база даних та IdP API-ядро. Система показала вдале опрацювання і журналювання подібних помилок.

### 4.4 Тестування роботи компоненту гейміфікації тесту Тюринга

Для компоненту гейміфікації тесту Тюринга передбачено тестування завдяки виконанню на пристроях з різною потужністю та швидкістю. Випробувано на бюджетному ПК з 4-х ядерним CPU Intel i3 та 8 GiB оперативної пам’яті з операційною системою Ubuntu 22.04 LTS (версія для ПК) на провідному інтернеті швидкістю 50 мбіт/с, на бюджетному ноутбуці Acer Aspire E1-510P з 4-х ядерним CPU Celeron N2920 та 8 GiB оперативної пам’яті з операційною системою Ubuntu 22.04 LTS (версія для ПК) на WiFi зі швидкістю 20 мбіт/с, а також на бюджетному смартфоні Xiaomi Redmi 2A з операційною системою Android 5.0 (Lollipop) з MIUI 6 на стільниковому 3G-інтернеті зі швидкістю 2-5 мбіт/с. Кадр із тестування гейміфікації показано на рисунку 4.2.


Рисунок 4.2 – Кадр із тестування гейміфікації на ПК

Для тестування було відкрито форму аутентифікації системи SSO та введено більше 10 неправильних спроб входу. Як наслідок система

запропонувала пройти верифікацію, і видала запрошення пройти гейміфікований тест Тюринга. Кожен із трьох учасників тесту пройшов тест. В результаті тестування гейміфікація добре себе показала на ПК та ноутбуці. Іншим чином справи склались на смартфоні. Гра на сервері гейміфікації працює схожим чином як ігри у стрімінгових сервісах: їм потрібен інтернет-зв'язок із низькими затримками. Оскільки на смартфоні був повільний інтернет-зв'язок та низькі обчислювальні параметри, частота оновлення кадрів гри коливалась між 2 та 10 FPS. Гейміфікація все ще потребує доопрацювання, оскільки не показала прийнятного по комфорту рівня користування на мобільному пристрої, незважаючи на наявність управління персонажем із сенсорних екранів.

### 4.5 Рекомендації щодо удосконалення системи централізованої авторизації для Вінницького національного технічного університету

Для повного впровадження розробленої в даній роботі SSO-системи MY у Вінницькому національному технічному університеті необхідно у першу чергу необхідно налагодити процес синхронізації даних між системами JetIQ, Електронного деканату, ERP-системою відділу кадрів та SSO-системою MY. Крім взаємних API-викликів рекомендується запровадити між цими системами обмін інформацією на основі механізму черги повідомлень (Message Queue). Тоді процес синхронізації користувачів займатиме набагато менше часу.

Для службового доступу рекомендується впровадити модель делегованого доступу, запропоновану у даній роботі. Застосування делегованого доступу здатне повністю вирішити проблему підбору та втрати паролів до службових акаунтів і як результат їх компрометацію. Нова модель дозволятиме доступ до служб лише за посередництвом входу звичайних співробітників і використання ними мультифакторної авторизації.

Також рекомендується запровадити бізнес-процес консолідованого доступу для студентів (рис. 4.3). Цей процес дозволить студентам не змінювати логін та електронну пошту при вступі з бакалаврату на магістратуру.

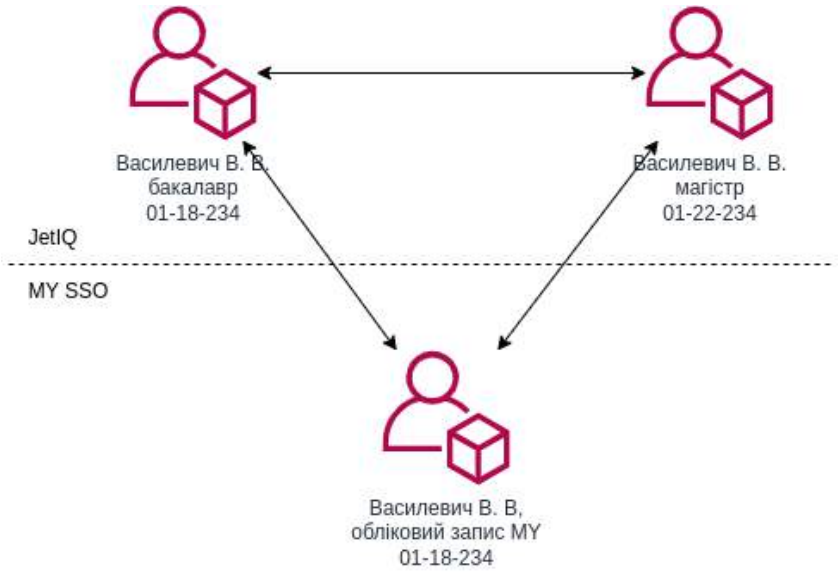


Рисунок 4.3 – Концепція консолідованого доступу для студентів

Для цього потрібно на рівні JetIQ, електронного деканату та системи MY налагодити зв’язок між старими та новими аккаунтами студентів. Крім того при завершенні бакалаврату необхідно тимчасово їх переміщати у тимчасовий архів, де вони будуть зберігатись та продовжуватимуть мати доступ до системи до вступу на магістратуру.

Впровадити гейміфікацію тесту Тюринга у освітнє середовище на даний момент поки що складно, оскільки зараз далеко не всі учасники освітнього процесу мають достатньо потужні пристрої та надійний і швидкий зв’язок для його успішного проходження.

Виконати всі рекомендації буде можливо після впровадження останньої розробленої версії SSO-системи MY та впровадження нових архітектурних підходів у решті систем, які з нею пов’язані.

Разом з експертами та розробниками Центру дистанційної освіти Вінницького національного технічного університету, було проведено тестування розробленої інформаційної системи централізованої авторизації для освітнього веб-середовища навчального закладу.

### 4.5 Висновки

У четвертому розділі виконані експериментальні дослідження у вигляді випробувань компонентів інформаційної системи централізованої авторизації для освітнього веб-середовища навчального закладу. Виконано тестування кіберзахищеності розробленої системи, в ході якого виявлено та виправлено незначні помилки, що в результаті можуть бути причиною більш складних проблем у користувачів системи. Також виконано тест на навантаження та стійкість, а також тест на відмову компонентів.

Випробування показали що система за виключенням деяких опціональних елементів повністю готова до подальшого впровадження у Вінницькому національному технічному університеті. Для розроблених модулів подано заявку на реєстрацію авторського права.

# 5. ЕКОНОМІЧНА ЧАСТИНА

## 5.1 Проведення технологічного аудиту розробки

При роботі над даною роботою розроблено програмне рішення для централізованої авторизації для освітнього веб-середовища навчального закладу. Для визначення оцінки комерційного потенціалу розробки проводять технологічний аудит. Це необхідно зробити для того щоб зрозуміти наскільки розроблене у цій роботі рішення буде приносити економічну вигоду.

Для проведення технологічного аудиту залучено трьох незалежних експертів: доцента з кафедри Комп'ютерних наук ВНТУ, професора з кафедри Обчислювальної техніки ВНТУ та доцента кафедри Комп'ютеризованих електромеханічних систем і комплексів ВНТУ. Оцінка науково-технічного рівня і комерційного потенціалу розробки буде здійснюватись за критеріями, наведеними у таблиці 5.1 [40].

Таблиця 5.1 – Критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки

| Критерії оцінювання та бали (за 5-ти бальною шкалою) | | | | | |
|---|---|---|---|---|---|
| Кри-тер. | 0 | 1 | 2 | 3 | 4 |
| Технічна здійсненність концепції: | | | | | |
| 1 | Достовірність концепції не підтверджена | Концепція підтверджена експертними висновками | Концепція підтверджена розрахунками | Концепція перевірена на практиці | Перевірено роботоздатність продукту в реальних умовах |
| Ринкові переваги (недоліки): | | | | | |
| 2 | Багато аналогів на малому ринку | Мало аналогів на малому ринку | Кілька аналогів на великому ринку | Один аналог на великому ринку | Продукт не має аналогів на великому ринку |
| 3 | Ціна продукту значно вища за ціни аналогів | Ціна продукту дещо вища за ціни аналогів | Ціна продукту приблизно дорівнює цінам аналогів | Ціна продукту дещо нижче за ціни аналогів | Ціна продукту значно нижче за ціни аналогів |

Продовження таблиці 5.1

| Критерії оцінювання та бали (за 5-ти бальною шкалою) | | | | | |
|---|---|---|---|---|---|
| Кри-тер. | 0 | 1 | 2 | 3 | 4 |
| 4 | Технічні та споживчі властивості продукту значно гірші, ніж в аналогів | Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів | Технічні та споживчі властивості продукту на рівні аналогів | Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів | Технічні та споживчі властивості продукту значно кращі, ніж в аналогів |
| 5 | Експлуатаційні витрати значно вищі, ніж в аналогів | Експлуатаційні витрати дещо вищі, ніж в аналогів | Експлуатаційні витрати на рівні експлуатаційних витрат аналогів | Експлуатаційні витрати трохи нижчі, ніж в аналогів | Експлуатаційні витрати значно нижчі, ніж в аналогів |
| Ринкові перспективи | | | | | |
| 6 | Ринок малий і не має позитивної динаміки | Ринок малий, але має позитивну динаміку | Середній ринок з позитивною динамікою | Великий стабільний ринок | Великий ринок з позитивною динамікою |
| 7 | Активна конкуренція великих компаній на ринку | Активна конкуренція | Помірна конкуренція | Незначна конкуренція | Конкурентів немає |
| Практична здійсненність | | | | | |
| 8 | Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї | Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців | Необхідне незначне навчання фахівців та збільшення їх штату | Необхідне незначне навчання фахівців | Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї |
| 9 | Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні | Потрібні незначні фінансові ресурси. Джерела фінансування відсутні | Потрібні значні фінансові ресурси. Джерела фінансування є | Потрібні незначні фінансові ресурси. Джерела фінансування є | Не потребує додаткового фінансування |
| 10 | Необхідна розробка нових матеріалів | Потрібні матеріали, що використовуються у військово-промисловому комплексі | Потрібні дорогі матеріали | Потрібні досяжні та дешеві матеріали | Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві |

Продовження таблиці 5.1

| Кри-тер. | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 11 | Термін реалізації ідеї більший за 10 років | Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років | Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років | Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років | Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років |
| 12 | Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту | Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу | Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу | Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту | Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту |

Результати оцінки науково-технічного рівня і комерційного потенціалу розробки експертами зведено в таблиці 5.2.

Таблиця 5.2 – Результати оцінювання потенціалу розробки експертами

| Критерії | Експерти | | |
|---|---|---|---|
| | Експерт 1 | Експерт 2 | Експерт 3 |
| | Бали, виставлені експертами: | | |
| 1. Технічна здійсненність концепції | 4 | 4 | 4 |
| Ринкові переваги (недоліки): | | | |
| 2. Наявність аналогів | 3 | 2 | 2 |
| 3. Ціна продукту | 4 | 4 | 4 |
| 4. Технічні властивості | 2 | 3 | 4 |
| 5. Експлуатаційні витрати | 4 | 4 | 4 |
| Ринкові перспективи | | | |
| 6. Розмір ринку | 2 | 3 | 3 |
| 7. Конкуренція | 3 | 2 | 3 |
| Практична здійсненність | | | |
| 8. Наявність фахівців | 4 | 3 | 3 |
| 9. Наявність фінансів | 4 | 4 | 4 |

Продовження таблиці 5.2

| | | | |
|---|---|---|---|
| 10. Необхідність нових матеріалів | 4 | 4 | 4 |
| 11. Термін реалізації | 4 | 3 | 4 |
| 12. Розробка документів | 4 | 4 | 4 |
| Сума балів | $СБ_1$=42 | $СБ_2$=40 | $СБ_3$=43 |
| Середньоарифметична сума балів $\underline{СБ}$ | 41,6 | | |

Згідно з результатами таблиці 5.2 можна встановити відповідний рівень комерційного потенціалу розробки. Зважаючи на отриманий результат, його можна порівняти з відповідними рівнями комерційного потенціалу розробки, що представлено в таблиці 5.3 [40].

Таблиця 5.3 – Рівні науково-технічного та комерційного потенціалу розробки

| Середньоарифметична сума балів $\underline{СБ}$, розрахована на основі висновків експертів | Рівень комерційного потенціалу розробки |
|---|---|
| 0 – 10 | Низький |
| 11 – 20 | Нижче середнього |
| 21 – 30 | Середній |
| 31 – 40 | Вище середнього |
| 41 – 48 | Високий |

Рівень комерційного потенціалу розробки, становить 41,6 балів, що відповідає рівню «високий».

В якості аналога для розробки було обрано хмарний сервіс Okta. Даний сервіс є одним із кращих на ринку з точки зору комерційних підприємств. В застосуванні у комерційній сфері він практично не має жодних помітних недостатків, що не властиві одночасно всім SSO-системам. Однак попри успіх його застосування у багатьох комерційних сферах, у інтеграції з освітніми веб-середовищами для навчальних закладів він не надто швидко розвивається. На

даний момент з ним можливо повінстю інтегрувати лише дві системи управління навчанням (LMS): Moodle та Canvas LMS. Для інтеграції інших систем необхідно мати високий рівень навичок і знань протоколів авторизації, або замовляти додаткові консультаційні послуги. Також до недоліків можна віднести високу вартість покупки їх хмарних послуг та надміру складність закупки їх послуг (та всіх їх основних конкурентів у цій сфері) через обмеження у виборі конкретних технологій у системі публічних закупівель Прозоро [41], через яку проводять закупівлі всі ЗВО державної форми власності. Вимоги до державних закупівель забороняють дискримінаційні вимоги до учасників торгів, серед яких обирається постачальник. На державних закупівлях замовнику може бути потрібен конкретний програмний продукт з певним набором характеристик. Однак досить часто трапляються ситуації, коли продавці не повністю сумісного, але дешевшого програмного забезпечення чи хмарних сервісів без реального наміру щось продати подають скарги в органи оскарження щоб завадити конкурентам які продають повністю сумісні товари або послуги виграти тендер [41]. Через всі подібні складнощі транснаціональні хмарні провайдери часто не бачать сенсу витрачати час та гроші на юристів, щоб продавати свої продукти та послуги бюджетному сектору, який є малодохідним для цих компаній.

У розробці проблема сумісності вирішується завдяки створення інструментів міграції та синхронізації даних з більшого числа систем управління навчанням (LMS) із застосуванням удосконалених бізнес-процесів для закладів освіти, розроблених у другому розділі. Модель продаж має полягати у наданні безкоштовного базового набору програмного забезпечення з платними модулями, в яких реалізовані ноу-хау методи та моделі, реалізовані у даній роботі.

Вирішити проблему з доступністю для державних ЗВО можливо завдяки заключенню угод на перепродаж (у вигляді хмарних послуг чи платних модулів) та сертифікації тех. підтримки компаній, які приймають участь у торгах на майданчиках системи державних закупівель Прозоро. Основні технічні

показники при застосуванні у сфері освіти аналога і нового програмного продукту наведені у таблиці 5.4.

Таблиця 5.4 - Основні технічні показники у сфері освіти аналога і нового програмного продукту

| Показники | Аналог | Нова розробка | Відношення параметрів нової розробки до параметрів аналога |
|---|---|---|---|
| Надійність | 99.9% | 99.9% | 1 |
| Простота використання | 90% | 90% | 1 |
| Сумісність з LMS | 77% | 88% | 1,14 |
| Покриття документацією | 75% | 70% | 0.94 |
| Швидкість інтеграції у освітній процес | 70% | 80% | 1,14 |
| Можливість впровадження наукової новизни даної роботи | 66% | 83% | 1,2 |

Нова розробка може використовуватися як державними так і приватними вищими, передвищими та професійно-технічними закладами освіти. Кінцевими користувачами розробленої системи у цих закладах виступають студенти, викладачі та рядові співробітники. Операторами системи у цих закладах виступають деканати, відділ кадрів та технічна підтримка ІТ-підрозділів. Технічна готовність складає 90%. Розроблено архітектуру програмного продукту, нові методи та моделі, їх програмну реалізацію та працюючий прототип системи, який у більшій мірі покриває необхідний функціонал. На даний існує декілька зацікавлених сторін з впровадження розробки. При успішному впровадженні також можлива і подальша комерціалізація.

### 5.2 Розрахунок витрат на здійснення науково-технічної розробки

Для визначення загальних витрат на здійснення науково-технічної розробки спочатку необхідно порахувати всі статті втрат. Основну заробітну $З_о$ розробника можна визначити за формулою (5.1):

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{Т_р}, \quad (5.1)$$

де k – кількість посад дослідників;

$M_{ni}$ – місячний посадовий оклад конкретного розробника, грн;

$t_i$ – кількість днів роботи розробника;

$Т_р$ – число робочих днів в місяці – 24.

Інженер-розробник, має місячний оклад 22 000 грн і кількість робочих днів у місяці – 24. Число днів роботи інженера – 72. Також пораховано витрати на заробітну плату менеджера проєктів, який планує та контролює виконану роботу раз на місяць.

$$З_о = \frac{21100 \cdot 72}{24} + \frac{22000 \cdot 72}{24} = 129\,300{,}48 \text{ (грн)}.$$

Результати розрахунків наведено у таблиці 5.5. Оскільки звичайні робітники відсутні, тому $З_Р = 0$.

Таблиця 5.5 – Основна заробітна плата розробників

| Найменування посади | Місячний посадовий оклад, грн. | Оплата за робочий день, грн. | Число днів роботи | Витрати на заробітну плату, грн. |
|---|---|---|---|---|
| Менеджер проєктів | 21 100 | 879,17 | 72 | 63 300,24 |
| Інженер-розробник | 22 000 | 916,67 | 72 | 66 000,24 |
| Всього, грн | | | | 129 300,48 |

Так як навчальний процес та наукова діяльність магістрів ВНТУ відбувається у дистанційному режимі, на виконавців покладається всі витрати на забезпечення свого робочого місця. У зв'язку з цим їм полагається додаткова заробітна плата $З_д$. Її можна розрахувати як як 15% від основної заробітної плати за формулою (5.2):

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}. \quad (5.2)$$

$$З_{дод} = (129\,300{,}48 + 0) \cdot 0{,}15 = 19\,395{,}072 \text{ (грн)}.$$

Згідно діючого законодавства нарахування на заробітну плату (Єдиний соціальний внесок) складають 22% від суми основної та додаткової заробітної плати, як подано формулою (5.3):

$$З_н = (З_{о\ р} + З_д) \cdot 0{,}22. \quad (5.3)$$

$$З_н = (129300{,}48 + 19395{,}072) \cdot 0{,}22 = 32\,713{,}02 \text{ (грн)}.$$

Для організації офлайн-зустрічей із стейкхолдерами передбачено закупівлю маркерів для малювання концептуальних схем на фліпчарті. Позиції таблиці 5.6 обирались за рейтингом торгівельного майданчику Prom.ua.

Таблиця 5.6 – Витрати на матеріали

| Найменування матеріалу | Ціна за одиницю, грн. | Витрачено, шт. | Вартість витраченого матеріалу, грн |
|---|---|---|---|
| Маркер Centropen для офісних дошок (чорний) | 28 | 1 | 28 |
| Маркер Centropen для офісних дошок (синій) | 32 | 1 | 32 |
| Маркер Centropen для офісних дошок (червоний) | 32 | 1 | 32 |
| Всього, грн | | | 92,00 |

Згідно інформації про витрати на матеріали, що використані при розробці програмного продукту у таблиці 5.6 можна розрахувати витрати на матеріали ($M$). Оскільки вартість відходів ($Ц_{вj}$) використаних матеріалів не перевищує 10 грн, тому при розрахунках нею можна знехтувати:

$$M = (28{,}00 \cdot 1) + (32{,}00 \cdot 1) + (32{,}00 \cdot 1) = 92{,}00 \text{ (грн)}.$$

Оскільки процес розробки відбувається у дистанційному режимі і всі інструменти для дистанційної розробки мають працювати у хмарі, витрати на програмне забезпечення та електроенергію покладаються на хмарного провайдера. В якості хмарного провайдера обрано сервіс Amazon Web Services, оскільки на ньому найбільш легко розгорнути середовище для розробки випробування SSO-системи, що розробляється. Найбільш зручно розраховувати вартість всіх необхідних ресурсів у онлайн-калькуляторі AWS (рис. 5.1), визначених при розробці архітектури системи у другому розділі даної роботи. Загальну вартість цих ресурсів пораховано у таблиці 5.7 по курсу американського долару (USD), що станом на 1.11.2023 становить 36.317 українських гривні (UAH).

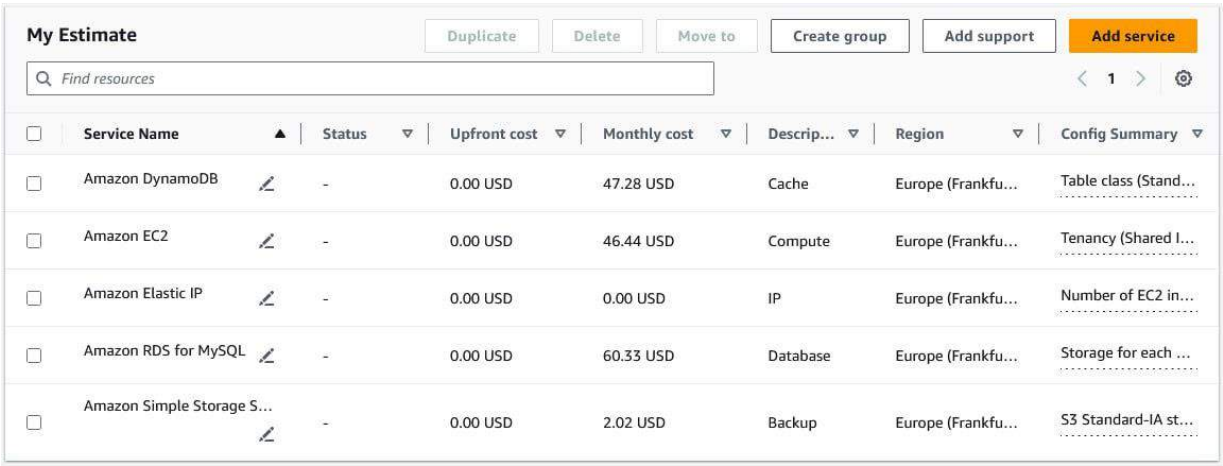


Рисунок 5.1 – Результати оцінки вибраних послуг та їх параметрів у онлайн-калькуляторі Amazon Web Services (у доларах USD)

Таблиця 5.7 – Витрати на хмарні послуги по кожному виду

| Найменування хмарної послуги | Технічні параметри | Кількість, шт | Ціна за одиницю | Вартість, грн |
|---|---|---|---|---|
| Amazon EC2 | t3a.small, 10 GB EBS | 3 | 562,2 | 1686,55 |
| Amazon RDS for MySQL | db.m3.medium, 10 GB Storage | 1 | 2191,00 | 2191,00 |
| Amazon DynamoDB | on-demand, 5 GB Storage | 1 | 1717,05 | 1717,05 |
| Amazon Elastic IP | - | 1 | 0 | 0 |
| Amazon S3 (для бекапів) | s3 standard-ia, 120 GB Storage | 1 | 73,36 | 73,36 |
| Всього, грн | | | | 5667,96 |

Крім періоду для розробки програмного забезпечення, який триватиме три місяці, знадобиться ще один місяць для демонстрації продукту стейкхолдерам, разом $C_{прг.і}$ буде становити 4 місяці. Згідно даним калькулятора (рис. 4.1), у даній конфігурації інсталяційні платежі не передбачені, тому $K_i$ буде дорівнювати 1. Отже величина витрат на програмне забезпечення у хмарі буде розраховуватись за формулою (5.4):

$$B_{прг} = \sum_{i=1}^{k} Ц_{прг} \cdot C_{прг.і} \cdot K_i. \quad (5.4)$$

$$B_{прг} = 5667{,}96 \cdot 4 \cdot 1 = 22\,671{,}84 \text{ (грн)}.$$

Так як хмара повністю покриває всі обчислювальні потреби, необхідності у покупці іншого комп’ютерного обладнання крім апаратних носіїв даних немає. Носії даних потрібні для резервного копіювання, а також передачі результатів стейкхолдерам. Для обчислення амортизаційних відрахувань можна використати формулу (5.5):

$$А_{обл} = \frac{Ц_б}{Т_В} \cdot \frac{t_{вик}}{12}, \quad (5.5)$$

де $Ц_б$ – загальна балансова вартість всього обладнання, комп’ютерів, приміщень тощо, грн;

t – фактична тривалість використання, міс;

$Т_В$ – термін використання обладнання, приміщень тощо, роки.

Для розрахунку амортизації відомо що балансова вартість зовнішнього жорсткого диску становить 2560 грн, а термін його використання – 4 років, а фактична тривалість використання 3 місяці. Величина амортизаційних відрахувань наведена в таблиці 5.8.

$$А_{обл} = \frac{2560}{4} \cdot \frac{3}{12} = 160 \text{ (грн)}$$

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

| # | Найменування | Балансова вартість, грн | Термін використання, р | Фактична тривалість використання, міс. | Величина амортизаційних відрахувань, грн |
|---|---|---|---|---|---|
| 1 | Зовнішній жорсткий диск 1TB Transcend | 2560 | 4 | 3 | 160 |
| | Всього, грн | | | | 160,00 |

Інші витрати ($В_{ін}$) охоплюють: витрати на Інтернет, управління організацією, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, тощо. Величина інших витрат має розраховуватись за формулою (5.6):

$$В_{ін} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}. \quad (5.6)$$

Знаючи зарплату співробітників, задіяних у розробці ($З_о$) та про відсутність звичайних працівників, можна розрахувати інші витрати ($В_{ін}$):

$$В_{ін} = (129300{,}48 + 0) \cdot 0{,}5 = 64\,650{,}24 \text{ (грн)}.$$

Отже, витрати на створення повноцінної версії інформаційної системи централізованої авторизації для освітнього веб-середовища навчального закладу, що є сумою всіх попередніх витрат, становлять:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + B_{прг} + A_{обл} + B_{ін} \quad (5.7)$$

У результаті додавання всіх витрат отримано:

$$B_{заг} = 129300{,}48 + 0 + 19395{,}072 + 32713{,}02 + 92 + 22671{,}84 + 160 + 64650{,}24 = 268\ 982{,}65 \text{ (грн)}$$

Для урахування нинішнього стану справ у даній роботі необхідно виконати прогнозування загальних витрат *(ЗВ)* на виконання та впровадження результатів виконаної науково-технічної роботи за формулою (5.8):

$$ЗВ = \frac{\text{В}_{\text{заг}}}{\eta}, \quad (5.8)$$

де $\eta$ – коефіцієнт, який характеризує етап (стадію) виконання даної роботи. Якщо розробка знаходиться на стадії впровадження, то $\eta \approx 0{,}9$.

$$\text{ЗВ} = 268\ 982{,}652 / 0{,}9 = 298\ 869{,}61 \text{ (грн)}$$

Отже, загальні витрати на виконання наукової роботи та впровадження її результатів становитимуть 298 869,61 грн.

## 5.3 Розрахунок економічної ефективності науково-технічної розробки від її впровадження у ЗВО

Хоча подальша комерціалізація системи що розробляється у даній роботі є потенційно можливою, але спершу планується її впровадити на рівні ЗВО, що надав технічне завдання для даної роботи. Для того щоб визначити наскільки буде ефективним впровадження розробленої у даній роботі інформаційної системи централізованої авторизації для освітнього веб-середовища навчального закладу необхідно провести розрахунок економічної ефективності даної розробки. Виконання даної науково-технічної роботи та впровадження її результатів складає приблизно до 1 року. Позитивні результати від впровадження розробки очікуються вже в перші місяці після впровадження.

Визначити зростання прибутку підприємства внаслідок зниження витрат на оплату праці працівників, які виконують окремі виробничі чи інформаційно-технічні управлінські функції можна за допомогою формули (5.8):

$$\Delta П_{я} = \frac{ЧП \cdot ЗП \cdot 12}{N} - \frac{0.5 \cdot ЗВ}{\Delta N_{i}} \quad (5.8)$$

де ЧП – чисельність працівників, які виконують певні функції вручну, осіб;

ЗП – середня заробітна плата працівника, який виконує відповідну функцію вручну, грн;

ЗВ – приблизні витрати на розробку автоматизованої системи управління, грн;

N – кількість функцій, які виконуються вручну у році до впровадження результатів нової науково-технічної розробки, шт;

$\Delta N_i$ – прогнозоване зростання кількості виробничих чи інформаційно-технічних управлінських функцій, виконання яких автоматизується, в аналізованому році (відносно року до впровадження цієї розробки), шт.

Проведено покращення основного якісного показника ($\Delta П_я$) в університеті, при якому співробітник відділу кадрів та по співробітнику у кожному деканаті, що вручну налаштовують доступ викладачам, вручну створюють пошту та тимчасовий пароль, вручну їх збивають, створюють прив'язку акаунтів на сайтах університету, прив'язують електронну пошту до кабінету студента. Разом вийде 8 співробітників, що отримують мінімальну ЗП, на основі чого вийде:

$$\Delta П_я = \frac{8 \cdot 6700 \cdot 12}{6} - \frac{0.6 \cdot 298869{,}61}{6} = 77\,313{,}04 \text{ (грн)}$$

Прибуток, який отримує підприємство від автоматизації виконання окремої виробничої чи інформаційно-технічної управлінської функції у кожному із років після впровадження науково-технічної розробки можна приблизно оцінити, виходячи з формули (5.9):

$$П_я = \frac{\Delta ЧП \cdot ЗП \cdot 12}{N} \quad (5.9)$$

де ΔЧП – економія чисельності працівників, виконання виробничої чи управлінської функції яких було автоматизовано в аналізованому році, осіб;

N – кількість функцій, які виконуються вручну у році до впровадження результатів нової науково-технічної розробки, шт. Отже,

$$П_я = \frac{6 \cdot 6700 \cdot 12}{6} = 80\,400 \text{ (грн)}$$

Для наведених випадків можливе збільшення чистого прибутку підприємства $\Delta П_і$ для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, що розраховується за формулою (5.10):

$$\Delta П_і = (\Delta П_я \cdot N + П_я \cdot \Delta N)_і, \quad (5.10)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження на підприємстві результатів науково-технічної розробки в аналізованому році;

N – основний кількісний показник, який визначає обсяг діяльності підприємства у році до впровадження результатів нової науково-технічної розробки; на даний момент це частка успішного тех. обслуговування (9 з 10);

$П_я$ – основний якісний показник, який визначає результати діяльності підприємства у кожному із років після впровадження науково-технічної розробки;

ΔN – зміна основного кількісного показника діяльності підприємства в результаті впровадження науково-технічної розробки в аналізованому році.

В результаті впровадження наукової розробки покращується якість обслуговування студентів та викладачів завдяки наявності системи самообслуговування. За кожен рік автоматизації ІТ-відділ зможе збільшувати свою продуктивність на 38%. Необхідно розрахувати показник прибутку впродовж трьох років відносно базового. Збільшення чистого прибутку $\Delta П_1$ протягом першого року складе:

$$\Delta П_1 = (99696{,}95 \cdot 9 + 77313{,}04 \cdot 0.38) = 926\,651{,}51 \text{ (грн)}.$$

Збільшення чистого прибутку $\Delta П_2$ протягом другого року:

$$\Delta П_2 = (99696{,}95 \cdot 9 + 77313{,}04 \cdot 0.76) = 956\,030{,}46 \text{ (грн)}.$$

Збільшення чистого прибутку $\Delta П_3$ протягом третього року становитиме:

$$\Delta П_3 = (99696{,}95 \cdot 9 + 77313{,}04 \cdot 1.14) = 985\,409{,}42 \text{ (грн)}.$$

Отже, розрахунки показують, що комерційний ефект від впровадження розробки виражається у суттєвому збільшенні чистого прибутку.

Приведену вартість збільшення всіх чистих прибутків (ПП), що їх може отримати університет від можливого впровадження науково-технічної розробки можна розрахувати за формулою (5.11).

$$ПП = \sum_1^T \frac{\Delta П_i}{(1+\tau)^t} \text{ [грн]}, \qquad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої роботи, грн;

т – період часу, протягом якого результати впровадженої роботи стають помітними, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки „0”.

$$ПП = \frac{926651,51}{(1+0,1)^1} + \frac{956030,46}{(1+0,1)^2} + \frac{985409,42}{(1+0,1)^3} = 2\ 773\ 400,27\ (грн)$$

Наступним чином необхідно розрахувати величину початкових інвестицій PV, які потрібно вкласти для впровадження науково-технічної розробки у закладі освіти. Для цього можна використати формулу:

$$PV = k_{інв} \cdot ЗВ$$

де $k_{інв}$ – коефіцієнт, що враховує витрати університету на впровадження науково-технічної розробки та її комерціалізацію, він має значення $k_{інв} = 1,1$.

$$PV = 1.1 \cdot 298\ 869,61 = 328\ 756,57\ (грн)$$

Розрахуємо абсолютну ефективність вкладених інвестицій $Е_{абс}$ за формулою (5.12):

$$Е_{абс} = (ПП - PV), \qquad (5.12)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає університет від реалізації результатів наукової розробки, грн.;

PV – поточна вартість інвестицій PV від ЗВ, грн.

$$E_{абс} = 2\ 773\ 400,27 - 328\ 756,57 = 2\ 444\ 643,70\ (грн).$$

Так як $E_{абс} > 0$ та є навіть більшим за PV, то це означає що у впровадженні науково-технічної розробки скоріше за все може бути доцільність. Однак одного цього факту може виявитись недостатньо для прийняття остаточного рішення.

Тому потрібно розрахувати щорічну відносну ефективність вкладених у наукову розробку інвестицій $E_в$. Для цього необхідно використати формулу (5.13):

$$\mathrm{E_в} = \sqrt[\text{Тж}]{1 + \frac{\mathrm{E_{абс}}}{PV}} - 1 \qquad (5.13)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

$PV$ теперішня вартість інвестицій PV = ЗВ, грн;

$T_{ж}$ – життєвий цикл наукової розробки, роки.

$$\mathrm{E_в} = \sqrt[3]{1 + \frac{312534{,}251}{82533{,}539}} - 1 = 1.03, \text{ або } 103\%.$$

Термін окупності вкладених у реалізацію науково-технічної розробки інвестицій $T_{ок}$ можна розрахувати за формулою (5.14):

$$\mathrm{T_{ок}} = \frac{1}{\mathrm{E_в}}, \qquad (5.14)$$

$$\mathrm{T_{ок}} = \frac{1}{1{,}03} = 0{,}97 \text{ (роки)}$$

Термін окупності становить 0,97 року (тобто $T_{ок} < 3…5$ років), що свідчить про те що впровадження даної науково-технічної розробки є економічно виправданим.

### 5.4 Висновки

В даному розділі виконано оцінку комерційного потенціалу науково-технічної розробки. Залучено трьох незалежних експертів та проведено ними

комерційний аудит. Визначено, що рівень комерційного потенціалу розробки є достатньо високим. В першу чергу розробка має бути впроваджена на рівні ЗВО [42], у результаті чого можна буде з’ясувати чи є сенс у подальшій комерціалізації.

В ході проведення аналізу комерційного потенціалу науково-технічної розробки встановлено, що основні конкуренти попри їх переваги у комерційному секторі мають менше переваг у автоматизації сфері освіти ніж створена науково-технічна розробка. Запропоновано модель подальшої комерціалізації, що дозволяє у майбутньому вийти за межі обмежень державних закупівель, які заважають конкурентам вийти на ринок державних ЗВО.

Згідно із розрахунками статей всіх витрат на завершення роботи над науково-технічною розробкою загальні витрати на розробку складають 298869,61 грн. Розрахована абсолютна ефективність вкладених інвестицій в сумі 2444643,70 грн свідчить про доцільність впровадження науково-технічної розробки. Крім того, щорічна ефективність вкладених в наукову розробку інвестицій складає 103%, що вище мінімальної бар'єрної ставки 19%.

Отже, розрахунок ефективності вкладених інвестицій та періоду їх окупності показав, що фінансування розробки є доцільним, адже інвестиції буде повернуто в термін до одного року.

# ВИСНОВКИ

Результати досліджень, аналіз існуючих розробок методів, моделей, а також хмарних та серверних SSO-рішень для централізованої ідентифікації, аутентифікації та авторизації довели, що актуальність питання вдосконалення методів та засобів, особливо у сфері освіти, не зменшуються із часом, а потребують подальшого розвитку..

У магістерській кваліфікаційній роботі отримані результати відповідно до мети роботи, яка полягає у підвищенні інтегрованості освітнього веб-середовища навчального закладу завдяки синхронізації користувачів різних сайтів у єдину базу та уніфікації доступу між ними; удосконалення методів та програмних модулів для захисту користувачів від компрометації доступу.

У магістерській кваліфікаційній роботі виконано:

- аналіз існуючих методів і засобів централізованої автентифікації, авторизації та ідентифікації;
- аналіз існуючих інформаційних систем централізованого входу в систему, а також ефективність їх використання у сфері освіти;
- запропоновано покращення для програмних API-компонентів існуючої SSO-системи для JetIQ, яку можна удосконалити та використати у новій розробці;
- розроблено нові методи для вдосконалення функціонування централізованої автентифікації, авторизації та ідентифікації у сфері освіти;
- розроблено модель гейміфікації для публічного тесту Тюринга;
- розроблено програмну архітектуру та компоненти системи централізованої авторизації для освітнього веб-середовища навчального закладу;
- проведено експериментальні дослідження кіберзахищеності та працездатності розробленої системи.

За результатами досліджень удосконалено метод проектування REST API-сервісів систем технології централізованої автентифікації, авторизації та ідентифікації, який тепер здатен зменшити ризик експлуатації найбільш

небезпечних типів атак, які можуть виникнути у будь-який момент життєвого циклу SSO-системи.

Подальшого розвитку отримав метод визначення довіри до клієнтського пристрою, що дозволяє визначити довіру по чотирьом параметрам: ідентифікатор сесії користувача, IP-адресі, номеру автономної системи, з якої походить IP-адреса, а також країна їх походження.

Запропоновано модель гейміфікації публічного тесту Тюринга, яка у свою чергу використовує математичну модель ReLU. Модель складається з п’яти рівнів: навчання, генерації, валідації, компонування та геймплею. Практична користь моделі полягає у можливості генерації рівнів гейміфікації з випадковим просторовим плануванням, але в якому зберігається можливість виконання завдань квесту тестування Тюринга.

Практична цінність одержаних результатів полягає в розроблених програмних компонентах, які пройшли практичне випробування та можуть бути впроваджені у закладах вищої освіти, що використовують систему керування навчальним процесом JetIQ.

Впровадження результатів досліджень підтверджуються актом впровадження Центру дистанційної освіти Вінницького національного технічного університету – використання розробленої SSO-системи в освітньому середовищі системи JetIQ.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Abhishek, K., Roshan, S., Kumar, P., & Ranjan, R. (2013). A comprehensive study on multifactor authentication schemes. *In Advances in Computing and Information Technology: Proceedings of the Second International Conference on Advances in Computing and Information Technology (ACITY)* July 13-15, 2012, Chennai, India-Volume 2 (pp. 561-568). Springer Berlin Heidelberg.

2. Pashalidis, A., & Mitchell, C. J. (2003). A taxonomy of single sign-on systems. *In Information Security and Privacy: 8th Australasian Conference, ACISP* 2003 Wollongong, Australia, July 9–11, 2003 Proceedings 8 (pp. 249-264). Springer Berlin Heidelberg.

3. Fett, D., Küsters, R., & Schmitz, G. (2015, October). Spresso: A secure, privacy-respecting single sign-on system for the web. *In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (pp. 1358-1369).

4. Малініч П. П., Коваленко О. О., Малініч І. П. Впровадження технологій централізованої ідентифікації, автентифікації та авторизації користувачів у освітніх інформаційних системах. *Інформаційні технології і автоматизація – 2023* : матеріали XVI міжнар. наук.-практ. конф. 19 – 20 жовтня 2023 р. Одеса : ОНТУ, 2023. С. 177–179.

5. Малініч П. П., Коваленко О. О., Малініч І. П. Актуальні проблеми кіберзахищеності систем центрального входу у багатосайтових освітніх інформаційних системах. *Сучасні тенденції розвитку техніки та технологій – 2023* : матер. міжнар. наук.-техн. конф. 31 жовтня 2023. Харків : Міжнародний центр технологічних інновацій, 2023. С. 10-13.

6. Малініч П. П., Коваленко О. О., Малініч І. П. Можливості гейміфікації публічного тесту Тюринга. *Розвиток науки та техніки України під час воєнного стану* : матер. міжнар. наук.-практ. інтернет-конф. 3 листопада 2023. Івано-Франківськ : Наука та практика, 2023. С. 92-95.

7. Hardt, Dick. The OAuth 2.0 authorization framework. No. RFC6749. 2012.

8. Campbell, B., Mortimore, C., & Jones, M. (2015). Security assertion markup language (SAML) 2.0 profile for OAuth 2.0 client authentication and authorization grants (No. RFC7522).

9. Howes, T., & Smith, M. (1995). RFC1823: The LDAP Application Program Interface.

10. Neuman, C., Yu, T., Hartman, S. and Raeburn, K., 2005. RFC 4120: The Kerberos network authentication service (V5).

11. Rigney, Carl, Steve Willens, Allan Rubens, and William Simpson. Remote authentication dial in user service (RADIUS). No. rfc2865. 2000.

12. Малініч П. П., Малініч І. П., Коваленко О. О. Негативні безпекові чинники у локальних Ethernet-мережах та абонентських мереж останньої милі. *LI Науково-технічна конференція підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2022) : матеріали Науково-технічної конференції.* Вінниця, 31 травня 2022 р. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15614.

13. Microsoft SMB Protocol Authentication. URL : https://learn.microsoft.com/en-us/windows/win32/fileio/microsoft-smb-protocol-authentication (дата звернення: 21.09.2023).

14. OpenID Connect Core 1.0 incorporating errata set 2 : https://openid.net/specs/openid-connect-core-1_0.html (дата звернення: 22.09.2023).

15. Marchel, M., & Bołdak, C. (2015). Assessment of LDAP services in high availability environment. *Advances in Computer Science Research*, (12), 21-35.

16. Vazquez, A., & Vazquez, A. (2019). FreeIPA AD Integration. Practical LPIC-3 300: *Prepare for the Highest Level Professional Linux Certification*, 663-701.

17. Kadlec, J., Jaros, D., & Kuchta, R. (2010, September). Implementation of an Advanced Authentication Method within Microsoft Active Directory Network Services. In *2010 6th International Conference on Wireless and Mobile Communications* (pp. 453-456). IEEE.

18. Butcher, Matt. Mastering OpenLDAP: Configuring, Securing, and Integrating Directory Services. Packt Publishing Ltd, 2007.

19. SimpleSAMLphp Installation and Configuration. URL: https://simplesamlphp.org/docs/stable/simplesamlphp-install.html (дата звернення: 24.09.2023).

20. Heyman, T., Preuveneers, D., & Joosen, W. (2014, August). Scalability analysis of the OpenAM access control system with the universal scalability law. In *2014 international conference on future internet of things and cloud* (pp. 505-512). IEEE.

21. Needleman, M. (2004). The Shibboleth authentication/authorization system. Serials Review, 30(3), 252-253.

22. A look into how Notre Dame is protecting and enabling campus users with Okta – Okta, 20.09.2020. URL: https://www.okta.com/resources/webinar-a-look-into-how-notre-dame-is-protecting-and-enabling-campus-users-with-okta/ (дата звернення: 28.09.2023).

23. Безкоштовний Microsoft Office 365 для шкіл та учнів - Освіта Microsoft. URL: https://www.microsoft.com/uk-ua/education/products/office (дата звернення: 28.09.2023).

24. Дистанційна освіта в університеті «КРОК» під час карантину – Освіта.UA, 31.07.2020. URL: https://osvita.ua/consultations/75374/ (дата звернення: 28.09.2023).

25. Best practices for REST API design. – Stack Overflow Blog, 02.03.2020. URL: https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/ (дата звернення: 29.09.2023).

26. Hoefling, S. (2022). Microsoft Graph, Web APIs, and MyFilesPage. In Getting Started with the Uno Platform and WinUI 3: Hands-On Building of Cross-Platform Desktop, Mobile, and Web Applications That Can Run Anywhere (pp. 337-400). Berkeley, CA: Apress.

27. Martin-Lopez, A. (2020, June). AI-driven web API testing. In *Proceedings of the ACM/IEEE 42nd international conference on software engineering: companion proceedings* (pp. 202-205).

28. Alan Duric. Three principles for security-first architecture. – Red Hat, 01.12.2020. URL: https://www.redhat.com/architect/three-principles-security-first-architecture (дата звернення: 29.09.2023).

29. Cloudflare WAF. A WAF for modern application security – Cloudflare, WAF product brief Fall – 2022. URL: https://www.cloudflare.com/static/9fd7253e4009caaa928a67ecca5d2709/WAF_product_brief_Fall_2022__1_.pdf (дата звернення: 29.09.2023).

30. Ritvik Gupta & Ankit Sahu. Software Architecture Patterns: What Are the Types and Which Is the Best One for Your Project. – Turing for employers. August 31, 2023. URL: https://www.turing.com/blog/software-architecture-patterns-types/ (дата звернення: 10.10.2023).

31. Lin, E., Aycock, J., & Mannan, M. (2012). Lightweight client-side methods for detecting email forgery. In *Information Security Applications: 13th International Workshop*, WISA 2012, Jeju Island, Korea, August 16-18, 2012, Revised Selected Papers 13 (pp. 254-269). Springer Berlin Heidelberg.

32. Hu, H., & Wang, G. (2018). Revisiting email spoofing attacks. arXiv preprint arXiv:1801.00853.

33. The general data protection regulation – European Council, 25.05.2018. URL: https://www.consilium.europa.eu/en/policies/data-protection/data-protection-regulation/ (дата звернення: 11.10.2023).

34. Lacey, D., Salmon, P., & Glancy, P. (2015). Taking the bait: a systems analysis of phishing attacks. Procedia Manufacturing, 3, 1109-1116.

35. OWASP Top Ten – OWASP Foundation, 2021. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 11.10.2023).

36. Madden, Neil. API security in action. Simon and Schuster, 2020.

37. Katt, B., & Prasher, N. (2018, September). Quantitative security assurance metrics: REST API case studies. In *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings* (pp. 1-7).

38. Блавт О.З. Онтологія метрологічного забезпечення контролю здібності до орієнтування в просторі студентів спеціальних медичних груп ВНЗ / О.З. Блавт // Науковий часопис Національного педагогічного університету імені М.П. Драгоманова. Серія № 15. “Науково-педагогічні проблеми фізичної культури /фізична культура і спорт/”36. Наукових праць / За ред. Г. М. Арзютова. - К.:Вид-во НПУ імені М.П. Драгоманова, 2014. - Випуск 3К(44)14 – С. 103-108.

39. Малініч П. П., Малініч І. П., Томчук. М. А. Реалізація двофакторної автентифікації для VPN-технології Cisco AnyConnect з використанням протоколу RADIUS / *Молодь в науці: дослідження, проблеми, перспективи (МН-2023)* : матер. всеукр. наук.-практ. інтернет-конф., 22 червня 2023 р. Вінниця : ВНТУ, 2023. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/view/18903.

40. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с

41. Тендерна документація - що це таке та як правильно скласти – E-tender.ua, 08.02.2023. URL: https://e-tender.ua/news/tenderna-dokumentaciya-yaki-obovyazkovi-ta-zaboroneni-vimogi-1219 (дата звернення: 27.11.2023).

42. Положення про кваліфікаційну роботу на другому (вищому) рівні вищої освіти. Розробники: А.О. Семенов, Л.П. Громова, Т.В. Макарова, О.В. Сердюк. Вінниця : ВНТУ, 2021. 60 с. URL: https://vntu.edu.ua/uploads/2021/mkr.pdf (дата звернення: 29.11.2023).

# ДОДАТКИ

**Додаток А – Технічне завдання**

**(обов'язковий)**

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

УЗГОДЖЕНО

Директор ЦДО ВНТУ

к.т.н., проф. каф. АІІТ

Паламарчук Є.А.

«19» вересня 2023 року

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

д.т.н., професор

Романюк О.Н.

«19» вересня 2023 р.

**Технічне завдання**

**на магістерську кваліфікаційну роботу «**Інформаційна система централізованої авторизації для освітнього веб-середовища навчального закладу**»**

**за спеціальністю**

**121 – Інженерія програмного забезпечення**

Керівник магістерської кваліфікаційної роботи:

к.т.н., доц. Коваленко О. О.

"19" 09 2023 р.

Виконав:

студент гр.1ПІ-22м Малініч П. П.

"19" 09 2023 р.

Вінниця – 2023 року

## 1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Інформаційна система централізованої авторизації для освітнього веб-середовища навчального закладу».

Галузь застосування –інформаційні системи управління освітнім процесом у закладах вищої освіти.

## 2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 247 від 18 вересня 2023 року ректора по ВНТУ про закріплення тем МКР.

## 3. Мета та призначення розробки.

Метою роботи є підвищення інтегрованості освітнього веб-середовища навчального закладу завдяки синхронізації користувачів різних сайтів у єдину базу та уніфікація доступу між ними; удосконалення методів та програмних модулів для захисту користувачів від компрометації доступу.

Призначення роботи – створення нової версії системи JetIQ MY із розробкою та застосуванням та розробкою нових методів та моделей для підвищення рівня захищеності системи.

## 4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР:

1.Abhishek, K., Roshan, S., Kumar, P., & Ranjan, R. (2013). A comprehensive study on multifactor authentication schemes. In Advances in Computing and Information Technology: Proceedings of the Second International Conference on Advances in Computing and Information Technology (ACITY) July 13-15, 2012, Chennai, India-Volume 2 (pp. 561-568). Springer Berlin Heidelberg.

2.Pashalidis, A., & Mitchell, C. J. (2003). A taxonomy of single sign-on systems. In Information Security and Privacy: 8th Australasian Conference, ACISP 2003 Wollongong, Australia, July 9–11, 2003 Proceedings 8 (pp. 249-264). Springer Berlin Heidelberg.

3. Fett, D., Küsters, R., & Schmitz, G. (2015, October). Spresso: A secure, privacy-respecting single sign-on system for the web. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (pp. 1358-1369).

4. Hardt, Dick. The OAuth 2.0 authorization framework. No. RFC6749. 2012.

5. Малініч П. П., Малініч І. П., Коваленко О. О. Негативні безпекові чинники у локальних Ethernet-мережах та абонентських мереж останньої милі. LI Науково-технічна конференція підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2022) : матеріали Науково-технічної конференції. м. Вінниця, 31 травня 2022 р. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15614.

**5. Технічні вимоги**

В ході виконання роботи проаналізувати недоліки та переваги попередніх пілотних версій SSO-системи JetIQ MY (0.1 та 0.2), проаналізувати можливості вдосконалення цієї системи чи створення нового коду сумісного з її API. Також необхідно запропонувати архітектурне та програмне рішення для випуску її нових версій. Необхідно запропонувати методику міграції даних із JetIQ.

Для функціонування бекенд-частини SSO-системи, що розробляється, необхідно передбачити можливість використання фреймворку Jet-сайтів другого покоління. Для функціонування фронтенд-частини необхідно забезпечити responsive-дизайн, який має бути доступним як у браузерах мобільних пристроїв, так і на десктопних версіях браузерів. Запропонувати модель гейміфікації для захисту від ботів, а також її програмну реалізацію.

Для розробки нових методів, моделей, архітектурного рішення та програмної реалізації додатково надаються Вимоги до програмної архітектури

SSO-системи для JetIQ версії 1.0 (2020 р.), розроблені спільно Центром дистанційної освіти ВНТУ та Центром електронних комунікацій "InterCEC" ВНТУ, без їх публікації у вільному доступі. Дозволяється цитування їх загального змісту та вимог:

- нової назви, яку матиме система: "JetIQ MY";
- передбаченого застосування мікросервісів;
- окремих вимог до сервісу IdP та сервіс-провайдеру A&A;
- описаного спосібу обміну інформацією з JetIQ та протокол авторизації у його сервісах SID;
- опису процесів аутентифікації та авторизації користувачів різних типів: звичайних користувачів (студентів і співробітників) та службових ролей;
- опису типів логінів: циферні (ідентифікатор співробітника), ПІБ (співробітники), номер залікової книжки (студенти), кастомний юзернейм (т.з. "нік" студента), а також вхід за email-адресами як основний і універсальний спосіб входу для всіх користувачів;
- опису автодоповнення логінів (для тих користувачів, у кого це увімкнено у налаштуваннях приватності) – для студентів та викладачів;
- вимоги реалізації сервіс-провайдеру A&A серед веб-додатків, в якому має бути реалізована панель самообслуговування службовими акаунтами, а також адмін-панель;
- вимоги: у панелі самообслуговування користувач повинен мати можливість змінювати свій пароль, способи відновлення паролю, прив’язку до способів мультифакторної автентифікації, параметри конфіденційності, а також централізоване управління акаунтами у всіх корпоративних сервісах та сайтах;
- вимоги: для всіх веб-додатків системи JetIQ MY має бути використаний фреймворк Jet-сайтів другого покоління (PHP);
- вимоги: для всіх веб-додатків системи JetIQ MY має бути забезпечено підтримку підходу blue-green deployment;

- вимоги: для звичайних користувачів (студентів і співробітників) та службових ролей мають існувати два окремі IdP-сервіси з різними вимогами до їх API.

**6. Конструктивні вимоги**

Архітектура програмного рішення повинна відповідати вимогам до програмної архітектури SSO-системи JetIQ MY версії 1.0. Рекомендується використовувати вже наявні у проєкті мови програмування: PHP, Python та JavaScript. При вдосконаленні існуючих API-модулів або при їх повному переписанні заново забезпечити їх сумісність із програмними інтерфейсами REST API системи управління навчальним процесом JetIQ. При проектуванні та розробці програмних компонент бекенд-частини рекомендується застосовувати принципи ООП. Графічна та текстова документація повинна відповідати діючим стандартам України.

**7. Перелік технічної документації, що пред'являється по закінченню робіт:**

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

**8. Вимоги до рівня уніфікації та стандартизації**

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

**9. Стадії та етапи розробки:**

| № з/п | Назва етапів магістерської кваліфікаційної роботи | Строк виконання етапів роботи |
|---|---|---|
| 1 | Аналіз сучасного стану використання технологій централізованого входу в систему у сфері освіти | 20.09.2023 – 09.10.2023 |
| 2 | Проектування архітектури, методів, засобів для централізованого входу в систему | 10.10.2023 – 20.10.2023 |
| 3 | Розробка компонентів системи централізованої авторизації для освітнього веб-середовища навчального закладу | 21.10.2023 – 14.11.2023 |
| 4 | Експериментальні дослідження кіберзахищеності та працездатності системи централізованої авторизації | 15.11.2023 – 24.11.2023 |
| 5 | Економічна частина | 25.11.2023 – 05.12.2023 |

**9. Порядок контролю та прийняття.**

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

## Додаток Б – Протокол перевірки на плагіат

**(Обов'язковий)**

## ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Інформаційна система централізованої авторизації для освітнього веб-середовища навчального закладу**

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра програмного забезпечення, ФІТКІ, 1ПІ – 22м

Науковий керівник: Коваленко О.О.

| **Unicheck** | |
|---|---|
| **Оригінальність** | **97,5%** |
| Схожість | 2,5% |

**Аналіз звіту подібності**

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Малініч П.П.

Керівник роботи _____ Коваленко О.О.

**Додаток В**

**Акт впровадження**

УЗГОДЖЕНО
Директор ІІДО ВНТУ
к.т.н., проф. каф. АІІТ
Паламарчук Є.А.
«05» 12 2023 року

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
Романюк О.Н.
«05» 12 2023 року

**АКТ ВПРОВАДЖЕННЯ № 1/ 5.12.2023**
**результатів науково-дослідних робіт**

Замовник Центр дистанційної освіти Вінницького національного технічного університету
*(найменування організації)*

Цим актом підтверджується, що результати роботи – «Інформаційна система централізованої авторизації для освітнього веб-середовища навчального закладу»
*(найменування теми)*

що виконав студент гр. 1ПІ – 22м, Малініч П.П., на громадських засадах
*(виконавець)*

відповідно до планів робіт Центру дистанційної освіти ВНТУ на 2020-2021 р., 2021-2022 р., а також на 2022-2023 р. та впроваджено у Вінницькому національному технічному університеті
*(строки виконання) (найменування організації, де здійснювалося впровадження)*

1. Вид впроваджених результатів: система "JetIQ MY"
*(експлуатація виробу, роботи, технології)*

2. Характеристика масштабу впровадження: одиничне
*(унікальне, одиничне, партія, масове, серійне)*

3. Форма впровадження: удосконаленні API-модулі, програмні бекенд- та фронтенд-компоненти системи «JetIQ MY»

4. Новизна результатів науково-дослідної роботи: модифікації методів використання інформаційних систем центрального входу у систему
*(піонерські, принципово нові, якісно нові, модифікації, модернізація старих розробок)*

5. Впроваджені: в освітньому середовищі системи JetIQ VNTU

6. Соціальний та науково-технічний ефект: уніфікація авторизованого доступу користувачів до різних сайтів ЗВО, як наслідок підвищення комфорту їх використання, а також підвищення захищеності їх доступу завдяки застосуванню запропонованих у роботі методів.
*(охорона навколишнього середовища, поліпшення й оздоровлення умов праці, удосконалення структури керування, науково-технічних напрямків, спеціальне призначення)*

7. Ліцензійні застереження: Всі результати магістерської кваліфікаційної роботи можуть бути впроваджені у навчальний процес, наукові дослідження, науково-методичне

забезпечення та ІТ-інфраструктуру Вінницького національного технічного університету на безоплатній основі без обмежень прав на використання, копіювання, редагування, доповнення, публікацію, поширення, субліцензування та/або продаж копій.

Програмний код розробки надається університету на безоплатній основі без обмежень прав на використання, копіювання, редагування, доповнення, публікацію, поширення, субліцензування та/або продаж копій за умовами ліцензії MIT:

"ЦЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ НАДАЄТЬСЯ «ЯК Є», БЕЗ ГАРАНТІЙ БУДЬ-ЯКОГО ВИДУ, ПРЯМИХ АБО НЕПРЯМИХ, ВКЛЮЧАЮЧИ, АЛЕ НЕ ОБМЕЖУЮЧИСЬ, ГАРАНТІЯМИ КОМЕРЦІЙНОЇ ВИГОДИ, ВІДПОВІДНОСТІ ЙОГО КОНКРЕТНОМУ ПРИЗНАЧЕННЮ Й ВІДСУТНОСТІ ПОРУШЕННЯ ПРАВ. У ЖОДНОМУ РАЗІ АВТОРИ АБО ВЛАСНИКИ АВТОРСЬКИХ ПРАВ НЕ ВІДПОВІДАЮТЬ ЗА БУДЬ-ЯКИМИ СУДОВИМИ ПОЗОВАМИ, ЩОДО ЗБИТКІВ АБО ІНШИХ ПРЕТЕНЗІЙ, ЧИ ДІЙ ДОГОВОРУ, ЦИВІЛЬНОГО ПРАВОПОРУШЕННЯ АБО ІНШИХ, ЩО ВИНИКАЮТЬ ПОЗА, АБО У ЗВ'ЯЗКУ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ АБО ВИКОРИСТАННЯМ ЧИ ІНШИМИ ДІЯМИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ"

Повний текст ліцензії знаходиться за адресою: https://opensource.org/license/mit/

Від виконавця:
студент групи 1ПІ-22м

_____ Малініч П.П.

Керівник: к.т.н., доцент кафедри ПЗ

Від Центру дистанційної
освіти ВНТУ, директор,
к.т.н., професор каф. АІІТ:

_____ Паламарчук Є.А.

_____ Коваленко О.О.