

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»**

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»

УДК 004.043

«До захисту допущено»

Завідувач кафедри

\_\_\_\_\_ Едуард ЖАРІКОВ

«\_\_» \_\_\_\_\_ 2024 р.

## **Магістерська дисертація**

**на здобуття ступеня магістра**

**за освітньо-професійною програмою «Інженерія програмного  
забезпечення інформаційних систем»**

**зі спеціальності 121 «Інженерія програмного забезпечення»**

**на тему: «Система уніфікації ідентифікації користувачів в АС ЗВО»**

Виконав:

студент II курсу, групи ПІ-32мп

Яцевський Олександр Ігорович

Керівник:

доцент, к.т.н. Фіногенов Олексій Дмитрович

Рецензент:

доцент, к.т.н. Катерина Мамедова.

Засвідчую, що у цій магістерській дисертації  
немає запозичень з праць інших авторів без  
відповідних посилань.

Студент \_\_\_\_\_

Київ – 2024 року

**Національний технічний університет України**  
**«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення інформаційних систем»

**ЗАТВЕРДЖУЮ**

Завідувач кафедри

\_\_\_\_\_ Едуард ЖАРІКОВ

« \_\_\_\_ » \_\_\_\_\_ 2024р.

**ЗАВДАННЯ**

**на практику студенту**

Яцевського Олександр Ігоровича

1. Тема дисертації «Система уніфікації ідентифікації користувачів в АС ЗВО», науковий керівник дисертації Фіногенов Олексій Дмитрович, затверджені наказом по університету від «8» жовтня 2024 р. № 5016-с.
2. Строк подання студентом дисертації «16» грудня 2024 р.
3. Об'єкт дослідження – процеси ідентифікації та аутентифікації користувачів в розподілених інформаційних системах закладів вищої освіти.
4. Предмет дослідження – методи, засоби та технології уніфікованої ідентифікації та аутентифікації користувачів для інтеграції з інформаційними системами університету.
5. Перелік завдань, які потрібно розробити: дослідити та реалізувати уніфіковану систему ідентифікації та автентифікації користувачів для інформаційного середовища університету, що включає аналіз існуючих рішень, розробку архітектури з підтримкою сучасних стандартів безпеки, проектування механізмів двофакторної аутентифікації, створення системи управління токенами та сесіями, реалізацію інтеграційних інтерфейсів з АС "Ядро" та іншими системами університету, а також впровадження механізмів моніторингу та масштабування.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу – 4 плакати.

7. Орієнтовний перелік публікацій – три публікації.

8. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання «1» вересня 2024 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
1	Аналіз предметної області, дослідження існуючих рішень SSO та формування вимог до системи	1.09.24-7.09.24	
2	Проектування архітектури системи та розробка механізмів інтеграції з АС "Ядро"	8.09.24-21.09.24	
3	Розробка базових компонентів системи автентифікації та авторизації	22.09.24-28.09.24	
4	Реалізація системи двофакторної автентифікації та управління токенами	29.09.24-05.10.24	
5	Створення користувацького інтерфейсу та API для інтеграції з іншими системами	06.10.24-19.10.24	
6	Впровадження системи безпеки, журналювання подій та моніторингу	20.10.24-09.11.24	
7	Виконання експериментальних досліджень	10.11.24-13.11.24	
8	Оформлення пояснювальної записки	14.11.24-18.11.24	
9	Подання дисертації на попередній захист	22.11.2024	
10	Подання дисертації на захист	16.12.2024	

Студент

Олександр ЯЦЕВСЬКИЙ

Науковий керівник дисертації

Олексій ФІНОГЕНОВ

## РЕФЕРАТ

Розмір пояснювальної записки – 167 аркушів, містить 16 ілюстрацій, 23 таблиці, 6 додатків, 22 посилання на джерела.

**Актуальність теми.** У роботі розглянуто проблему уніфікації ідентифікації користувачів в інформаційному середовищі закладу вищої освіти. Показано основні особливості існуючих рішень, їх переваги та недоліки в контексті специфічних потреб університету. Виявлено потребу в розробці системи єдиної точки входу, що забезпечить централізоване управління автентифікацією користувачів та інтеграцію з існуючими інформаційними системами університету.

**Мета дослідження.** Основною метою є розробка та впровадження системи уніфікованої ідентифікації користувачів для забезпечення єдиної точки входу до інформаційних ресурсів університету.

**Об'єкт дослідження:** процеси ідентифікації та автентифікації користувачів в розподілених інформаційних системах закладів вищої освіти.

**Предмет дослідження:** методи, засоби та технології уніфікованої ідентифікації та автентифікації користувачів для інтеграції з інформаційними системами університету.

Для реалізації поставленої мети **сформульовані наступні завдання:**

- аналіз існуючих рішень SSO та формування вимог до системи;
- проектування архітектури системи та розробка механізмів інтеграції з АС "Ядро";
- розробка системи двофакторної автентифікації та управління токенами;
- створення користувацького інтерфейсу та API для інтеграції з іншими системами.

**Наукова новизна** результатів магістерської дисертації полягає в розробці архітектури системи централізованої автентифікації для розподілених інформаційних систем закладу вищої освіти, яка, на відміну від існуючих рішень, забезпечує інтеграцію з наявними системами без порушення їх автономності та специфічних механізмів авторизації,

можливість поетапного впровадження єдиної точки входу з підтримкою паралельного функціонування різних механізмів автентифікації, а також гнучке управління ролями користувачів на основі даних з центрального каталогу довідників університету. Такий підхід дозволяє зберегти працездатність існуючої інфраструктури під час міграції на нову систему автентифікації та забезпечити необхідний рівень безпеки при роботі з розподіленими інформаційними ресурсами закладу вищої освіти.

**Практичне значення** отриманих результатів полягає у створенні системи, що забезпечує єдину точку входу для всіх інформаційних ресурсів університету, підвищує безпеку облікових даних користувачів та спрощує процес автентифікації. Система може бути адаптована для використання в інших закладах вищої освіти.

**Зв'язок з науковими програмами, планами, темами.** Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського" в рамках програми цифровізації університету.

**Апробація.** Основні положення роботи доповідалися та обговорювалися на науково-технічній конференції "Інформаційні технології в освіті" (Київ, 2024). Розроблене ПЗ впроваджено в КБ ІС КПІ ім. Ігоря Сікорського (акт впровадження).

**Публікації.** Наукові положення дисертації опубліковані в:

1) Фіногенов О.Д. Проблеми ідентифікації персони в автоматизованих системах закладів вищої освіти / Ковтунець О.В., Бабич Б.Б., Губський А.М., Ромашкевич Я.О., Фіногенов О.Д., М'яч Д.О., Яцевський О.І. // Вступна кампанія до закладів вищої освіти України: проблеми та перспективи : 7-а Всеукраїнська науково-практична конференція, 21 червня 2024, Київ : матеріали. — 2024. — С. 72-74.

2) Фіногенов О.Д. Пікові навантаження на інформаційні системи забезпечення процесу прийому до університету та на інформаційні системи

автоматизації університетських процесів в КПІ ім. Ігоря Сікорського / Ковтунець О.В., Бабич Б.Б., Губський А.М., Ромашкевич Я.О., Фіногенов О.Д., М'яч Д.О., Яцевський О.І. // Вступна кампанія до закладів вищої освіти України: проблеми та перспективи : 7-а Всеукраїнська науково-практична конференція, 21 червня 2024, Київ : матеріали. — 2024. — С. 74-76.

Фіногенов О.Д. Оцінювання роботи в приймальній комісії за результатами підсистеми рейтингування науково-педагогічних працівників / Ковтунець О.В., Бабич Б.Б., Губський А.М., Ромашкевич Я.О., Фіногенов О.Д., М'яч Д.О., Яцевський О.І. // Вступна кампанія до закладів вищої освіти України: проблеми та перспективи : 7-а Всеукраїнська науково-практична конференція, 21 червня 2024, Київ : матеріали. — 2024. — С. 77-79.

**Ключові слова:** ЄДИНА ТОЧКА ВХОДУ, АВТЕНТИФІКАЦІЯ, АВТОРИЗАЦІЯ, OAUTH 2.0, OPENID CONNECT, ДВОФАКТОРНА АВТЕНТИФІКАЦІЯ, ІНФОРМАЦІЙНІ СИСТЕМИ УНІВЕРСИТЕТУ.

## ABSTRACT

The explanatory note size is 167 pages, contains 16 illustrations, 23 tables, 6 appendices, 22 references.

**Relevance of the topic.** The paper addresses the problem of unifying user identification in the information environment of higher education institutions. The main features of existing solutions, their advantages and disadvantages in the context of specific university needs are shown. The need to develop a single sign-on system that will provide centralized user authentication management and integration with existing university information systems has been identified.

**Research objective.** The main goal is to develop and implement a unified user identification system to provide a single entry point to university information resources.

**Object of research:** processes of identification and authentication of users in distributed information systems of higher education institutions.

**Subject of research:** methods, tools and technologies for unified identification and authentication of users for integration with university information systems.

To achieve this goal, **the following tasks** were formulated:

- Analysis of existing SSO solutions and system requirements formation
- System architecture design and development of integration mechanisms with AS "Core"
- Development of two-factor authentication system and token management
- Creation of user interface and API for integration with other systems

The **scientific novelty** of the master's thesis results lies in the development of a centralized authentication system architecture for distributed information systems of higher education institutions, which, unlike existing solutions, provides integration with existing systems without compromising their autonomy and specific authorization mechanisms, the possibility of phased implementation of

single sign-on with support for parallel functioning of different authentication mechanisms, as well as flexible user role management based on data from the university's central directory catalog. This approach allows maintaining the operability of existing infrastructure during migration to the new authentication system and ensuring the required level of security when working with distributed information resources of higher education institutions.

The **practical significance** of the obtained results lies in creating a system that provides a single entry point for all university information resources, enhances user account security, and simplifies the authentication process. The system can be adapted for use in other higher education institutions.

**Connection with scientific programs, plans, themes.** The work was performed at the Department of Informatics and Software Engineering of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" as part of the university digitalization program.

**Approbation.** The main provisions of the work were reported and discussed at the scientific and technical conference "Information Technologies in Education" (Kyiv, 2024). The developed software has been implemented at the Information Systems Design Bureau of Igor Sikorsky Kyiv Polytechnic Institute (implementation act).

**Publications.** The scientific provisions of the dissertation were published in:

1) Finogenov O.D. Problems of person identification in automated systems of higher education institutions / Kovtunets O.V., Babych B.B., Hubskeyi A.M., Romashkevych Ya.O., Finogenov O.D., Miach D.O., Yatsevskyi O.I. // Admission campaign to higher education institutions of Ukraine: problems and prospects: 7th All-Ukrainian Scientific and Practical Conference, June 21, 2024, Kyiv: proceedings. — 2024. — P. 72-74.

2) Finogenov O.D. Peak loads on information systems supporting the university admission process and on information systems for automation of university processes at Igor Sikorsky KPI / Kovtunets O.V., Babych B.B., Hubskeyi A.M., Romashkevych Ya.O., Finogenov O.D., Miach D.O., Yatsevskyi O.I. //

Admission campaign to higher education institutions of Ukraine: problems and prospects: 7th All-Ukrainian Scientific and Practical Conference, June 21, 2024, Kyiv: proceedings. — 2024. — P. 74-76.

**Keywords:** SINGLE SIGN-ON, AUTHENTICATION, AUTHORIZATION, OAUTH 2.0, OPENID CONNECT, TWO-FACTOR AUTHENTICATION, UNIVERSITY INFORMATION SYSTEM

## ЗМІСТ

<b>ЗМІСТ .....</b>	<b>11</b>
<b>ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ СКОРОЧЕНЬ ТА ТЕРМІНІВ.....</b>	<b>13</b>
<b>ВСТУП .....</b>	<b>15</b>
<b>1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....</b>	<b>17</b>
1.1 Опис предметної області.....	17
1.2 Аналіз існуючих рішень.....	23
1.3 Аналіз вимог до програмного забезпечення .....	33
1.4 Постановка задачі .....	41
Висновки до розділу .....	44
<b>2 ПРОЄКТУВАННЯ РІШЕННЯ.....</b>	<b>46</b>
2.1 Опис процесу діяльності .....	46
2.2 Аналіз та проектування сутностей системи .....	50
2.3 Проектування компонентів системи .....	57
Висновки до розділу .....	68
<b>3 КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ .....</b>	<b>70</b>
3.1 Опис використаних технологій .....	71
3.2 Розробка бази даних .....	77
3.3 Розробка серверної частини.....	81
Висновки до розділу .....	84
<b>4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ</b>	
<b>86</b>	
4.1 Аналіз якості програмного забезпечення .....	86
4.2 Опис контрольного прикладу .....	88
Висновки до розділу .....	95
<b>5 МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП ПРОЄКТУ .....</b>	<b>97</b>
5.1 Технологічний аудит ідеї проєкту .....	98
Висновки до розділу .....	100
<b>СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....</b>	<b>102</b>
<b>ДОДАТОК А .....</b>	<b>104</b>
<b>ДОДАТОК Б .....</b>	<b>105</b>
<b>ДОДАТОК В .....</b>	<b>106</b>

	12
<b>ДОДАТОК Г</b> .....	<b>107</b>
<b>ДОДАТОК Е</b> .....	<b>117</b>
<b>ДОДАТОК Ж</b> .....	<b>165</b>

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ СКОРОЧЕНЬ ТА ТЕРМІНІВ

2FA	– двофакторна автентифікація (Two-Factor Authentication), додатковий рівень захисту, що вимагає підтвердження особи користувача через другий фактор
API	– прикладний програмний інтерфейс, набір визначень для взаємодії різних компонентів програмного забезпечення
JWT	– відкритий стандарт для створення токенів доступу, що базується на форматі JSON;
OAuth 2.0	– протокол авторизації, який дозволяє додаткам отримувати обмежений доступ до призначених для користувача даних на сервері
OpenID Connect	– протокол автентифікації, побудований поверх OAuth 2.0, що дозволяє клієнтам перевіряти особу користувача
SMTP	– протокол передачі електронної пошти
SSO	– технологія єдиного входу, що дозволяє користувачу пройти автентифікацію один раз та отримати доступ до кількох пов'язаних систем
БД	– база даних
OTP	– одноразовий пароль, що використовується для підтвердження особи користувача
TOTP	– одноразовий пароль, що генерується на основі поточного часу
АС	– автоматизована система
UI	– користувацький інтерфейс
URL	– уніфікований локатор ресурсів, адреса веб-ресурсу

Аутентифікація	– процес перевірки достовірності пред'явленого користувачем ідентифікатора, підтвердження того, що користувач дійсно є тим, за кого себе видає
Ідентифікація	– процес розпізнавання користувача системою через перевірку існування його облікових даних в базі зареєстрованих користувачів, присвоєння суб'єкту унікального ідентифікатора
Авторизація	– процес надання користувачу прав на виконання певних дій в системі, перевірка та контроль прав доступу користувача до захищених ресурсів системи на основі його ролей та політик безпеки

## ВСТУП

У зв'язку з інтенсивним розвитком інформаційних технологій та потребою в ефективному управлінні освітніми процесами, ЗВО зтикаються з викликами, пов'язаними з управлінням ідентифікацією користувачів. Однією з ключових проблем є відсутність уніфікованої системи, яка б інтегрувала авторизацію користувачів з існуючими освітніми та адміністративними сервісами. Це призводить до необхідності розробки нової системи ідентифікації користувачів, яка стане єдиною точкою входу для доступу до усіх університетських ресурсів.

Метою даної роботи є створення та впровадження системи "University ID" у КПІ ім. Ігоря Сікорського яка буде використовувати принципи OpenID для забезпечення безпечного, надійного та зручного доступу до усіх інформаційних ресурсів університету. Система має забезпечити інтеграцію з існуючими освітніми платформами та можливість масштабування для обслуговування значної кількості користувачів.

Основні завдання, що вирішуються в процесі розробки системи, включають:

- аналіз існуючих рішень у різних підрозділах КПІ ім. Ігоря Сікорського для виявлення поточних методів ідентифікації та управління доступом;
- вибір методу аутентифікації для користувачів університету, забезпечення балансу між безпекою та зручністю;
- розробка архітектури системи, що забезпечить гнучке управління доступом до ресурсів та безпечний обмін даними;
- приклад інтеграції системи з уже наявними інформаційними системами університету;
- тестування роботи системи в умовах масштабованого навантаження.

Наукова новизна роботи полягає у вдосконаленні методів ідентифікації та авторизації, а також у дослідженні шляху для інтеграції систем з власними

системами аутентифікації, що дозволяє оптимізувати доступ до інформаційних ресурсів.

Практичне значення роботи виявляється у підвищенні безпеки та зручності користувачів освітніх систем, забезпеченні стабільного доступу до необхідних ресурсів університету.

# 1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 1.1 Опис предметної області

В умовах стрімкого розвитку цифрових технологій і зростання обсягів доступних даних, питання ідентифікації та управління доступом користувачів в освітніх установах, зокрема в КПІ ім. Ігоря Сікорського набуває особливого значення. Ці системи спрощують доступ до навчальних та адміністративних ресурсів, відіграючи ключову роль у повсякденній академічній діяльності [1,2].

Однією з актуальних проблем у сфері ідентифікації користувачів є необхідність уніфікації доступу до різноманітних підсистем університету, які використовують різні джерела даних.

На прикладі КПІ ім. Ігоря Сікорського входять власні механізми авторизації існують у таких системах адміністрування та підтримки навчального процесу:

- АС “Е-кампус”;
- АС “my.kpi.ua”;
- АС “Деканат”;;
- Moodle;
- [Google Classroom](#);
- [Автоматизована система формування розкладу](#);
- Система “Працевлаштування”.

У кожна система має власні вимоги до авторизації та зберігає різні дані користувача. Це призводить до значних незручностей для користувачів, ускладнює підтримку переліку користувачів інформаційних систем у актуальному стані, знижує ефективність взаємодії з університетськими ресурсами тощо.

Проблема ускладнюється тим, що різні підсистеми підтримують різні типи ролей користувачів. Наприклад, студент може мати одні права доступу в навчальній системі та зовсім інші в системі бібліотеки чи наукових ресурсів. Це вимагає складної координації даних між системами та забезпечення консистенції інформації про користувача, а також додає необхідність перевизначити трактування профілю користувача окремо для кожного сервісу. Важливою особливістю є те, що практично всі системи є системами закритого типу, тобто не передбачають можливості реєстрації для зовнішнього користувача – створення, оновлення, видалення аккаунту відбувається виключно з боку адміністратора. Це означає, що кожна система, яка несе відповідальність за певну кількість навчальних чи організаційних процесів має необхідність окремого адміністрування й менеджменту доступу. Відповідно, команда розробки кожного сервісу має брати на себе відповідальність над такими аспектами:

- управління даними користувачів;
- розділ до доступу інформації відповідно до профілю користувача;
- забезпечення протоколів безпеки для комунікації з кінцевим користувачем;
- узгодження даних користувацької бази з іншими підсистемами;
- розподіл відповідальності за окремі процеси;
- адміністрування облікових записів.

Також важливим аспектом є відсутність уніфікованого стандарту багатофакторної аутентифікації у більшості існуючих систем. Деякі системи використовують протоколи доступу з використанням логіну та паролю, окремий ряд систем підтримує підтвердження за допомогою мобільного телефону або ж електронної пошти, АС "Відділ Кадрів" та АС "Деканат" мають підтримку апаратного захисту. Кожен тип багатофакторної аутентифікації сильно підвищує ступінь захищеності системи загалом, але різниця в використаних стандартах не дозволить ефективно управляти даними [3].

Таким чином доступ за допомогою лише логіна та пароля відкриває велику вразливість в системі, тому що такий тип доступу може бути дуже легко компрометований навіть без використання технічних засобів чи компрометації пристроїв користувача. Тому додавання захисту за допомогою електронної пошти чи мобільного телефону сильно знизить рівень вразливості системи, але підтримка таких рішень може бути ускладнена з ряду причин.

#### Технічні фактори:

- можливі збої в роботі мобільного зв'язку та проблеми з доставкою SMS;
- затримки в отриманні електронних листів через перевантаженість поштових серверів;
- складності синхронізації часу при використанні додатків-аутентифікаторів;
- проблеми сумісності з різними моделями телефонів та версіями програмного забезпечення.

#### Організаційні фактори

- необхідність підтримки актуальної бази контактних даних користувачів;
- додаткові витрати на SMS-розсилки та технічну підтримку;
- потреба в навчанні персоналу новим процедурам автентифікації;
- Складність масштабування системи при збільшенні кількості користувачів;
- необхідність розробки процедур відновлення доступу.

#### Фактори, що пов'язані з користувачем:

##### Можлива втрата доступу до телефону чи пошти;

- зміна номеру телефону без своєчасного оновлення в системі;
- складнощі для користувачів похилого віку при роботі з додатковими засобами автентифікації;
- небажання користувачів використовувати додаткові методи захисту через їх незручність;
- можливість помилкового введення кодів підтвердження.

### Фактори безпеки:

- можливість перехоплення SMS-повідомлень злоумисниками;
- вразливість електронної пошти до фішингових атак;
- ризик компрометації резервних каналів відновлення доступу;
- можливість соціальної інженерії для отримання доступу до додаткових факторів автентифікації;
- складність забезпечення належного рівня захисту баз даних з контактною інформацією.

-

Таким чином, при впровадженні двофакторної автентифікації необхідно враховувати наступні вимоги:

- технічна надійність та стабільність роботи каналів доставки кодів автентифікації;
- організаційні витрати на підтримку та адміністрування системи;
- зручність використання та простота освоєння для кінцевих користувачів;
- стійкість до різних типів атак на канали передачі даних;
- масштабованість та гнучкість системи при зростанні кількості користувачів;
- наявність ефективних процедур відновлення доступу.

Ці фактори повинні бути детально проаналізовані та враховані при проектуванні системи багатфакторної автентифікації для забезпечення балансу між безпекою та зручністю використання. Відсутність або недостатня підтримка такої функціональності може підвищити ризик несанкціонованого доступу до особистих та навчальних даних користувачів, особливо в контексті зростаючих кіберзагроз.

Аналіз відкритих літературних джерел показав, що хоча проблема ідентифікації та управління доступом є добре відомою та широко обговорюваною в академічних колах, існуючі рішення часто не забезпечують достатньої універсальності та гнучкості для комплексного вирішення викликів сучасних освітніх установ. Це зумовлює потребу в розробці нових, більш ефективних підходів, що можуть бути інтегровані в єдине інформаційне середовище університету, яке забезпечить єдину точку входу та гармонійну взаємодію різних підсистем.

Серед наявних рішень для даного типу задач є системи “Single Sign-On”, “OAuth 2.0”, “OpenID Connect”, “OTP”, “TOTP” [2].

Single Sign-On (SSO) є концепцією, яка дозволяє користувачам отримувати доступ до багатьох систем за допомогою однієї авторизації. У контексті університетських середовищ це особливо важливо для полегшення доступу до навчальних, адміністративних та інших інформаційних ресурсів без потреби постійно вводити різні облікові дані для кожної підсистеми [1].

OAuth 2.0 Authorization Framework є широко використовуваним стандартом для надання стороннім додаткам обмеженого доступу до ресурсів без передачі паролів користувача. Він надає можливість розробникам реалізувати безпечний механізм авторизації, що забезпечує різні рівні доступу залежно від типу клієнта (конфіденційні чи публічні) [1].

OpenID Connect є протоколом, який розширює можливості OAuth 2.0, додаючи рівень ідентифікації, що дозволяє забезпечити перевірку особистості користувача. Це робить OpenID Connect особливо корисним у контексті освітніх установ, оскільки він спрощує процес входу в систему для користувачів, забезпечуючи зручність і уніфікований доступ до різних ресурсів [3].

OpenID Connect не розглядається як спосіб комунікації між машинами. Натомість він орієнтований на аутентифікацію користувачів і забезпечення безпечного входу до ресурсів на основі стандартизованих механізмів.

Для створення системи єдиної точки входу (Single Sign-On, SSO) на базі стандарту OAuth 2.0 Authorization Framework в умовах КПІ ім. Ігоря Сікорського необхідно дотримуватися наступних вимог:

- кожен додаток, який використовуватиме систему SSO, має бути зареєстрований на сервері авторизації OAuth 2.0, і реєстрація повинна включати вказівку типів клієнтів (конфіденційні або публічні), URI перенаправлення та інші необхідні деталі, такі як назва додатку та вебсайт;

- система повинна підтримувати кілька типів дозволів, які відповідають різним умовам клієнта, серед яких дозвіл на авторизаційний код, неявний дозвіл, дозвіл на пароль власника ресурсу та дозвіл на повноваження клієнта для комунікацій між машинами;

- централізований сервер авторизації має керувати всіма токенами доступу, дозволяючи інтеграцію та контроль над різними платформами, такими як `campus.kpi.ua`, `my.kpi.ua`, забезпечуючи безпечні методи аутентифікації клієнтів та видачу токенів;

- система повинна забезпечити єдину точку входу для всіх користувачів університету, незалежно від того, до яких саме підсистем вони звертаються, централізуючи аутентифікацію та скорочуючи кількість облікових записів і паролів, які потрібно пам'ятати користувачам;

- система також повинна мати механізми для управління різними ролями користувачів, такими як студенти, викладачі та адміністративний персонал, з відповідними правами доступу;

- права доступу повинні бути чітко розмежовані на основі ролей, що забезпечить коректний і безпечний доступ до інформаційних ресурсів.

Масштабованість системи є важливим аспектом, вона має бути здатною до обслуговування значної кількості користувачів та інтеграції з усіма існуючими освітніми платформами університету, підтримуючи нові сервіси і підсистеми в майбутньому.

Простота у використанні є важливим критерієм для такої системи. Вона повинна бути інтуїтивно зрозумілою для користувачів, зручна у використанні та не вимагати додаткових знань або спеціальних навичок. Багатофакторна аутентифікація повинна бути реалізована таким чином, щоб забезпечити зручність для користувачів без додаткових ускладнень. Також система повинна відповідати вимогам безпеки OAuth 2.0, включаючи шифрування токенів доступу та захист від можливих атак, таких як повторна атака або атака злоумисника.

## 1.2 Аналіз існуючих рішень

Існуючі рішення для реалізації єдиної точки входу (SSO) в освітніх установах є ключовим елементом для забезпечення зручності доступу до різноманітних систем. Загальна ідея таких рішень полягає в об'єднанні кількох різних підсистем університету через єдиний механізм аутентифікації, що дозволяє користувачам увійти в усі сервіси університету, використовуючи одні й ті ж облікові дані.

Існуючі рішення SSO для освітніх систем здебільшого реалізуються на основі технологій OAuth 2.0 та OpenID Connect, які забезпечують стандартизований спосіб доступу до ресурсів. Багато університетів у світі впровадили такі системи для спрощення користування своїми інформаційними ресурсами. Наприклад, у системах Moodle та Canvas LMS застосовується інтеграція з SSO для забезпечення централізованого входу для студентів і викладачів, що значно спрощує процес взаємодії з різними освітніми платформами.

Одним із популярних рішень є використання Microsoft Azure Active Directory для реалізації SSO в освітніх установах [19]. Це рішення дозволяє забезпечити централізовану аутентифікацію для різних сервісів, таких як електронна пошта, платформи для спільної роботи та навчальні системи. Також Google Workspace for Education пропонує функціональність SSO, що дозволяє користувачам отримувати доступ до всіх інструментів Google за допомогою однієї авторизації.

Інші університети впроваджують локальні рішення на базі CAS (Central Authentication Service) – відомого протоколу для SSO, який широко застосовується у вищих навчальних закладах. CAS забезпечує централізовану аутентифікацію користувачів і підтримує інтеграцію з багатьма різноманітними підсистемами, такими як бібліотечні системи, системи управління навчальним процесом і адміністративні портали [20].

Основною перевагою таких рішень є зручність для користувачів, оскільки вони отримують можливість доступу до всіх сервісів через одну точку входу, що значно скорочує час і зусилля, потрібні для роботи з університетськими ресурсами. Крім того, централізована система аутентифікації дозволяє краще контролювати та забезпечувати безпеку даних користувачів.

Рішення, які наразі забезпечують аутентифікацію й авторизацію користувачів в КІІ ім. Ігоря Сікорського не мають багатьох ключових характеристик перелічених вище, а саме:

- не надають єдиної точки входу в систему;
- не мають уніфікації для потрібної кількості ролей;
- мають технічні обмеження для подальшої інтеграції;
- є системами закритими для розробки і внесення змін;
- не підтримують аутентифікацію користувача загалом.

Це відбувається переважно через розділення призначень для кожної з підсистем, що призводить до розділення користувацького профіля на велику кількість аккаунтів, кожному з яких наданий доступ лише до певної частини даних користувача. Це породжує такі проблеми як :

- необхідність синхронізації профілів користувачів, наприклад під час початку нового семестру або у випадку припинення навчання в університеті;
- необхідність розділяти рівні доступу користувача згідно наявної структури даних окремо для кожного продукту;

- розмиття відповідальності за авторизацію користувача між різними частинами інформаційного середовища;
- існування кількох реалізацій профілю користувача з використанням різних технологій, підходів та команд, що надалі може ускладнювати процес розширення університету через необхідність підтримки нових стандартів та вимог різними кодовими базами;
- необхідність узгодження стандартів на ролей користувачів між різними командами розробників.

Також важливо відзначити, що для великих установ із розгалуженою структурою впровадження єдиної точки входу може потребувати значних інвестицій у модернізацію інформаційних систем.

### 1.2.1 Azure Active Directory.

Одним із популярних рішень для реалізації SSO в освітніх установах є Microsoft Azure Active Directory. Це рішення надає централізовану аутентифікацію для різних сервісів, включаючи електронну пошту, платформи для спільної роботи та навчальні системи. Основною перевагою Azure Active Directory є його інтеграція з іншими продуктами Microsoft, такими як Microsoft 365, що дозволяє користувачам зручно взаємодіяти з різними освітніми та адміністративними інструментами в одному середовищі. Крім того, система підтримує багатофакторну аутентифікацію, що підвищує рівень захисту даних користувачів.

При інтеграції з Azure Active Directory постає питання портативності облікових даних університету, таких як списки студентів, викладачів, формування груп. Тому надаючи перевагу в централізованій аутентифікації користувачів дана система вимагатиме глибокої інтеграції й порушення консистентності інформаційного середовища університету, також порушується питання захищеності даних користувачів, які не можуть передаватися стороннім сервісам без відповідних гара

Однією з переваг Azure AD є також можливість налаштування політик доступу, які враховують різні ролі користувачів, такі як студенти, викладачі, адміністративний персонал. Це дозволяє створювати диференційовані права доступу до ресурсів залежно від потреб конкретної категорії користувачів. Використання ролей та груп у Azure AD значно спрощує управління доступом та дозволяє університету автоматизувати процеси надання та відкликання прав доступу.

Проте, використання Azure Active Directory має і свої недоліки. Один із них - це вартість впровадження та підтримки. Для університетів із обмеженим бюджетом ліцензії на використання Azure AD можуть бути занадто дорогими. Крім того, інтеграція Azure AD із сервісами, що не належать до екосистеми Microsoft, може вимагати додаткових технічних зусиль та спеціальних налаштувань, що ускладнює розгортання системи у більш гетерогенних середовищах. Додатковою складністю є необхідність навчання персоналу для роботи з Azure AD, а також забезпечення підтримки користувачів, що може вимагати додаткових ресурсів.

Таким чином, Microsoft Azure Active Directory є потужним рішенням для впровадження SSO в університетському середовищі завдяки своїй інтеграції з іншими продуктами Microsoft, можливості багатофакторної аутентифікації та підтримці оновлення даних у реальному часі. Однак, висока вартість та певні технічні складнощі можуть стати перешкодою для його використання, особливо в умовах обмеженого бюджету та різномірної інфраструктури.

### 1.2.2 Central Authentication Service (CAS)

Central Authentication Service (CAS) — це відкрите програмне забезпечення, яке забезпечує реалізацію єдиної точки входу (SSO) для автентифікації користувачів. CAS було розроблено в Єльському університеті для забезпечення безпечного та спрощеного доступу до численних підсистем університету, що дозволяє студентам, викладачам та адміністративному персоналу використовувати один набір облікових даних для доступу до всіх освітніх і адміністративних

ресурсів. З моменту свого створення система активно розвивалася та була впроваджена у багатьох навчальних закладах по всьому світу.

Основні переваги CAS включають:

Відкритість та гнучкість:

- відкритий вихідний код забезпечує можливість адаптації під власні потреби;
- можливість інтеграції з існуючими інформаційними системами;
- підтримка власної бази даних користувачів університету;
- гнучкі налаштування політик безпеки та доступу;
- можливість модифікації та розширення функціоналу.

Функціональні можливості:

- підтримка різних методів автентифікації (паролі, OTP, багатофакторна автентифікація);
- автоматична синхронізація даних користувачів ;
- централізоване управління правами доступу;
- можливість інтеграції з різними протоколами та стандартами;
- підтримка масштабування системи.

Безпека:

- захищений механізм єдиного входу;
- підтримка сучасних криптографічних протоколів;
- можливість аудиту та моніторингу подій безпеки;
- контроль сесій та автоматичний вихід;
- захист від поширених атак.

База даних регулярно оновлюється, що гарантує актуальність інформації про користувачів та їхні права доступу. Так, наприклад, зміни в реєстрації студентів, оцінки, та інші події автоматично синхронізуються з системою CAS, що значно полегшує управління доступом. Система також забезпечує:

#### Управління користувачами:

- централізоване зберігання облікових даних;
- автоматична синхронізація з іншими системами;
- гнучке управління ролями та правами;
- підтримка групових політик;
- можливість делегування прав адміністрування.

Одним із ключових аспектів CAS є можливість підтримки численних методів автентифікації, включаючи паролі, одноразові паролі (OTP), та інші форми багатофакторної автентифікації. Це підвищує рівень безпеки і забезпечує захист інформації користувачів, що є критично важливим для освітніх установ.

CAS також надає підтримку для масштабування системи. Оскільки це відкрите програмне забезпечення, університет може безперешкодно розширювати кількість користувачів та інтегрувати нові підсистеми без значних фінансових витрат. Відкритість і можливість налаштування системи дозволяє гнучко змінювати її функціонал відповідно до потреб університету.

Проте, використання CAS має певні обмеження та недоліки:

#### Технічні складнощі:

- потреба у спеціалізованих знаннях для налаштування;
- складність інтеграції з legacy-системами;
- необхідність постійного моніторингу та обслуговування;
- можливі проблеми сумісності з деякими додатками;
- вимоги до інфраструктури та продуктивності.

#### Організаційні виклики:

- необхідність навчання технічного персоналу;
- витрати на впровадження та підтримку;
- залежність від спільноти розробників;
- відсутність офіційної технічної підтримки;

#### Обмеження функціоналу:

- можливі обмеження при інтеграції з деякими типами систем;
- необхідність додаткової розробки для специфічних вимог;
- залежність від стандартних протоколів;
- складність кастомізації деяких компонентів;
- обмеження при роботі з legacy-системами.

Також існує певна залежність від спільноти розробників та підтримки CAS. Оскільки це відкрите програмне забезпечення, офіційної підтримки як такої немає, і користувачі повинні покладатися на документацію та допомогу з боку спільноти. Це може створити труднощі у разі виникнення складних проблем, особливо коли потрібна негайна підтримка.

Таким чином, Central Authentication Service є ефективним рішенням для реалізації єдиної точки входу в освітніх установах, що має наступні ключові характеристики:

- відкритість та можливість модифікації під власні потреби;
- підтримка різних методів автентифікації та авторизації;
- можливість масштабування та інтеграції з існуючими системами;
- активна спільнота розробників та користувачів;
- економічна ефективність впровадження;
- гнучкість налаштування та адміністрування.

Проте, необхідно враховувати технічні та організаційні виклики при впровадженні системи, особливо для установ з обмеженими ресурсами або недостатньою технічною експертизою.

### 1.2.3 Google Classroom

Google Classroom - це безкоштовний веб-сервіс, розроблений Google для освітніх установ, який спрощує створення, поширення і класифікацію завдань безпаперовим шляхом. Основною метою сервісу є упорядкування процесу обміну файлами між викладачами та студентами.

Система має ряд суттєвих переваг, зокрема доступність та простоту використання. Google Classroom надає безкоштовний доступ для освітніх установ, має інтуїтивно зрозумілий інтерфейс та дозволяє швидко розпочати роботу без складних налаштувань. Важливою перевагою є можливість доступу з будь-якого пристрою та глибока інтеграція з іншими сервісами Google.

Функціональні можливості Google Classroom включають створення віртуальних класів, публікацію завдань та навчальних матеріалів, оцінювання робіт студентів, а також забезпечення комунікації між викладачами та студентами. Система автоматично створює необхідні папки на Google Drive для зберігання матеріалів курсу.

В контексті управління навчальним процесом Google Classroom забезпечує можливості для організації матеріалів за темами, встановлення термінів виконання завдань, відстеження прогресу студентів та надання швидкого зворотного зв'язку. Викладачі можуть ефективно перевіряти роботи та комунікувати зі студентами.

Проте, система має ряд суттєвих обмежень та недоліків. З технічної точки зору, Google Classroom сильно залежить від стабільного інтернет-з'єднання, має обмеження на розмір файлів та не дозволяє глибоку кастомізацію інтерфейсу. Можливості для адміністрування та аналітики також досить обмежені. Критичним недоліком є неможливість прямої інтеграції з ЄІС університету через відсутність відповідних API та механізмів синхронізації даних із закритими системами.

Функціональні недоліки включають дуже обмежений контекст взаємодії в рамках окремих класів, що ускладнює створення складних навчальних сценаріїв та групову роботу. Відсутність інтеграції з університетськими системами та неможливість автоматичної синхронізації даних між Google Classroom та ЄІС створюють додаткові складнощі в роботі.

Особливу складність представляє взаємодія з ЄІС університету. Закрита архітектура ЄІС та обмежений доступ до її даних унеможливають автоматичне оновлення інформації про контингент. Відсутність стандартизованих протоколів

обміну даними вимагає підтримки двох паралельних систем обліку, що створює додаткове навантаження на персонал та ризики при передачі конфіденційних даних.

З організаційної точки зору виникають проблеми з необхідністю створення та управління окремими класами, відсутністю централізованого управління на рівні закладу та складністю масштабування на велику кількість користувачів. Ручне внесення даних, які вже існують в ЄІС, та підтримка їх актуальності вимагають значних ресурсів.

Таким чином, Google Classroom, попри свої переваги у вигляді простоти використання та доступності, має суттєві обмеження для використання як основної платформи електронного навчання у великих освітніх установах. Обмежений контекст взаємодії, відсутність глибокої інтеграції з університетськими системами та складність синхронізації даних з ЄІС роблять його більш придатним для використання як додаткового інструменту, а не повноцінної системи управління навчальним процесом.

Важливо відзначити, що для повноцінної інтеграції Google Classroom у навчальний процес університету необхідна розробка додаткових програмних компонентів та адаптерів. Досвід використання Google Classroom в різних університетах світу показує, що найбільш ефективним є його застосування як допоміжного інструменту паралельно з основними системами управління навчальним процесом. У зв'язку з цим, при проектуванні нових систем автентифікації важливо передбачити можливість інтеграції з Google Classroom на рівні облікових записів користувачів. Такий підхід дозволить зберегти переваги обох систем, забезпечуючи при цьому єдину точку входу для користувачів.

Порівняння розглянутих рішень виконано в таблиці 1.1.

Таблиця 1.1 – Порівняння існуючих рішень

Характеристика	Azure AD	CAS	Google Classroom
Інтеграція з існуючими системами університету	Обмежена, потребує додаткових адаптерів	Висока, підтримує різні протоколи	Низька, закрита екосистема
Підтримка багатофакторної автентифікації	Так, повна підтримка	Так, через модулі розширення	Так, тільки через Google
Можливість кастомізації	Середня	Висока	Низька
Вартість впровадження	Висока, ліцензійна	Низька, відкрите ПЗ	Безкоштовно для освіти
Автономність даних	Низька, дані в хмарі Microsoft	Висока, локальне розгортання	Низька, дані в Google
Підтримка ролей користувачів	Стандартні ролі AD	Гнучка система ролей	Обмежена (викладач/студент)
Технічна підтримка	Професійна від Microsoft	Спільнота розробників	Підтримка Google
Синхронізація з АС "Ядро"	Складна	Можлива через API	Неможлива
Відкритість коду	Закрита система	Відкритий код	Закрита система
Аудит безпеки	Повний	Базовий	Базовий
Інтеграція з мобільними додатками	Так	Через API	Так
Можливість поетапного впровадження	Складно	Так	Ні

У цілому, аналіз існуючих рішень показує, що реалізація SSO в університетських середовищах на основі стандартів OAuth 2.0 та OpenID Connect є ефективним способом забезпечення зручності, безпеки та уніфікації доступу до освітніх і адміністративних ресурсів. може скористатися досвідом інших університетів, адаптуючи ці технології до своїх потреб, щоб забезпечити централізовану та ефективну систему ідентифікації користувачів.

### 1.3 Аналіз вимог до програмного забезпечення

У процесі проектування системи University ID було проведено детальний аналіз вимог до програмного забезпечення. На основі аналізу існуючої інфраструктури університету та організаційної структури були виявлені основні варіанти використання системи. Центральне місце в системі займає процес автентифікації користувачів та управління їхніми ролями, при цьому важливим аспектом є взаємодія з АС "Ядро" як джерелом достовірних даних про користувачів університету.

Діаграма варіантів використання (рис. 1.1) демонструє ключових акторів системи та їх взаємодію з основними функціями. На діаграмі представлено чотири основних актори: надавачі освітніх послуг, отримувачі освітніх послуг, клієнтські системи та АС "Ядро". Система розділяє користувачів на дві основні категорії:

- надавачі освітніх послуг (викладачі, адміністративний персонал);
- отримувачі освітніх послуг (студенти, аспіранти).

Кожен з акторів має свій набір варіантів використання, що відображає їхні специфічні ролі та права в системі. Надавачі освітніх послуг мають доступ до всіх основних функцій системи, включаючи аутентифікацію, отримання даних користувача, управління ролями, перегляд доступних ресурсів та можливість відкриття доступу. Отримувачі освітніх послуг мають більш обмежений набір

функцій, зосереджений на базових операціях аутентифікації, отриманні власних даних та перегляді доступних їм ресурсів.

Важливим елементом діаграми є відображення взаємозв'язків між варіантами використання. Зокрема, більшість операцій включають в себе процес аутентифікації користувача що підкреслює центральну роль механізму аутентифікації в системі. Клієнтські системи взаємодіють з функціями аутентифікації та отримання даних користувача, в той час як АС "Ядро" забезпечує доступ до даних користувачів та управління ролями.

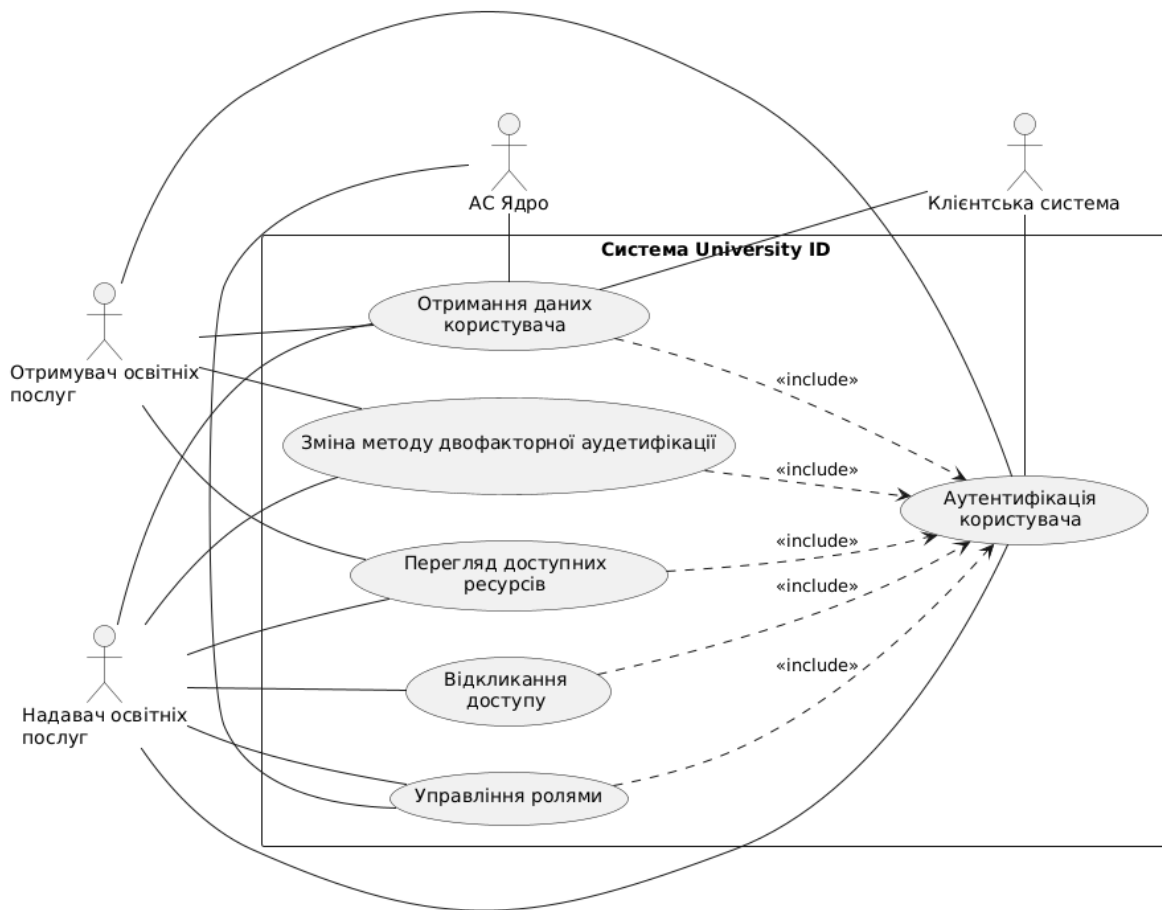


Рисунок 1.1 – Діаграма варіантів використання

Аналіз варіантів використання наведений в таблицях 1.1 - 1.7:

Таблиця 1.2 – Варіант використання UC-01

Use case name	Автентифікація користувача
Use case ID	UC-01
Goals	Надати користувачу доступ до системи університету через єдину точку входу
Actors	Надавач освітніх послуг, Отримувач освітніх послуг, Клієнтська система
Trigger	Спроба доступу до захищеного ресурсу
Pre-conditions	1. Користувач має обліковий запис в системі 2. Клієнтська система зареєстрована в University ID
Flow of Events	<ul style="list-style-type: none"> <li>– Користувач переходить до клієнтської системи</li> <li>– Система перенаправляє на сторінку автентифікації University ID</li> <li>– Користувач вводить облікові дані</li> <li>– Система верифікує дані через АС "Ядро"</li> <li>– Система генерує токен доступу</li> <li>– Користувач перенаправляється назад до клієнтської системи з токеном</li> <li>– Система записує інформацію про успішну автентифікацію в журнал подій</li> <li>– Система надсилає сповіщення про новий вхід на електронну пошту користувача</li> </ul>
Extension	<ul style="list-style-type: none"> <li>– Якщо дані невірні, система показує помилку</li> <li>– Якщо сесія вже існує, система пропускає кроки 3-4</li> </ul>
Post-Condition	<ul style="list-style-type: none"> <li>– Користувач отримує доступ до системи</li> <li>– Токен доступу збережено в клієнтській системі</li> </ul>

Таблиця 1.3 – Варіант використання UC-02

Use case name	Отримання профілю користувача
Use case ID	UC-02
Goals	Отримати актуальні дані про користувача з АС "Ядро"
Actors	Надавач освітніх послуг, Отримувач освітніх послуг, Клієнтська система, АС "Ядро"
Trigger	Запит клієнтської системи на отримання даних користувача
Pre-conditions	1. Наявний валідний токен доступу 2. Клієнтська система має необхідні права
Flow of Events	1. Клієнтська система надсилає запит з токеном 2. Система верифікує токен 3. Система запитує дані з АС "Ядро" 4. Система форматує та повертає дані відповідно до ролі користувача
Extension	– Якщо токен невалідний, повертається помилка – Якщо дані в АС "Ядро" недоступні, використовується кеш
Post-Condition	– Клієнтська система отримує актуальні дані користувача

Таблиця 1.4 – Варіант використання UC-03

Use case name	Управління ролями
Use case ID	UC-03
Goals	Забезпечити актуальність ролей користувачів відповідно до даних АС "Ядро"
Actors	Надавач освітніх послуг, АС "Ядро"
Trigger	Необхідність зміни ролей користувача
Pre-conditions	1. Користувач існує в системі 2. Клієнтська система має необхідні права

Таблиця 1.4 – Варіант використання UC-03 (Продовження)

Flow of Events	<ol style="list-style-type: none"> <li>1. Система отримує оновлені дані про ролі з АС "Ядро"</li> <li>2. Система порівнює поточні ролі з отриманими</li> <li>3. Система оновлює інформацію про ролі користувача</li> <li>4. Система інвалідує відповідні кеші та токени доступу</li> </ol>
Extension	<ul style="list-style-type: none"> <li>– Якщо з'єднання з АС "Ядро" недоступне, зберігаються поточні ролі</li> <li>– При критичних змінах ролей (наприклад, видалення всіх ролей) відбувається примусове завершення активних сесій</li> </ul>
Post-Condition	<ul style="list-style-type: none"> <li>– Ролі користувача синхронізовано з АС "Ядро"</li> <li>– Права доступу оновлено відповідно до нових ролей</li> </ul>

Таблиця 1.5 – Варіант використання UC-04

Use case name	Перегляд ресурсів
Use case ID	UC-04
Goals	Надати користувачу інформацію про доступні інформаційні ресурси
Actors	Надавач освітніх послуг, Отримувач освітніх послуг
Trigger	Необхідність зміни ролей користувача
Pre-conditions	Користувач автентифікований в системі
Flow of Events	<ol style="list-style-type: none"> <li>1. Користувач запитує список доступних ресурсів</li> <li>2. Система перевіряє права доступу користувача</li> <li>3. Система формує список доступних ресурсів відповідно до ролі</li> <li>4. Система відображає список з можливістю переходу до ресурсів</li> </ol>
Extension	<ul style="list-style-type: none"> <li>– Якщо немає доступних ресурсів, показується відповідне повідомлення</li> </ul>
Post-Condition	<ul style="list-style-type: none"> <li>– Користувач бачить список доступних ресурсів</li> </ul>

Таблиця 1.6 – Варіант використання UC-05

Use case name	Заміна методу двофактної аутентифікації
Use case ID	UC-05
Goals	Надати користувачу можливість змінити метод двофакторної аутентифікації
Actors	Надавач освітніх послуг, Отримувач освітніх послуг
Trigger	<ul style="list-style-type: none"> <li>– Бажання користувача обрати інший спосіб аутентифікації</li> <li>– Підозра на компрометацію облікового запису</li> </ul>
Pre-conditions	<ul style="list-style-type: none"> <li>– Користувач аутентифікований в системі</li> <li>– У користувача є активні сесії</li> </ul>
Flow of Events	<ol style="list-style-type: none"> <li>1. Користувач обирає новий метод 2FA (TOTP, SMS або Email) в налаштуваннях безпеки Система відображає наявний спосіб двофакторної аутентифікації</li> <li>2. Система ініціює налаштування обраного методу та надсилає необхідні дані для верифікації</li> <li>3. Користувач підтверджує новий метод, ввівши тестовий код верифікації</li> <li>4. Система деактивує старий метод, активує новий та генерує нові резервні коди</li> </ol>
Extension	<ul style="list-style-type: none"> <li>– Користувач може обрати термінове відкликання всіх сесій у випадку підозри на злом</li> </ul>
Post-Condition	<ul style="list-style-type: none"> <li>– Доступ користувача оновлено відповідно до даних АС "Ядро"</li> <li>– Завершено всі сесії, які не відповідають оновленим засобам аутентифікації</li> </ul>

Таблиця 1.7 – Варіант використання UC-06

Use case name	Відкликання доступу
Use case ID	UC-06
Goals	Надати користувачу можливість припинити сесію
Actors	Надавач освітніх послуг, Отримувач освітніх послуг
Trigger	<ul style="list-style-type: none"> <li>– підозра на компрометацію облікового запису;</li> <li>– необхідність термінового припинення доступу;</li> <li>– зміна статусу користувача в АС "Ядро".</li> </ul>
Pre-conditions	<ul style="list-style-type: none"> <li>– наявність активної сесії користувача;</li> <li>– наявність відповідних прав для відкликання доступу.</li> </ul>
Flow of Events	<ol style="list-style-type: none"> <li>1. Ініціювання процесу відкликання доступу</li> <li>2. Система показує список активних сесій та пристроїв</li> <li>3. Система верифікує права на виконання операції Система запитує підтвердження дії</li> <li>4. Система інвалідує всі активні токени доступу</li> <li>5. Система сповіщає користувача про відкликання доступу</li> </ol>
Extension	<ul style="list-style-type: none"> <li>– Користувач може обрати термінове відкликання всіх сесій у випадку підозри на злом</li> </ul>
Post-Condition	<ul style="list-style-type: none"> <li>– Доступ користувача оновлено відповідно до даних АС "Ядро"</li> <li>– Всі невідповідні сесії завершено</li> </ul>

Дослідимо функціональні вимоги до системи (табл. 1.7), та на основі визначених варіантів використання побудуємо матрицю трасування вимог (табл. 1.8):

Таблиця 1.8 – Функціональні вимоги

Код вимоги	Опис вимоги
FR-01	Єдина точка входу для автентифікації користувачів
FR-02	Інтеграція з АС "Ядро" для отримання даних користувачів
FR-03	Синхронізація ролей користувачів з АС "Ядро"
FR-04	Забезпечення токена доступу для клієнтських систем
FR-05	Управління сесіями користувачів
FR-07	Надання API для клієнтських систем
FR-08	Керування власним доступом користувачами

Таблиця 1.9 – Матриця трасування вимог

	FR-01	FR-02	FR-03	FR-04	FR-05	FR-06	FR-07
UC-1	+	+	+	+	+	+	
UC-2		+	+	+		+	
UC-3		+	+			+	
UC-4		+	+	+	+	+	
UC-5					+		+
UC-6	+						

На основі проведеного аналізу та розробленої матриці трасування вимог можна побачити, що всі ідентифіковані варіанти використання системи покриваються визначеними функціональними вимогами. Матриця демонструє тісний взаємозв'язок між сценаріями використання та функціональними можливостями системи, що підтверджує повноту та узгодженість визначених вимог.

#### 1.4 Постановка задачі

На основі проведеного аналізу предметної області та існуючих рішень встановлено, що наявна інфраструктура інформаційних систем закладу вищої освіти характеризується роз'єднаністю підходів до ідентифікації користувачів. Кожна з систем, таких як `campus.kpi.ua`, `my.kpi.ua`, `directory.kpi.ua` реалізує власні механізми автентифікації та зберігання даних користувачів, що створює надлишковість та ускладнює процеси управління доступом.

Метою роботи є розробка та впровадження системи єдиної точки входу (SSO) для уніфікації ідентифікації користувачів в інформаційному середовищі закладу вищої освіти на основі стандарту OAuth 2.0 для забезпечення централізованого доступу до всіх інформаційних ресурсів університету. Важливим аспектом є забезпечення можливості поступової міграції існуючих систем без порушення їх функціонування.

Проект University ID спрямований на створення єдиного джерела достовірної інформації для всіх підсистем та додатків університету щодо даних користувачів та їх прав доступу. В якості первинного джерела даних про користувачів та точки доступу до існуючих реєстрів використовується АС "Ядро", яка забезпечує уніфікацію та класифікацію даних університету та впроваджує єдину систему ролей. Таким чином, проект University ID бере на себе відповідальність за дані користувачів та їх автентифікацію, що дозволяє знизити навантаження на інші підсистеми університету та усуває необхідність дублювання функціоналу управління користувачами в кожній окремій системі. Це не лише спрощує розробку та підтримку нових сервісів, але й забезпечує цілісність та актуальність даних про користувачів у масштабах всього університету.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- проаналізувати вимоги до системи уніфікованої ідентифікації користувачів з урахуванням специфіки освітнього закладу, включаючи дослідження існуючих потоків даних між різними підсистемами та визначення точок інтеграції;
- розробити архітектуру системи єдиної точки входу на основі протоколу OAuth 2.0 з підтримкою OpenID Connect, що дозволить реалізувати сучасні підходи до управління ідентифікацією та забезпечить сумісність з широким спектром технологій;
- розробити механізм інтеграції з існуючими інформаційними системами університету, включаючи створення адаптерів для різних протоколів автентифікації та форматів даних;
- оцінити розроблене рішення через аналіз процесів інтеграції та управління користувачами.

Створена система повинна відповідати наступним функціональним вимогам:

- забезпечення єдиної точки входу для всіх інформаційних систем університету через централізований сервер авторизації, який стане основним джерелом інформації про користувачів та їх права;
- управління ролями для різних категорій користувачів з можливістю розширення, що дозволить гнучко налаштовувати права доступу відповідно до організаційної структури університету;
- підтримка стандартних протоколів OAuth 2.0 та OpenID Connect для забезпечення сумісності з існуючими та майбутніми системами, що спростить процес розробки нових сервісів;
- можливість поетапного впровадження в існуючу інфраструктуру університету через підтримку паралельного функціонування старих та нових механізмів автентифікації.

Нефункціональні вимоги до системи включають:

- розширюваність архітектури для додавання нових типів авторизації та інтеграції з різними протоколами автентифікації;

- підтримка різних джерел даних для зберігання інформації про користувачів, що дозволить використовувати існуючі бази даних та поступово переходити до нової архітектури;
- можливість інтеграції з існуючими системами без значних змін у їхній архітектурі через використання проміжних адаптерів та шлюзів;
- повна документація щодо інтеграції для розробників нових систем, включаючи приклади реалізації та типові сценарії використання.

Обмеження та припущення при розробці системи:

- система розробляється з використанням відкритих технологій та стандартів, що забезпечить незалежність від конкретних постачальників та можливість модифікації;
- необхідна підтримка існуючої інфраструктури університету, включаючи збереження працездатності критично важливих систем під час міграції;
- рішення має забезпечувати можливість поступового переходу існуючих систем на нову модель авторизації без необхідності одночасної міграції всіх підсистем.

Результатом роботи має стати комплексне рішення для централізованої ідентифікації користувачів, що дозволить уніфікувати процеси управління доступом до інформаційних ресурсів університету. Система забезпечить єдиний підхід до автентифікації та авторизації, спростить процеси інтеграції різних підсистем та створить фундамент для подальшого розвитку цифрової інфраструктури закладу вищої освіти.

Впровадження розробленої системи дозволить вирішити проблему фрагментації користувацьких даних, зменшити складність адміністрування та забезпечити більш ефективне використання інформаційних ресурсів університету.

## Висновки до розділу

У першому розділі було проведено комплексний аналіз предметної області систем ідентифікації користувачів в освітніх закладах. В результаті дослідження виявлено ряд суттєвих проблем в існуючій інфраструктурі КПІ ім. Ігоря Сікорського, зокрема відсутність уніфікованої системи автентифікації для різних підсистем університету, що призводить до необхідності підтримки окремих баз користувачів у кожній системі. Встановлено, що поточний стан призводить до значних складнощів при синхронізації даних між різними компонентами та відсутності єдиних стандартів багатфакторної автентифікації.

В ході роботи було досліджено існуючі рішення для реалізації єдиної точки входу (SSO). Розглянуто Microsoft Azure Active Directory як потужне корпоративне рішення, що однак вимагає значних фінансових інвестицій та глибокої інтеграції з екосистемою Microsoft. Проаналізовано можливості Central Authentication Service (CAS) - відкритого рішення з гнучкими можливостями налаштування, яке при цьому характеризується підвищеною складністю впровадження та потребує значних технічних компетенцій. Також досліджено Google Classroom як приклад простого у використанні сервісу, котрий, втім, має суттєво обмежені можливості інтеграції з університетськими системами та не може повністю задовольнити потреби закладу вищої освіти.

Важливою частиною дослідження став детальний аналіз вимог до програмного забезпечення. Визначено та описано основні варіанти використання системи та сформульовано функціональні вимоги. В результаті аналізу виявлено критичну необхідність забезпечення єдиної точки входу для всіх інформаційних систем університету з обов'язковою інтеграцією з АС "Ядро" як джерелом достовірних даних. Особливу увагу приділено необхідності підтримки різних ролей користувачів та забезпечення можливості поступової міграції існуючих систем без порушення їх функціонування.

На підставі проведеного аналізу було сформульовано постановку задачі розробки системи University ID. Визначено, що система повинна забезпечувати централізоване управління ідентифікацією користувачів та надавати уніфікований доступ до всіх інформаційних ресурсів університету. Важливою вимогою стала підтримка сучасних стандартів OAuth 2.0 та OpenID Connect, що забезпечить сумісність з широким спектром існуючих та майбутніх систем. Особливий акцент зроблено на необхідності забезпечення можливості поетапного впровадження системи без порушення роботи існуючої інфраструктури університету.

Проведене дослідження переконливо доводить необхідність розробки системи уніфікованої ідентифікації користувачів для інформаційного середовища КПІ ім. Ігоря Сікорського " та формує чітке розуміння вимог і обмежень, які необхідно врахувати при її проектуванні та реалізації. Отримані результати аналізу створюють надійний фундамент для подальшої розробки архітектури системи та її технічної реалізації.

## 2 ПРОЄКТУВАННЯ РІШЕННЯ

### 2.1 Опис процесу діяльності

В даному розділі розглядаються ключові аспекти проектування системи уніфікованої ідентифікації користувачів University ID, що розробляється для інформаційного середовища КПІ ім. Ігоря Сікорського. Основна увага приділяється архітектурним рішенням, які забезпечують надійну інтеграцію з існуючими системами університету та відповідають визначеним у попередньому розділі вимогам.

Проектування системи базується на сучасних підходах до розробки програмного забезпечення та враховує специфіку освітнього середовища. Важливим аспектом є вибір оптимальних технологій та архітектурних патернів, які забезпечать необхідну гнучкість та масштабованість системи. Особлива увага приділяється проектуванню механізмів взаємодії з АС "Ядро" як основним джерелом даних про користувачів, а також розробці уніфікованого API для інтеграції з іншими підсистемами університету.

В розділі детально розглядаються структурні компоненти системи, їх взаємозв'язки та принципи функціонування. Описуються механізми автентифікації та авторизації, що базуються на стандартах OAuth 2.0 та OpenID Connect. Значна увага приділяється проектуванню системи управління ролями користувачів, яка повинна забезпечити гнучке налаштування прав доступу відповідно до організаційної структури університету.

Також розглядаються механізми кешування для оптимізації продуктивності та стратегії масштабування системи. Представлені архітектурні рішення спрямовані на створення надійної та ефективної системи, що відповідає сучасним вимогам до програмного забезпечення корпоративного рівня та забезпечує централізоване управління доступом користувачів до інформаційних ресурсів університету.

Ключовим завданням проектування системи є розробка механізмів компонування та синхронізації даних користувачів. АС "Ядро" виступає як центральний каталог довідників, що забезпечує уніфікований доступ до даних з обох систем. Такий підхід дозволяє стандартизувати процес отримання інформації та забезпечити її валідність на рівні першоджерела.

Системи АС "Деканат" та АС "Відділ кадрів" мають принципово різні структури даних та підходи до зберігання інформації про користувачів. АС "Деканат" містить інформацію про студентів, їх академічні групи, навчальні плани та успішність, тоді як АС "Відділ кадрів" оперує даними про професійні навички, досвід роботи та кар'єрні прагнення студентів. При цьому всі довідники та класифікатори, які використовуються в обох системах, централізовано зберігаються та управляються через АС "Ядро".

Для ефективного об'єднання цих даних необхідно спроектувати гнучку систему маппінгу, яка дозволить створити єдине представлення профілю користувача на основі інформації з обох джерел через АС "Ядро". Важливим аспектом є розробка механізмів узгодження даних у випадку їх розбіжності та визначення пріоритетів при оновленні інформації. Система повинна коректно обробляти ситуації, коли дані про одного й того ж користувача в різних системах можуть відрізнятися або бути неповними, використовуючи АС "Ядро" як джерело достовірної інформації.

Проектування даної системи вимагає ретельного аналізу структур даних обох систем та розробки універсальної моделі даних, яка зможе акомодувати всю необхідну інформацію, враховуючи особливості взаємодії з АС "Ядро". При цьому важливо забезпечити можливість подальшого розширення моделі для інтеграції з іншими системами університету в майбутньому. Особлива увага приділяється розробці механізмів валідації та нормалізації даних, що надходять з різних джерел через АС "Ядро", для забезпечення їх цілісності та консистентності.

При проектуванні системи єдиної автентифікації важливо враховувати специфіку існуючих інформаційних систем університету, які можна розділити на три основні категорії за можливістю модифікації їх механізмів взаємодії з даними користувача.

До першої категорії належать системи з відкритим для розробки кодом, які підтримуються та розвиваються безпосередньо університетом. Такі системи, як "campus.kpi.ua", "my.kpi.ua" та інші внутрішні розробки, надають можливість модифікації та розширення функціоналу автентифікації. Це дозволяє реалізувати пряму інтеграцію з проектованою системою "University ID" шляхом впровадження необхідних програмних компонентів та адаптації існуючих механізмів авторизації.

Друга категорія включає закриті системи, що функціонують на основі придбаних ліцензій та мають обмежені можливості для модифікації. На даному етапі розробки прийнято рішення не включати такі системи область дослідження, оскільки їх модифікація може виявитися технічно складною або економічно недоцільною. Це дозволить зосередитись на розробці сучасного та технологічно актуального рішення, не обмежуючись застарілими технологічними обмеженнями закритих систем.

Третя категорія охоплює системи з частковою відкритістю, які надають можливість інтеграції через API або прямий доступ до бази даних. Такі системи, хоча й не надають повного доступу до свого програмного коду, все ж дозволяють реалізувати базову інтеграцію з системою "University ID" через стандартизовані інтерфейси обміну даними. Це створює можливість для поетапного впровадження єдиної системи автентифікації, починаючи з базового рівня інтеграції через доступні API, з перспективою подальшого розширення функціональності при появі нових можливостей для взаємодії. Такий підхід особливо актуальний для систем, які активно використовуються в навчальному процесі, але мають обмеження щодо модифікації основного коду.

Основні характеристики систем для інтеграції підсумовано в таблиці 2.1:

Таблиця 2.1 –Типи систем, що використовуються

Назва системи	Тип системи	Тип ідентифікатора користувача	Тип інтеграції
АС “Є-Кампус”	Відкрита	Числовий	HTTP API
АС “Деканат”	Закрита	Рядок	Інтеграція на рівні бази даних
АС “Відділ кадрів”	Закрита	Ідентифікатор формується за системою правил	Інтеграція на рівні бази даних
АС “Розклад”	Відкрита	Відсутній	HTTP API
Moodle	Частково відкрита	Рядок	HTTP API

Через таку різницю між системами виникає проблема визначення користувача на програмному рівні, оскільки немає єдиного ідентифікатора користувача, який би можна було використовувати при інтеграції однієї системи з іншою. Через це інтеграція кожної системи вимагає процесу узгодження цих даних з кожною залежною системою.

Така структура взаємодії ускладнює додання чи виключення облікових засобів через необхідність розробки окремого процесу для кожної пари систем [8].

Проект “University ID” передбачає інтеграцію з АС “Ядро”, яка може бути центральною ланкою, що зможе підтримувати зв’язки між спорідненими сутностями в різних системах, що в свою чергу робить можливим уніфікацію процесу оновлення даних. Для забезпечення цієї інтеграції, уніфікована система аутентифікації має визначати єдиний ідентифікатор для облікового запису користувача.

При проектуванні системи особлива увага приділяється використанню сучасних технологічних платформ та стандартів, які активно розвиваються та мають

перспективу довгострокової підтримки, а також врахуванню технічних особливостей взаємодії користувачів із системою. Важливим аспектом є забезпечення зручного доступу для різних груп користувачів, які можуть мати різні вподобання щодо способів автентифікації та пристроїв доступу. Частина користувачів віддає перевагу використанню мобільних пристроїв та відповідних додатків, інші працюють переважно з десктопними версіями через браузер, а деякі користувачі звикли до автентифікації через електронну пошту. Система повинна забезпечувати однаково якісний досвід автентифікації для всіх цих сценаріїв використання.

Важливим критерієм при виборі технологій є їх відкритість до змін та можливість адаптації під нові вимоги, що виникають у процесі розвитку інформаційної інфраструктури університету та потреб користувачів. Такий підхід забезпечить створення гнучкого та масштабованого рішення, здатного еволюціонувати разом з технологічним прогресом та змінами у потребах користувачів щодо методів автентифікації.

## 2.2 Аналіз та проектування сутностей системи

У цьому розділі проводиться детальний аналіз сутностей, які взаємодіють з системою “University ID”. Ключовою особливістю проектованої системи є коректне відображення ролей користувачів в освітньому процесі для забезпечення відповідного рівня доступу до інформаційних ресурсів університету.

Основною сутністю системи є користувач, який може виступати в одній з двох ключових ролей або їх комбінації:

- надавач освітніх послуг (викладачі, адміністративний персонал);
- отримувач освітніх послуг (студенти).

Особливу категорію становлять користувачі, які одночасно належать до обох ролей. Наприклад, аспіранти, які можуть як навчатися (отримувачі послуг), так і проводити практичні заняття (надавачі послуг). Така дуальність ролей

відображається в системі на основі даних, що надходять з АС "Ядро", при цьому сама система "University ID" не бере участі в процесах призначення чи модифікації ролей користувачів, а лише використовує цю інформацію для забезпечення коректної автентифікації та авторизації.

Система працює виключно з фактичними даними про ролі користувачів, які вже визначені в інформаційному середовищі університету. Це забезпечує чітке розмежування відповідальності: управління ролями користувачів здійснюється на рівні відповідних адміністративних систем, тоді як University ID забезпечує лише механізми автентифікації та авторизації на основі цих даних.

У контексті КПІ ім. Сікорського розвиток інформаційної інфраструктури відбувався органічно, відповідаючи на конкретні потреби різних підрозділів університету, які виникали у різний час та за різних обставин. Такий підхід "від периферії до центру" був зумовлений декількома факторами:

По-перше, впровадження систем відбувалося у відповідь на нагальні потреби окремих підрозділів. Наприклад, автоматизація роботи деканатів та кафедр вимагала створення АС "Деканат", необхідність ефективного управління кадрами призвела до впровадження АС "Відділ кадрів". Кожна з цих систем розроблялася як відносно автономне рішення для конкретних задач.

По-друге, різні періоди розробки систем призвели до використання різних технологічних стеків та архітектурних рішень. Системи, що розроблялися раніше, базувалися на технологіях свого часу та відповідали тогочасним уявленням про архітектуру програмного забезпечення. Більш нові системи, такі як АС "my.kpi", використовують сучасніші підходи та технології.

По-третє, відсутність початкового єдиного плану розвитку цифрової інфраструктури призвела до того, що кожна система формувала власні підходи до зберігання та обробки даних, власні каталоги користувачів та специфічні механізми управління доступом. В результаті сформувалася складна екосистема взаємопов'язаних, але технологічно різнорідних систем.

Такий історично сформований підхід мав свої переваги на етапі становлення окремих систем, оскільки дозволяв швидко вирішувати конкретні задачі автоматизації. Проте з часом, при зростанні кількості систем та необхідності їх інтеграції, відсутність централізованого підходу створила ряд викликів, зокрема в області уніфікованої автентифікації та авторизації користувачів.

Аналіз існуючої структури сервісів університету демонструє їх значну автономність, що проявляється у специфічних підходах до організації доступу та управління даними в кожній системі:

АС “Деканат” реалізує складну ієрархічну структуру ролей, пов'язану з організацією навчального процесу [8]:

- працівники деканату мають повний доступ до даних свого факультету/інституту;
- методисти кафедр можуть керувати навчальними планами та розкладом для своїх спеціальностей;
- викладачі отримують доступ до інформації про свої дисципліни та групи;
- куратори мають розширений доступ до даних своїх академічних груп;
- студенти можуть переглядати власні дані про успішність та відвідуваність;

АС "Відділ кадрів" використовує власну систему ролей для кадрового обліку;

- працівники відділу кадрів з повним доступом до особових справ;
- табельники підрозділів з обмеженим доступом до даних про робочий час;
- керівники підрозділів з можливістю перегляду даних своїх підлеглих;
- співробітники з доступом тільки до власних персональних даних;
- спеціальні ролі для доступу до даних про відпустки, лікарняні та інші кадрові процеси.

АС "Е-кампус" має розвинену структуру ролей для забезпечення освітньої взаємодії [8]:

- адміністратори системи з повними правами управління;
- модератори контенту для перевірки навчальних матеріалів;
- викладачі з правами створення та оцінювання завдань;
- студенти з доступом до навчальних матеріалів та можливістю відправки робіт;
- спеціальні ролі для роботи з рейтингами та опитуваннями.

АС "my.kpi" реалізує специфічну модель прав для управління навчальним процесом [8]:

- члени методичних комісій з правами формування каталогів дисциплін;
- відповідальні за планування навчального процесу;
- куратори з правами управління вибором дисциплін;
- студенти з можливістю вибору дисциплін та перегляду індивідуальних планів;
- спеціальні ролі для управління розкладом та навчальними потоками.

Кожна з цих систем має не лише власний набір ролей, але й специфічні механізми їх призначення та зміни, власні правила валідації доступу та обмеження на комбінації ролей. Крім того, системи використовують різні підходи до зберігання та оновлення даних про права користувачів, що ускладнює синхронізацію цієї інформації між системами.

Така глибока автономність систем у питаннях управління доступом є результатом їх незалежного розвитку та необхідності забезпечення специфічних вимог безпеки для різних типів даних та процесів. Це створює суттєві виклики при спробах уніфікації системи автентифікації та авторизації на рівні всього університету. В умовах складної децентралізованої інфраструктури університету, спроба повної централізації управління правами доступу через систему "University ID" була б надмірно складною та потенційно небезпечною. Це пов'язано з тим, що

кожна підсистема має власні критично важливі механізми контролю доступу, які формувались відповідно до специфічних вимог безпеки та бізнес-процесів.

Враховуючи це, пропонується розділення відповідальності між компонентами системи:

АС "Ядро" виступає як:

- каталог довідників, що зберігає та надає доступ до базової інформації про структуру університету;
- джерело актуальних даних про користувачів;
- центральне сховище для синхронізації довідникової інформації між підсистемами.

Система "University ID" забезпечує (рис 2.1):

1. Єдину точку входу з уніфікованою автентифікацією користувачів.
2. Базову авторизацію, яка визначає фундаментальні ролі користувача:
  - надавач освітніх послуг (викладачі, адміністративний персонал);
  - отримувач освітніх послуг (студенти);
  - можливі комбінації цих ролей (наприклад, для аспірантів).
3. Механізм передачі інформації про аутентифікованого користувача та його базові ролі до підсистем.

При цьому кожна підсистема зберігає автономію в наступних аспектах:

- визначення та управління власною деталізованою системою прав доступу;
- контроль над своїми специфічними реєстрами та довідниками;
- реалізація унікальних бізнес-правил та обмежень;
- впровадження додаткових механізмів безпеки відповідно до своїх потреб.

### Архітектура системи University ID

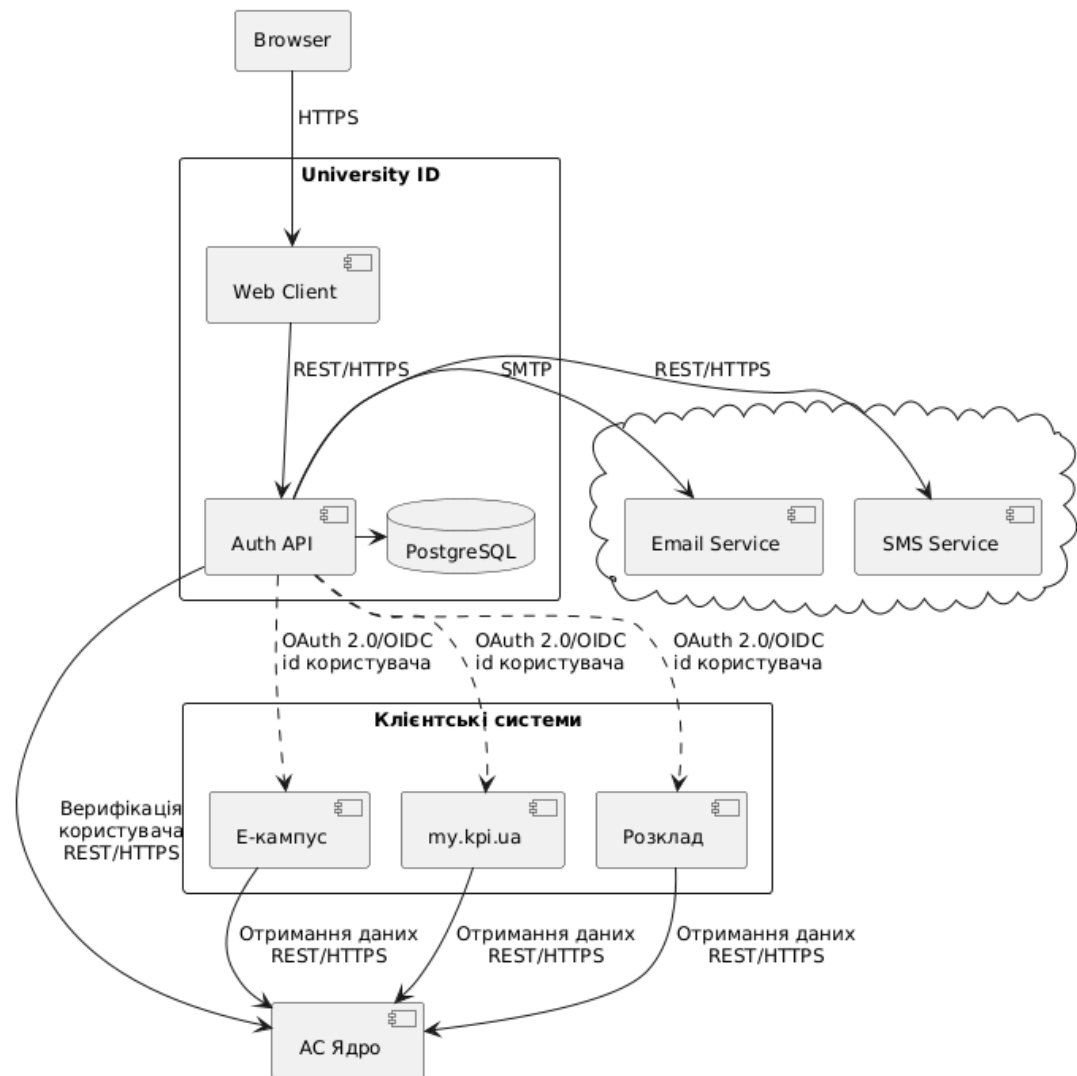


Рисунок 2.1 – Концептуальна схема архітектури системи

Така архітектура забезпечує:

1. Збереження існуючих механізмів безпеки в кожній підсистемі, які вже довели свою ефективність.
2. Уніфікований вхід користувачів через "University ID" без порушення автономності підсистем.
3. Можливість поетапної модернізації та уніфікації систем у майбутньому.

Взаємодія компонентів відбувається наступним чином (рис 2.1):

- користувач автентифікується через "University ID";

- "University ID" звертається до АС "Ядро" для отримання базової інформації про користувача;
- підсистеми отримують від "University ID" дані про автентифікованого користувача;
- за необхідності підсистеми можуть звертатися до АС "Ядро" для отримання додаткової довідникової інформації;
- на основі отриманих даних підсистеми застосовують власні механізми авторизації та контролю доступу.

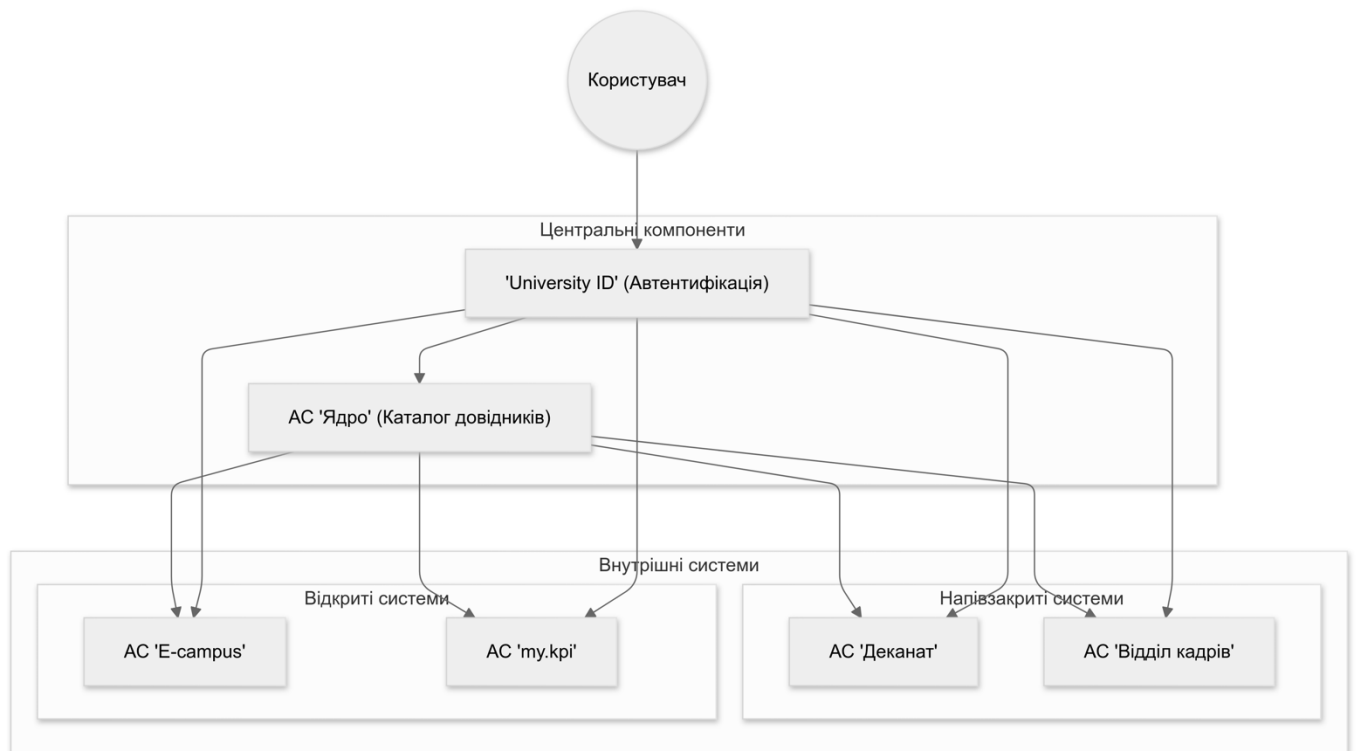


Рисунок 2.2 – Схема компонентів

Такий підхід забезпечує оптимальний баланс між централізацією процесу автентифікації та збереженням необхідної гнучкості у реалізації специфічних механізмів безпеки на рівні окремих підсистем.

### 2.3 Проектування компонентів системи

На основі проведеного аналізу вимог до системи єдиної точки входу та дослідження існуючих механізмів інтеграції інформаційних систем університету, було визначено ключові компоненти системи University ID. Особлива увага приділяється механізмам формування та валідації токенів доступу, оскільки саме вони забезпечують надійну та безпечну взаємодію між різними підсистемами університету. Відповідно до специфікації OAuth 2.0 [18], токен доступу є ключовим елементом авторизації, що представляє собою набір облікових даних, які використовуються для доступу до захищених ресурсів. В рамках проектування було прийнято рішення використовувати JWT (JSON Web Token) як основний механізм авторизації [6].

Цей вибір обумовлений кількома ключовими факторами, які відповідають вимогам специфікації OAuth 2.0 [18] щодо безпечного делегування доступу. По-перше, JWT забезпечує самодостатність токена, що дозволяє проводити валідацію без звернення до центральної бази даних, що особливо важливо в умовах розподіленої інфраструктури університету. По-друге, формат JWT підтримує включення додаткових claims, що дозволяє зберігати специфічну для університету інформацію про користувача та його права, що відповідає вимогам RFC 6749 щодо можливості розширення стандартних атрибутів доступу. По-третє, використання криптографічного підпису гарантує цілісність даних та захист від підробки, що є критичним вимогою для безпечної передачі токенів через незахищені канали зв'язку [18].

Додатковою перевагою є широка підтримка JWT в сучасних фреймворках та бібліотеках, що спрощує інтеграцію з існуючими та майбутніми системами університету. Це особливо важливо в контексті вимог специфікації OAuth 2.0 щодо сумісності та можливості розширення протоколу [18]. Важливим фактором у виборі JWT стала також можливість горизонтального масштабування системи - оскільки

токени є самодостатніми, можна розгортати додаткові сервери авторизації без необхідності синхронізації стану сесій між ними. Це особливо актуально для університетської системи, де кількість одночасних користувачів може досягати десятків тисяч.

Крім того, JWT дозволяє реалізувати механізм делегування прав доступу між різними підсистемами без необхідності повторної автентифікації користувача, що повністю відповідає основній меті OAuth 2.0 щодо спрощення процесу авторизації між різними сервісами [18]. Такий підхід значно покращує користувацький досвід при роботі з різними сервісами університету, зберігаючи при цьому високий рівень безпеки та контролю доступу.

### 2.3.1 Структура JWT токена

JWT токен в системі University ID складається з трьох основних частин: заголовка, корисного навантаження та підпису. Кожна з цих частин кодується в форматі Base64URL та розділяється крапками. Заголовок містить мета-інформацію про тип токена та алгоритм підпису. В системі використовується алгоритм RS256, що базується на асиметричному шифруванні та забезпечує надійний захист від модифікації токена.

Корисне навантаження (payload) токена містить набір тверджень (claims) про користувача та контекст його автентифікації. Стандартні (registered) claims включають унікальний ідентифікатор користувача (sub), час створення (iat) та закінчення дії токена (exp), ідентифікатор емітента (iss). Для потреб університетської системи було розширено стандартний набір claims специфічними для предметної області даними.

Ідентифікація користувача реалізується через унікальний ідентифікатор, який відповідає запису в АС "Ядро". Цей ідентифікатор дозволяє однозначно визначити користувача в усіх підсистемах університету. Додатково в токені зберігається ім'я

користувача для зручності логування та відображення в інтерфейсах клієнтських додатків.

Важливою частиною токена є інформація про ролі користувача. В системі University ID використовується ієрархічна модель ролей, яка відображає організаційну структуру університету. Наприклад, для студента вказується його належність до конкретної групи, факультету та форми навчання. Для викладачів фіксується їх приналежність до кафедри, наукові звання та адміністративні посади.

Специфічні для університету claims включають ідентифікатори структурних підрозділів (факультет, кафедра, група), до яких належить користувач. Ця інформація використовується клієнтськими системами для налаштування інтерфейсу та обмеження доступу до даних відповідного підрозділу. Наприклад, працівник деканату має доступ лише до інформації свого факультету.

Система прав доступу реалізована через окремий claim permissions, який містить перелік конкретних дозволів для різних підсистем. Такий підхід дозволяє гнучко налаштовувати доступ користувача до різних функцій та ресурсів. Права доступу формуються на основі ролей користувача та можуть бути додатково обмежені адміністративними політиками.

Для забезпечення цільового використання токена використовується claim audience (aud), який містить перелік систем, для яких призначений даний токен. Це дозволяє обмежити використання токена лише визначеним набором сервісів та запобігти його несанкціонованому використанню в інших системах.

Кожен токен має обмежений термін дії, який визначається значенням claim expires (exp). В системі University ID використовується короткий термін дії токенів (до 60 хвилин) для мінімізації ризиків при компрометації токена. Для подовження сесії користувача використовується механізм refresh токенів.

### 2.3.2 Валідація та обробка токенів

Процес валідації JWT токена включає кілька рівнів перевірок. В першу чергу перевіряється цілісність токена шляхом верифікації його цифрового підпису з використанням публічного ключа системи. Цей етап гарантує, що токен не був модифікований після його створення.

Наступним кроком є перевірка часових обмежень токена. Токен вважається дійсним лише в проміжку між часом створення (*iat*) та часом закінчення терміну дії (*exp*). Додатково перевіряється, чи не був токен відкликаний адміністратором системи через компрометацію або зміну прав доступу користувача (рис 2.3).

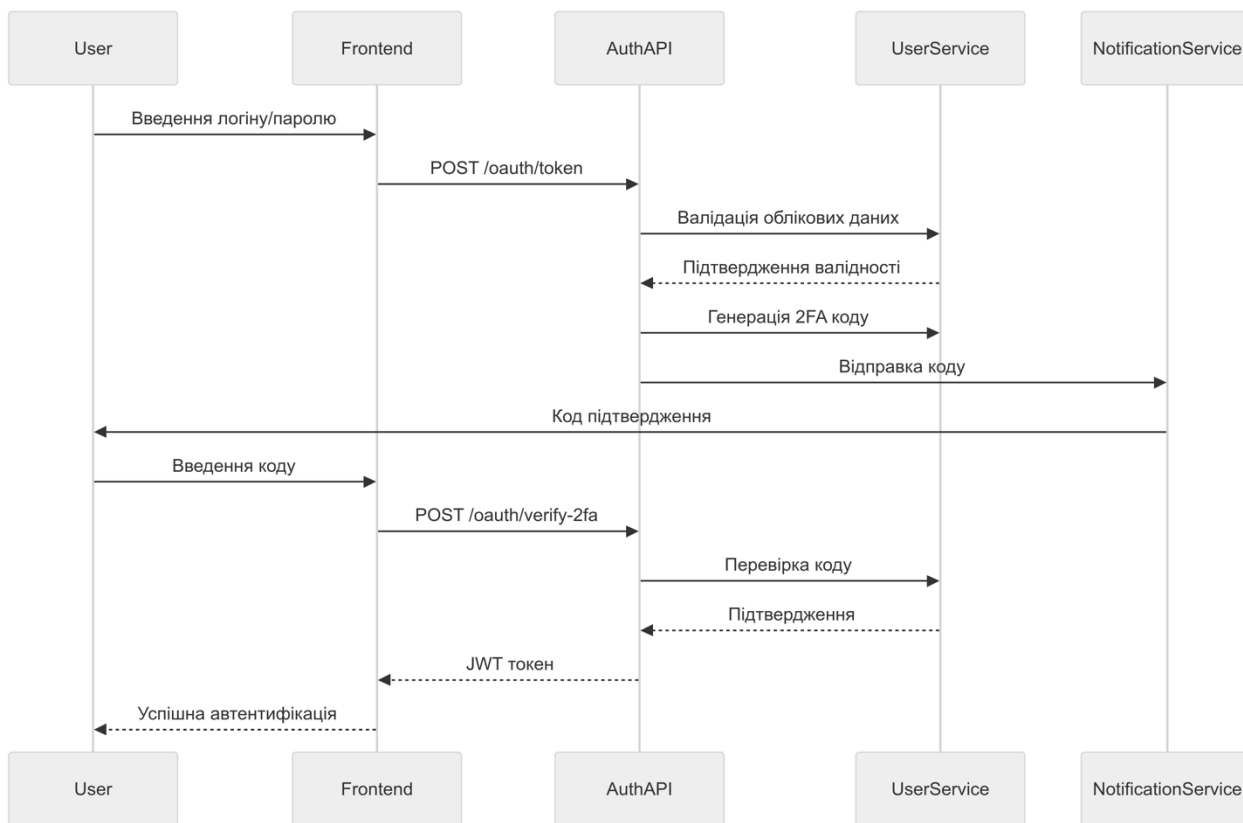


Рисунок 2.3 – Діаграма процесу входу в систему

При використанні токена клієнтськими системами виконується перевірка відповідності системи значенню в полі *audience*. Це забезпечує, що токен

використовується лише тими системами, для яких він був виданий. Додатково перевіряється джерело видачі токена через поле issuer, щоб гарантувати, що токен був виданий довіреним сервером авторизації.

Така структура токена та механізми його обробки забезпечують надійну автентифікацію та авторизацію користувачів в розподіленому середовищі інформаційних систем університету, зберігаючи при цьому гнучкість та зручність використання.

### 2.3.3 Серверна реалізація системи двофакторної автентифікації

Серверна частина системи автентифікації реалізує гнучкий механізм двофакторної автентифікації з підтримкою різних методів верифікації. У системі передбачено три основні канали для другого фактору автентифікації: TOTP, SMS та електронна пошта. Користувач може обрати один або декілька методів верифікації відповідно до своїх потреб та вимог безпеки (рис. 2.4).

В рамках реалізації механізму автентифікації за протоколом TOTP окрема увага приділяється безпеці при генерації та зберіганні секретних ключів. Система використовує криптографічно стійкий генератор випадкових чисел для створення унікального ключа для кожного користувача. Цей ключ зберігається в базі даних у зашифрованому вигляді з використанням алгоритму AES-256, а його розшифрування відбувається тільки в момент верифікації коду. Для додаткового захисту, система обмежує кількість спроб введення невірного коду.

Важливим аспектом реалізації TOTP є забезпечення надійної синхронізації часу між сервером та клієнтським пристроєм. Система враховує можливі розбіжності в часі та приймає коди з невеликим допустимим відхиленням, зазвичай  $\pm 1$  інтервал (30 секунд). Для випадків, коли користувач втрачає доступ до пристрою з аутентифікатором, система генерує набір одноразових резервних кодів, які зберігаються з використанням одностороннього хешування та можуть бути

використані для відновлення доступу. Ці коди автоматично анулюються після використання для запобігання їх повторного застосування.

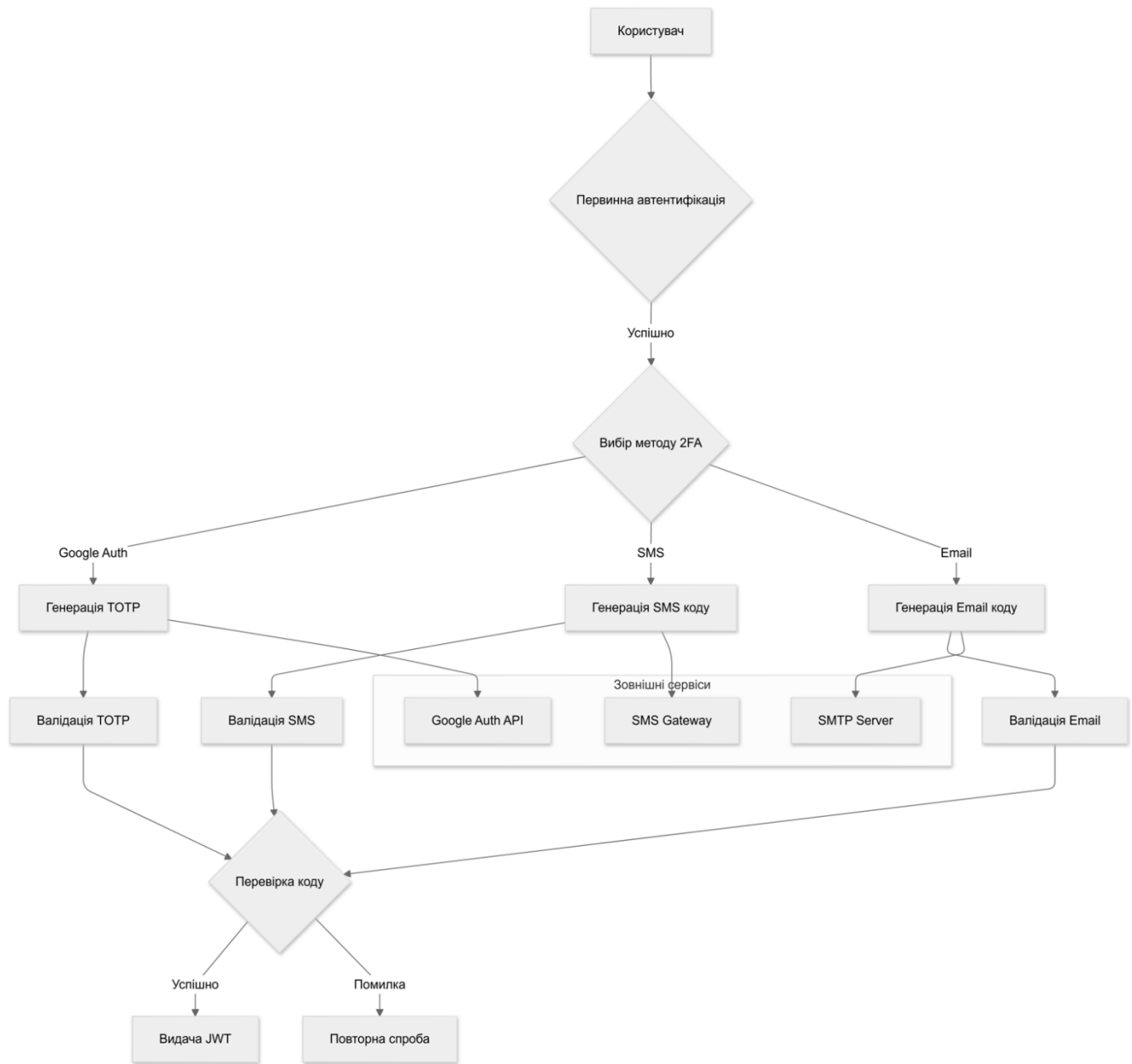


Рисунок 2.4 –Алгоритм двофакторної автентифікації

TOTP використовує стандарт TOTP (Time-based One-Time Password) для генерації тимчасових кодів [22]. При налаштуванні цього методу система генерує унікальний секретний ключ для користувача та QR-код, який можна відсканувати за

допомогою додатку Google Authenticator. Цей метод не потребує додаткової інфраструктури для доставки кодів і працює навіть без підключення до інтернету, що робить його особливо надійним.

SMS-автентифікація реалізована через інтеграцію із SMS-шлюзом. Система генерує випадковий код та надсилає його через захищений канал до SMS-провайдера. Важливою особливістю є механізм відстеження статусу доставки повідомлення та можливість повторної відправки у разі збоїв. Для підвищення безпеки використовується обмеження на кількість SMS, які можуть бути надіслані протягом визначеного періоду.

Верифікація через електронну пошту використовує захищене SMTP-з'єднання для надсилання кодів. Система використовує HTML-шаблони для форматування листів, що покращує користувацький досвід. Також впроваджено механізм відстеження відкриття листів та переходів за посиланнями для виявлення потенційних проблем з доставкою.

Система реалізує механізм вибору методу верифікації. Якщо користувач налаштував декілька методів, система може автоматично перемикатися між ними у разі недоступності основного методу. Наприклад, якщо виникли проблеми з доставкою SMS, система може запропонувати використати протокол TOTP або надіслати код на електронну пошту.

Для забезпечення безпеки всі коди верифікації мають обмежений термін дії - 5 хвилин для SMS та email, та 30 секунд для TOTP. Система також відслідковує аномальну активність, наприклад, велику кількість невдалих спроб введення коду або спроби верифікації з підозрілих IP-адрес.

Для додаткового захисту від автоматизованих атак система передбачає механізм адаптивної складності перевірки. Це може включати збільшення затримки між спробами введення коду, додаткову перевірку за допомогою CAPTCHA, або тимчасове блокування доступу з підозрілих IP-адрес. Система також веде детальний журнал всіх спроб автентифікації, включаючи інформацію про використаний метод

верифікації, IP-адресу, тип пристрою та результат спроби. У випадку виявлення критичних аномалій система автоматично сповіщає адміністраторів безпеки та може ініціювати додаткові заходи захисту. Послідовність верифікації зображено на рисунку 2.5.

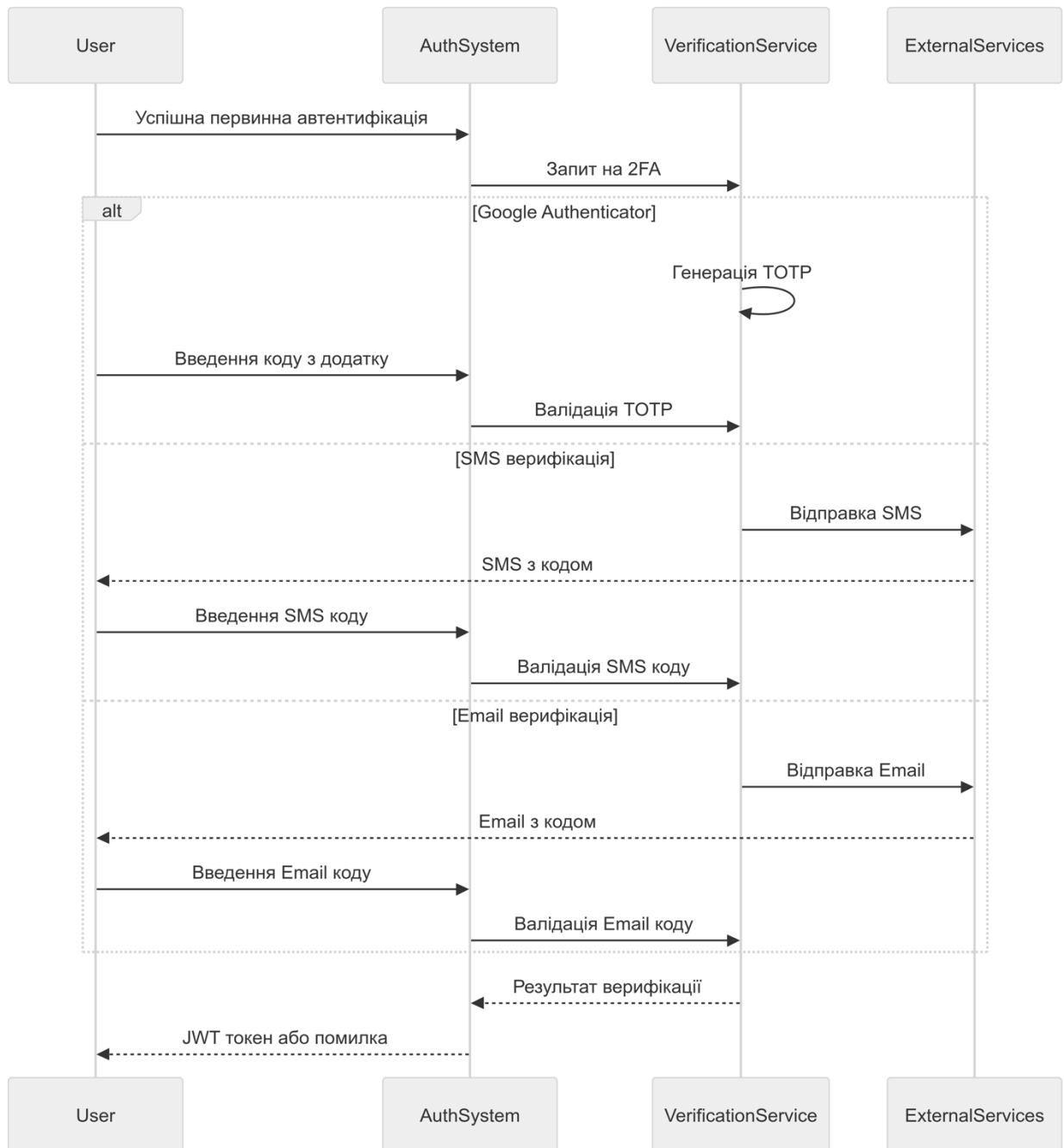


Рисунок 2.5 – Діаграма послідовності верифікації

Важливим аспектом є процес відновлення доступу у випадку втрати доступу до засобів двофакторної автентифікації. Система передбачає можливість генерації резервних кодів, які користувач може використати для входу в систему у надзвичайних ситуаціях. Ці коди зберігаються в зашифрованому вигляді та можуть бути використані лише один раз.

#### 2.3.4 Клієнтська реалізація системи автентифікації

Клієнтська частина системи автентифікації реалізована як веб-додаток, що забезпечує зручний та безпечний інтерфейс для автентифікації користувачів. Система підтримує два основних сценарії використання: прямий доступ користувача до системи та інтеграційний потік для сторонніх сервісів університету.

При прямому доступі користувача до сервісу автентифікації, після успішного проходження всіх етапів автентифікації, система відображає профіль користувача. Особистий кабінет містить наступні основні розділи:

1. Інформаційна панель з загальними даними користувача
2. Налаштування безпеки, де можна:
  - 2.3 Змінити пароль
  - 2.4 Налаштувати методи двофакторної автентифікації
  - 2.5 Переглянути історію входів
3. Управління методами двофакторної автентифікації:
  - 3.3 Налаштування TOTP
  - 3.4 Підтвердження номеру телефону для SMS
  - 3.5 Верифікація email для кодів підтвердження

Ці сценарії зображені на рисунку 2.6.

При інтеграційному сценарії, коли користувач перенаправляється з іншої системи університету, процес автентифікації відбувається наступним чином:

Зовнішня система формує URL з необхідними параметрами:

- `client_id`: ідентифікатор системи;
- `redirect_uri`: URL для повернення;
- `state`: параметр для захисту від CSRF атак.

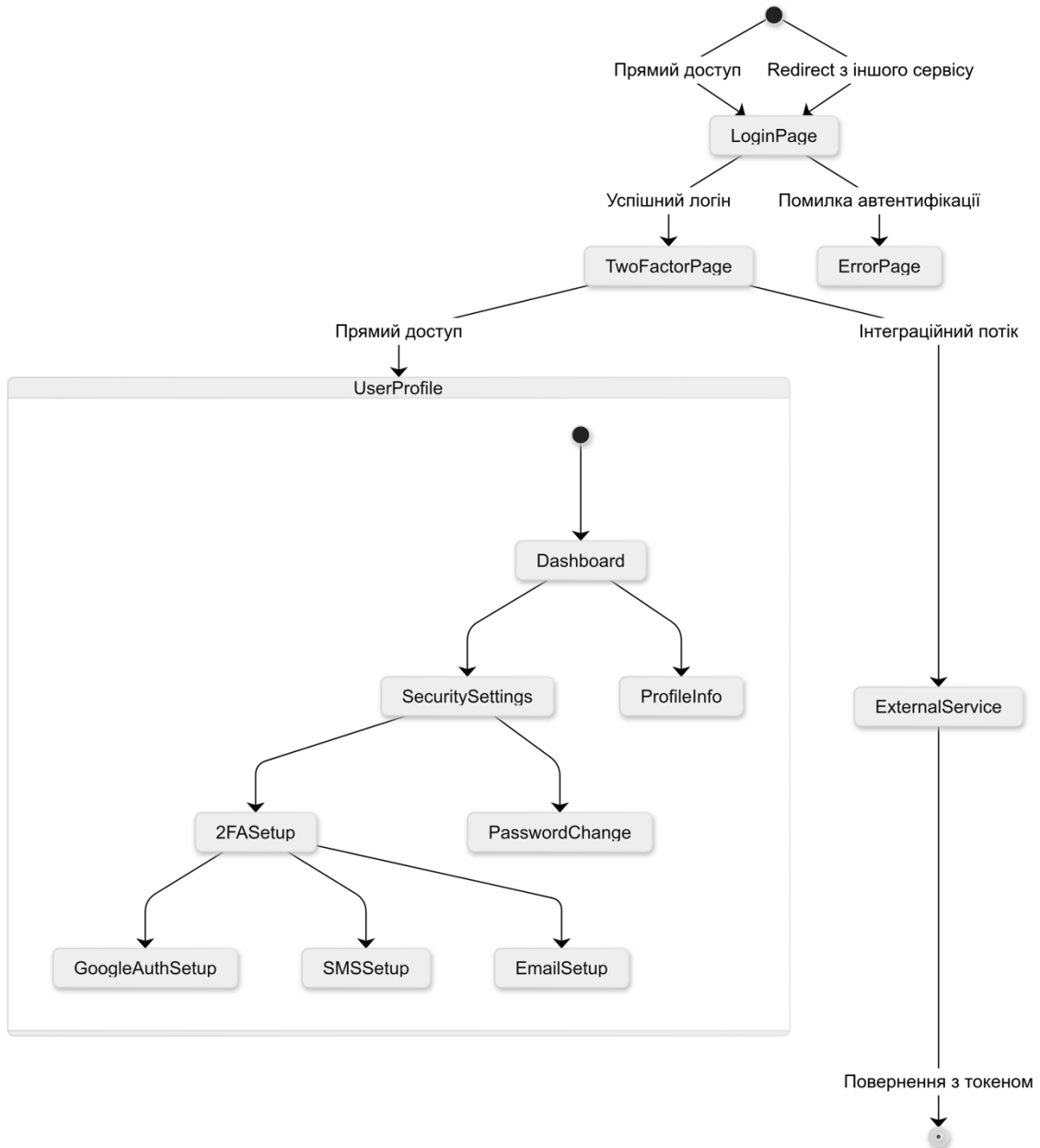


Рисунок 2.6 – Логіка інтерфейсу користувача

Користувач перенаправляється на сторінку автентифікації, де проходить всі необхідні етапи верифікації

Після успішної автентифікації система:

1. Генерує токен доступу
2. Формує URL для повернення з токеном
3. Автоматично перенаправляє користувача назад до системи-ініціатора

Інтерфейс системи розроблений з урахуванням сучасних вимог до юзабіліті та доступності. Всі форми мають валідацію на стороні клієнта, інформативні повідомлення про помилки та підказки для користувачів. Система також адаптивна та коректно відображається на різних пристроях.

Особлива увага приділена безпеці клієнтської частини:

- використання HTTPS для всіх комунікацій;
- захист від XSS атак;
- механізми запобігання CSRF атакам;
- автоматичний вихід при бездіяльності;
- безпечне зберігання токенів у браузері.

Така реалізація забезпечує зручний та безпечний процес автентифікації як для прямих користувачів системи, так і для інтегрованих сервісів університету.

### 2.3.5 Деталі технічної реалізації

Система реалізована з використанням сучасного технологічного стеку, що забезпечує надійність, безпеку та високу продуктивність. Серверна частина побудована на платформі .NET 8, що надає потужний фреймворк для створення веб-додатків. Для управління користувачами та їх ідентифікацією використовується Microsoft Identity, який забезпечує готові рішення для автентифікації та авторизації. Клієнтська частина розроблена з використанням Next.js, що дозволяє створювати швидкі та інтерактивні користувацькі інтерфейси. Для зберігання даних обрано PostgreSQL - надійну та масштабовану реляційну базу даних.

## Висновки до розділу

У другому розділі було детально розглянуто процес проектування системи уніфікованої ідентифікації користувачів "University ID". В результаті проведеної роботи було розроблено архітектурне рішення, яке враховує специфіку інформаційного середовища КПІ ім. Ігоря Сікорського та забезпечує надійну інтеграцію з існуючими системами.

Ключовим аспектом розробленого рішення є інтеграція з АС "Ядро" як джерелом достовірних даних про користувачів. Такий підхід дозволяє уникнути дублювання інформації та забезпечити консистентність даних у всіх підсистемах університету. При цьому система "University ID" зберігає автономність в управлінні сесіями, токенами та налаштуваннями безпеки, що забезпечує необхідну гнучкість та контроль над процесами автентифікації.

Особлива увага була приділена розробці механізмів двофакторної автентифікації з підтримкою різних методів верифікації - Google Authenticator, SMS та електронна пошта. Це дозволяє користувачам обирати найбільш зручний спосіб захисту свого облікового запису, при цьому система зберігає можливість легкого розширення для підтримки нових методів автентифікації в майбутньому.

Розроблена архітектура бази даних забезпечує ефективне зберігання та управління даними безпеки, включаючи історію входів, налаштування двофакторної автентифікації та токени доступу. Схема бази даних оптимізована для швидкого доступу до даних та підтримки масштабування системи при зростанні кількості користувачів.

Серверна частина системи реалізована з використанням сучасного стеку технологій (.NET 8, PostgreSQL) та слідує принципам чистої архітектури, що забезпечує:

- чітке розділення відповідальності між компонентами;

- легкість тестування та підтримки коду;
- можливість незалежного масштабування окремих компонентів;
- гнучкість у розширенні функціональності.

Клієнтська частина, розроблена на базі Next.js, надає зручний та інтуїтивно зрозумілий інтерфейс для користувачів, забезпечуючи при цьому необхідний рівень безпеки при роботі з конфіденційними даними. Особлива увага приділена підтримці різних сценаріїв використання - від прямого доступу користувачів до інтеграції зі сторонніми системами університету.

Запропоноване рішення створює надійний фундамент для подальшого розвитку єдиної системи автентифікації в університеті та може бути легко адаптоване під нові вимоги та технології. Система спроектована з урахуванням можливості поетапного впровадження, що дозволяє здійснювати міграцію існуючих сервісів без порушення їх роботи.

Таким чином, розроблена архітектура системи "University ID" повністю відповідає поставленим вимогам та забезпечує надійне підґрунтя для створення єдиного простору автентифікації в інформаційному середовищі університету.

### 3 КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

На основі проведеного аналізу предметної області та спроектованої архітектури виконується практична реалізація системи уніфікованої ідентифікації користувачів University ID для інформаційного середовища КПІ ім. Ігоря Сікорського. Розробка програмного забезпечення здійснюється з використанням сучасного технологічного стеку, що забезпечує надійність, безпеку та високу продуктивність системи.

В основі реалізації лежить створення серверної частини на базі .NET 8[13] та ASP.NET Core[15], яка забезпечує всю функціональність автентифікації та авторизації. Паралельно розробляється клієнтський веб-додаток на Next.js[16] для зручної взаємодії користувачів із системою. Важливим аспектом є реалізація інтеграційних механізмів для взаємодії з АС "Ядро" та іншими системами університету, а також впровадження надійної системи двофакторної автентифікації та механізмів безпечного зберігання та обробки токенів доступу.

Технічна реалізація охоплює множину аспектів, від структурування програмного коду та організації модулів до реалізації складної бізнес-логіки та механізмів обробки даних. Значна увага приділяється роботі з базою даних PostgreSQL [12], забезпеченню необхідного рівня безпеки та захисту даних. Окремим напрямком є створення інтуїтивного та зручного користувацького інтерфейсу, а також налагодження процесів розгортання та контейнеризації додатку.

Важливою складовою розробки є забезпечення високої якості програмного забезпечення через впровадження комплексного тестування, включаючи модульні та інтеграційні тести. Система розробляється з урахуванням необхідності постійного моніторингу та логування роботи, а також можливості горизонтального масштабування для забезпечення стабільної роботи при зростанні навантаження. Практики безперервної інтеграції та розгортання (CI/CD) забезпечують надійний процес оновлення та розширення функціональності системи.

Практична реалізація включає розгортання системи в інфраструктурі університету, що вимагає ретельного налаштування серверів, конфігурації мережевої взаємодії та інтеграції з існуючими сервісами. При цьому вирішуються технічні виклики, пов'язані з особливостями наявної інфраструктури та вимогами до безпеки даних.

Результатом проведеної розробки стає повноцінна система уніфікованої ідентифікації користувачів, що відповідає сучасним стандартам безпеки та зручності використання. Система забезпечує надійну автентифікацію та авторизацію користувачів у масштабах всього університету, створюючи єдиний захищений простір для доступу до інформаційних ресурсів КПП ім. Ігоря Сікорського.

### 3.1 Опис використаних технологій

#### 3.1.1 Реалізація серверної частини на базі .NET та ASP.NET Core

В якості основної платформи для розробки серверної частини системи University ID було обрано .NET 8 та фреймворк ASP.NET Core. Цей вибір обумовлений потребою у високопродуктивній та надійній платформі, що здатна обробляти значну кількість одночасних запитів на автентифікацію та забезпечувати безпечну роботу з конфіденційними даними користувачів.

Архітектура серверної частини побудована за принципами чистої архітектури (Clean Architecture) з чітким розділенням на шари та дотриманням принципів SOLID. Система розділена на кілька основних проєктів, кожен з яких відповідає за свій аспект функціональності.

Основний проєкт UniversityID.API представляє собою веб-додаток ASP.NET Core, який надає REST API для всіх операцій, пов'язаних з автентифікацією та авторизацією користувачів. Цей компонент обробляє вхідні HTTP-запити, валідує вхідні дані та координує роботу інших компонентів системи. Тут реалізовані всі необхідні контролери для обробки процесів автентифікації, управління токенами та

двофакторної верифікації. Важливою особливістю є використання вбудованих механізмів ASP.NET Core для забезпечення безпеки, включаючи захист від CSRF-атак, XSS-вразливостей та інших типових загроз веб-додатків.

В проєкті UniversityID.Core зосереджена вся доменна логіка системи. Тут визначені основні бізнес-сутності, такі як користувач, сесія, токен доступу, та описані інтерфейси для взаємодії між різними частинами системи. Використання інтерфейсів забезпечує слабе зв'язування компонентів та полегшує тестування. Цей проєкт не має залежностей від конкретних технологій, що робить його стійким до змін в інфраструктурі.

Проєкт UniversityID.Infrastructure містить всі реалізації взаємодії з зовнішніми системами та сервісами. Тут реалізована робота з базою даних PostgreSQL через Entity Framework Core, інтеграція з АС "Ядро" через відповідні API, взаємодія з системами відправки email та SMS для двофакторної автентифікації. Важливим аспектом є використання патерну Repository для абстрагування роботи з даними та Unit of Work для забезпечення транзакційності операцій.

Для роботи з JWT токенами використовуються можливості бібліотеки Microsoft.AspNetCore.Authentication.JwtBearer, яка забезпечує генерацію, валідацію та управління токенами доступу. Реалізовано механізми оновлення токенів, їх відкликання та перевірки терміну дії.

В системі активно використовується вбудована в .NET система впровадження залежностей (Dependency Injection), що дозволяє легко керувати життєвим циклом об'єктів та їх взаємозв'язками. Це особливо важливо для реалізації різних стратегій автентифікації та авторизації.

Для забезпечення надійності та відмовостійкості системи використовуються вбудовані механізми .NET для обробки помилок та логування. Реалізовано глобальну обробку винятків, що дозволяє централізовано обробляти всі помилки та надавати клієнтам зрозумілі повідомлення про проблеми. Для логування

використовується Serilog, що забезпечує гнучке налаштування рівнів логування та форматів виведення інформації.

Особлива увага приділена продуктивності системи. Використовуються асинхронні методи обробки запитів, що дозволяє ефективно використовувати ресурси сервера при високому навантаженні. Реалізовано кешування часто використовуваних даних за допомогою IMemoryCache, що знижує навантаження на базу даних.

Система розгортається в контейнерах Docker, що забезпечує однакове середовище виконання на різних серверах та спрощує процес масштабування. Конфігурація системи винесена у файли appsettings.json та змінні середовища, що дозволяє легко налаштовувати систему для різних середовищ розгортання.

Завдяки використанню можливостей .NET 8 та ASP.NET Core, система University ID демонструє високу продуктивність, надійність та безпеку, забезпечуючи стабільну роботу служби автентифікації для всієї інформаційної інфраструктури університету.

### 3.1.2 Використання бази даних в системі University ID

Для забезпечення надійного зберігання та управління даними в системі University ID використовується реляційна система управління базами даних PostgreSQL. Вибір PostgreSQL обумовлений її надійністю, продуктивністю та широкими можливостями для забезпечення цілісності даних, що критично важливо для системи автентифікації.

Для забезпечення високої продуктивності впроваджено ряд оптимізацій на рівні бази даних. Створено необхідні індекси для прискорення пошуку за часто використовуваними полями, зокрема для пошуку користувачів за ідентифікаторами та перевірки токенів доступу. Налаштовано відповідні налаштування PostgreSQL для оптимального використання доступних ресурсів сервера.

Особлива увага приділена безпеці даних. Усі конфіденційні дані, такі як секретні ключі та хеші токенів, зберігаються в зашифрованому вигляді. Використовуються вбудовані механізми PostgreSQL для шифрування та хешування даних. Доступ до бази даних обмежений мінімально необхідним набором привілеїв для кожного компонента системи.

Для забезпечення цілісності даних використовуються транзакції та механізм Unit of Work. Всі операції, що змінюють стан кількох пов'язаних сутностей, виконуються в межах однієї транзакції, що гарантує узгодженість даних навіть у випадку збоїв. Реалізовано механізм відстеження змін (Change Tracking) для аудиту модифікацій критично важливих даних.

Для забезпечення відмовостійкості налаштовано регулярне резервне копіювання бази даних. Розроблено стратегію відновлення даних у випадку збоїв. Система моніторингу відстежує продуктивність бази даних та автоматично сповіщає адміністраторів про потенційні проблеми.

В системі реалізовано механізм очищення застарілих даних (data cleanup), який автоматично видаляє прострочені токени, невикористані коди верифікації та інші тимчасові дані. Це запобігає надмірному розростанню бази даних та підтримує її продуктивність на належному рівні.

Для розробки та тестування використовуються окремі instances бази даних. Тестове середовище регулярно оновлюється замаскованими даними з продуктивного середовища, що дозволяє проводити тестування в умовах, максимально наближених до реальних, але без ризику компрометації конфіденційних даних.

Система підтримує можливість горизонтального масштабування бази даних. При необхідності можливе налаштування реплікації для розподілу навантаження між кількома серверами баз даних. Архітектура системи передбачає можливість майбутнього переходу на шардинг для обробки надвеликих об'ємів даних.

Для полегшення діагностики проблем налаштовано детальне логування всіх критичних операцій з базою даних. Логи містять інформацію про тривалість виконання запитів, що дозволяє виявляти та оптимізувати повільні операції. При цьому конфіденційні дані в логах автоматично маскуються.

Взаємодія з базою даних інкапсульована на рівні репозиторіїв, що забезпечує єдиний інтерфейс доступу до даних та можливість легкої заміни конкретної реалізації сховища даних без змін у бізнес-логіці системи. Такий підхід також спрощує процес тестування, дозволяючи легко підміняти реальну базу даних mock-об'єктами при модульному тестуванні.

### 3.1.3 Реалізація клієнтської частини системи University ID

Клієнтська частина системи University ID реалізована з використанням сучасного фреймворку Next.js версії 14, що забезпечує оптимальну продуктивність та зручність розробки. Next.js було обрано через його потужні можливості для створення серверного рендерингу (SSR), що критично важливо для швидкого початкового завантаження сторінок та SEO-оптимізації.

Архітектура клієнтської частини побудована за принципами компонентного підходу, де кожен елемент інтерфейсу є незалежним та перевикористовуваним компонентом. Це забезпечує модульність системи та полегшує її подальшу підтримку та розширення. Використання TypeScript додає строгу типізацію, що значно зменшує кількість потенційних помилок під час розробки та покращує якість коду.

Для стилізації компонентів використовується Tailwind CSS, що дозволяє створювати адаптивний та естетичний інтерфейс без написання власних CSS-стилів. Tailwind забезпечує уніфікований підхід до дизайну через систему утилітарних класів, що прискорює розробку та гарантує консистентність візуального

оформлення. Важливою перевагою є автоматичне очищення невикористовуваних стилів при збірці проекту, що оптимізує розмір фінального пакета.

В системі реалізовано кілька ключових користувацьких потоків.

Процес автентифікації користувачів включає форму входу з валідацією даних на клієнтській стороні, обробку різних сценаріїв помилок та підтримку двофакторної автентифікації. Інтерфейс адаптивно реагує на різні стани процесу автентифікації, надаючи користувачу чіткий зворотний зв'язок про поточний статус операції.

Панель управління безпекою надає інтерфейс для налаштування методів двофакторної автентифікації, перегляду історії входів та управління активними сесіями. Реалізовано інтерактивні форми для налаштування Google Authenticator, підтвердження номеру телефону та email-адреси.

Оптимізація продуктивності досягається через:

- використання вбудованого в Next.js механізму розділення коду (code splitting);
- автоматичну оптимізацію зображень через Next.js Image компонент;
- попереднє завантаження критичних ресурсів;
- кешування компонентів та результатів запитів;
- ліниве завантаження другорядних елементів інтерфейсу.

Безпека на клієнтській стороні забезпечується через:

- захист від XSS-атак за допомогою вбудованих механізмів React;
- безпечне зберігання токенів у захищених HTTP-only cookies;
- валідацію всіх користувацьких вводів;
- автоматичний вихід при неактивності;
- захист від CSRF-атак.

Для моніторингу роботи клієнтської частини впроваджено систему логування помилок та відстеження користувацьких дій. Це дозволяє оперативно виявляти та виправляти проблеми у роботі інтерфейсу. При цьому забезпечується анонімність користувачів та захист їх персональних даних.

Система збірки налаштована таким чином, щоб забезпечити оптимальний розмір бандлу та швидке завантаження сторінок. Використовується автоматична оптимізація зображень, мінімізація CSS та JavaScript.

Завдяки використанню сучасних технологій та підходів до розробки, клієнтська частина University ID забезпечує швидкий, надійний та зручний інтерфейс для взаємодії користувачів з системою автентифікації, одночасно підтримуючи високі стандарти безпеки та доступності.

### 3.2 Розробка бази даних

Для забезпечення ефективного зберігання та управління даними в системі University ID розроблено реляційну базу даних на основі PostgreSQL. Структура бази даних спроектована з урахуванням всіх вимог до системи єдиної автентифікації та необхідності забезпечення цілісності даних про користувачів та їхні сесії.

База даних системи повинна забезпечувати надійне зберігання інформації про користувачів, їхні методи автентифікації, активні сесії та токени доступу. Особлива увага при проектуванні приділялася аспектам безпеки та продуктивності, враховуючи що система працює з конфіденційними даними та повинна обробляти значну кількість запитів на автентифікацію.

Основними сутностями в базі даних є користувачі, їхні методи двофакторної автентифікації, сесії, токени доступу та історія подій безпеки. Кожна з цих сутностей має свій набір атрибутів та зв'язків, що забезпечують повноцінну роботу системи автентифікації. Важливим аспектом є підтримка зв'язку з АС "Ядро" через

відповідні зовнішні ключі, що дозволяє синхронізувати дані про користувачів між системами.

В структурі бази даних передбачено механізми для:

- зберігання та управління обліковими записами користувачів;
- підтримки різних методів двофакторної автентифікації;
- відстеження активних сесій користувачів;
- управління токенами доступу та оновлення;
- збереження історії автентифікації та подій безпеки;
- налаштування клієнтських додатків та їх областей доступу.

Схема бази даних зображено в Додатку А.

Опишемо окремі таблиці в базі даних:

Таблиця 3.1 -- Опис таблиці Users (Користувачі)

Назва поля	Тип даних	Обмеження	Опис
id	int	PK	Ідентифікатор з АС "Ядро"
username	string	NOT NULL	Ім'я користувача з АС "Ядро"
email	string	NOT NULL	Електронна пошта з АС "Ядро"
last_login	timestamp		Час останнього входу
is_active	boolean	NOT NULL	Статус активності облікового запису
two_factor_required	boolean	NOT NULL	Необхідність двофакторної автентифікації

Таблиця 3.2 -- Опис таблиці UserSessions (Сесії користувачів)

Назва поля	Тип даних	Обмеження	Опис
id	uuid	PK	Унікальний ідентифікатор сесії
user_id	int	FK	Зовнішній ключ на таблицю Users
session_token	string	NOT NULL	Токен сесії
ip_address	string	NOT NULL	IP-адреса користувача
user_agent	string	NOT NULL	Інформація про браузер/пристрій
expires_at	timestamp	NOT NULL	Час закінчення сесії
is_active	boolean	NOT NULL	Статус активності сесії
created_at	timestamp	NOT NULL	Час створення сесії

Таблиця 3.3 -- Опис таблиці TwoFactorMethods (Методи двофакторної автентифікації)

Назва поля	Тип даних	Обмеження	Опис
id	uuid	PK	Унікальний ідентифікатор методу
user_id	int	FK	Зовнішній ключ на таблицю Users
method_type	string	NOT NULL	Тип методу (google_auth/sms/email)
identifier	string		Телефон або email
is_primary	boolean	NOT NULL	Чи є метод основним
secret_key	string		Секретний ключ для Google Auth
is_confirmed	boolean	NOT NULL	Статус підтвердження методу
setup_completed_at	timestamp		Час завершення налаштування
created_at	timestamp	NOT NULL	Час створення запису

Таблиця 3.4 -- Опис таблиці LoginHistory (Історія входів)

Назва поля	Тип даних	Обмеження	Опис
id	uuid	PK	Унікальний ідентифікатор запису
user_id	int	FK	Зовнішній ключ на таблицю Users
login_time	timestamp	NOT NULL	Час спроби входу
ip_address	string	NOT NULL	IP-адреса
user_agent	string	NOT NULL	Інформація про браузер/пристрій
success	boolean	NOT NULL	Успішність входу
failure_reason	string		Причина невдачі
auth_method	string	NOT NULL	Метод автентифікації
created_at	timestamp	NOT NULL	Час створення запису

Таблиця 3.5 -- Опис таблиці AuthClients (Клієнтські додатки)

Назва поля	Тип даних	Обмеження	Опис
id	uuid	PK	Унікальний ідентифікатор клієнта
client_id	string	UNIQUE	Публічний ідентифікатор клієнта
client_name	string	NOT NULL	Назва клієнтського додатку
client_secret_hash	string	NOT NULL	Хеш секретного ключа
redirect_uris	string[]	NOT NULL	Дозволені URL перенаправлення
is_active	boolean	NOT NULL	Статус активності клієнта
created_at	timestamp	NOT NULL	Час створення
updated_at	timestamp	NOT NULL	Час останнього оновлення

Таблиця 3.6 -- Опис таблиці ClientScores (Області доступу клієнтів)

Назва поля	Тип даних	Обмеження	Опис
id	uuid	PK	Унікальний ідентифікатор
client_id	uuid	FK	Зовнішній ключ на таблицю AuthClients
scope_name	string	NOT NULL	Назва області доступу

Таблиця 3.7 -- Опис таблиці RefreshTokens (Токени оновлення)

Назва поля	Тип даних	Обмеження	Опис
id	uuid	PK	Унікальний ідентифікатор токена
user_id	int	FK	Зовнішній ключ на таблицю Users
token_hash	string	NOT NULL	Хеш токена оновлення
client_id	uuid	FK	Зовнішній ключ на таблицю AuthClients
expires_at	timestamp	NOT NULL	Час закінчення дії токена
is_revoked	boolean	NOT NULL	Статус відкликання
replaced_by	string		Ідентифікатор нового токена
created_at	timestamp	NOT NULL	Час створення

Таблиця 3.8 -- Опис таблиці TwoFactorCodes (Коди двофакторної автентифікації)

Назва поля	Тип даних	Обмеження	Опис
id	uuid	PK	Унікальний ідентифікатор коду
user_id	int	FK	Зовнішній ключ на таблицю Users
method_id	uuid	FK	Зовнішній ключ на TwoFactorMethods
code_hash	string	NOT NULL	Хеш коду підтвердження
method_type	string	NOT NULL	Тип методу двофакторної автентифікації
expires_at	timestamp	NOT NULL	Час закінчення дії коду
attempts_count	int	NOT NULL	Кількість спроб використання
is_used	boolean	NOT NULL	Статус використання
created_at	timestamp	NOT NULL	Час створення

Таблиця 3.9 -- Опис таблиці SecurityEvents (Події безпеки)

Назва поля	Тип даних	Обмеження	Опис
id	uuid	PK	Унікальний ідентифікатор події
user_id	int	FK	Зовнішній ключ на таблицю Users
event_type	string	NOT NULL	Тип події безпеки
description	string	NOT NULL	Опис події
metadata	json		Додаткові дані про подію
created_at	timestamp	NOT NULL	Час створення запису

Між таблицями встановлені наступні зв'язки:

- Users має зв'язок "один до багатьох" з таблицями UserSessions, LoginHistory, TwoFactorMethods, RefreshTokens, TwoFactorCodes та SecurityEvents;
- AuthClients має зв'язок "один до багатьох" з таблицею ClientScopes;
- TwoFactorMethods має зв'язок "один до багатьох" з таблицею TwoFactorCodes.

### 3.3 Розробка серверної частини

Серверна частина системи University ID реалізована з використанням об'єктно-орієнтованого підходу та принципів чистої архітектури. Центральним компонентом системи є AuthController, який координує всі процеси автентифікації та взаємодіє з різними сервісами для забезпечення необхідної функціональності.

AuthController виступає точкою входу для всіх запитів, пов'язаних з автентифікацією. Він обробляє базову автентифікацію користувачів, налаштування та валідацію двофакторної автентифікації, а також перевірку поточних сесій. Контролер використовує впровадження залежностей для взаємодії з іншими компонентами системи, що забезпечує гнучкість та тестованість коду.

UserAccountService відповідає за основну логіку роботи з обліковими записами користувачів. Цей сервіс виконує автентифікацію користувачів на основі наданих облікових даних, генерує необхідні токени та ідентифікатори сесій, а також надає доступ до інформації про користувачів. Сервіс тісно взаємодіє з доменною моделлю User, яка містить всю необхідну інформацію про користувача.

Клас User представляє собою доенну модель користувача системи та містить основні атрибути, такі як ідентифікатор, повне ім'я, фотографія, особисте кредо, ім'я користувача, електронна пошта, телефон та інформація про групу. Ця модель використовується для передачі даних між різними компонентами системи.

`AuthClientService` забезпечує роботу з клієнтськими додатками, які можуть використовувати систему автентифікації. Він працює з моделлю `AuthClient`, яка містить інформацію про зареєстровані клієнтські додатки, включаючи їх ідентифікатори, назви та дозволені URL для перенаправлення.

`SecondFAService` реалізує логіку двофакторної автентифікації. Цей сервіс взаємодіє з `UniIdRepository` для зберігання та отримання інформації про методи двофакторної автентифікації користувачів. Він надає можливості для включення двофакторної автентифікації, валідації кодів та генерації нових кодів підтвердження.

Для відправки повідомлень використовується система сервісів, що реалізують інтерфейс `INotificationService`. `EmailService` та `SMSService` є конкретними реалізаціями цього інтерфейсу, що забезпечують відправку повідомлень через електронну пошту та SMS відповідно. Такий підхід дозволяє легко розширювати систему новими каналами комунікації.

`UniIdRepository` забезпечує взаємодію з базою даних, надаючи методи для отримання та оновлення інформації про користувачів, їхні методи двофакторної автентифікації та історію входів. Репозиторій інкапсулює всю логіку роботи з базою даних, забезпечуючи єдиний інтерфейс для доступу до даних.

Взаємодія між компонентами системи побудована таким чином, щоб забезпечити слабке зв'язування та високу модульність. `AuthController` залежить від абстракцій сервісів, а не від їх конкретних реалізацій, що відповідає принципу інверсії залежностей. Всі залежності впроваджуються через конструктор, що спрощує тестування та забезпечує гнучкість при заміні реалізацій.

Система обробки запитів побудована з урахуванням необхідності асинхронної роботи. Всі операції, що можуть займати значний час (наприклад, відправка повідомлень або звернення до бази даних), реалізовані як асинхронні методи, що забезпечує ефективне використання ресурсів сервера при високому навантаженні.

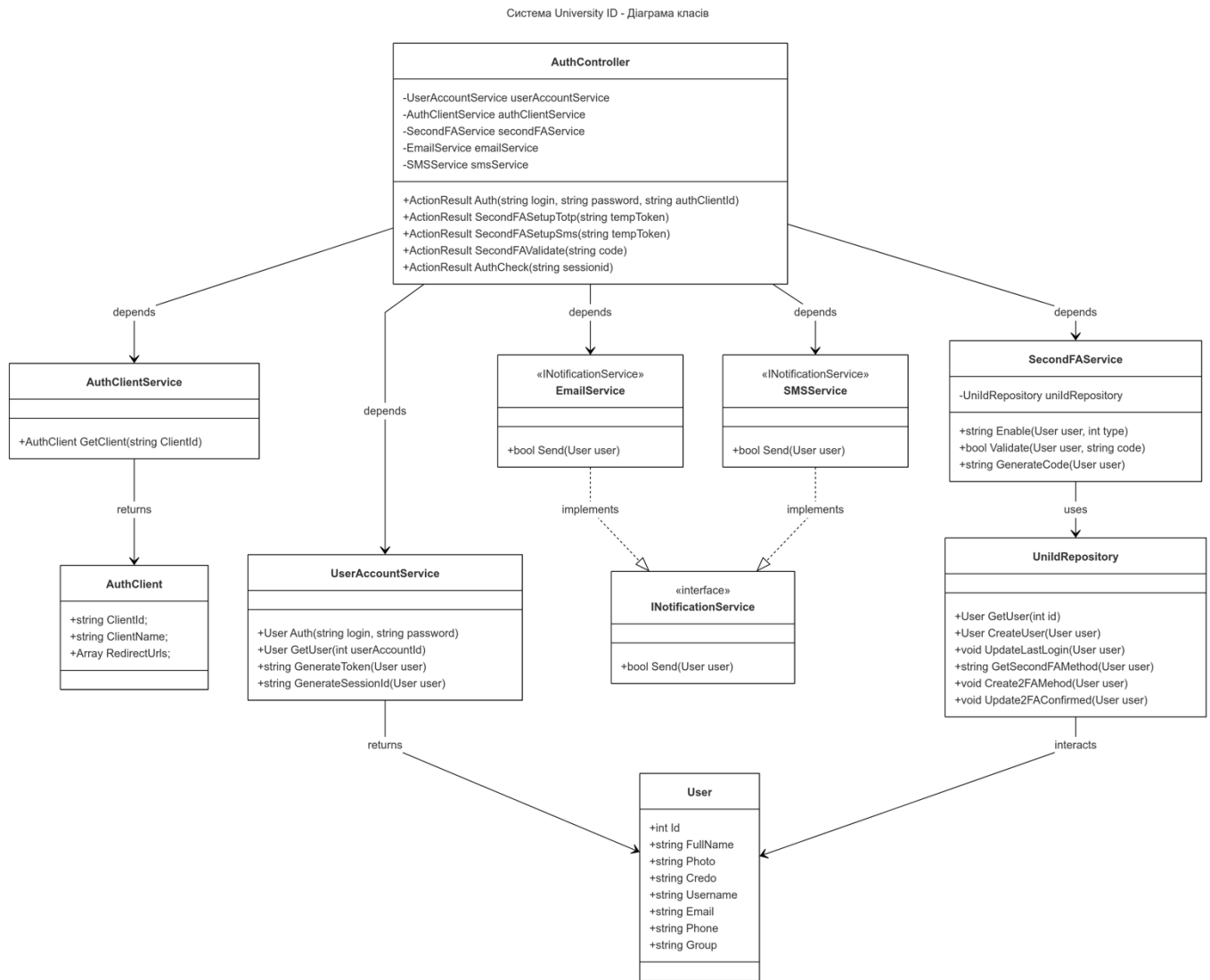


Рисунок 3.1 – Діаграма класів

Безпека системи забезпечується на різних рівнях, починаючи від валідації вхідних даних у контролері та закінчуючи безпечним зберіганням конфіденційної інформації в базі даних. Всі критичні операції логуються для подальшого аудиту, а система двофакторної автентифікації забезпечує додатковий рівень захисту облікових записів користувачів.

## Висновки до розділу

У третьому розділі було детально розглянуто процес конструювання програмного забезпечення системи уніфікованої ідентифікації користувачів University ID. В результаті виконаної роботи було створено надійне та масштабоване рішення, що відповідає всім визначеним вимогам та забезпечує ефективну автентифікацію користувачів в інформаційному середовищі університету.

Серверна частина системи реалізована на базі сучасної платформи .NET 8 та фреймворку ASP.NET Core, що забезпечує високу продуктивність та надійність. Архітектура серверної частини побудована за принципами чистої архітектури з чітким розділенням на шари, що спрощує подальшу підтримку та розширення системи. Впровадження паттернів проектування, таких як Repository та Unit of Work, забезпечило гнучкість роботи з даними та можливість легкої заміни компонентів системи.

Клієнтська частина розроблена з використанням фреймворку Next.js та бібліотеки Tailwind CSS, що дозволило створити сучасний, адаптивний та зручний користувацький інтерфейс. Особлива увага приділена оптимізації продуктивності через використання серверного рендерингу та ефективного управління станом додатку. Реалізовані інтерфейси забезпечують інтуїтивно зрозумілу взаємодію користувачів з системою при виконанні всіх основних операцій.

База даних системи, реалізована на PostgreSQL, спроектована з урахуванням вимог до безпеки та продуктивності. Структура бази даних забезпечує ефективне зберігання та управління інформацією про користувачів, їхні сесії та методи автентифікації. Впроваджені механізми індексації та оптимізації запитів гарантують високу швидкість роботи системи навіть при значному навантаженні.

Особлива увага в процесі розробки була приділена питанням безпеки. Реалізовано надійний механізм двофакторної автентифікації з підтримкою різних методів верифікації, включаючи SMS та електронну пошту. Всі критичні дані

зберігаються в зашифрованому вигляді, а система токенів забезпечує безпечну передачу даних між компонентами.

Важливим досягненням є успішна реалізація інтеграції з АС "Ядро" як джерелом достовірних даних про користувачів. Це дозволило забезпечити актуальність інформації про користувачів та їхні ролі в масштабах всього університету. Розроблені механізми синхронізації гарантують коректну роботу системи навіть при тимчасовій недоступності зовнішніх сервісів.

Таким чином, в результаті виконаної розробки створено повноцінну систему уніфікованої ідентифікації користувачів, яка не лише відповідає поточним потребам університету, але й має потенціал для подальшого розвитку та масштабування. Використання сучасних технологій та архітектурних рішень забезпечило створення надійної та ефективної системи, що може стати основою для подальшого розвитку цифрової інфраструктури університету.

## 4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 4.1 Аналіз якості програмного забезпечення

В даному розділі розглядаються питання забезпечення якості та тестування системи уніфікованої ідентифікації користувачів University ID. Оскільки система відповідає за критично важливі процеси автентифікації та авторизації користувачів в інформаційному середовищі університету, особлива увага приділяється всебічній перевірці її надійності, безпеки та відповідності функціональним вимогам.

Метою тестування є перевірка коректності роботи всіх компонентів системи, включаючи процеси автентифікації, управління токенами доступу, двофакторну верифікацію та інтеграцію з АС "Ядро". Важливим аспектом є валідація механізмів безпеки системи, зокрема перевірка надійності зберігання конфіденційних даних, захищеності каналів комунікації та стійкості до потенційних атак.

В рамках тестування інтеграційних можливостей системи особлива увага приділяється взаємодії з двома ключовими системами університету - АС "Е-кампус" та системою "Розклад". Ці системи обрані як пілотні для демонстрації можливостей інтеграції, оскільки вони активно використовуються всіма категоріями користувачів та мають різні вимоги до автентифікації та авторизації. Успішна інтеграція з цими системами слугуватиме підтвердженням ефективності розробленого рішення та створить основу для подальшого розширення єдиного простору автентифікації на інші системи університету.

Слід зазначити, що представлені в роботі інтерфейси користувача є прототипами і не повністю відповідають офіційному брендбуку КПІ ім. Ігоря Сікорського. Проте це жодним чином не впливає на функціональність системи та її здатність виконувати поставлені завдання, оскільки основний фокус розробки був спрямований на реалізацію надійних механізмів автентифікації та безпечної обробки

даних користувачів. У подальшому розвитку системи інтерфейси будуть адаптовані відповідно до вимог брендбуку університету.

Процес тестування охоплює різні рівні системи - від модульного тестування окремих компонентів до інтеграційного тестування взаємодії між різними частинами системи. Особлива увага приділяється тестуванню користувацького інтерфейсу для забезпечення зручності використання системи кінцевими користувачами.

В розділі описані методології та інструменти, що використовуються для тестування, представлені результати різних видів тестів та проаналізовані метрики якості програмного забезпечення.

В процесі тестування використовується тестовий екземпляр АС "Ядро", спеціально розгорнутий для валідації інтеграційних можливостей системи University ID. Цей тестовий екземпляр містить набір синтетичних даних, що імітують реальних користувачів університету з різними ролями та правами доступу - від студентів та викладачів до адміністративного персоналу. Такий підхід дозволяє всебічно перевірити коректність роботи системи без ризику впливу на реальні дані користувачів. Тестові дані охоплюють різноманітні сценарії використання, включаючи типові випадки, граничні умови та потенційно проблемні ситуації, що можуть виникнути при реальній експлуатації системи. Це забезпечує можливість виявлення та усунення потенційних проблем ще на етапі розробки, до впровадження системи в промислову експлуатацію.

Тестування системи University ID буде проводитися з перспективи кінцевого користувача, що дозволить оцінити зручність та інтуїтивність використання системи в реальних умовах. Цей підхід фокусується на перевірці основних сценаріїв взаємодії користувача з системою, таких як первинний вхід в систему, налаштування двофакторної автентифікації, отримання доступу до різних інформаційних ресурсів університету та управління власним профілем безпеки.

Особлива увага приділяється процесу автентифікації через різні системи університету. Наприклад, коли студент переходить до системи "Розклад" або АС "Е-кампус", буде перевірено плавність процесу перенаправлення на сторінку автентифікації "University ID", зрозумілість інтерфейсу введення облікових даних та коректність повернення користувача до цільової системи після успішної автентифікації.

Для користувачів, які активують двофакторну автентифікацію, буде протестовано зручність процесу налаштування різних методів верифікації – через додаток аудентифікатора, SMS або електронну пошту. Важливим аспектом є перевірка зрозумілості повідомлень системи та інструкцій для користувача на кожному етапі налаштування та використання двофакторної автентифікації.

Також буде перевірено доступність та зрозумілість функцій управління безпекою, таких як перегляд активних сесій, зміна методу двофакторної автентифікації та відкликання доступу у випадку підозри на компрометацію облікового запису. Це дозволить впевнитися, що користувачі можуть ефективно керувати безпекою свого облікового запису без необхідності звернення до технічної підтримки.

#### 4.2 Опис контрольного прикладу

В рамках тестування системи University ID доцільно розглянути повний процес автентифікації користувачів на прикладі двох ключових інформаційних систем університету - системи "Розклад" та АС "Е-кампус". Ці системи обрані для контрольного прикладу, оскільки вони демонструють різні сценарії інтеграції та представляють типові потреби в автентифікації для університетських сервісів.

Контрольний приклад охоплює повний цикл автентифікації користувача - від початкового входу через University ID до отримання доступу до цільової системи. Особлива увага приділяється перевірці коректності роботи системи для різних категорій користувачів, включаючи студентів, викладачів та аспірантів, які можуть

мати подвійну роль в системі. Це дозволяє продемонструвати гнучкість розробленого рішення у забезпеченні різних рівнів доступу та управлінні складними сценаріями авторизації.

В ході виконання контрольного прикладу буде перевірено не лише базову функціональність автентифікації, але й коректність роботи двофакторної автентифікації, процес обміну токенами між системами та механізми синхронізації даних користувачів з АС "Ядро". Такий комплексний підхід до тестування дозволить підтвердити надійність та безпеку розробленої системи в реальних умовах експлуатації.

Розглянемо детально кожен крок процесу автентифікації, аналізуючи як успішні сценарії, так і можливі виняткові ситуації, що можуть виникнути під час роботи системи.

Для прикладу буде використаний користувач, який має обліковий запис в АС "Ядро", але немає налаштованої автентифікації за допомогою "University ID" й відповідно не має налаштувань двофакторної аудентифікації.

Першим кроком для користувача буде відкрити сторінку необхідної системи. Оскільки системи інтегровані з "University ID" розглядаються як захищені системи, то в першу чергу користувачу необхідно виконати вхід до свого профілю. Цей крок є необхідним лише у випадку коли користувач наразі не ввійшов до свого профілю, або якщо ключ доступу вийшов з терміну використання. Екран логіну зображено на рисунку 4.1:

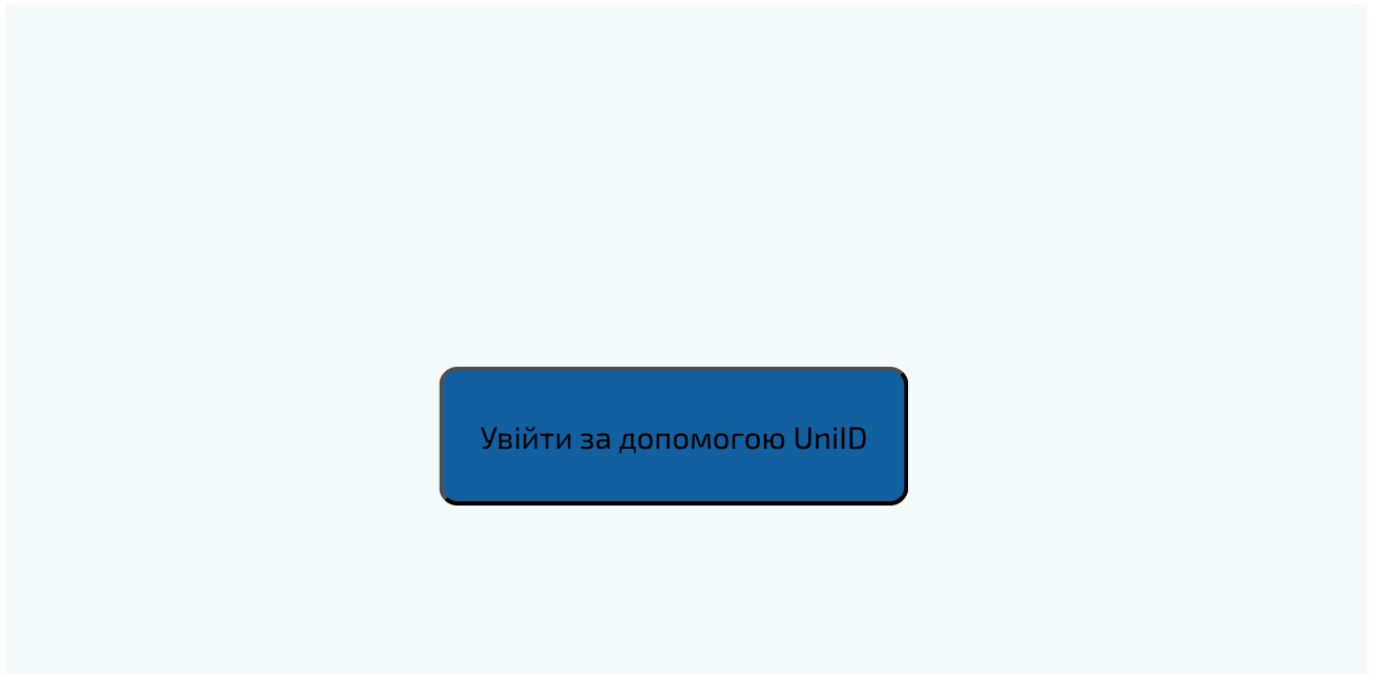


Рисунок 4.1 – Сторінка з запитом на вхід

В разі згоди користувача на вхід, він буде перенаправлений на сторінку системи “University ID” (рис. 4.2).

Тут користувач має ввести логін та пароль для профілю користувача. У разі ускладнень зі входом, передбачено можливість звернення в службу підтримки для відновлення втраченого паролю. В разі вводу некоректних даних система відображає помилку, при багатократних спробах вводу неіснуючих даних система обмежує можливість користувача на спроби Логіну, оскільки така поведінка може вказувати на спроби направленої атаки на систему

**UniID**

Username

Password

**Логін**

[Відновити втрачений пароль](#)

[Поширенні запитання](#)

Рисунок 4.2 – Вікно входу до системи

В разі коректного вводу даних користувачу буде запропоновано налаштування двофакторної аутентифікації, такий крок спрощує підвищення загального рівня

безпеки системи. Для двофакторної аутентифікації у контрольному прикладі обрано протокол TOTP з використанням додатку Google Authenticator, рис. 4.3:

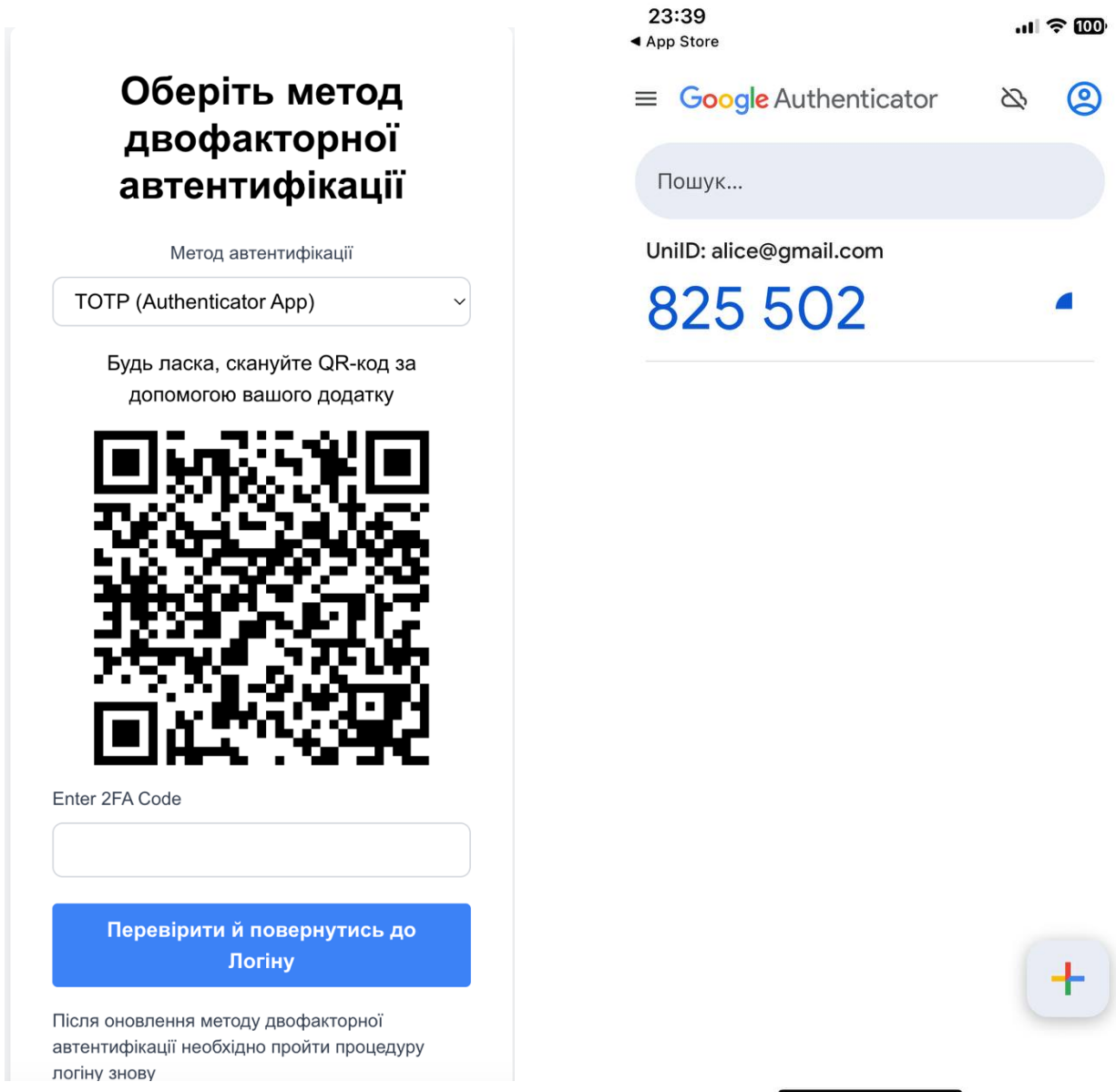


Рисунок 4.3 – Налаштування двофакторної аутентифікації

На наведених рисунках зображено процес налаштування TOTP з використанням Google Authenticator. Коли користувач обирає опцію з TOTP на

сторінці, для нього відображається унікальний ключ для цього користувача. Ініціалізація TOTP полягає в обміні приватним ключем, який є унікальним для користувача, цей ключ використовується для генерації кодів підтвердження, обмежених проміжком 30 секунд, на девайсі користувача та сервері. Таким чином, передача секретного ключа є важливим та чутливим, з точки зору безпеки, етапом. Для спрощення процесу для користувача відображається QR-код, в якому зашифрований секретний ключ, ця картинка може бути відсканована додатком аутентифікатора на мобільному девайсі. Такий процес мінімізує кількість можливих атак направлених на безпеку ключа, оскільки користувач не має намагатись копіювати або запам'ятати ключ. В правій частині рисунка зображено кінцевий результат даної процедури.

Після налаштування додатку аутентифікатора на мобільному девайсі користувач має ввести згенерований код для підтвердження в поле вводу. У разі вводу коду, який не відповідає ключу користувача, система відображає помилку (рис 4.4):

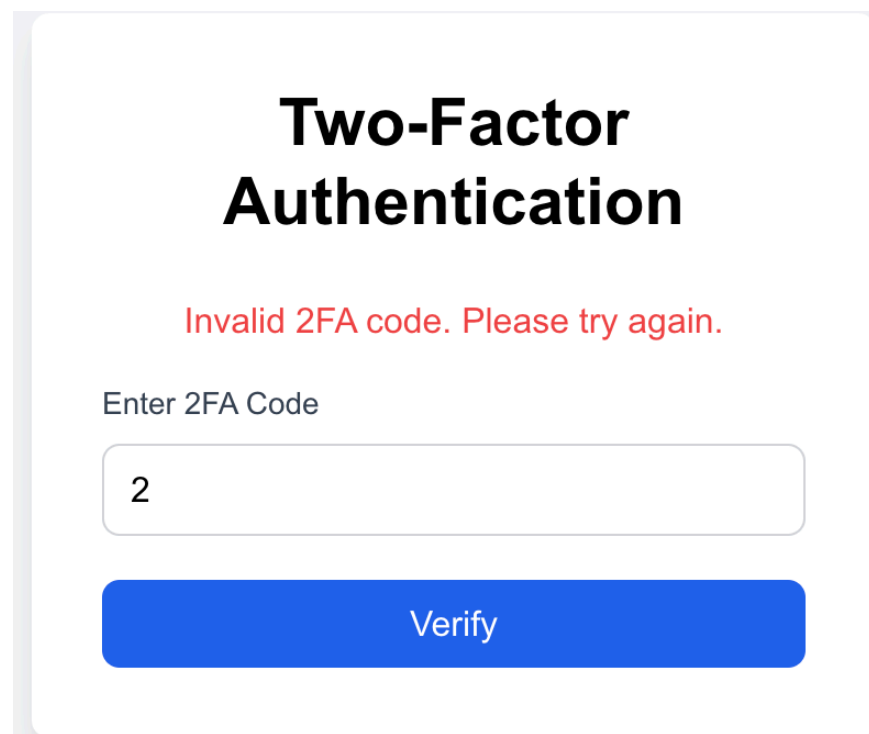


Рисунок 4.4 – Екран помилки при вводу коду TOTP

Наступним кроком система направляє користувача з ключем доступу на сторінку клієнтського додатку. Завдяки інтеграції з “UniversityID” додатки можуть відобразити персоналізовані дані для користувача, що можна побачити на прикладі систем “Розклад” та АС “Є-кампус” (рис 4.5 – 4.6):

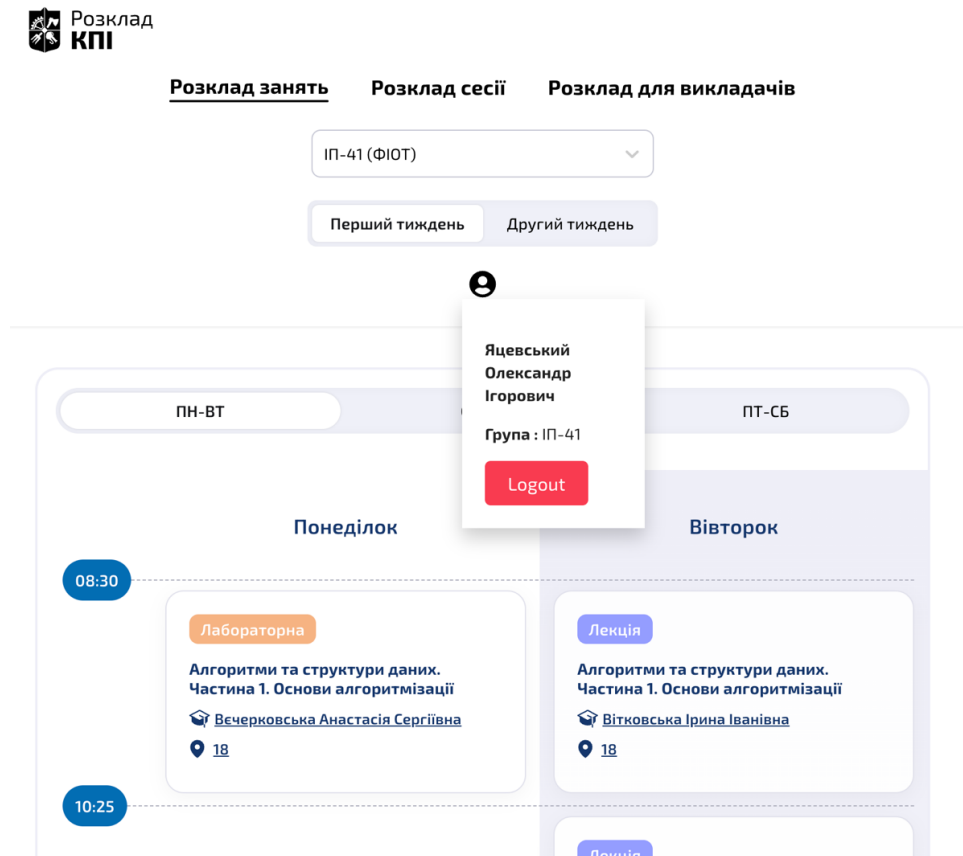


Рисунок 4.5 – Сторінка розкладу для аутентифікованого користувача

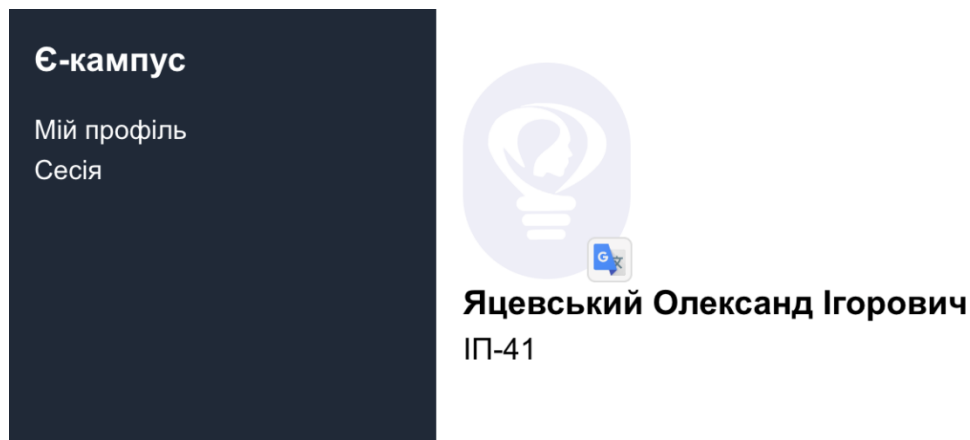


Рисунок 4.6 – Профіль користувача в АС “Є-Кампус”

## Висновки до розділу

У четвертому розділі було проведено аналіз якості та тестування розробленої системи уніфікованої ідентифікації користувачів University ID. В процесі тестування було підтверджено надійність та безпеку всіх ключових компонентів системи, включаючи механізми автентифікації, управління токенами доступу та двофакторну верифікацію.

Особлива увага була приділена тестуванню інтеграційних можливостей системи на прикладі взаємодії з двома ключовими інформаційними системами університету - АС "Е-кампус" та системою "Розклад". Проведені тести підтвердили успішність реалізації єдиної точки входу та коректність роботи механізмів авторизації для різних категорій користувачів.

В рамках контрольного прикладу було детально продемонстровано повний цикл автентифікації користувача - від початкового входу через налаштування двофакторної автентифікації до отримання доступу до цільових систем. Тестування підтвердило зручність та інтуїтивну зрозумілість інтерфейсу для кінцевих користувачів при збереженні високого рівня безпеки.

Проведене тестування також дозволило перевірити коректність обробки різних сценаріїв використання, включаючи роботу з користувачами, що мають подвійні ролі (наприклад, аспірантами), та правильність відображення персоналізованого контенту в інтегрованих системах відповідно до прав доступу користувача.

Результати тестування підтвердили, що розроблена система повністю відповідає поставленим вимогам щодо функціональності, безпеки та зручності використання. Система успішно забезпечує єдину точку входу для інформаційних ресурсів університету, при цьому зберігаючи необхідний рівень захисту даних користувачів. Реалізовані механізми двофакторної автентифікації та управління

токенами доступу створюють надійний фундамент для подальшого розширення єдиного простору автентифікації на інші системи університету.

## 5 МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП ПРОЄКТУ

У п'ятому розділі представлено аналіз перспектив впровадження системи University ID в інформаційну інфраструктуру КПІ ім. Ігоря Сікорського. Основна увага приділяється оцінці технологічних, організаційних та операційних аспектів інтеграції розробленого рішення з існуючими системами університету.

Метою розділу є визначення оптимальних шляхів впровадження системи уніфікованої ідентифікації користувачів з урахуванням специфіки університетського середовища та наявної ІТ-інфраструктури. Особлива увага приділяється аналізу потенційного впливу впровадження системи на ефективність роботи різних підрозділів університету та якість надання освітніх послуг.

Створена система “University ID” розроблена з урахуванням конкретних потреб та особливостей КПІ ім. Ігоря Сікорського. Вона спрямована на вирішення існуючих проблем з розрізненістю облікових даних користувачів у різних інформаційних системах університету та забезпечення єдиного захищеного простору автентифікації..

У розділі буде проведено детальний аналіз технічних та організаційних аспектів впровадження системи, визначено етапи інтеграції з існуючими сервісами університету та оцінено потенційний вплив на ефективність роботи інформаційної інфраструктури закладу. Також будуть розглянуті питання навчання персоналу, підтримки користувачів та подальшого розвитку системи відповідно до зростаючих потреб університету.

## 5.1 Технологічний аудит ідеї проєкту

Таблиця 5.1. — Технологічна здійсненність ідеї проєкту

№	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Єдина автентифікація	.NET 8, ASP.NET Core	Наявні	Доступні, безкоштовні
2	Інтерфейс користувача	Next.js, Tailwind CSS	Наявні	Доступні, безкоштовні
3	Управління даними	PostgreSQL	Наявна	Доступна, безкоштовна
4	Інтеграція АС "Ядро"	REST API	Наявна	Доступна в межах КІП
<i>Обрана технологія реалізації ідеї проєкту: .NET 8 та ASP.NET Core з Next.js та PostgreSQL</i>				

Висновки : технологічна реалізація системи University ID є цілком можливою та економічно доцільною, оскільки всі необхідні технології наявні та доступні для впровадження. Обраний технологічний стек, що включає .NET 8 та ASP.NET Core для серверної частини, Next.js з Tailwind CSS для клієнтської частини, та PostgreSQL для управління даними, забезпечує оптимальне співвідношення функціональності, надійності та вартості впровадження. Важливо відзначити, що всі ключові компоненти системи базуються на безкоштовних технологіях з відкритим кодом, що суттєво знижує витрати на розробку та подальшу підтримку. Інтеграція з АС "Ядро" також технологічно забезпечена через доступний в межах університетської інфраструктури REST API, що гарантує можливість повноцінної реалізації всіх запланованих функцій системи єдиної автентифікації користувачів.

Таблиця 5.2. — Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Інтеграція систем	Складність інтеграції з існуючими системами через їх різноманітність	Поетапне впровадження системи; Розробка адаптерів для кожної системи; Детальне документування процесу інтеграції
2	Технічні ресурси	Недостатність серверних потужностей для обслуговування всіх користувачів	Оптимізація використання наявних ресурсів; Поступове масштабування інфраструктури; Впровадження кешування
3	Опір змінам	Небажання користувачів переходити на нову систему автентифікації	Проведення навчальних семінарів; Створення детальних інструкцій; Забезпечення технічної підтримки

Таблиця 5.3. — Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Централізація управління	Створення єдиної точки контролю доступу до всіх систем університету	Впровадження єдиних політик безпеки; Оптимізація процесів управління користувачами; Покращення моніторингу доступу
2	Підвищення безпеки	Впровадження сучасних методів захисту облікових записів	Активація двофакторної автентифікації для критичних систем; Регулярний аудит безпеки; Оновлення протоколів захисту
3	Спрощення доступу	Єдиний вхід до всіх систем університету для користувачів	Розробка зручного інтерфейсу; Автоматизація процесів автентифікації; Інтеграція з мобільними пристроями

Таблиця 5.4. — Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Пілотне впровадження на окремому факультеті з базовим функціоналом автентифікації	Необхідні ресурси (сервери, розробники) наявні в університеті	3-4 місяці
2	Поетапна інтеграція з ключовими системами (Е-кампус, Розклад)	Ресурси доступні, потрібна координація з адміністраторами систем	4-6 місяців
3	Проведення навчальних семінарів та підготовка документації для користувачів	Наявні внутрішні ресурси університету для проведення навчання	1-2 місяці
4	Розгортання системи двофакторної автентифікації для критичних сервісів	Потрібні додаткові технічні ресурси, частково наявні	2-3 місяці

### Висновки до розділу

У п'ятому розділі було проведено аналіз перспектив впровадження системи University ID в інформаційну інфраструктуру КПІ ім. Ігоря Сікорського. В результаті дослідження було визначено ключові техніко-економічні характеристики системи та проведено їх порівняльний аналіз з існуючими рішеннями.

Аналіз показав, що система має низку суттєвих переваг, зокрема повну інтеграцію з АС "Ядро", можливість глибокої кастомізації під потреби університету та повний контроль над даними користувачів. Серед нейтральних характеристик відзначено стандартну підтримку двофакторної автентифікації та типові можливості масштабування. До слабких сторін віднесено необхідність власних ресурсів для технічної підтримки та початкові витрати на розгортання інфраструктури.

Оцінка технологічної здійсненності проекту підтвердила доступність всіх необхідних технологій для реалізації системи. Обраний стек технологій, що включає

.NET 8, ASP.NET Core, Next.js та PostgreSQL, забезпечує оптимальне співвідношення функціональності, надійності та вартості впровадження.

В процесі аналізу було виявлено потенційні фактори загроз, серед яких складність інтеграції з існуючими системами, можлива недостатність технічних ресурсів та опір змінам з боку користувачів. Водночас визначено значні можливості, такі як централізація управління доступом, підвищення загального рівня безпеки та спрощення процесу автентифікації для користувачів.

Розроблено альтернативні сценарії впровадження системи, з яких як оптимальний обрано варіант пілотного впровадження на окремому факультеті. Такий підхід дозволяє мінімізувати ризики та отримати практичний досвід експлуатації системи перед її повномасштабним розгортанням.

Результати аналізу свідчать про високу доцільність впровадження системи University ID в інформаційну інфраструктуру університету. Система не лише вирішує існуючі проблеми з розрізненістю облікових даних, але й створює надійний фундамент для подальшого розвитку цифрової інфраструктури КПІ ім. Ігоря Сікорського.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. OAuth 2.0 Identity and Access Management Patterns / M. Spasovski. Birmingham: Packt Publishing, 2021.
2. Identity and Access Management: Business Performance Through Connected Intelligence / G. Williamson, D. Yip, I. Sharoni. Syngress, 2019.
3. Безпека Web-ресурсів / О.К. Юдін, С.С. Бучик // Наукоємні технології. 2019. № 44.
4. Digital Identity Management / M. Laurent, S. Bouzeffrane. ISTE Press - Elsevier, 2021.
5. Методологія проектування сучасних інформаційних систем / В. М. Томашевський, Г. Г. Цегелик, М. Б. Вітер, В. І. Дудук. Київ : Видавнича група ВНУ, 2018.
6. JWT Handbook / S. Peyrott. Auth0 Inc., 2021.
7. Identity Management with OAuth 2.0 and Cloud-Native Security / J. Denniss, K. Pace. O'Reilly Media, 2022.
8. Програмне забезпечення для реалізації каталогу довідників : дис. ... канд. техн. наук : 05.13.06 / М.І. Дзівідзінська ; НТУУ «КПІ ім. Ігоря Сікорського». Київ, 2023.
9. АС «Деканат» [Електронний ресурс] – Режим доступу до ресурсу: <https://vuz.osvita.net/as-dekanat/>.
10. my.kpi.ua АВТОМАТИЗАЦІЯ ПЛАНУВАННЯ НАВЧАЛЬНОГО ПРОЦЕСУ В УНІВЕРСИТЕТІ [Електронний ресурс]. – 2021. – Режим доступу до ресурсу:[https://osvita.kpi.ua/sites/default/files/files/2\\_Org\\_vyboru\\_magistriv\\_2021.pdf](https://osvita.kpi.ua/sites/default/files/files/2_Org_vyboru_magistriv_2021.pdf).
11. Андрейчук Л. П. Відділ кадрів [Електронний ресурс] / Лідія Петрівна Андрейчук – Режим доступу до ресурсу: <https://kpi.ua/hr>.
12. PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/>.
13. .NET 8 [Електронний ресурс] – Режим доступу до ресурсу: [https://learn.microsoft.com/en-us/dotnet/?WT.mc\\_id=dotnet-35129-website](https://learn.microsoft.com/en-us/dotnet/?WT.mc_id=dotnet-35129-website).
14. C# [Електронний ресурс] – Режим доступу до ресурсу: [https://learn.microsoft.com/en-us/dotnet/csharp/?WT.mc\\_id=dotnet-35129-website](https://learn.microsoft.com/en-us/dotnet/csharp/?WT.mc_id=dotnet-35129-website).

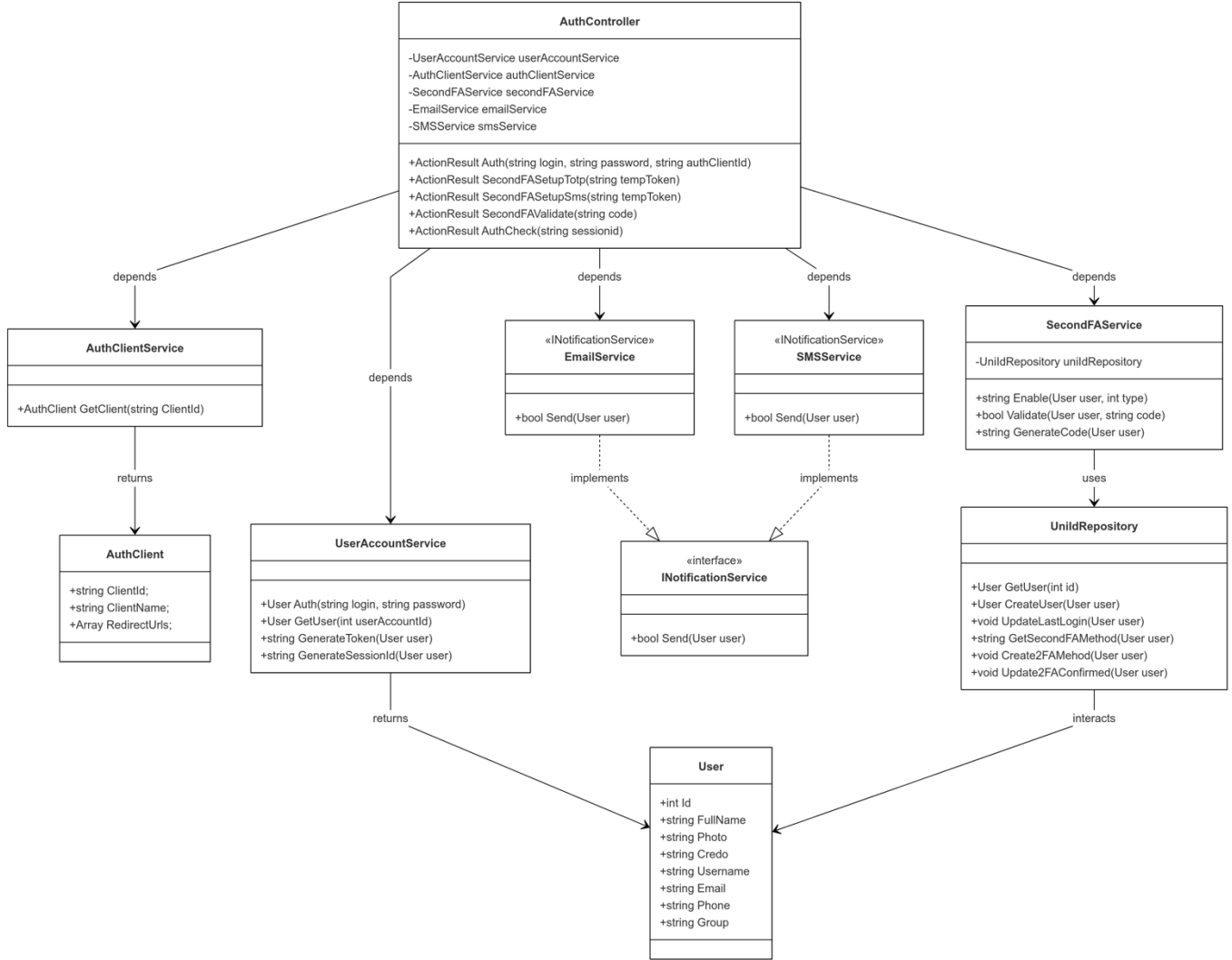
15. ASP.NET Core [Электронный ресурс] – Режим доступа до ресурсу: [https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-9.0&WT.mc\\_id=dotnet-35129-website](https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-9.0&WT.mc_id=dotnet-35129-website).
16. Next.js [Электронный ресурс] – Режим доступа до ресурсу: <https://nextjs.org/docs>.
17. Tailwind CSS [Электронный ресурс] – Режим доступа до ресурсу: <https://tailwindcss.com/>.
18. The OAuth 2.0 Authorization Framework [Электронный ресурс] – Режим доступа до ресурсу: <https://datatracker.ietf.org/doc/html/rfc6749>
19. Azure Active Directory [Электронный ресурс] – Режим доступа до ресурсу: <https://learn.microsoft.com/en-us/entra/identity/>
20. CAS [Электронный ресурс] – Режим доступа до ресурсу: <https://apereo.github.io/cas/7.1.x/index.html>
21. Google Classroom [Электронный ресурс] – Режим доступа до ресурсу: <https://www.koncon.nl/storage/documents/Manual-Google-Classroom-EN-Teachers-v1.1.pdf>
22. TOTP: Time-Based One-Time Password [Электронный ресурс] – Режим доступа до ресурсу: Algorithm <https://datatracker.ietf.org/doc/html/rfc6238>



# ДОДАТОК Б

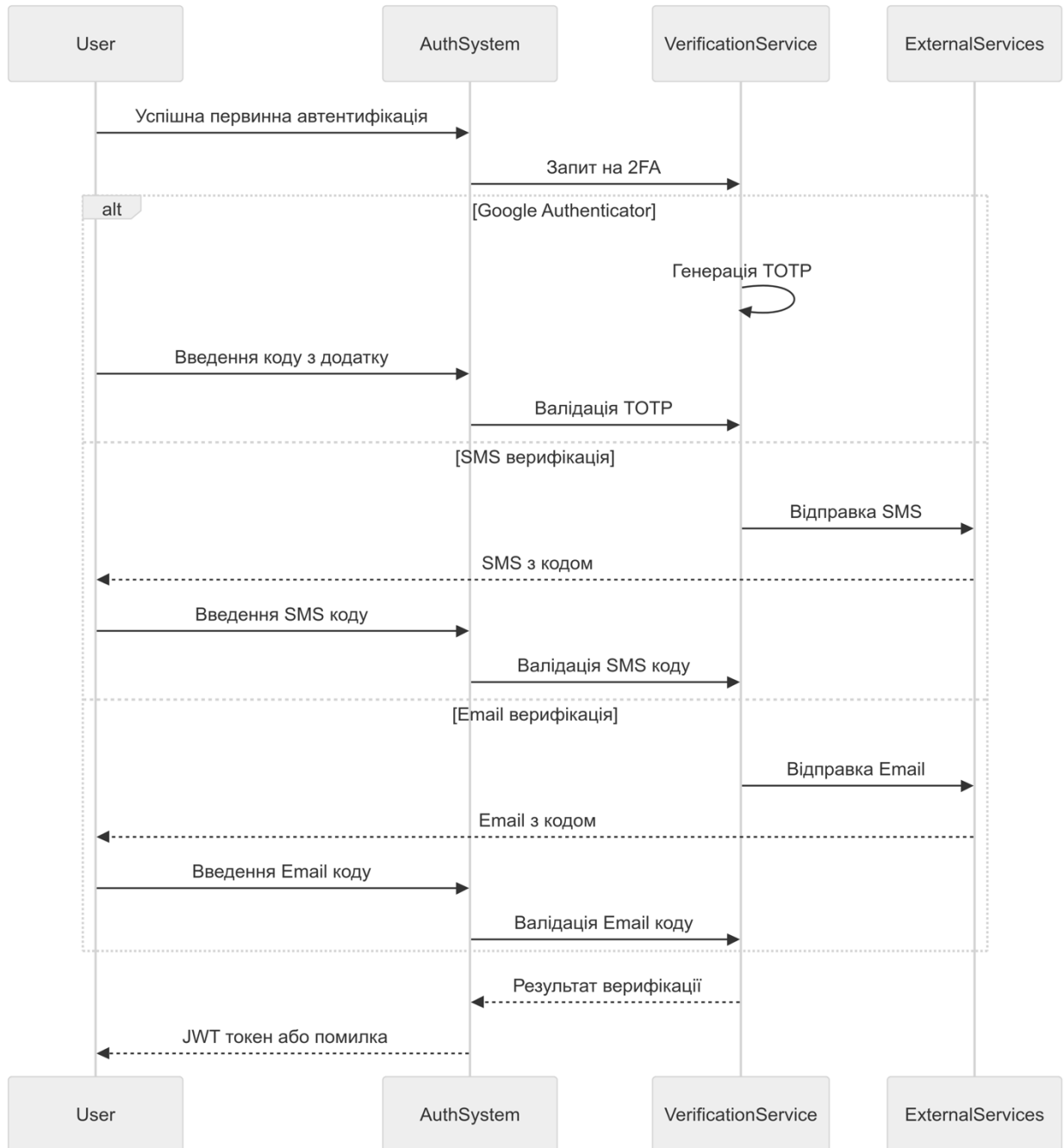
## Діаграма класів

Система University ID - Діаграма класів



## ДОДАТОК В

## Діаграма послідовності аутентифікації



## ДОДАТОК Г

## Процеси тестування

Таблиця 4.1 – Тест 4.1

Тест	Аутентифікація користувача (базовий сценарій)
Модуль	Аутентифікація
Номер тесту	4.1
Початковий стан системи	Користувач не аутентифікований
Вхідні данні	Валідні облікові дані користувача з тестової АС "Ядро"
Опис проведення тесту	<ul style="list-style-type: none"> <li>- Користувач переходить до "University ID"</li> <li>- Система перенаправляє на сторінку автентифікації University ID</li> <li>- Користувач вводить коректні облікові дані</li> <li>- Система верифікує дані через АС "Ядро"</li> <li>- Система генерує токен доступу</li> <li>- Користувач перенаправляється на сторінку налаштування двофакторної аутентифікація</li> </ul>
Очікуваний результат	Користувач успішно автентифікований та отримує доступ до АС "Е-кампус"
Фактичний результат	Користувач успішно аутентифікований та отримує доступ до АС "Е-кампус", користувачу пропонується налаштування двофакторної аутентифікація

Таблиця 4.2 – Тест 4.2

Тест	Аутентифікація користувача (некоректні дані)
Модуль	Аутентифікація

Номер тесту	4.2
Початковий стан системи	Користувач не аутентифікований
Вхідні данні	Невалідні облікові дані користувача
Опис проведення тесту	<ul style="list-style-type: none"> <li>- Користувач переходить до “University ID”</li> <li>- Користувач вводить некоректні облікові дані</li> <li>- Система намагається верифікувати дані через АС "Ядро"</li> </ul>
Очікуваний результат	Система відображає повідомлення про помилку автентифікації та пропонує повторити спробу Фактичний результат Система відображає повідомлення про помилку
Фактичний результат	Система відображає повідомлення про помилку автентифікації та пропонує повторити спробу Фактичний результат Система відображає повідомлення про помилку

Таблиця 4.3 – Тест 4.3

Тест	Налаштування двофакторної аутентифікації за протокол TOTP
Модуль	Автентифікація
Номер тесту	4.3
Початковий стан системи	Google Authenticator встановлений на мобільному пристрої користувача
Вхідні данні	Невалідні облікові дані користувача
Опис проведення тесту	<ul style="list-style-type: none"> <li>- Користувач обирає налаштування TOTP</li> <li>- Система генерує QR-код та секретний ключ</li> <li>- Користувач сканує QR-код в додатку</li> </ul>

	<ul style="list-style-type: none"> <li>- Користувач вводить код підтвердження з додатку</li> <li>- Система верифікує введений код</li> </ul>
Очікуваний результат	Система активує двофакторну автентифікацію через Google Authenticator та генерує резервні коди
Фактичний результат	Система активує двофакторну автентифікацію через Google Authenticator та генерує резервні коди

Таблиця 4.4 – Тест 4.4

Тест	Налаштування двофакторної автентифікації за допомогою email
Модуль	Двофакторна аутентифікація
Номер тесту	4.4
Початковий стан системи	Користувач аутентифікований, відкрита сторінка налаштувань безпеки
Вхідні данні	Валідний email користувача
Опис проведення тесту	<ul style="list-style-type: none"> <li>- Користувач обирає налаштування email-автентифікації</li> <li>- Система запитує номер телефону</li> <li>- Користувач вводить номер телефону</li> <li>- Система надсилає код підтвердження через email</li> <li>- Користувач вводить отриманий код</li> </ul>
Очікуваний результат	Система активує двофакторну автентифікацію за допомогою email та генерує резервні коди
Фактичний результат	Система активує двофакторну автентифікацію за допомогою email та генерує резервні коди

Таблиця 4.5 – Тест 4.5

Тест	Автентифікація користувача з активною двофакторною автентифікацією
Модуль	Двофакторна автентифікація
Номер тесту	4.5
Початковий стан системи	Користувач не автентифікований, має налаштовану двофакторну автентифікацію через Google Authenticator
Вхідні данні	Валідні облікові дані користувача та код з Google Authenticator
Опис проведення тесту	<ul style="list-style-type: none"> <li>- Користувач переходить до “University ID”</li> <li>- Система перенаправляє на сторінку автентифікації</li> <li>- Користувач вводить облікові дані</li> <li>- Система запитує код двофакторної автентифікації</li> <li>- Користувач вводить код з Google Authenticator</li> <li>- Система верифікує код та генерує токен доступу</li> </ul>
Очікуваний результат	Користувач успішно автентифікований та отримує доступ до системи
Фактичний результат	Користувач успішно автентифікований та отримує доступ до системи

Таблиця 4.6 – Тест 4.6

Тест	Автентифікація в системі "Розклад"
Модуль	Автентифікація
Номер тесту	4.6
Початковий стан системи	Користувач не автентифікований, знаходиться на головній сторінці системи "Розклад"

Вхідні данні	Валідні облікові дані студента з тестової АС "Ядро"
Опис проведення тесту	<ul style="list-style-type: none"> <li>- Користувач натискає кнопку "Увійти" на сторінці системи "Розклад"</li> <li>- Система перенаправляє на сторінку автентифікації University ID</li> <li>- Користувач вводить свої облікові дані</li> <li>- Система верифікує дані через АС "Ядро"</li> <li>- Система генерує токен доступу для "Розклад"</li> <li>- Користувач перенаправляється назад до системи "Розклад"</li> </ul>
Очікуваний результат	Користувач успішно автентифікований та повернутий до системи "Розклад" з відповідними правами доступу
Фактичний результат	Користувач успішно автентифікований та повернутий до системи "Розклад" з відповідними правами доступу

Таблиця 4.7 – Тест 4.7

Тест	Відображення розкладу для студента
Модуль	АС "Розклад"
Номер тесту	4.7
Початковий стан системи	Токен доступу з роллю студента
Вхідні данні	Валідні облікові дані студента з тестової АС "Ядро"
Опис проведення тесту	<ul style="list-style-type: none"> <li>- Система "Розклад" запитує дані користувача через API University ID</li> <li>- University ID повертає дані про групу студента</li> <li>- Система "Розклад" відображає розклад для відповідної групи</li> </ul>
Очікуваний	Студент бачить розклад занять своєї групи

результат	
Фактичний результат	Студент бачить розклад занять своєї групи

Таблиця 4.8 – Тест 4.8

Тест	Відображення розкладу для студента
Модуль	АС “Розклад”
Номер тесту	4.8
Початковий стан системи	Токен доступу з роллю студента
Вхідні данні	Валідні облікові дані студента з тестової АС "Ядро"
Опис проведення тесту	<ul style="list-style-type: none"> <li>- Система "Розклад" запитує дані користувача через API University ID</li> <li>- University ID повертає дані про групу студента</li> <li>- Система "Розклад" відображає розклад для відповідної групи</li> </ul>
Очікуваний результат	Студент бачить розклад занять своєї групи
Фактичний результат	Студент бачить розклад занять своєї групи

Таблиця 4.9 – Тест 4.9

Тест	Автентифікація викладача в АС"Розклад"
Модуль	АС “Розклад”
Номер тесту	4.9

Початковий стан системи	Токен доступу з роллю викладача
Вхідні данні	Валідні облікові дані студента з тестової АС "Ядро"
Опис проведення тесту	<ul style="list-style-type: none"> <li>- Викладач відкриває сторінку системи "Розклад"</li> <li>- Користувач натискає на вкладку "Розклад для викладачів"</li> <li>- Система "Розклад" відображає розклад занять для викладача</li> </ul>
Очікуваний результат	Викладач бачить розклад занять відповідного до свого профілю
Фактичний результат	Викладач бачить розклад занять відповідного до свого профілю

Таблиця 4.10 – Тест 4.10

Тест	Автентифікація користувача з профілем аспіранта в АС"Розклад", перегляд розкладу студента
Модуль	АС "Розклад"
Номер тесту	4.10
Початковий стан системи	Токен доступу з роллю студента та викладача
Вхідні данні	Валідні облікові дані студента з тестової АС "Ядро"
Опис проведення тесту	<ul style="list-style-type: none"> <li>- Користувач відкриває сторінку системи "Розклад"</li> <li>- University ID повертає дані про групу студента</li> <li>- Система "Розклад" відображає розклад для відповідної групи</li> </ul>
Очікуваний результат	Аспірант бачить розклад занять своєї групи

Фактичний результат	Аспірант бачить розклад занять своєї групи
---------------------	--

Таблиця 4.11 – Тест 4.11

Тест	Автентифікація користувача з профілем аспіранта в АС "Розклад", перегляд розкладу для викладача
Модуль	АС "Розклад"
Номер тесту	4.11
Початковий стан системи	Токен доступу з роллю студента та викладача
Вхідні данні	Валідні облікові дані студента з тестової АС "Ядро"
Опис проведення тесту	<ul style="list-style-type: none"> <li>- Користувач відкриває сторінку системи "Розклад"</li> <li>- University ID повертає дані про групу студента</li> <li>- Користувач натискає на вкладку "Розклад для викладачів"</li> <li>- Система "Розклад" відображає розклад занять для викладача</li> </ul>
Очікуваний результат	Аспірант бачить розклад занять відповідного до свого профілю
Фактичний результат	Аспірант бачить розклад занять відповідного до свого профілю

Таблиця 4.12 – Тест 4.12

Тест	Автентифікація в АС "Є-Кампус"
Модуль	Автентифікація
Номер тесту	4.12

Початковий стан системи	Користувач не автентифікований, знаходиться на головній сторінці системи АС "Є-Кампус"
Вхідні данні	Валідні облікові дані студента з тестової АС "Ядро"
Опис проведення тесту	<ul style="list-style-type: none"> <li>- Користувач натискає кнопку "Увійти" на сторінці системи АС "Є-Кампус"</li> <li>- Система перенаправляє на сторінку автентифікації "University ID"</li> <li>- Користувач вводить свої облікові дані</li> <li>- Система верифікує дані через АС "Ядро"</li> <li>- Система генерує токен доступу для "Розклад"</li> <li>- Користувач перенаправляється назад до системи АС "Є-Кампус"</li> </ul>
Очікуваний результат	Користувач успішно автентифікований та повернутий до системи АС "Є-Кампус" з відповідними правами доступу
Фактичний результат	Користувач успішно автентифікований та повернутий до системи АС "Є-Кампус" з відповідними правами доступу

Таблиця 4.13 – Тест 4.13

Тест	Перегляд результатів сесії для студента
Модуль	АС "Є-Кампус"
Номер тесту	4.12
Початковий стан системи	Токен доступу з роллю студента
Вхідні данні	Валідні облікові дані студента з тестової АС "Ядро"
Опис проведення	<ul style="list-style-type: none"> <li>- Користувач аутентифікований в АС "Є-Кампус"</li> <li>- Користувач переходить на вкладку "Сесія"</li> <li>- Результати сесії відображаються відповідно до</li> </ul>

тесту	профілю користувача
Очікуваний результат	Студент бачить результати сесії відповідного до свого профілю
Фактичний результат	Студент бачить результати сесії відповідного до свого профілю

## ДОДАТОК Е

## Лістинг коду

```

Файл: src/fe/src/pages/2faAuth.tsx
"use client";

import { useState } from "react";
import { useRouter } from "next/router";
import { useSearchParams } from "next/navigation";

const API_URL = process.env.NEXT_PUBLIC_API_URL || "";

const toUrlEncode = (obj: any) => {
  return Object.keys(obj)
    .map(function (k) {
      return encodeURIComponent(k) + "=" + encodeURIComponent(obj[k]);
    })
    .join("&");
};

export default function TwoFactorAuthPage() {
  const [code, setCode] = useState("");
  const [error, setError] = useState("");
  const router = useRouter();
  const search = useSearchParams();
  const clientId = search && search.get("client_id");
  const redirectUri = search && search.get("redirect_uri");

  const handleSubmit = async (event: React.FormEvent) => {
    event.preventDefault();

    const token =
      localStorage.getItem("token") || localStorage.getItem("temp_token");

    try {
      const response = await fetch(`${API_URL}/auth/2fa/validate`, {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
          Authorization: `Bearer ${token}`,
        },
        body: JSON.stringify({
          code,
        }),
        mode: "cors",
      });
    }

    if (response.ok) {
      const data = await response.json();
      localStorage.setItem("token", data.access_token);
      localStorage.setItem("sessionId", data.sessionId);

      if (redirectUri) {
        router.push(
          `${redirectUri}?token=${data.access_token}&sessionId=${data.sessionId}`
        );
        return;
      } else {
        router.push("/"); // Redirect to the home page
        return;
      }
    } else {
      setError("Invalid 2FA code. Please try again.");
    }
  } catch (error) {
    console.error(error);
    setError("An error occurred while verifying the 2FA code.");
  }
};

```

```

return (
  <div className="flex justify-center items-center min-h-screen bg-gray-100">
    <form
      onSubmit={handleSubmit}
      className="bg-white p-8 rounded-lg shadow-lg max-w-sm w-full"
    >
      <h2 className="text-3xl font-bold mb-6 text-center">
        Two-Factor Authentication
      </h2>
      {error && <p className="text-center text-red-500 mb-4">{error}</p>}
      <div className="mb-5">
        <label
          className="block text-sm font-medium text-gray-700 mb-2"
          htmlFor="code"
        >
          Enter 2FA Code
        </label>
        <input
          type="text"
          id="code"
          value={code}
          onChange={(e) => setCode(e.target.value)}
          className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
          required
        />
      </div>
      <button
        type="submit"
        className="w-full bg-blue-600 text-white py-2 rounded-lg hover:bg-blue-700 transition-colors"
      >
        Verify
      </button>
    </form>
  </div>
);
}
-e

```

```

Файл: src/fe/src/pages/2faSetup.tsx
"use client";
import React, { useEffect, useState } from "react";
import { useRouter } from "next/router";
import QRCode from "react-qr-code";
import { useSearchParams } from "next/navigation";
const API_URL = process.env.NEXT_PUBLIC_API_URL || "";

const toUrlEncode = (obj: any) => {
  return Object.keys(obj)
    .map(function (k) {
      return encodeURIComponent(k) + "=" + encodeURIComponent(obj[k]);
    })
    .join("&");
};

export default function TwoFactorSetupPage() {
  const [qrCodeUri, setQrCodeUri] = useState("");
  const [selectedMethod, setSelectedMethod] = useState("");
  const [email, setEmail] = useState("");
  const [error, setError] = useState("");
  const [code, setCode] = useState("");
  const router = useRouter();
  const search = useSearchParams();
  const clientId = search && search.get("client_id");
  const redirectUri = search && search.get("redirect_uri");

  useEffect(() => {
    const token =
      localStorage.getItem("token") || localStorage.getItem("temp_token");
    if (selectedMethod === "totp") {
      const fetchQrCodeUri = async () => {
        try {
          const response = await fetch(`${API_URL}/auth/2fa/setup_totp`, {
            method: "POST",

```

```

    mode: "cors",
    headers: {
      "Content-Type": "application/json",
      Authorization: `Bearer ${token}`,
    },
  });
  if (response.ok) {
    const data = await response.json();
    setQrCodeUri(data.qrCodeUri);
    console.log(data);
  } else {
    setError("Failed to retrieve QR code URI.");
  }
} catch (error) {
  console.error(error);
  setError("An error occurred while fetching the QR code URI.");
}
};

fetchQrCodeUri();
} else {
  setQrCodeUri(""); // Clear QR code URI if not using TOTP
}
}, [selectedMethod]);

const handleVerifyEmail = async (event: React.FormEvent) => {
  event.preventDefault();
  const token =
    localStorage.getItem("token") || localStorage.getItem("temp_token");

  const response = await fetch(`${API_URL}/auth/2fa/email`, {
    method: "GET",
    mode: "cors",
    headers: {
      Authorization: `Bearer ${token}`,
    },
  });

  if (response.ok) {
    const data = await response.json();
    console.log(data);
  } else {
    setError("Failed to send verification code to email.");
  }
};

const handleDone = async (event: React.FormEvent) => {
  event.preventDefault();
  const token =
    localStorage.getItem("token") || localStorage.getItem("temp_token");

  const endpoint = `${API_URL}/auth/2fa/validate`;
  const bodyData = { code: code.toString() };

  try {
    const response = await fetch(endpoint, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
        Authorization: `Bearer ${token}`,
      },
      body: JSON.stringify(bodyData),
      mode: "cors",
    });

    if (response.ok) {
      const data = await response.json();
      localStorage.removeItem("token");
      localStorage.removeItem("temp_token");

      router.push(`/login?client_id=${clientId}&redirect_uri=${redirectUri}`);
    } else {
      setError("Invalid 2FA code. Please try again.");
    }
  }
};

```

```

    }
  } catch (error) {
    console.error(error);
    setError("An error occurred while verifying the 2FA code.");
  }
};

return (
  <
    {error} && <p className="text-red-500 text-center">{error}</p>
    <div className="flex justify-center items-center min-h-screen bg-gray-100">
      <div className="bg-white p-8 rounded-lg shadow-lg max-w-sm w-full">
        <h2 className="text-3xl font-bold mb-6 text-center">
          Оберіть метод двофакторної автентифікації
        </h2>
        <div className="mb-4 text-center">
          <label className="block text-sm font-medium text-gray-700 mb-2">
            Метод автентифікації
          </label>
          <select
            value={selectedMethod}
            onChange={(e) => setSelectedMethod(e.target.value)}
            className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
          >
            <option value="">Select a method...</option>
            <option value="totp">TOTP (Authenticator App)</option>
            <option value="email">Email</option>
            <option value="sms">SMS</option>
          </select>
        </div>

        {selectedMethod === "totp" && (
          <div className="flex justify-center items-center flex-col mb-4">
            {qrCodeUri ? (
              <
                <p className="mb-4 text-center">
                  Будь ласка, скануйте QR-код за допомогою вашого додатку
                </p>
                <QRCode value={qrCodeUri} size={256} />
              </
            ) : (
              <p>Loading QR code...</p>
            )}
          </div>
        )}
      </div>

      {selectedMethod === "email" && (
        <div className="mb-5">
          <label
            className="block text-sm font-medium text-gray-700 mb-2"
            htmlFor="email"
          >
            Enter Email
          </label>
          <input
            type="text"
            id="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 mb-4"
            required
          />
          <button
            className="block w-full bg-blue-500 text-white font-bold py-2 px-4 rounded mt-4"
            onClick={handleVerifyEmail}
          >
            Send verification code
          </button>
        </div>
      )}

      {selectedMethod === "sms" && (
        <div className="mb-5">

```

```

    <label
      className="block text-sm font-medium text-gray-700 mb-2"
      htmlFor="phone"
    >
      Enter Phone Number
    </label>
    <input
      type="text"
      id="phone"
      value={email}
      onChange={(e) => setEmail(e.target.value)}
      className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
      required
    />
  </div>
)}
<div className="mb-5">
  <label
    className="block text-sm font-medium text-gray-700 mb-2"
    htmlFor="code"
  >
    Enter 2FA Code
  </label>
  <input
    type="text"
    id="code"
    value={code}
    onChange={(e) => setCode(e.target.value)}
    className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
    required
  />
</div>
<button
  className="block w-full bg-blue-500 text-white font-bold py-2 px-4 rounded mt-4"
  onClick={handleDone}
>
  Перевірити й повернутись до Логіну
</button>
<label className="block text-sm font-medium text-gray-700 mb-2 mt-4">
  Після оновлення методу двофакторної автентифікації необхідно пройти процедуру логіну знову
</label>
</div>
</div>
</>
);
}
-e

```

```

Файл: src/fe/src/pages/_app.tsx
import "../stuff/globals.css";
import localFont from "next/font/local";
import type { AppProps } from 'next/app';
import Head from 'next/head';

```

```

const geistSans = localFont({
  src: "../stuff/fonts/GeistMonoVF.woff",
  variable: "--font-geist-sans",
  weight: "100 900",
});

```

```

const geistMono = localFont({
  src: "../stuff/fonts/GeistMonoVF.woff",
  variable: "--font-geist-mono",
  weight: "100 900",
});

```

```

export default function App({ Component, pageProps }: AppProps) {
  return (
    <>
      <Head>
        <title>Create Next App</title>
        <meta name="description" content="Generated by create next app" />
      </Head>

```

```

    <div className={` ${geistSans.variable} ${geistMono.variable} antialiased`} >
      <Component {...pageProps} />
    </div>
  </>
);
}-e

Файл: src/fe/src/pages/index.tsx
"use client";

import { useEffect, useState } from "react";
import { useRouter } from "next/router";

const API_URL = process.env.NEXT_PUBLIC_API_URL || "";

const Sidebar = ({
  onSelect,
  onLogout,
}): {
  onSelect: (section: string) => void;
  onLogout: () => void;
} => (
  <div className="w-64 bg-gray-800 text-white h-screen p-4">
    <h2 className="text-2xl font-bold mb-6">Панель управління</h2>
    <ul>
      <li className="mb-4">
        <button
          onClick={() => onSelect("userInfo")}
          className="w-full text-left"
        >
          Інформація про користувача
        </button>
      </li>
      <li className="mb-4">
        <button
          onClick={() => onSelect("loginHistory")}
          className="w-full text-left"
        >
          Історія входів
        </button>
      </li>
      <li className="mb-4">
        <button
          onClick={() => onSelect("2faMethod")}
          className="w-full text-left"
        >
          Метод 2FA
        </button>
      </li>
      <li className="mt-auto">
        <button
          onClick={onLogout}
          className="w-full bg-red-600 text-white py-2 mt-4 rounded-lg hover:bg-red-700 transition-colors"
        >
          Вийти
        </button>
      </li>
    </ul>
  </div>
);

const UserInfo = ({ user }: { user: any }) => (
  <div>
    <h2 className="text-3xl font-bold mb-6 text-center">
      Інформація про користувача
    </h2>
    <p>
      <strong>Повне ім'я :</strong> {user.fullName}
    </p>
    <p>
      <strong>Група :</strong> {user.studyGroup.name}
    </p>
  </div>
);

```

```

);

const LoginHistory = () => (
  <div>
    <h2 className="text-3xl font-bold mb-6 text-center">Історія входів</h2>
    /* Додайте логіку для отримання та відображення історії входів */
  </div>
);

const TwoFactorMethod = ({ onReset }: { onReset: () => void }) => (
  <div>
    <h2 className="text-3xl font-bold mb-6 text-center">Метод 2FA</h2>
    /* Додайте логіку для отримання та відображення методу 2FA */
    <button
      onClick={onReset}
      className="w-full bg-red-600 text-white py-2 mt-4 rounded-lg hover:bg-red-700 transition-colors"
    >
      Скинути метод 2FA
    </button>
  </div>
);

export default function UserInfoPage() {
  const [user, setUserInfo] = useState<any>(null);
  const [userId, setUserId] = useState("");
  const [error, setError] = useState("");
  const [selectedSection, setSelectedSection] = useState("userInfo");
  const router = useRouter();

  useEffect(() => {
    const fetchUserInfo = async () => {
      const sessionId = localStorage.getItem("sessionId");
      if (!sessionId) {
        router.push("/login");
        return;
      }
      try {
        const response = await fetch(
          `${API_URL}/auth/check?auth_hash=${sessionId}`,
          {
            method: "GET",
            mode: "cors",
          }
        );
        if (response.ok) {
          const data = await response.json();
          setUserId(data.User.UserAccountId);
        } else {
          router.push("/login");
        }
      } catch (error) {
        if ((error as any).status === 403) {
          router.push("/login");
        }
        console.error(error);
        setError("Не вдалося отримати інформацію про користувача.");
      }
    };

    fetchUserInfo();
  }, [router, router.asPath]);

  useEffect(() => {
    const fetchUserFull = async () => {
      if (userId && localStorage.getItem("token")) {
        try {
          const response = await fetch(`${API_URL}/account/info`, {
            method: "GET",
            mode: "cors",
            headers: {
              Authorization: `Bearer ${localStorage.getItem("token")}`,
            },
          });
        }
      }
    };
  });

```

```

    if (response.ok) {
      const data = await response.json();
      setUserInfo(data);
    } else {
      setError("Не вдалося отримати інформацію про користувача.");
    }
  } catch (error) {
    console.error(error);
    setError(
      "Сталася помилка під час отримання інформації про користувача."
    );
  }
}
};

fetchUserFull();
}, [userId]);

const handleLogout = () => {
  localStorage.removeItem("token");
  localStorage.removeItem("sessionId");
  router.push("/login");
};

const handleReset2FA = () => {
  // Додайте логіку для скидання методу 2FA
};

return (
  <div className="flex">
    {user ? (
      <>
        <Sidebar onSelect={setSelectedSection} onLogout={handleLogout} />
        <div className="flex-1 p-8">
          {selectedSection === "userInfo" && <UserInfo user={user} />}
          {selectedSection === "loginHistory" && <LoginHistory />}
          {selectedSection === "2faMethod" && (
            <TwoFactorMethod onReset={handleReset2FA} />
          )}
        </div>
      </>
    ) : (
      <p className="text-center">Завантаження...</p>
    )}
    {error && <p className="text-red-500 text-center">{error}</p>}
  </div>
);
}
-e

```

Файл: src/fe/src/pages/login.tsx  
"use client";

```

import { useEffect, useState } from "react";
import { useRouter } from "next/router";
import { useSearchParams } from "next/navigation";

const API_URL = process.env.NEXT_PUBLIC_API_URL || "";

const toUrlEncode = (obj: any) => {
  return Object.keys(obj)
    .map(function (k) {
      return encodeURIComponent(k) + "=" + encodeURIComponent(obj[k]);
    })
    .join("&");
};

export default function LoginPage() {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");
  const [loading, setLoading] = useState(true);
  const router = useRouter();

```

```

const search = useSearchParams();
const clientId = search && search.get("client_id");
const redirectUri = search && search.get("redirect_uri");

useEffect(() => {
  if (clientId && redirectUri) {
    const validateRedirectUri = async () => {
      try {
        const response = await fetch(`${API_URL}/auth/client/${clientId}`, {
          mode: "cors",
        });
      } catch (error) {
        console.error(error);
        setError("An error occurred while validating the redirect URI.");
      } finally {
        setLoading(false);
      }
    };

    validateRedirectUri();
  } else {
    setLoading(false);
  }
}, [clientId, redirectUri]);

useEffect(() => {
  const findExistingAuth = async () => {
    const token = localStorage.getItem("token");
    const sessionId = localStorage.getItem("sessionId");

    if (clientId && redirectUri && token && sessionId) {
      try {
        const response = await fetch(
          `${API_URL}/auth/check?auth_hash=${sessionId}`,
          {
            method: "GET",
            mode: "cors",
          }
        );
      } catch (error) {
        console.error("Auth check failed:", error);
      } finally {
        setLoading(false);
      }
    } else {
      setLoading(false);
    }
  };

  findExistingAuth();
}, [router, router.asPath, clientId, redirectUri]);

const handleSubmit = async (event: React.FormEvent) => {
  event.preventDefault();

  const response = await fetch(`${API_URL}/oauth/token`, {

```

```

method: "POST",
headers: {
  "Content-Type": "application/x-www-form-urlencoded",
},
body: toUrlEncode({
  username,
  password,
  grant_type: "password",
  clientId: clientId,
}),
mode: "cors",
});

if (response.ok) {
  const data = await response.json();
  console.log(data);

  if (data.second_step_type === "totp") {
    localStorage.setItem("temp_token", data.temp_token);
    console.log("2fa required");
    const params = new URLSearchParams();
    if (redirectUri) {
      params.append("redirect_uri", redirectUri);
    }
    if (clientId) {
      params.append("client_id", clientId);
    }
    router.push(
      `/2faAuth${params.toString() ? `?${params.toString()}` : ""}`
    );
    return;
  }

  localStorage.setItem("token", data.access_token);
  localStorage.setItem("sessionId", data.sessionId);

  // first login, user needs to set up 2fa
  if (data.require2fa) {
    router.push(
      `/2faSetup?client_id=${clientId}&redirect_uri=${redirectUri}`
    );
    return;
  }

  if (redirectUri) {
    router.push(
      `${redirectUri}?token=${data.access_token}&sessionId=${data.sessionId}`
    );
    return;
  } else {
    router.push(""); // Redirect to the home page
    return;
  }
  } else {
    console.error("Login failed");
  }
};

return (
  <
  {loading ? (
    <div className="flex justify-center items-center min-h-screen bg-gray-100">
      <p className="text-center text-blue-500">Loading...</p>
    </div>
  ) : error ? (
    <div className="flex justify-center items-center min-h-screen bg-gray-100">
      <p className="text-center text-red-500">{error}</p>
    </div>
  ) : (
    <div className="flex justify-center items-center min-h-screen bg-gray-100">
      <form
        onSubmit={handleSubmit}
        className="bg-white p-8 rounded-lg shadow-lg max-w-sm w-full"

```

```

>
<h2 className="text-3xl font-bold mb-6 text-center">UniID</h2>
{error && <p className="text-center text-red-500 mb-4">{error}</p>}
<div className="mb-5">
  <label
    className="block text-sm font-medium text-gray-700 mb-2"
    htmlFor="username"
  >
    Username
  </label>
  <input
    type="text"
    id="username"
    value={username}
    onChange={(e) => setUsername(e.target.value)}
    className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
    required
  />
</div>
<div className="mb-5">
  <label
    className="block text-sm font-medium text-gray-700 mb-2"
    htmlFor="password"
  >
    Password
  </label>
  <input
    type="password"
    id="password"
    value={password}
    onChange={(e) => setPassword(e.target.value)}
    className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
    required
  />
</div>
<button
  type="submit"
  className="w-full bg-blue-600 text-white py-2 rounded-lg hover:bg-blue-700 transition-colors"
>
  Логін
</button>
<div className="mt-4 text-center">
  <a
    href="/forgot-password"
    className="text-blue-500 hover:underline"
  >
    Відновити втрачений пароль
  </a>
</div>
<div className="mt-4 text-center">
  <a
    href="/forgot-password"
    className="text-blue-500 hover:underline"
  >
    Поширенні запитання
  </a>
</div>
</form>
</div>
  )}
</>
);
}
-e

```

```

Файл: src/UniID.API/Controllers/AccountController.cs
using System;
using System.Collections.Generic;
using System.IO;
using System.Net;
using System.Threading.Tasks;
using Campus.API.Infrastructure;
using Campus.API.Models;

```

```

using Campus.Core.Services;
using Campus.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;

namespace Campus.API.Controllers;

/// <summary>
/// Provides functionality to implement actions with user account
/// </summary>
public class AccountController : ControllerBase
{
    private readonly IUserAccountService _userAccountService;
    private readonly IRecaptchaService _recaptchaService;
    private readonly IUserProfileImageService _userProfileImageService;
    private readonly Settings _settings;
    private readonly ILogger _logger;

    public AccountController(
        IRecaptchaService recaptchaService,
        IUserAccountService userAccountService,
        IUserProfileImageService userProfileImageService,
        Settings settings,
        ILogger<AccountController> logger)
    {
        _settings = settings;
        _logger = logger;
        _userProfileImageService = userProfileImageService;
        _recaptchaService = recaptchaService;
        _userAccountService = userAccountService;
    }

    [HttpGet]
    [Route("account")]
    public IReadOnlyCollection<string> Get() => new[] { "All users" };

    /// <summary>
    /// Get a user full name e.g. Lastname Firstname Patronymic
    /// </summary>
    [HttpGet]
    [Route("account/{userAccountId}/photo")]
    [Route("account/{userAccountId}/ProfileImage")]
    public async Task<IActionResult> GetProfileImage(int userAccountId)
    {
        var url = await _userProfileImageService.GetPublicUrl(userAccountId);

        return Redirect(url.ToString());
    }

    [HttpPost]
    [Authorize]
    [Route("account/{userAccountId}/photo")]
    [Route("account/{userAccountId}/ProfileImage")]
    public async Task<IActionResult> UpdatePhoto(UploadMultipartModel model)
    {
        var userAccountId = User.GetUserAccountId();

        if (userAccountId == null)
        {
            return StatusCode((int)HttpStatusCode.Forbidden);
        }

        var stream = model.File.OpenReadStream();

        if (stream.Length == 0 || stream.Length > _settings.MaxUserProfileFileSize)
        {
            return BadRequest("Size limit exceeded");
        }

        using (var memoryStream = new MemoryStream())
        {
            await stream.CopyToAsync(memoryStream);
        }
    }
}

```

```

var bytes = memoryStream.ToArray();

try
{
    await _userProfileImageService.Set(userAccountId.Value, bytes);

    // var url = await _userProfileImageService.GetPublicUrl(userAccountId.Value);
    // return Created(url, new { url, size = bytes.Length });

    return Ok();
}
catch (Exception ex)
{
    _logger.LogError(ex, ex.Message);

    return StatusCode((int)HttpStatusCode.InternalServerError);
}
}
}

/// <summary>
/// Get base user data
/// </summary>
[HttpGet]
[Route("account/{userAccountId}")]
public async Task<IActionResult> GetUser(int userAccountId)
{
    var user = await _userService.GetUser(userAccountId);

    return user != null ? Ok(new UserBase(user)) : NotFound();
}

/// <summary>
/// Get user full public profile
/// </summary>
[HttpGet]
[Route("account/public/{userIdentifier}")]
public async Task<IActionResult> GetPublicProfile(string userIdentifier)
{
    var profile = await _userService.GetPublicProfile(userIdentifier);

    return profile != null ? Ok(profile) : NotFound();
}

/// <summary>
/// Get users public files
/// </summary>
[HttpGet]
[Route("account/public/{userIdentifier}/files")]
public async Task<IActionResult> GetAccountPublicProfileFiles(string userIdentifier)
{
    var result = await _userService.GetUserPublicFiles(userIdentifier);

    return result != null ? Ok(result) : NotFound();
}

/// <summary>
/// Get full information about account.
/// </summary>
/// <returns>information about account</returns>
[HttpGet]
[Route("account/check/{sessionId}")]
[PascalCase]
public async Task<IActionResult> CheckAccount(string sessionId)
{
    var user = await _userService.GetUserBySessionId(sessionId);

    if (user == null)
    {
        return Forbid();
    }
}

```

```

var result = new
{
    User = new
    {
        UserAccountId = user.Id.ToString(),
        Login = user.Username,
        user.FullName,
        UserId = user.Id.ToString()
    }
};

return Ok(result);
}

/// <summary>
/// Get full information about account.
/// </summary>
/// <returns>information about account</returns>
[HttpGet]
[Authorize]
[Route("account/info")]
public async Task<ActionResult> GetAccountFullInformation()
{
    var userAccountId = User.GetUserAccountId();

    if (userAccountId == null)
    {
        return BadRequest();
    }

    try
    {
        var result = await _userService.GetAccountInformation(userAccountId.Value);

        if (result != null)
        {
            return Ok(result);
        }
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }

    return NotFound();
}

/// <summary>
/// Update user information
/// </summary>
/// <returns>information about account</returns>
[HttpPut]
[Authorize]
[Route("account/info")]
public async Task<ActionResult> UpdateAccountInformation([FromBody] AccountInformationUpdateRequest request)
{
    var userAccountId = User.GetUserAccountId();

    if (userAccountId == null)
    {
        return Forbid();
    }

    try
    {
        if (!string.IsNullOrEmpty(request.CurrentPassword) || !string.IsNullOrEmpty(request.Password))
        {
            var updatePasswordResult = await _userService.UpdateUserPassword(
                request.CurrentPassword,
                request.Password,

```

```

        userAccountId.Value);

        if (!updatePasswordResult)
        {
            return Forbid();
        }
    }

    var user = await _userService.GetUser(userAccountId);
    user.Email = request.Email;
    user.ScientificInterest = request.ScientificInterest;
    user.Credo = request.Credo;
    user.FullName = user.FullName;

    var result = await _userService.UpdateUser(user);

    if (result)
    {
        return Ok();
    }
}
catch (Exception ex)
{
    _logger.LogError(ex, ex.Message);

    return StatusCode((int)HttpStatusCode.InternalServerError);
}

return BadRequest();
}

[HttpPost]
[Route("account/recovery")]
public async Task<IActionResult> RecoveryPassword([FromBody] PasswordRecoveryRequest request)
{
    if (string.IsNullOrEmpty(request?.UserIdentifier) || string.IsNullOrEmpty(request.Captcha))
    {
        return BadRequest("UserIdentifier and Captcha is required fields");
    }

    try
    {
        if (await _recaptchaService.VerifyRequest(request.Captcha))
        {
            var user = await _userService.GetUserByEmailOrLogin(request.UserIdentifier);

            if (user == null)
            {
                _logger.LogError($"User not found: {request.UserIdentifier}");

                return NotFound();
            }

            if (string.IsNullOrEmpty(user.Email))
            {
                _logger.LogWarning($"This user has no associated email: {request.UserIdentifier}");

                return Conflict("This user has no associated email.");
            }

            if (await _userService.SendRecoveryLink(user))
            {
                _logger.LogInformation($"Recovery link was sent to user: {request.UserIdentifier}");

                return Accepted("Recovery link was sent.");
            }

            return StatusCode((int)HttpStatusCode.InternalServerError);
        }
    }
    else
    {
        _logger.LogError($"Captcha validation failed for user: {request.UserIdentifier}. Captcha: {request.Captcha}");
    }
}

```

```

        return BadRequest("Invalid captcha");
    }
}
catch (Exception ex)
{
    _logger.LogError(ex, ex.Message);

    return StatusCode((int)HttpStatusCode.InternalServerError);
}

return Forbid();
}

[HttpGet]
[Route("account/recovery/verify/{key}")]
public async Task<IActionResult> VerifyPasswordRecoveryLink(Guid key)
{
    try
    {
        if (await _userService.GenerateAndSendNewPasswordForUser(key))
        {
            return Ok("Mail with new password was sent to your email.");
        }
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }

    return StatusCode((int)HttpStatusCode.Forbidden, "Recovery link already activated or expired. Please request new recovery link.");
}

[HttpGet]
[Route("account/employee/responsibility/{id}")]
public async Task<IActionResult> GetUserResponsibilities(int id)
{
    try
    {
        var result = await _userService.GetUserResponsibilities(id);

        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}
}
}-e

```

```

Файл: src/UniID.API/Controllers/AuthController.cs
using System;
using System.ComponentModel.DataAnnotations;
using System.Net;
using System.Threading.Tasks;
using Campus.API.Infrastructure;
using Campus.API.Models;
using Campus.Core.Services;
using Campus.Core.Services.Email;
using Campus.Models;
using Dapper;
using JetBrains.Annotations;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Cors;
using Microsoft.AspNetCore.Identity.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;

namespace Campus.API.Controllers;

```

```

class TotpValidateRequest
{
    public string Code { get; set; }
}

public class AuthController : ControllerBase
{
    private readonly IUserAccountService _userAccountService;
    private readonly Settings _settings;
    private readonly ILogger _logger;
    private readonly IAuthClientService _authClientService;
    private readonly ITotpService _totpService;
    private readonly IUnildRepository _unildRepository;
    private readonly ISecondFaService _secondFaService;
    private readonly IEmailService _emailService;

    public AuthController(
        IUserAccountService userAccountService,
        Settings settings,
        ILogger<AccountController> logger,
        IAuthClientService authClientService,
        ITotpService totpService,
        IUnildRepository unildRepository,
        ISecondFaService secondFaService,
        IEmailService emailService)
    {
        _settings = settings;
        _logger = logger;
        _userAccountService = userAccountService;
        _authClientService = authClientService;
        _totpService = totpService;
        _unildRepository = unildRepository;
        _secondFaService = secondFaService;
        _emailService = emailService;
    }

    [HttpGet]
    [Route("/auth/client/{clientId}")]
    public IActionResult GetAuthClient(string clientId)
    {
        //TODO: from db
        var client = _authClientService.GetClient(clientId);

        if (client == null)
        {
            return BadRequest("Invalid client ID");
        }

        return Ok(client);
    }

    [HttpPost]
    [Route("/oauth/token")]
    public async Task<IActionResult> Auth(AuthorizationRequest request)
    {
        var client = _authClientService.GetClient(request.ClientId);
        if (client == null)
        {
            return BadRequest("Invalid client ID");
        }

        if (request.RedirectUri != null && !client.RedirectUrls.AsList().Contains(request.RedirectUri))
        {
            return BadRequest("Invalid redirect URL");
        }

        var user = await _userAccountService.Auth(request.Username, request.Password);
        if (user == null)
        {
            return Forbid();
        }

        var userKnown = _unildRepository.GetUser(user.Id);
    }
}

```

```

string token;
string sessionId;

if (userKnown == null)
{
    // Create user record and redirect to 2FA setup
    userKnown = new UniIdUser
    {
        Id = user.Id,
        Username = user.Username,
        CreatedAt = DateTime.UtcNow,
        Email = user.Email,
        IsActive = true,
        LastLogin = DateTime.UtcNow,
        TwoFactorRequired = true
    };

    _uniIdRepository.CreateUser(userKnown);

    token = await _userService.GenerateToken(user);
    sessionId = await _userService.GetUserSessionId(user.Id);

    return Ok(new AuthorizationResponse(token, sessionId, true));
}

if (userKnown.TwoFactorRequired)
{
    //TODO: restrict that token to access only validate endpoint
    var secondFactor = _uniIdRepository.GetTwoFactorMethod(user.Id);

    var tempToken = await _userService.GenerateToken(user);
    return Ok(new Require2FA(tempToken, SecondStepType.Totp));
}

token = await _userService.GenerateToken(user);
sessionId = await _userService.GetUserSessionId(user.Id);

return Ok(new AuthorizationResponse(token, sessionId, false));
}

[HttpPost]
[Authorize]
[Route("/auth/2fa/setup_totp")]
public async Task<ActionResult> SecondFASetupTotp([FromBody] string secondFAType)
{
    var userAccountId = User.GetUserAccountId();
    if (userAccountId == null)
    {
        return Forbid();
    }

    var user = await _userService.GetUser(userAccountId.Value);
    if (user == null)
    {
        return Forbid();
    }

    var response = _secondFaService.Enable((int)userAccountId, SecondFAType.Totp);

    return Ok(response);
}

[HttpPost]
[Authorize]
[Route("/auth/2fa/email")]
public async Task<ActionResult> SecondFASetupSms([FromBody] string userEmail)
{
    var userAccountId = User.GetUserAccountId();
    if (userAccountId == null)
    {
        return Forbid();
    }
}

```

```

var user = await _userService.GetUser(userAccountId.Value);
if (user == null)
{
    return Forbid();
}

_secondFaService.Enable((int)userAccountId, SecondFAType.Email);
var code = _secondFaService.GenerateCode((int)userAccountId);
await _emailService.Send(userEmail, "verification", code);
return Ok();
}

[PublicAPI]
public class ValidateRequest
{
    [Required]
    public string Code { get; set; }
}

[HttpPost]
[Authorize]
[Route("/auth/2fa/validate")]
public async Task<IActionResult> SecondFAValidate([FromBody] ValidateRequest request)
{
    var code = request.Code;
    if (code == null)
    {
        return BadRequest("Missing TOTP code");
    }
    var userAccountId = User.GetUserAccountId();
    if (userAccountId == null)
    {
        return Forbid();
    }

    var user = await _userService.GetUser(userAccountId.Value);
    if (user == null)
    {
        return Forbid();
    }

    if (!_secondFaService.Validate(user.Id, code))
    {
        return BadRequest("Invalid TOTP code");
    }

    var token = await _userService.GenerateToken(user);
    var sessionId = await _userService.GetUserSessionId(user.Id);

    return Ok(new AuthorizationResponse(token, sessionId, false));
}
//TODO: post setup mobile number
//TODO: post setup email

//TODO: submit

//TODO: httppost
//TODO: save secret only if was verified
// [HttpGet]
// [Authorize]
// [Route("/auth/totp/setup")]
// public async Task<IActionResult> TotpSetup()
// {
//     var userAccountId = User.GetUserAccountId();
//     if (userAccountId == null)
//     {
//         return Forbid();
//     }
// }
//
// var user = await _userService.GetUser(userAccountId.Value);
// if (user == null)

```

```

// {
//     return Forbid();
// }
//
// if (!_totpService.HasEnabledTotp(user.Id.ToString()))
// {
//     return BadRequest("TOTP already set up");
// }
//
// var response = _totpService.GenerateTotp(user.Id.ToString());
// return Ok(response);
// }
//
// [HttpPost]
// [Authorize]
// [Route("/auth/totp/validate")]
// public async Task<IActionResult> TotpValidate(string code)
// {
//     //TODO: make as body
//     var userAccountId = User.GetUserAccountId();
//     if (userAccountId == null)
//     {
//         return Forbid();
//     }
//
//     var user = await _userService.GetUser(userAccountId.Value);
//     if (user == null)
//     {
//         return Forbid();
//     }
//
//     if (!_totpService.HasTotp(user.Id.ToString()))
//     {
//         return BadRequest("TOTP not set up");
//     }
//
//     // if (!_totpService.ValidateTotp(user.Id.ToString(), code))
//     // {
//     //     return BadRequest("Invalid TOTP code");
//     // }
//     if (!_totpService.HasEnabledTotp(user.Id.ToString()))
//     {
//         _totpService.SetTotpEnabled(user.Id.ToString(), true);
//     }
//
//     var token = await _userService.GenerateToken(user);
//     var sessionId = await _userService.GetUserSessionId(user.Id);
//
//     return Ok(new AuthorizationResponse(token, sessionId, false));
// }

[HttpGet]
[Route("/auth/check")]
[PascalCase]
// ReSharper disable once InconsistentNaming
public async Task<IActionResult> AuthCheck(string auth_hash)
{
    try
    {
        if (string.IsNullOrEmpty(auth_hash))
        {
            return Forbid();
        }

        var user = await _userService.GetUserBySessionId(auth_hash);

        if (user == null)
        {
            return Forbid();
        }

        var token = await _userService.GenerateToken(user);
        var sessionId = await _userService.GetUserSessionId(user.Id);
    }
}

```

```

        return Ok(new AuthCheckResponse(user, token, sessionId));
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

/// <summary>
/// Set fresh cookies and redirect to old portal.
/// </summary>
/// <returns></returns>
[HttpGet]
[Authorize]
[Route("auth/refresh")]
public async Task<IActionResult> AuthRefresh()
{
    try
    {
        var userAccountId = User.GetUserAccountId();

        var user = await _userService.GetUser(userAccountId);

        if (user == null)
        {
            return StatusCode((int)HttpStatusCode.Forbidden);
        }

        var token = await _userService.GenerateToken(user);
        var sessionId = await _userService.GetUserSessionId(user.Id);

        return Ok(new AuthorizationResponse(token, sessionId));
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

[HttpGet]
[HttpPost]
[Route("auth/logout")]
[Route("account/logout")]
public async Task<IActionResult> Logout()
{
    try
    {
        var userAccountId = User.GetUserAccountId();

        if (userAccountId != null)
        {
            _logger.LogInformation($"Logout user with ID `{userAccountId}`");

            // Clear all user's active sessions
            await _userService.Logout(userAccountId.Value);
        }

        return Ok();
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}
}
-e

```

```

Файл: src/UniID.API/Controllers/GroupController.cs
using System;
using System.Net;
using System.Threading.Tasks;
using Campus.Core;
using Campus.Core.Services;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;

namespace Campus.API.Controllers;

public class GroupController : ControllerBase
{
    private readonly ILogger _logger;
    private readonly IGroupService _groupService;

    public GroupController(IGroupService groupService, ILogger<GroupController> logger)
    {
        _logger = logger;
        _groupService = groupService;
    }

    [Obsolete("Should be used only one endpoint for group searching")]
    [HttpGet("group/find")]
    public async Task<IActionResult> GetGroups(string name = "")
    {
        try
        {
            var result = await _groupService.GetActualStudyGroups(name);

            return Ok(result);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, ex.Message);

            return StatusCode((int)HttpStatusCode.InternalServerError);
        }
    }

    [HttpGet]
    [Route("schedule/groups")]
    public async Task<IActionResult> GetGroupsList([FromQuery] string filter)
    {
        try
        {
            var result = await _groupService.GetGroupsList(filter);

            return Ok(PagedListResult.Create(result));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, ex.Message);

            return StatusCode((int)HttpStatusCode.InternalServerError);
        }
    }
}
-e

```

```

Файл: src/UniID.API/Controllers/ScheduleController.cs
using System;
using System.Collections.Generic;
using System.Net;
using System.Threading.Tasks;
using Campus.Core;
using Campus.Core.Services;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;

namespace Campus.API.Controllers;

[ApiController]
public class ScheduleController : ControllerBase

```

```

{
private readonly ILogger _logger;
private readonly IScheduleService _scheduleService;

public ScheduleController(IScheduleService scheduleService, ILogger<ScheduleController> logger)
{
    _logger = logger;
    _scheduleService = scheduleService;
}

[HttpGet]
[Route("schedule/lessons")]
public async Task<ActionResult> GetLessonsById([FromQuery] string groupId, [FromQuery] string groupName)
{
    try
    {
        if (string.IsNullOrEmpty(groupId) && string.IsNullOrEmpty(groupName))
        {
            return BadRequest($"One of required parameters '{nameof(groupId)}' or '{nameof(groupName)}' missing");
        }

        var result = await _scheduleService.GetLessonsById(groupId, groupName);

        return Ok(PagedListResult.Create(result));
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

[HttpGet]
[Route("schedule/lecturer/list")]
public async Task<ActionResult> GetLecturersList([FromQuery] string filter)
{
    try
    {
        var lecturers = filter != null
            ? await _scheduleService.GetLecturers(filter)
            : await _scheduleService.GetLecturers();

        return Ok(PagedListResult.Create(lecturers));
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

[HttpGet]
[Route("schedule/lecturer/profile/{lecturerId}")]
public async Task<ActionResult> GetLecturerProfile(Guid lecturerId)
{
    try
    {
        var result = await _scheduleService.GetLecturerPublicProfile(lecturerId);

        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

[HttpGet]
[Route("schedule/lecturer")]

```

```

public async Task<IActionResult> GetLessonsList([FromQuery] Guid lecturerId)
{
    try
    {
        var result = await _scheduleService.GetLecturerLessons(lecturerId);

        return Ok(PagedListResult.Create(result));
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

[HttpGet]
[Route("schedule/status")]
public async Task<IActionResult> GetStatus()
{
    try
    {
        var result = await _scheduleService.GetStatus();

        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

[HttpGet]
[Route("exams/group")]
public async Task<IActionResult> GetExamsById([FromQuery] string groupId)
{
    try
    {
        var result = await _scheduleService.ExamWithDeadlines(groupId);

        return Ok(PagedListResult.Create(result));
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}
}
-e

```

Файл: src/UniID.API/Controllers/ServiceController.cs

```

using System;
using System.Net;
using System.Threading.Tasks;
using Campus.Core.Repositories;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;

namespace Campus.API.Controllers;

/// <summary>
/// Auxiliary functions for API methods.
/// </summary>
[ApiController]
public class ServiceController : ControllerBase
{
    private readonly ILogger _logger;
    private readonly IServiceRepository _repository;

    public ServiceController(IServiceRepository repository, ILogger<ServiceController> logger)

```

```

{
    _logger = logger;
    _repository = repository;
}

/// <summary>
/// Get list of profiles.
/// </summary>
/// <returns>list of profiles</returns>
[HttpGet]
[Route("roles")]
public async Task<IActionResult> GetProfiles()
{
    try
    {
        var result = await _repository.GetProfilesAsync();

        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

/// <summary>
/// Get list of blocks.
/// </summary>
/// <returns>list of blocks</returns>
[HttpGet]
[Route("blocks")]
public async Task<IActionResult> GetBlocks()
{
    try
    {
        var result = _repository.GetBlocks();

        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

/// <summary>
/// Get list of cycles.
/// </summary>
/// <returns>list of cycles</returns>
[HttpGet]
[Route("cycles")]
public async Task<IActionResult> GetCycles()
{
    try
    {
        var result = _repository.GetCycles();

        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

/// <summary>
/// Get list of study forms.

```

```

/// </summary>
/// <returns>list of study forms</returns>
[HttpGet]
[Route("study-forms")]
[Route("studyForms")]
public async Task<IActionResult> GetStudyForms()
{
    try
    {
        var result = _repository.GetStudyForms();

        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}
}
}-e

```

Файл: src/UniID.API/Controllers/SomeController.cs  
using Microsoft.AspNetCore.Mvc;  
using System.Threading.Tasks;  
using Campus.API.Models;  
using Campus.Core.Services;  
using JetBrains.Annotations;  
using Microsoft.AspNetCore.Authorization;

```

namespace Campus.API.Controllers;

public class SomeController : Controller
{
    private readonly IUserAccountService _userAccountService;

    public SomeController(IUserAccountService userAccountService)
    {
        _userAccountService = userAccountService;
    }

    [AllowAnonymous]
    [HttpGet("/login")]
    public IActionResult Login(string returnUrl)
    {
        ViewBag.ReturnUrl = returnUrl;
        return View();
    }

    [HttpPost("/login")]
    public async Task<IActionResult> Login(LoginViewModel model)
    {
        // if (!ModelState.IsValid)
        // {
        //     // Return the form with validation errors
        //     return PartialView("LoginPartial", model);
        // }
        //
        // var user = await _userAccountService.AuthenticateAsync(model.Username, model.Password);
        //
        // if (user != null)
        // {
        //     // Sign in the user
        //     await _userAccountService.SignInAsync(HttpContext, user);
        //
        //     // Redirect to the return URL
        //     if (!string.IsNullOrEmpty(model.ReturnUrl))
        //     {
        //         return Redirect(model.ReturnUrl);
        //     }
        //
        //     return RedirectToAction("Index", "Home");
        // }
    }
}

```

```

        ModelState.AddModelError(string.Empty, "Invalid login attempt.");
        return PartialView("LoginPartial", model);
    }
}
-e

```

Файл: src/UniID.API/Controllers/StatisticController.cs

```

using System;
using System.Linq;
using System.Net;
using System.Threading.Tasks;
using Campus.API.Infrastructure;
using Campus.Core.Repositories;
using Campus.Models;
using Campus.Models.Security;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;

namespace Campus.API.Controllers;

/// <summary>
///
/// </summary>
public class StatisticController : ControllerBase
{
    private readonly ILogger _logger;
    private readonly IStatisticRepository _statisticRepository;
    private readonly IAccountRepository _accountRepository;

    public StatisticController(
        IStatisticRepository statisticRepository,
        IAccountRepository accountRepository,
        ILogger<StatisticController> logger)
    {
        _logger = logger;
        _statisticRepository = statisticRepository;
        _accountRepository = accountRepository;
    }

    [HttpGet]
    [Route("Statistic/Subdivisions/{subdivisionId}/Employees/Loaded")]
    [ProfileAuthorization(Profile = Profile.Lecturer)]
    public async Task<ActionResult> GetUserUploadedLoad(int subdivisionId)
    {
        if (!await UserHasPermission(subdivisionId))
        {
            return Forbid();
        }

        try
        {
            var result = _statisticRepository.GetUserUploadedLoad(subdivisionId);

            return Ok(result);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, ex.Message);

            return StatusCode((int)HttpStatusCode.InternalServerError);
        }
    }

    /// <summary>
    /// 1.1a Общее количество ЭИР по кредитным модулям, которые читает данная кафедра для себя
    /// Statistic/Cathedras/{cathedraId}/Modules/ByItself/Count
    /// </summary>
    /// <returns></returns>
    [HttpGet]
    [Route("Statistic/Cathedras/{cathedraId}/Modules/ByItself/Count")]
    [ProfileAuthorization(Profile = Profile.Lecturer)]
    public async Task<ActionResult> GetCatherdaTotalCountCreditModulesByItself(int cathedraId)
    {

```

```

if (!await UserHasPermission(cathedraId))
{
    return Forbid();
}

try
{
    var result = _statisticRepository.GetCatherdaTotalCountCreditModulesByItself(cathedraId);

    return Ok(result);
}
catch (Exception ex)
{
    _logger.LogError(ex, ex.Message);

    return StatusCode((int)HttpStatusCode.InternalServerError);
}
}

private async Task<bool> UserHasPermission(int subdivisionId)
{
    var userAccountId = User.GetUserAccountId();

    if (userAccountId == null)
    {
        return false;
    }

    bool CheckSubsystem(Responsible responsible)
    {
        return responsible.Subsystem == Subsystem.RNP ||
            responsible.Subsystem == Subsystem.KM ||
            responsible.Subsystem == Subsystem.Zavkaf ||
            responsible.Subsystem == Subsystem.Methodical_providing;
    }

    var responsibilities = await _accountRepository.GetUserResponsibilities(userAccountId.Value);
    var hasPermission = responsibilities.Any(o => o.Subdivision.Id == subdivisionId && CheckSubsystem(o));

    return hasPermission;
}

/// <summary>
/// 1.16 Общее количество ЭИР кредитных модулей для которых загружено мат. обесп.
/// Statistic/Cathedras/{cathedraId}/Modules/WithMethodicalmanualByItself/Count
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/WithMethodicalmanualByItself/Count")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<ActionResult> GetCatherdaTotalCountCreditModulesWithMethodicalmanualByItself(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    try
    {
        var result =
            _statisticRepository.GetCatherdaTotalCountCreditModulesWithMethodicalmanualByItself(cathedraId);
        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

/// <summary>
/// 1.1в Общее количество ЭИР кредитных модулей для которых НЕ загружено мат. обесп.

```

```

/// </summary>
/// <returns></returns>
///
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/WithoutMethodicalManualByItself/Count")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetCatherdaTotalCountCreditModulesWithoutMethodicalManualByItself(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    try
    {
        var result =
            _statisticRepository.GetCatherdaTotalCountCreditModulesWithoutMethodicalManualByItself(cathedraId);

        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

/// <summary>
/// 1.3 Кількісна відомість завантажених інф. рес. за видами
/// </summary>
/// <returns>
/// При осутствии значений вернётся {"null": 0}
/// </returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/EIRSByItself/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetEIRSByCreditModulesByTypeByItself(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    try
    {
        var result = _statisticRepository.GetEIRSByCreditModulesByTypeByItself(cathedraId);

        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

/// <summary>
/// 1.2 Количество загруженных информационных ресурсов по кредитным модулям которые кафедра читает сама для себя
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/EIRSByItself/Count")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetTotalEIRCCountByCreditModulesByItself(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    try

```

```

    {
        var number = _statisticRepository.GetEIRSByCreditModulesByTypeByItself(cathedraId);
        var counter = number.ContainsKey("Null") ? 0 : number.Count;

        return Ok(counter);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

/// <summary>
/// 1.4 Список кредитних модулів що читає кафедра сама для себе, та для яких завантажено метод. забезп.
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/WithMethodicalManual/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetCreditModulesWithMethodicalManual(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    try
    {
        var result = _statisticRepository.GetCreditModulesWithMethodicalManual(cathedraId);

        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

/// <summary>
/// 1.5 Список кредитних модулів для яких відсутнє метод. забезпечення. (Новый №1.1)
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/WithoutMethodicalManul/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetCreditModulesWithoutMethodicalManul(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetCreditModulesWithoutMethodicalManul(cathedraId)); // []
}

/// <summary>
/// 1.6 Список кредитних модулів що читає кафедра сама для себе, та які частково забезпечені МЗ
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/WithPartialMethodicalManul/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetCreditModulesWithPartialMethodicalManul(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }
}

```

```

    return Ok(_statisticRepository.GetCreditModulesWithPartialMethodicalManul(cathedraId));
}

/// <summary>
/// 1.7 Список кредитних модулів що читає кафедра сама для себе, та для яких відсутні завантажені файли та посилання
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/WithoutFiles/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<ActionResult> GetCreditModulesWithoutFiles(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetCreditModulesWithoutFiles(cathedraId));
}

/// <summary>
/// 2.1a Кількість КМ, що читають інші кафедри для даної
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/FromForeignCathedras/Count")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<ActionResult> GetCreditModulesCountFromForeignCathedras(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetCreditModulesCountFromForeignCathedras(cathedraId));
}

/// <summary>
/// 2.1b Завантажено МЗ іншими кафедрами для даної
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/MethodicalManual/FromForeignCathedras/Count")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<ActionResult> GetMethodicalManualFromForeignCathedras(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetMethodicalManualFromForeignCathedras(cathedraId));
}

/// <summary>
/// 2.2 Кількість завантажених ЕІР по КМ, що читають інші кафедри для
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/EIRByCreditModules/FromForeignCathedras/Count")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<ActionResult> GetTotalEIRCCountByCreditModulesByForeignCatherdas(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    var number = _statisticRepository.GetEIRSByCreditModulesByTypeByForeignCatherdas(cathedraId);
    var counter = number.ContainsKey("Null") ? 0 : number.Count();
    return Ok(counter);
}

```

```

}

/// <summary>
/// 2.3 Кількісна відомість завантажених ЕІР за видами по КМ, що читають інші кафедри для даної
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/EIRSByCreditModules/FromForeignCathedras/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetEIRSByCreditModulesByTypeByForeignCatherdas(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    try
    {
        var result = _statisticRepository.GetEIRSByCreditModulesByTypeByForeignCatherdas(cathedraId);

        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

/// <summary>
/// 2.4 Список кредитних модулів що читають інші кафедри для даної, та для яких завантажено МЗ
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/WithMethodicalManual/FromForeignCathedras/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetCreditModulesFromForeignCathedrasWithMethodicalManual(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    try
    {
        var result = _statisticRepository.GetCreditModulesFromForeignCathedrasWithMethodicalManual(cathedraId);

        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

/// <summary>
/// 2.5 Список кредитних модулів що читають інші кафедри для даної кафедри, та для яких відсутнє метод. забезпечення. (Новый №2.1)
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/WithoutMethodicalManual/FromForeignCathedras/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetCreditModulesFromForeignCathedrasWithoutMethodicalManul(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetCreditModulesFromForeignCathedrasWithoutMethodicalManul(cathedraId));
}

```

```

}

///<summary>
/// 2.6 Список кредитних модулів що читають інші кафедри для даної кафедри, та для які частково забезпечені МЗ
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/WithPartialMethodicalManual/FromForeignCathedras/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetCreditModulesFromForeignCathedrasWithPartialMethodicalManul(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetCreditModulesFromForeignCathedrasWithPartialMethodicalManul(cathedraId));
}

/// <summary>
/// 2.7 Список кредитних модулів що читають інші кафедри для даної кафедри, та для яких відсутні завантажені файли та посилання
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/WithoutFiles/FromForeignCathedras/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetCreditModulesFromForeignCathedrasWithoutFiles(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetCreditModulesFromForeignCathedrasWithoutFiles(cathedraId));
}

//РОЗДІЛ 3. Статистичні дані по {0}, яка читає іншим кафедрам університету

/// <summary>
/// 3.1а Кількість КМ, що читає кафедра для інших кафедр
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/ForForeignCathedras/Count")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetCreditModulesCountForForeignCathedras(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetCreditModulesCountForForeignCathedras(cathedraId)); // []
}

/// <summary>
/// 3.1б Завантажено МЗ даної кафедри для інших
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/WithMethodicalManual/ForForeignCathedras/Count")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetMethodicalManualForForeignCathedras(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetMethodicalManualForForeignCathedras(cathedraId));
}

```

```

/// <summary>
/// 3.1в НЕ Завантажено МЗ даної кафедри для інших
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/WithoutMethodicalManual/ForForeignCathedras/Count")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetWithoutMethodicalManualForForeignCathedras(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetWithoutMethodicalManualForForeignCathedras(cathedraId));
}

/// <summary>
/// 3.2 Кількість завантажених ЕІР по КМ, що читає дана кафедра для інших кафедр
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/EIRByCreditModules/ForForeignCathedras/Count")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetTotalEIRCountByCreditModulesForForeignCatherdas(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    var number = _statisticRepository.GetEIRSByCreditModulesByTypeForForeignCatherdas(cathedraId);
    var counter = number.ContainsKey("Null") ? 0 : number.Count;

    return Ok(counter); // []
}

/// <summary>
/// 3.3 Кількісна відомість завантажених ЕІР за видами по КМ, що читає дана кафедра для інших кафедр
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/EIRByCreditModules/ForForeignCathedras/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetEIRSByCreditModulesByTypeForForeignCatherdas(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetEIRSByCreditModulesByTypeForForeignCatherdas(cathedraId));
}

/// <summary>
/// 3.4 Список кредитних модулів що читає кафедра для інших та для яких завантажено метод. забезп.
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/WithMethodicalManual/ForForeignCathedras/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetCreditModulesForForeignCathedrasWithMethodicalManual(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    try
    {
        var result = _statisticRepository.GetCreditModulesForForeignCathedrasWithMethodicalManual(cathedraId);

        return Ok(result);
    }
}

```

```

    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

/// <summary>
/// 3.5 Список кредитних модулів що читає дана кафедра для інших кафедр, та для яких відсутнє метод. забезпечення. (Новий №3.1)
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/WithoutMethodicalManual/ForForeignCathedras/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetCreditModulesForForeignCathedrasWithoutMethodicalManul(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetCreditModulesForForeignCathedrasWithoutMethodicalManul(cathedraId));
}

/// <summary>
/// 3.6 Список кредитних модулів що читає дана кафедра для інших, та які частково забезпечені МЗ
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/WithPartialMethodicalManual/ForForeignCathedras/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetCreditModulesForForeignCathedrasWithPartialMethodicalManul(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetCreditModulesForForeignCathedrasWithPartialMethodicalManul(cathedraId));
}

/// <summary>
/// 3.7 Список кредитних модулів що читає дана кафедра для інших, та для яких відсутні завантажені файли та посилання
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Modules/WithoutFiles/ForForeignCathedras/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetCreditModulesForForeignCathedrasWithoutFiles(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetCreditModulesForForeignCathedrasWithoutFiles(cathedraId));
}

// РОЗДІЛ 4

/// <summary>
/// 4.1 Статистичні дані по співробітникам, яким надано доступ до МЗ {0} (назва кафедри)
/// </summary>
/// <param name="cathedraId"></param>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Employers/WithMethodicalManualPermission/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetStatisticByEmployersWithMethodicalManualPermission(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))

```

```

    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetStatisticByEmployersWithMethodicalManualPermission(cathedraId));
}

/// РОЗДІЛ 5
/// <summary>
/// 5 Статистичні дані по співробітникам, які завантажили МЗ для {0} (назва кафедри)
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Employers/WithMethodicalManual/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetStatisticByEmployersWithMethodicalManual(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetStatisticByEmployersWithMethodicalManual(cathedraId));
}

/// Розділ 4
/// <summary>
/// 4. Індивідуальне навантаження
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/Employers/WithIndividualLoad/List")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetEmployeeWhoUploadIndividualLoad(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetEmployeeWhoUploadIndividualLoad(cathedraId));
}

/// <summary>
/// РОЗДІЛ 1. Статистичні дані по кафедрі, що сама себе забезпечує, тобто сама собі читає кредитні модулі(KM).
/// </summary>
/// <param name="cathedraId"></param>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/StatisticForFirstSubjectProviderSection")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetStatisticForFirstSubjectProviderSection(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {
        return Forbid();
    }

    return Ok(_statisticRepository.GetStatisticForFirstSubjectProviderSection(cathedraId));
}

/// <summary>
/// РОЗДІЛ 2. Статистичні дані по кафедрі, для якої читають інші кафедри університету.
/// </summary>
/// <param name="cathedraId"></param>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedraId}/StatisticForSecondSubjectProviderSection")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<IActionResult> GetStatisticForSecondSubjectProviderSection(int cathedraId)
{
    if (!await UserHasPermission(cathedraId))
    {

```

```

        return Forbid();
    }

    try
    {
        var result = _statisticRepository.GetStatisticForSecondSubjectProviderSection(cathedralId);

        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}

/// <summary>
/// РОЗДІЛ 3. Статистичні дані по кафедрі, яка читає іншим кафедрам університету.
/// </summary>
/// <param name="cathedralId"></param>
/// <returns></returns>
[HttpGet]
[Route("Statistic/Cathedras/{cathedralId}/StatisticForThirdSubjectProviderSection")]
[ProfileAuthorization(Profile = Profile.Lecturer)]
public async Task<ActionResult> GetStatisticForThirdSubjectProviderSection(int cathedralId)
{
    if (!await UserHasPermission(cathedralId))
    {
        return Forbid();
    }

    try
    {
        var result = _statisticRepository.GetStatisticForThirdSubjectProviderSection(cathedralId);

        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, ex.Message);

        return StatusCode((int)HttpStatusCode.InternalServerError);
    }
}
}-e

```

Файл: src/UniID.API/Controllers/SystemController.cs  
using Microsoft.AspNetCore.Mvc;

namespace Campus.API.Controllers;

```

/// <summary>
/// General information about API
/// </summary>
[ApiController]
public class SystemController : ControllerBase
{
    private readonly Settings _settings;

    public SystemController(Settings settings)
    {
        _settings = settings;
    }

    [HttpGet]
    [Route("")]
    public ActionResult Get()
    {
        return Ok(new
        {
            Name = "campus API",
            _settings.Build,

```

```

        _settings.Environment
    });
}
}-e

```

Файл: src/UniID.API/Infrastructure/ClaimsPrincipalExtensions.cs

```

using System.Security.Claims;
using Campus.Core.Infrastructure;

```

```

namespace Campus.API.Infrastructure;

```

```

public static class ClaimsPrincipalExtensions
{
    public static int? GetUserAccountId(this ClaimsPrincipal user)
    {
        if (user == null)
        {
            return null;
        }

        var accountId = int.TryParse(user.FindFirstValue(TokenClaim.Id), out var result) ? result : (int?)null;

        return accountId;
    }
}
}-e

```

Файл: src/UniID.API/Infrastructure/PascalCase.cs

```

using System;
using System.Text.Json;
using System.Text.Json.Serialization.Metadata;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Filters;
using Microsoft.AspNetCore.Mvc.Formatters;

```

```

namespace Campus.API.Infrastructure;

```

```

[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method)]
public class PascalCase : ActionFilterAttribute
{
    public override void OnActionExecuted(ActionExecutedContext context)
    {
        if (context?.Result == null)
        {
            return;
        }

        var options = new JsonSerializerOptions
        {
            PropertyNamingPolicy = null,
            WriteIndented = true,
            TypeInfoResolver = new DefaultJsonTypeInfoResolver()
        };

        if (context.Result is OkObjectResult result)
        {
            result.Formatters.Clear();
            result.Formatters.Add(new SystemTextJsonOutputFormatter(options));
        }
    }
}
}-e

```

Файл: src/UniID.API/Infrastructure/ProfileAuthorizationAttribute.cs

```

using System;
using System.Linq;
using Campus.Models.Security;
using Campus.Core.Services;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Filters;
using Microsoft.Extensions.DependencyInjection;

```

```

namespace Campus.API.Infrastructure;

```

```

[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, AllowMultiple = true)]

```

```

public class ProfileAuthorizationAttribute : Attribute, IAuthorizationFilter
{
    public Profile Profile { get; set; }

    public void OnAuthorization(AuthorizationFilterContext authorizationFilterContext)
    {
        var context = authorizationFilterContext.HttpContext;

        var userManager = context.RequestServices.GetService<IUserAccountService>();
        var userId = context.User.GetUserId();

        if (userId.HasValue)
        {
            var profiles = userManager.GetUserProfiles(userId.Value);

            if (profiles.Any(profile => profile.Profile == this.Profile))
            {
                return;
            }
        }

        authorizationFilterContext.Result = new ForbidResult();
    }
}

```

Файл: src/UniID.API/Models/AccountInformationUpdateRequest.cs  
using JetBrains.Annotations;

namespace Campus.API.Models;

```

[PublicAPI]
public record AccountInformationUpdateRequest
{
    public string Password { get; set; }
    public string CurrentPassword { get; set; }
    public string Email { get; set; }
    public string FullName { get; set; }
    public string Credo { get; set; }
    public string ScientificInterest { get; set; }
}

```

Файл: src/UniID.API/Models/AuthCheckResponse.cs  
using System.Collections.Generic;  
using Campus.Models;  
using JetBrains.Annotations;

namespace Campus.API.Models;

```

[PublicAPI]
public record AuthCheckResponse
{
    public AuthCheckResponseUser User { get; set; }
    public string Token { get; }
    public string SessionId { get; }

    public AuthCheckResponse(IUser user, string token, string sessionId)
    {
        Token = token;
        SessionId = sessionId;
        User = new AuthCheckResponseUser(user);
    }
}

```

```

[PublicAPI]
public record AuthCheckResponseUser
{
    public AuthCheckResponseUser(IUser user)
    {
        UserId = user.Id;
        Login = user.Username;
        FullName = user.FullName;
        Email = user.Email;
        Groups = new List<string>();
    }
}

```

```

    }

    public int UserAccountId { get; set; }
    public string Login { get; set; }
    public string FullName { get; set; }
    public List<string> Groups { get; set; }
    public string EMail { get; set; }
}-e

```

```

Файл: src/UniID.API/Models/AuthorizationRequest.cs
using System.ComponentModel.DataAnnotations;
using System.Linq;
using JetBrains.Annotations;

```

```

namespace Campus.API.Models;

```

```

[PublicAPI]
public record AuthorizationRequest
{
    [Required]
    public string ClientId { get; init; }
    [Required]
    public string RedirectUri { get; init; }
    [Required]
    public string Username { get; set; }
    [Required]
    public string Password { get; set; }
    public string TotpCode { get; set; }

    public bool BehalfAnotherUser => Username?.Contains(":") ?? false;

    public BehalfAnotherUserAuthorizationRequest ToBehalfAnotherUserAuthorizationRequest() =>
        !BehalfAnotherUser
        ? null
        : new BehalfAnotherUserAuthorizationRequest
        {
            Password = Password,
            Username = Username.Split(':').ElementAt(0),
            BehalfUsername = Username.Split(':').ElementAt(1)
        };
}

```

```

[PublicAPI]
public record BehalfAnotherUserAuthorizationRequest
{
    public string BehalfUsername { get; set; }
    public string Username { get; set; }
    public string Password { get; set; }
}-e

```

```

Файл: src/UniID.API/Models/AuthorizationResponse.cs
using System.Text.Json.Serialization;
using JetBrains.Annotations;

```

```

namespace Campus.API.Models;

```

```

[PublicAPI]
public record AuthorizationResponse
{
    public AuthorizationResponse()
    {
    }

    public AuthorizationResponse(string token, string sessionId)
    {
        AccessToken = token;
        SessionId = sessionId;
    }

    public AuthorizationResponse(string token, string sessionId, bool require2Fa)
    {
        AccessToken = token;
        SessionId = sessionId;
    }
}

```

```

    Require2Fa = require2Fa;
}

[JsonPropertyName("access_token")]
public string AccessToken { get; set; }

[JsonPropertyName("sessionId")]
public string SessionId { get; set; }

[JsonPropertyName("token_type")]
public string TokenType => "bearer";

[JsonPropertyName("expires_in")]
public int ExpiresIn => 86399;

[JsonPropertyName("require2fa")]
public bool Require2Fa { get; set; }
}-e

```

Файл: src/UniID.API/Models/EmploymentSystemValidationResponse.cs  
using Newtonsoft.Json;

```

namespace Campus.API.Models;

public record EmploymentSystemValidationResponse
{
    public string FullName { get; }
    public int UserAccountId { get; }
    public string StudentId { get; }

    public EmploymentSystemValidationResponse()
    {
    }

    public EmploymentSystemValidationResponse(string fullName, int userAccountId, string studentId)
    {
        FullName = fullName;
        UserAccountId = userAccountId;
        StudentId = studentId;
    }

    /// <summary>
    /// This serialization use specific naming convention: field names should be equal to DB column names.
    /// </summary>
    /// <returns></returns>
    public string ToJson()
    {
        var obj = new
        {
            FullName,
            UserAccountId,
            STUDENT_ID = StudentId
        };

        var json = JsonConvert.SerializeObject(obj);

        return json;
    }
}-e

```

Файл: src/UniID.API/Models/EnforceTotpResponse.cs  
using System.Text.Json.Serialization;  
using JetBrains.Annotations;

```

namespace Campus.API.Models;

public enum SecondStepType
{
    Totp,
    Sms,
    Email
}

```

```
[PublicAPI]
public class Require2FA
{
    public Require2FA(string tempToken, SecondStepType secondStepType)
    {
        TempToken = tempToken;
        SecondStepType = secondStepType.ToString().ToLower();
    }

    [JsonPropertyName("temp_token")]
    public string TempToken { get; set; }
    [JsonPropertyName("second_step_type")]
    public string SecondStepType { get; set; }
}-e
```

Файл: src/UniID.API/Models/LoginViewModel.cs  
using System.ComponentModel.DataAnnotations;

```
namespace Campus.API.Models;

public class LoginViewModel
{
    [Required]
    public string Username { get; set; }

    [Required]
    public string Password { get; set; }

    public string returnUrl { get; set; }
}
-e
```

Файл: src/UniID.API/Models/PasswordRecoveryRequest.cs  
using System.ComponentModel.DataAnnotations;

```
namespace Campus.API.Models;

public record PasswordRecoveryRequest
{
    [Required]
    public string UserIdentifier { get; set; }

    [Required]
    public string Captcha { get; set; }
}-e
```

Файл: src/UniID.API/Models/UploadMultipartModel.cs  
using JetBrains.Annotations;  
using Microsoft.AspNetCore.Http;

```
namespace Campus.API.Models;

[PublicAPI]
public record UploadMultipartModel
{
    public IFormFile File { get; set; }

    public string Name { get; set; }
}-e
```

Файл: src/UniID.API/Program.cs  
using System.Collections.Generic;  
using System.Text;  
using System.Text.Json.Serialization;  
using Campus.API;  
using Campus.Core;  
using Campus.Core.Data;  
using Campus.Core.Infrastructure;  
using Campus.Core.Mappers;  
using Campus.Core.Repositories;  
using Campus.Core.Services;  
using Campus.Core.Services.Email;  
using Campus.Core.Services.Social;

```

using Campus.DAL;
using Campus.DAL.Queries;
using Campus.Storage;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.HttpOverrides;
using Microsoft.Extensions.Caching.Distributed;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;
using Microsoft.OpenApi.Models;
using Microsoft.Extensions.Hosting;
using Microsoft.IdentityModel.Tokens;
using UniID.Core.Mappers;

const string CorsPolicy = "cors-policy";
const string SwaggerDocumentName = "api_description";

static Settings LoadSettings(IConfiguration configuration)
{
    var settings = new Settings();
    configuration.Bind(settings);
    settings.ConnectionString = configuration.GetConnectionString("DefaultConnection");

    return settings;
}

var builder = WebApplication.CreateBuilder(args);

builder.Configuration
    .AddEnvironmentVariables()
    .AddJsonFile("appsettings.local.json", true);

var settings = LoadSettings(builder.Configuration);

builder.Services.AddLogging(b =>
{
    b.ClearProviders();
    b.AddSimpleConsole();
});

builder.Services.AddDistributedMemoryCache(options => { });

builder.Services.AddAuthentication();

builder.Services
    .AddSingleton<Settings>(settings)
    .AddSingleton<QueryManager>(p =>
    {
        var logger = p.GetService<ILogger<QueryManager>>();
        return new QueryManager(builder.Environment.ContentRootPath, logger);
    })
    .AddSingleton<AccountQueryManager>()
    .AddSingleton<BulletinBoardQueryManager>()
    .AddSingleton<FileQueryManager>()
    .AddSingleton<GroupQueryManager>()
    .AddSingleton<IntellectQueryManager>()
    .AddSingleton<StatisticQueryManager>()
    .AddSingleton<ServiceQueryManager>()
    .AddSingleton<SubdivisionQueryManager>()
    .AddScoped<IEmploymentSystemSessionManager, EmploymentSystemSessionManager>()
    .AddScoped<IAccountRepository, AccountRepository>()
    .AddScoped<IntellectRepository, IntellectRepository>()
    .AddScoped<ISubdivisionRepository, SubdivisionRepository>()
    .AddScoped<IStatisticRepository, StatisticRepository>()
    .AddScoped<IServiceRepository, ServiceRepository>()
    .AddScoped<IFileRepository, FileRepository>()
    .AddScoped<IBulletinRepository, BulletinRepository>()
    .AddScoped<IUserAccountService>(p =>
    {
        var accountRepository = p.GetService<IAccountRepository>();
        var emailService = p.GetService<IEmailService>();
    }

```

```

var jwtTokenGenerator = p.GetService<IJwtTokenGenerator>();
var hashingService = p.GetService<IHashingService>();
var logger = p.GetService<ILogger<UserAccountService>>();
var cache = p.GetService<IDistributedCache>();

return new UserAccountService(
    accountRepository,
    emailService,
    jwtTokenGenerator,
    hashingService,
    settings.CanAuthBehalfAnotherUser,
    settings.RecoveryLinkTTL,
    settings.ApiEndpoint,
    cache,
    logger);
})
.AddScoped<ISubdivisionService, SubdivisionService>()
.AddScoped<IScheduleService, ScheduleService>()
.AddScoped<IGroupService, GroupService>()
.AddScoped<IBulletinService, BulletinService>()
.AddScoped<IRecaptchaService>(ctx => new RecaptchaService(settings.ReCaptchaSecret))
.AddScoped<IAuthClientService, AuthClientService>()
.AddScoped<IEmailService>(p =>
{
    // return new AWSRawEmailService(
    //     settings.AWSESSES.Sender,
    //     settings.AWSESSES.AccessKeyId,
    //     settings.AWSESSES.SecretAccessKey,
    //     settings.AWSESSES.Region,
    //     p.GetService<ILogger<AWSEmailService>>());

    return new SmtplibEmailService(
        settings.Smtplib.Sender,
        settings.Smtplib.Host,
        settings.Smtplib.Login,
        settings.Smtplib.Password,
        settings.Smtplib.Port,
        settings.Smtplib.EnableSsl,
        p.GetService<ILogger<SmtplibEmailService>>());
})
.AddScoped<IJwtTokenGenerator>(p =>
{
    return new JwtTokenGenerator(settings.Jwt.Secret, settings.Jwt.Issuer, settings.Jwt.Audience);
})
.AddScoped<IStorage>(p =>
{
    return new S3Storage(
        settings.AWS.S3BucketName,
        settings.AWS.CDNRoot,
        settings.AWS.Region,
        settings.AWS.AccessKeyId,
        settings.AWS.SecretAccessKey,
        p.GetService<ILogger<S3Storage>>());
})
.AddScoped<ITelegramService>(p => new TelegramService(settings.Telegram.BotToken))
.AddScoped<IHashingService, HashingService>()
.AddScoped<IUserProfileImageService>(p =>
{
    return new UserProfileImageService(
        p.GetService<IStorage>(),
        p.GetService<IDistributedCache>(),
        settings.UserProfileImagePlaceholderUrl,
        p.GetService<ILogger<UserProfileImageService>>());
})
.AddSingleton<IDatabase>(p =>
{
    return new MySqlDatabase(settings.ConnectionString, p.GetService<ILogger<MySqlDatabase>>());
})
.AddSingleton<UserProfileImageBuilder>(p => { return new UserProfileImageBuilder(settings.ApiEndpoint); })
.AddSingleton<ResponsibleMapper>()
.AddSingleton<FileMapper>()
.AddSingleton<PositionMapper>()
.AddSingleton<StatisticMapper>()

```

```

.AddSingleton<GroupedEmployeeListMapper>()
.AddSingleton<SubdivisionMapper>()
.AddSingleton<GroupInfoMapper>()
.AddSingleton<ItemMapper>()
.AddSingleton<UserMapper>()
.AddSingleton<PublicProfileMapper>()
.AddSingleton<ScheduleQueryManager>()
.AddSingleton<ScheduledMapper>()
.AddSingleton<LecturerMapper>()
.AddSingleton<GroupMapper>()
.AddSingleton<FacultyMapper>()
.AddSingleton<ITotpService, TotpService>()
.AddSingleton<ISecndFAService, SecndFAService>()
.AddScoped<IScheduleRepository, ScheduleRepository>(p =>
{
    var logger = p.GetService<ILogger<ScheduleQueryManager>>();
    var databaseLogger = p.GetService<ILogger<PostgresDatabase>>();

    var database = new PostgresDatabase(settings.Schedule.ConnectionString, databaseLogger);
    var queryManager = p.GetService<QueryManager>();
    var scheduleQueryManager = new ScheduleQueryManager(queryManager);
    var scheduledMapper = new ScheduledMapper();
    var lecturerMapper = new LecturerMapper();
    var groupMapper = new GroupMapper();
    var facultyMapper = new FacultyMapper();

    return new ScheduleRepository(
        database,
        scheduleQueryManager,
        scheduledMapper,
        lecturerMapper,
        groupMapper,
        logger,
        facultyMapper);
})
.AddScoped<IUnidRepository, UnidRepository>(p =>
{
    var logger = p.GetService<ILogger<UnidRepository>>();
    var databaseLogger = p.GetService<ILogger<PostgresDatabase>>();

    var database = new PostgresDatabase(settings.Unid.ConnectionStrings, databaseLogger);
    var unidQueryManager = new UnidQueryManager();
    var unidMapper = new UnidMapper();

    return new UnidRepository(database, unidQueryManager, unidMapper, logger);
})
.AddScoped<ScheduleWeekService>(p => new ScheduleWeekService(settings.Schedule));

builder.Services.AddControllers().AddJsonOptions(options =>
{
    options.JsonSerializerOptions.Converters.Add(new JsonStringEnumConverter());
});

builder.Services.AddCors(options =>
{
    options.AddPolicy(CorsPolicy, policyBuilder =>
    {
        policyBuilder
            .AllowAnyHeader()
            .AllowAnyMethod()
            .AllowAnyOrigin()
            ;
    });
});

builder.Services
.AddAuthentication(config =>
{
    config.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    config.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{

```

```

options.SaveToken = true;
options.RequireHttpsMetadata = false;

options.TokenValidationParameters = new TokenValidationParameters
{
    ValidateIssuer = true,
    ValidateAudience = true,
    ValidateLifetime = true,
    ValidateIssuerSigningKey = true,
    ValidIssuer = settings.Jwt.Issuer,
    ValidAudience = settings.Jwt.Audience,
    IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(settings.Jwt.Secret))
};
});

// Add services to the container.
builder.Services.AddControllersWithViews();

if (settings.EnableSwagger)
{
    builder.Services.AddSwaggerGen(options =>
    {
        var info = new OpenApiInfo { Title = "campus API", Version = settings.Build };

        options.SwaggerDoc(SwaggerDocumentName, info);

        options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
        {
            Name = "Authorization",
            Description = "JWT Authorization header using the Bearer scheme. \r\n\r\n " +
                "Enter 'Bearer' [space] and then your token in the text input below.\r\n\r\n " +
                "Example: 'Bearer 12345abcdef'",
            In = ParameterLocation.Header,
            Type = SecuritySchemeType.ApiKey,
            Scheme = "Bearer"
        });

        options.AddSecurityRequirement(new OpenApiSecurityRequirement()
        {
            {
                new OpenApiSecurityScheme
                {
                    Reference = new OpenApiReference { Type = ReferenceType.SecurityScheme, Id = "Bearer" },
                    Scheme = "oauth2",
                    Name = "Bearer",
                    In = ParameterLocation.Header
                },
                new List<string>()
            }
        });
    });
}

var app = builder.Build();

app.UseRouting();

var isDevelopment = app.Environment.IsDevelopment();

if (isDevelopment)
{
    app.UseDeveloperExceptionPage();
}

if (settings.EnableSwagger)
{
    // Enable middleware to serve generated Swagger as a JSON endpoint.
    app.UseSwagger();

    // Enable middleware to serve swagger-ui-0pp (HTML, JS, CSS, etc.),
    // specifying the Swagger JSON endpoint.
    app.UseSwaggerUI(c => { c.SwaggerEndpoint($"/swagger/{SwaggerDocumentName}/swagger.json", "campus API"); });
}

```

```

app.UseForwardedHeaders(new ForwardedHeadersOptions
{
    ForwardedHeaders = ForwardedHeaders.XForwardedFor | ForwardedHeaders.XForwardedProto
});

app.UseAuthentication();
app.UseAuthorization();

app.UseCors(CorsPolicy);

app.UseEndpoints(endpoints =>
{
    endpoints.MapControllers();
});

app.Run();-e

    Файл: src/UniID.API/Settings.cs
using System;
using System.Collections.Generic;
using System.Collections.Immutable;
using System.IO;
using Campus.Models.Schedule;

namespace Campus.API;

/// <summary>
/// Class that provides access to configuration setting.
/// </summary>
public sealed record Settings
{
    public const int DefaultPageSize = 30;

    public string Build
    {
        get
        {
            var revision = File.ReadAllText("revision.txt").Trim();

            return revision;
        }
    }

    public IReadOnlyCollection<string> AppDomains { get; set; }

    public string Environment { get; set; }

    public TimeSpan RecoveryLinkTTL => TimeSpan.FromMinutes(20);

    public string ApiEndpoint { get; set; }
    public long MaxUserProfileFileSize { get; set; }

    public AWSS3Settings AWS { get; set; } = new();

    public AWSESSettings AWSESE { get; set; } = new();

    public string ReCaptchaSecret { get; set; }

    public SmtSettings Smt { get; set; } = new();

    public Uri UserProfileImagePlaceholderUrl { get; set; }

    public string ConnectionString { get; set; }

    public TelegramSettings Telegram { get; set; } = new();

    public JwtSettings Jwt { get; set; } = new();

    public IReadOnlyCollection<int> CanAuthBehalfAnotherUser { get; set; } = ImmutableList<int>.Empty;

    public string QueryDirectory { get; set; }

```

```

public EmploymentSystemSettings EmploymentSystem { get; set; } = new();

public ScheduleSettings Schedule { get; set; }
public UniIdSettings UniId { get; set; }

public bool EnableSwagger { get; set; }

public class EmploymentSystemSettings
{
    public string AuthEndpoint { get; set; } = "";
    public List<string> WhiteList { get; set; } = new();
}

public class TelegramSettings
{
    public string BotToken { get; set; }
}

public class JwtSettings
{
    public string Issuer { get; set; }
    public string Audience { get; set; }
    public string Secret { get; set; }

    public TimeSpan AccessTokenExpire => TimeSpan.FromDays(1);
}

public class SntpSettings
{
    public string Host { get; set; }
    public string Sender { get; set; }
    public string Password { get; set; }
    public int Port { get; set; } = 465;
    public bool EnableSsl { get; set; }
    public string Login { get; set; }
}

public record AWSS3Settings
{
    public string AccessKeyId { get; set; }
    public string SecretAccessKey { get; set; }
    public string Region { get; set; }
    public string CDNRoot { get; set; }
    public string S3BucketName => "kpi-public-files";
}

/// <summary>
/// Settings for AWS Simple Email Service
/// </summary>
public record AWSSSESSettings
{
    public string Sender { get; set; }
    public string AccessKeyId { get; set; }
    public string SecretAccessKey { get; set; }
    public string Region { get; set; }
}

public record UniIdSettings
{
    public string ConnectionStrings { get; set; }
}
}-e

```

ДОДАТОК Ж

Результати перевірки роботи на співпадіння

## Звіт подібності

## метадані

Заголовок

ІП-32мп\_Яцевський\_ПЗ

Науковий корізіник / Експорт

Автор






Фіногенов О.Д.

підрозділ

ФІОТ, К-а інформатики та програмної інженерії

## Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		1
Інтервали		0
Мікропробіли		1
Білі знаки		43
Парафрази (SmartMarks)		0

## Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



10

Довжина фрази для коефіцієнта подібності 2

13776

Кількість слів


111370



Кількість символів

## Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз		Колір тексту	
ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	<a href="https://www.oksim.ua/2023/12/08/10-najkrashnih-lms-dlya-osvitnih-zakladiv/">https://www.oksim.ua/2023/12/08/10-najkrashnih-lms-dlya-osvitnih-zakladiv/</a>	11	0.08 %
2	<a href="https://naurok.com.ua/realizaciya-principiv-pedagogiki-partnerstva-v-umovah-distancijnogo-navchannya-332507.html">https://naurok.com.ua/realizaciya-principiv-pedagogiki-partnerstva-v-umovah-distancijnogo-navchannya-332507.html</a>	6	0.04 %
3	<a href="https://pedscience.spu.edu.ua/wp-content/uploads/2024/09/%D0%92%D0%BE%D0%BB%D0%BE%D1%88%D0%B8%D0%BD.pdf">https://pedscience.spu.edu.ua/wp-content/uploads/2024/09/%D0%92%D0%BE%D0%BB%D0%BE%D1%88%D0%B8%D0%BD.pdf</a>	5	0.04 %

з домашньої бази даних (0.00 %) 

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
<b>з програми обміну базами даних (0.00 %)</b>			
ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
<b>з Інтернету (0.16 %)</b>			
ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	<a href="https://www.okslm.ua/2023/12/08/10-najkrashhih-ims-dlya-osvitnih-zakladiv/">https://www.okslm.ua/2023/12/08/10-najkrashhih-ims-dlya-osvitnih-zakladiv/</a>	11 (1)	0.08 %
2	<a href="https://naurok.com.ua/realizaciya-principiv-pedagogiki-partnerstva-v-umovah-distancijnogo-navchannya-332507.html">https://naurok.com.ua/realizaciya-principiv-pedagogiki-partnerstva-v-umovah-distancijnogo-navchannya-332507.html</a>	6 (1)	0.04 %
3	<a href="https://pedscience.sspu.edu.ua/wp-content/uploads/2024/09/%D0%92%D0%BE%D0%BB%D0%BE%D1%88%D0%B8%D0%BD.pdf">https://pedscience.sspu.edu.ua/wp-content/uploads/2024/09/%D0%92%D0%BE%D0%BB%D0%BE%D1%88%D0%B8%D0%BD.pdf</a>	5 (1)	0.04 %

### Список прийнятих фрагментів

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)	
	<a href="https://naurok.com.ua/realizaciya-principiv-peda...">https://naurok.com.ua/realizaciya-principiv-peda...</a>	6 (0.04%)	
1	Також Google Workspace for Education пропонує	6 (0.04%)	