

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
МЕТОДІВ ІНТЕГРАЦІЇ З СИСТЕМОЮ ДИСТАНЦІЙНОГО
НАВЧАННЯ MOODLE
RESEARCH AND SOFTWARE IMPLEMENTATION
OF METHODS FOR INTEGRATION WITH MOODLE
DISTANCE LEARNING SYSTEM

Виконав: студент(ка) 2 курсу, групи КНТ-219м
Спеціальності _____

122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

Давиденко Р.Б.

(прізвище та ініціали)

Керівник Каплієнко Т.І.

(прізвище та ініціали)

Рецензент Шитікова О.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 122 Комп'ютерні науки
 (код і найменування)
 Освітня програма (спеціалізація) Системи штучного інтелекту
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Давиденка Родіона Борисовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів інтеграції з системою дистанційного навчання Moodle. Research and Software Implementation of Methods for Integration with Moodle Distance Learning System

керівник проєкту (роботи) Каплієнко Тетяна Ігорівна, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “03” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) 14 грудня 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Аналіз і специфікація вимог. 3. Матеріали і методи. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програми. 6. Економіко-організаційна частина. 7. Охорона праці та безпека в надзвичайних ситуаціях. 8. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Каплієнко Т.І., доцент		
6 Економіко-організаційна частина	Пожуєва Т.О., професор		
7 Охорона праці та безпека в надзвичайних ситуаціях	Коробко О.В., ст. викладач		
Нормоконтролер	Андрєєв М.О., асистент		

7. Дата видачі завдання “ 4 ” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2-3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4-5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7-8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 5
7	Розробка економіко-організаційної частини.	10 тиждень	Розділ 6
8	Розробка заходів з охорони праці та безпеки в надзвичайних ситуаціях.	11 тиждень	Розділ 7
9	Оформлення пояснювальної записки та документів до неї.	12-13 тиждень	Додатки
10	Нормоконтроль та рецензування.	14-15 тижні	
11	Захист роботи.	16 тиждень	

Студент(ка)

Давиденко Р.Б.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

Каплієнко Т.І.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
118 с., 12 табл., 29 рис., 2 дод., 26 джерел.

ІНТЕГРАЦІЯ ІНФОРМАЦІЙНИХ СИСТЕМ, СИСТЕМА
ДИСТАНЦІЙНОЇ ОСВІТИ MOODLE, ПРОТОКОЛ ВЗАЄМОДІЇ, SOAP,
REST.

Об'єкт дослідження – методи інтеграції інформаційних систем.

Предмет дослідження – методи інтеграції з системою дистанційної освіти Moodle.

Мета роботи – дослідження та програмна реалізація методів інтеграції з системою дистанційного навчання Moodle для спрощення та автоматизації процесів взаємодії.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, інтерфейс прикладного програмування Application Programming Interface, протокол передачі даних Simple Object Access Protocol, мова програмування PHP, персональний комп'ютер під управлінням операційної системи Microsoft Windows 7 або Linux.

Результати. Створено інтеграційну систему для передачі даних через Simple Object Access Protocol до методів Moodle через Application Programming Interface, що отримує запити через HTTP протокол, оброблює їх та спрощує взаємодію з системою дистанційного навчання Moodle.

Висновки. Розроблено програмну систему для реалізації методів інтеграції з СДН Moodle для спрощення та автоматизації процесів взаємодії з нею.

Галузь використання – автоматизація взаємодії з системою дистанційного навчання Moodle у закладах вищої освіти.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 118 pages, 12 figures, 29 tables, 2 appendixes, 26 sources.

INTEGRATION OF INFORMATION SYSTEMS, DISTANCE EDUCATION SYSTEM MOODLE, INTERACTION PROTOCOL, SOAP, REST.

Object of study are methods of information systems integration.

Subject of study are methods of integration with the distance education system Moodle.

The purpose of the work is research and software implementation of methods of integration with the distance learning system Moodle to simplify and automate the interaction processes.

Materials, methods, and tools: structured and object-oriented programming, Application Programming Interface, Simple Object Access Protocol, PHP programming language, personal computer running Microsoft Windows 7 or Linux.

Results. An integration system for data transfer via Simple Object Access Protocol to Moodle methods via Application Programming Interface has been created, which receives requests via HTTP protocol, processes them and simplifies interaction with the Moodle distance learning system.

Conclusions A software system A software system for the integration methods implementation with the distance education system Moodle to simplify and automate the processes of interaction with it.

The field of use is an automation of interaction with the distance learning system Moodle in higher education institutions.

ЗМІСТ

С.

Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Вибір топології інтегрованої системи.....	11
1.2 Основні методи інтеграції програмних систем.....	13
1.3 Вибір підходу до інтеграції з СДН Moodle	20
1.4 Висновки за розділом 1	24
2 Аналіз і специфікація вимог	26
2.1 Специфікація системних вимог	26
2.2 Специфікація функціональних вимог	26
2.3 Специфікація нефункціональних вимог	27
3 Матеріали і методи.....	29
3.1 Протокол XML-RPC	30
3.2 Протокол SOAP	31
3.3 Протокол REST	32
3.4 Порівняння протоколів.....	33
3.5 Мова програмування.....	35
3.5 Висновки за розділом 3	36
4 Основні рішення щодо реалізації компонентів системи.....	38
4.1 API -функції Moodle	40
4.2 Основні розроблені алгоритми	41
4.3 Висновок за розділом 4.....	46
5 Експлуатація, тестування та експериментальне дослідження програми	47
5.1 Порядок дій для підключення інтеграційної системи.....	47
5.2 Тестування та експериментальне дослідження.....	51
5.3 Обробка виключних ситуацій.....	53
6 Економіко-організаційна частина.....	55
6.1 Планування розробки програмного продукту.....	55

6.2 Визначення витрат на розробку програми	56
6.3 Розрахунок економічної ефективності програмного продукту.....	63
7 Охорона праці та безпека в надзвичайних ситуаціях.....	66
7.1 Аналіз потенційних небезпек.....	66
7.2 Заходи із забезпечення безпеки	68
7.3 Заходи з виробничої санітарії та гігієни праці.....	69
7.4 Заходи з забезпечення безпеки у надзвичайних ситуаціях.....	73
7.4.1 Заходи з пожежної безпеки.....	73
7.4.2 Заходи з цивільного захисту. Єдина державна система цивільного .	
захисту, її складові та режим функціонування	75
Висновки	799
Перелік джерел посилання	81
Додаток А Текст програми.....	84
Додаток Б Слайди презентації.....	110

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
CSV	– Comma-Separated Values;
LCMS	– Learning Content Management Systems;
Moodle	– Modular Object Oriented Distance Learning Environment
PDM	– Product Data Management
REST	– Representational state transfer;
SOAP	– Simple Object Access Protocol;
XML-RPC	– Extensible Markup Language Remote Procedure Call;
WSDL	– Web Services Description Language;
АС	– автоматизована система;
БД	– база даних;
ІС	– інформаційна система;
ОС	– операційна система;
ПЗ	– програмне забезпечення;
СДН	– система дистанційного навчання

ВСТУП

В даний час швидко удосконалюються засоби інформатизації та телекомунікацій отримують світове визнання і все більш широке поширення. Інтерес до впровадження технологій дистанційного навчання в навчальний процес існує вже давно. Ця форма отримання знань була затребувана і в попередні роки, а зараз, в епоху бурхливого розвитку телекомунікаційних технологій та світової пандемії Covid-19, дистанційному навчанню, як новій форми навчання на відстані, приділяється особлива увага.

Застосування дистанційних освітніх технологій в університеті забезпечується системою дистанційного навчання (СДН) Moodle (Modular Object Oriented Distance Learning Environment). Moodle – це система управління навчальним контентом (LCMS – Learning Content Management Systems), яка володіє широким спектром функціональних можливостей [1]. За допомогою даної системи можна створювати електронні навчальні курси і проводити як аудиторне (очне) навчання, так і навчання на відстані (заочне/дистанційне).

Для коректної роботи в цій системі необхідне чітке розділення ролей користувачів на викладачів (авторів курсів, асистентів тощо) та студентів. Кожен з викладачів має обмежений доступ до курсів своєї кафедри, а кожний студент має доступ до тих курсів, на які було зараховано його академічну групу. Проте перелік викладачів та контингент студентів є мінливим явищем, особливо студенти часто змінюють групи, зараховуються, уходять в академічну відпустку або відраховуються. Підтримувати коректний стан контингенту в Moodle за рахунок ручного створення акаунтів студентів та груп, редагування зв'язків студентів з групами, видалення неактуальних даних є дуже трудомістким завданням без гарантії відсутності помилок.

Тому одним із завдань інтеграції систем в корпоративному інформаційному середовищі університету є забезпечення взаємодії між

інформаційною системою з обліку контингенту і системи дистанційної освіти Moodle [2], що дозволить:

- здійснювати автоматичну реєстрацію студентів в СДН Moodle, раніше внесених в систему з обліку контингенту;
- автоматично формувати глобальні групи в СДН Moodle у відповідності з академічними групами в системі з обліку контингенту;
- автоматизувати управління обліковими записами студентів в СДН Moodle при виконанні наказів по студентському контингенту в системі з обліку контингенту: видалення облікового запису при відрахуванні, блокування на період академічної відпустки, відновлення, переміщення / виключення студента з глобальної групи;
- організувати автоматизовану передачу результатів внутрисеместрових атестацій з СДН Moodle.

Метою роботи є дослідження та програмна реалізація методів інтеграції з системою дистанційного навчання Moodle для спрощення та автоматизації процесів взаємодії.

Для досягнення цієї мети було поставлено такі задачі:

- дослідження методів інтеграції програмних систем та варіантів реалізації топології інтегрованих систем;
- порівняння методів взаємодії з СДН Moodle;
- виявлення вимог для реалізації методів інтеграції;
- розроблення методів для інтеграції з СДН Moodle;
- розробка програмного продукту;
- проведення тестування та експериментального дослідження;
- створення відповідної програмної документації;
- аналіз економіки та організації виробництва щодоекономічного обґрунтування доцільності розробки;
- аналіз стану охорони і умов праці об'єкта проектування.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

Не існує інформаційних систем (ІС), які поодиночі могли б покрити потреби сучасного закладу вищої освіти. Середні і великі університети зазвичай експлуатують як мінімум десяток багатокористувацьких систем. У цих системах часто обробляються однакові дані – починаючи з довідників і класифікаторів. Звичайні ситуації, коли в рамках одного процесу задіяні різні ІС. Багато ІС з самого початку орієнтовані на отримання інформації з інших застосунків і баз даних (БД) (наприклад, системи формування зведеної звітності, системи управління і моніторингу). Тому жодний застосунок не може розглядатися як щось автономне, а завжди є частиною великого механізму під назвою «інформаційна система університету».

Наслідками відсутності належного вирішення проблеми інтеграції є:

- повторне ручне введення даних (довідники, дані про студентів, викладачів, фінансові транзакції і т.п.);
- багаторазові і нескінченні «звірки та коригування», які не виключають помилок;
- непомірні витрати на формування зведеної звітності;
- завищені терміни і собівартість виконання навіть звичайних завдань.

Перераховане вище визначає необхідність інтеграції застосунків університету, зокрема СДН Moodle та системи з обліку контингенту, які містять загальну інформацію про студентів та їх групи.

1.1 Вибір топології інтегрованої системи

Існує два підходи до організації маршрутів взаємодії інтегрованих системи. Перший - пряму взаємодію інтегрованих систем за принципом «кожна з кожною», або «точка-точка». Другий - взаємодія через центральний вузол; подібну зіркоподібну архітектуру зазвичай називають «хаб +

спиці» [3]. Топологія не залежить від фізичної архітектури ІС, а визначає логічні маршрути взаємодії і передачі даних між інтегрованими системами.

1.1.1 Топологія «точка-точка»

При використанні підходу топології «точка-точка» інтегровані системи взаємодіють безпосередньо. Переваги підходу - простота, прозорість і відсутність необхідності в додатковому програмному забезпеченні. Однак, є і недоліки. По-перше, інтегровані застосунки повинні спілкуватися з використанням однакових методів взаємодії та форматів викликів / даних. При зміні одного з застосунків (якщо воно спричинило за собою зміну інтерфейсу взаємодії цього застосунку) доводиться модифікувати або як мінімум перенастроювати всі інтегровані з ним системи. По-друге, в ІС підприємства з'являється занадто багато зв'язків, кожен з яких потрібно контролювати і підтримувати в працездатному стані.

Якщо взаємодіючих застосунків багато, вартість супроводу інтегрованої таким чином ІС стає неприйнятно високою. Проте підхід «точка-точка» широко використовується. Це відбувається, як правило, в тих випадках, коли при взаємодії конкретних програм необхідно передавати великі обсяги даних або забезпечувати нормований час взаємодії, а також якщо експлуатовані на підприємстві застосунки мають вбудовані засоби взаємодії (це часто трапляється при впровадженні декількох систем від одного постачальника, а також якщо при розробці замовних програмних систем або впровадженні нових до них спочатку ставиться вимога щодо взаємодії з уже наявними системами).

Тут, однак, криється небезпека «повзучої» інтеграції, яка робить можливою ситуації, коли при необхідності поміняти систему XYZ несподівано виявляється, що зробити цього не можна, оскільки довідник структури і співробітників підприємства, історично ведеться в XYZ, щодня реплікується ще в десяток систем.

1.1.2 Топологія «хаб + спиці»

Взаємодія по типу «точка-точка» створює в інфраструктурі підприємства занадто багато зв'язків і вимагає узгодження інтерфейсів і форматів даних між взаємодіючими застосунками. Ці недоліки покликана вирішити архітектура взаємодії, в якій всі застосунки безпосередньо з'єднані тільки з центральним вузлом, який вирішує наступні завдання:

- організація маршрутизації взаємодії між інтегрованими програмами;
- перетворення форматів файлів і даних;
- забезпечення взаємодії застосунків з використанням різних методів і протоколів взаємодії.

Завдяки введенню проміжної ланки, зменшується число зв'язків між застосунками, усуваються прямі зв'язки, а система інтеграції стає більш гнучкою і дешевою в експлуатації. Якщо змінюється одна з інтегрованих програм, то - за умови правильно спроектованої системи інтеграції - потрібно буде модифікувати тільки один зв'язок, між даними застосунком і хабом.

Недоліком топології «хаб + спиці» є висока вартість придбання і складність програмного інструментарію, що грає роль хаба, а також брак фахівців, які мають досвід застосування подібних програмних засобів.

1.2 Основні методи інтеграції програмних систем

Для взаємодії застосунків зазвичай використовуються такі методи, як обмін файлами, загальна БД, віддалений виклик і асинхронний обмін повідомленнями. У цьому списку немає прямого обміну даними між базами даних застосунків, тому що цей метод ближче не до інтеграції застосунків, а до переміщення даних. З точки зору інтеграції застосунків важлива можливість в процесі обміну даними виконувати якусь змістовну обробку. Прямий обмін даними, який зазвичай виконується засобами класу ETL

(Extract, Transfer, Load) або саморобними утилітами, зазвичай такої можливості не надає.

1.2.1 Обмін файлами

Обмін файлами мабуть найпоширеніший підхід до організації взаємодії. Це пов'язано з відносною простотою реалізації, а також існуванням стандартних (або «майже» стандартних) форматів обміну. Наприклад, велика частина корпоративних ІС дозволяє завантажувати і вивантажувати файли, наприклад, у форматі CSV (Comma-Separated Values - «поля, розділені комами»). Але у цього підходу є і недоліки - якщо необхідно оперувати складними структурами, то прості формати обміну вже не придатні. Виникаючи в таких випадках спеціалізовані формати файлів повинні «розуміти» взаємодіючі системи, що веде до жорсткої залежності систем один від одного. Цей недолік зазвичай долають всілякими утилітами конвертації даних. Крім того, зазвичай обмін файлами передбачає участь людини – хтось повинен вивантажити файл, скопіювати його на інший комп'ютер, завантажити. Однак якщо інтегровані методом обміну файлами системи мають можливість автоматичного завантаження / розвантаження (наприклад, за розкладом), то даний підхід дозволяє побудувати повністю автоматизоване рішення, яке внаслідок своєї простоти має високу надійність і пропускну здатність.

1.2.2 Спільна база даних

Підхід з використанням спільної БД концептуально дуже простий – кілька ІС або застосунків використовують одну базу даних. Головний його недолік - зв'язок між інтегрованими програмами настільки тісний, що іноді неможливо помітити межу між ними (зазвичай так інтегруються продукти одного виробника). Прикладом такого підходу можуть служити більшість

ERP-систем, де різні модулі системи використовують одну базу. Однак занадто тісний зв'язок перетворює конгломерат інтегрованих програм в моноліт, в «суперсистему», окремі частини якої практично не піддаються самостійної модернізації та заміні. З цим борються, використовуючи механізми серверів БД (представлення даних, проміжні таблиці і т.п.), але далеко не завжди ефективно.

1.2.3 Віддалений виклик

Стандарти на віддалений виклик процедур виникли два десятиліття тому, дозволяючи програмного коду, який виконується на одному комп'ютері, викликати код на іншому. В різний час існували різні стандарти (RPC, CORBA, DCOM, RMI тощо), останнім в цьому ряду став протокол SOAP, основа сучасних вебсервісів. Власне в підході до інтеграції з використанням віддалених викликів за ці роки нічого принципово не змінилося - якщо застосунку А щось потрібно від застосунку Б, то А одним з перерахованих способів викликає функцію застосунку Б.

Основний недолік віддаленого виклику – вимога працездатності всіх задіяних застосунків в момент взаємодії. Аналіз прикладів застосування віддаленого виклику показує, що цей підхід прийнятний тільки в тих випадках, коли взаємодія застосунків ініціюється користувачем, який сам контролює результат. Для автоматичного взаємодії без участі людини даний підхід практично непридатний. В цьому немає нічого дивного: віддалені виклики спочатку були придумані не для інтеграції різних застосунків, а для створення розподілених систем, коли компоненти однієї системи можуть працювати на різних комп'ютерах.

Сьогодні технологія вебсервісів позиціонується як зручний засіб інтеграції застосунків, оскільки дозволяє реалізовувати, розгортати і забезпечити просту міжплатформну взаємодію. Проте поки вебсервіси виявляються непридатні для обробки великих обсягів інформації; в них

відсутні засоби підтримки транзакцій; взаємодіючі системи повинні перебувати в працездатному стані на момент взаємодії.

Непридатність до обробки великих обсягів - вроджена вада, який пов'язаний з XML-природою вебсервісів. Всі дані і параметри викликів вебсервісів перетворюються в формат XML, що у багато разів збільшує обсяг даних, що призводить до наслідків у вигляді зростаючої потреби в оперативній пам'яті і навантаження на процесори. На жаль, мова йде не про гігабайти переданої інформації: вебсервіси не витримують, коли за одну взаємодію між інтегрованими системами потрібно передавати вже сотню кілобайт.

У ситуації, де в процесі взаємодії безлічі ІС необхідно провести узгоджені зміни в деяких з них, тобто виконати логічну транзакцію, це зачіпає кілька систем. Якщо для інтеграції використовуються вебсервіси, то такої можливості немає: в своєму «вихідному» варіанті вебсервіси не підтримують виведення в рамки однієї транзакції кількох викликів методів одного або декількох сервісів. Постачальники інтеграційного програмного забезпечення в цьому випадку зазвичай пропонують використовувати так звані «компенсаційні операції» (на кожну операцію передбачається операція, яка скасовує її). Скажімо, якщо з трьох застосувань, які беруть участь в логічній транзакції, два змогли виконати якусь операцію, а одне - ні, то в двох системах потрібно виконати дії відкату, що скасовують результат успішно виконаних операцій. В теорії підхід хороший, але, по-перше, ніхто не гарантує успіх компенсаційної операції, а по-друге, деякі системи (наприклад, фінансові) не допускають видалення даних, внесених в базу.

Попередній стандарт WS-Transactions, покликаний вирішити дану проблему, існує, але відсутні його повноцінні реалізації. Так, корпорація IBM заявила про підтримку цього стандарту в своїх інтеграційних продуктах, заснованих на WebSphere Application Server, але з масою обмежень; найголовніше з яких - всі взаємодіючі вебсервіси повинні працювати під управлінням WebSphere Application Server. Аналогічна ситуація і у Microsoft

в .Net 3.0, що втрачає сенс вебсервісів - простоту міжплатформної взаємодії, створення, розгортання, підтримки. Підтримка WS-Transactions, WS-Attachments і інших стандартів, що розширюють можливості вебсервісів, може різко ускладнити їх створення, налагодження та розгортання.

1.2.4 Асинхронний обмін повідомленнями

Асинхронний обмін повідомленнями –підхід, який створювався спеціально для інтеграції ІС. Ідея концептуально проста і нагадує роботу електронної пошти. Коли застосункуА необхідно викликати якусь дію в застосункуБ, воно формує відповідне повідомлення з даними і інструкціями і відправляє його за допомогою системи доставки повідомлень. Слово «асинхронний» означає, що застосункуА не потрібно чекати, поки повідомлення дійде до Б, буде оброблено, сформована відповідь і т.п. Повідомлення гарантовано доставляється завдяки механізму черг повідомлень, які знімають з взаємодіючих систем турботу про надійність мережі передачі даних, працездатності взаємодіючих систем в конкретні моменти часу і т.д.

Недолік даного підходу - висока ціна. Система гарантованої доставки на основі черг повідомлень зазвичай сама по собі недешева. Існують і вільно поширювані безкоштовні (наприклад, ActiveMQ), які, тим не менш, потрібно розгорнути, навчити фахівців, підтримувати, написати адаптери між системою доставки та застосунками і т.д.

1.2.5 Інтеграції через сервіси інтерфейсу прикладного програмування

Через сервіси інтерфейсу прикладного програмування (application programming interface, API)зазвичай інтегруються між собою прикладна автоматизована система (АС) і Product Data Management(PDM)-система.

Інтерфейс, як правило, реалізується на стороні прикладної АС у вигляді додаткових вкладок стрічки, панелей інструментів, меню і т.п., що відповідають за виклики функцій PDM-системи. API - опис способів (набір класів, процедур, функцій, структур або констант), якими одна комп'ютерна програма може взаємодіяти з іншою програмою, зазвичай входить в опис будь-якого інтернет-протоколу, програмного каркаса (фреймворка) або стандарту викликів функцій операційної системи. API часто реалізується окремої програмної бібліотекою або сервісом операційної системи[4].

Програмні компоненти взаємодіють один з одним за допомогою API. При цьому зазвичай компоненти утворюють ієрархію - високорівневі компоненти використовують API низькорівневих компонентів, а ті, в свою чергу, використовують API ще більш низькорівневих компонентів. За таким принципом побудовані протоколи передачі даних по Internet. Стандартний протокол Internet (мережева модель OSI) містить 7 рівнів (від фізичного рівня передачі пакетів біт до рівня протоколів програм, подібних протоколів HTTP і IMAP). Кожен рівень користується функціональністю попереднього рівня передачі даних і, в свою чергу, надає потрібну функціональність наступного рівня. Важливо зауважити, що поняття протоколу близьке за змістом до поняття API. І те й інше є абстракцією функціональності, тільки в першому випадку мова йде про передачу даних, а в другому - про побудову комп'ютерних застосунків. API бібліотеки функцій і класів включає в себе опис сигнатур і семантики функцій.

Програмний інтерфейс API взаємодії між системами дозволяє:

- отримувати доступ до бізнес-сервісів підприємства;
- обмінюватися інформацією між системами і застосунками;
- спростити взаємодію між компаніями, партнерами, розробниками і клієнтами.

В індустрії програмного забезпечення загальні стандартні API для стандартної функціональності мають важливу роль, так як вони гарантують, що всі програми, які використовують загальний API, будуть працювати

однаково добре або, принаймні, типовим звичним чином. У разі API графічних інтерфейсів це означає, що програми будуть мати схожий користувацький інтерфейс, що полегшує процес освоєння нових програмних продуктів.

З іншого боку, відмінності в API різних операційних систем (ОС) істотно ускладнюють перенесення застосунків між платформами. Існують різні методи обходу цієї складності - написання «проміжних» API (API графічних інтерфейсів Qt, Gtk, і т. П.), написання бібліотек, які відображають системні виклики однієї ОС в системні виклики іншої ОС (такі середовища виконання, як Wine, cygwin і т.п.), введення стандартів кодування в мовах програмування (наприклад, стандартна бібліотека мови C написання різних мов, що реалізуються на різних платформах (sh, perl, php, tcl, Java, і т. д.).

Також необхідно відзначити, що в розпорядженні програміста часто знаходиться кілька різних API, що дозволяють добитися одного і того ж результату. При цьому кожен API зазвичай реалізований з використанням API програмних компонент нижчого рівня абстракції.

Основними труднощами існуючих багаторівневих систем API, таким чином, є:

- складність портування програмного коду з однієї системи API на іншу (наприклад, при зміні ОС);
- втрата функціональності при переході з нижчого рівня на вищий. Кожен «шар» API створюється для полегшення виконання деякого стандартного набору операцій, але при цьому стає принципово неможливим виконання деяких інших операцій, які надає більш низький рівень API.

Проте існує як мінімум чотири причини, за якими варто впроваджувати API [4]:

- взаємодія в рамках загальної інфраструктури (такі фундаментальні елементи API-рішень, як посвідчення особи, аутентифікація, безпека і управління мережевими з'єднаннями, повинні узгоджуватися на галузевому рівні, а не між окремими застосуваннями);

- координація стандартів API для підтримки сумісності (потреба в єдиному стандарті API, який буде працювати між усіма провайдерами);
- підтримка мережових API, а не p2p-рішень (мережеве рішення буде підтримувати конвергенцію як для пояснення даних, так і для інших характеристик API);
- відповідність суворим стандартам безпеки і стійкості (для успішного розвитку API-рішення повинно мати високий рівень захисту даних і стійкості).

1.3 Вибір підходу до інтеграції з СДН Moodle

Для повноцінного впровадження СДН Moodle виникає завдання її інтеграції у вже сформоване інформаційне середовище університету. Комплексна інтеграція включає управління процесами, моделювання процесів і технологічний процес, який охоплює різні завдання, процедури, архітектури, необхідну вхідну і вихідну інформацію.

Розглянуті у розділі 1.1 підходи можна розділити на 3 типи (рис. 1.1):

- інтеграція застосунків (віддалений виклик, асинхронний обмін повідомленнями,);
- інтеграція даних (спільна база даних, обмін файлами);
- інтеграція платформ (інтеграції через сервіси інтерфейсу прикладного програмування).

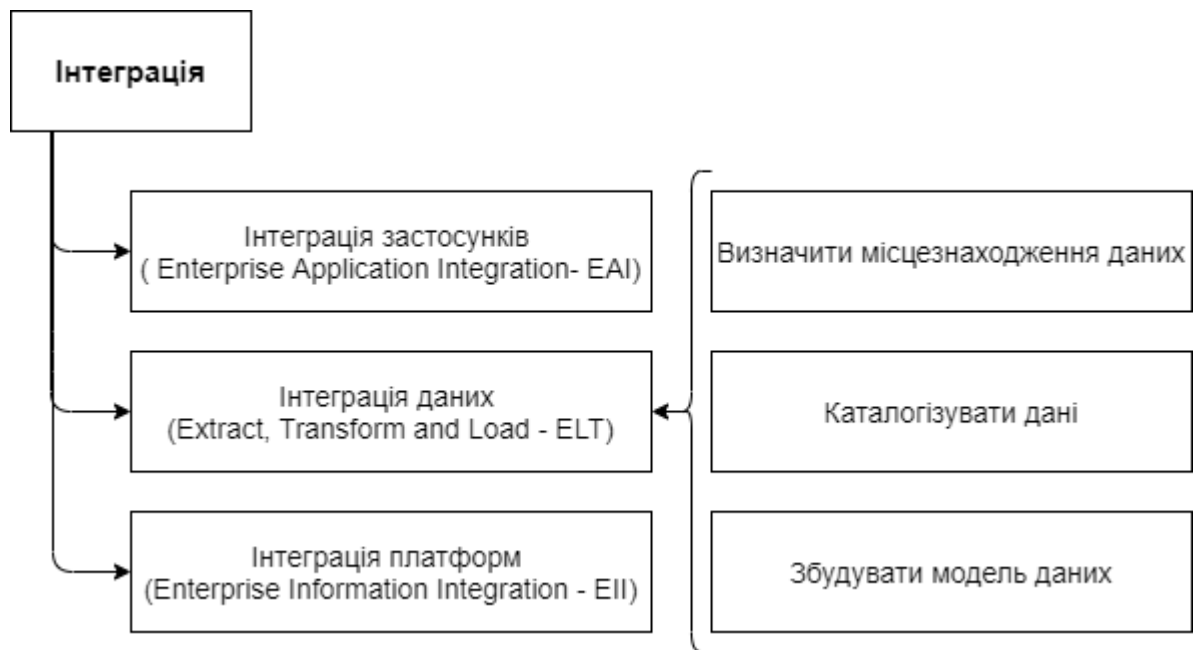


Рисунок 1.1 – Методи інтеграції

Для ефективного вирішення задач інтеграції та вибору методу інтеграції необхідно співвідносити вимоги, які висуваються до інтеграційного рішення, з реальними характеристиками існуючих технологій.

На даний момент існують наступні методи інтеграції з СДН Moodle:

- за допомогою інтеграційних плагінів (інтеграції з Openmeetings для відеоконференцій, з WordPress за допомогою Edwiser Bridge, плагін Arlo для комерційних цілей та ін.) [4];
- за допомогою імпорту з CSV-файлу (завантаження даних про користувачів);
- за рахунок підтримки міжнародного стандарту SCORM (Sharable Content Object Reference Model) для обміну електронними курсами, стандарт забезпечує перенесення ресурсів до інших систем;
- за допомогою API веб-служб Moodle можливо додати функції самостійно розробленого плагіну (як правило, зовнішні функції) як веб-служби, що дозволить реалізувати функції цього плагіна для інших систем через веб-служби за допомогою одного з ряду протоколів, таких як XML-RPC, REST або SOAP.

Під час внесення даних усіх студентів після оголошення загальнодержавного карантину було використано спосіб завантаження даних про студентів до Moodle через імпортування CSV-файлів. CSV є загальним форматом для текстового обміну даними між різними інформаційними системами. Для того, щоб імпортувати користувачів з CSV-файлу, потрібно дотримуватися певних вимог його форматування. Перш за все, необхідно записати обов'язкові поля (поля, які повинні обов'язково бути включеними до першого рядка та заповнені для кожного користувача): `username`, `password`, `firstname`, `lastname`, `email`. Далі за необхідністю вказуються типові поля (поля, які є необов'язковими, якщо їх не задавати, то будуть підставлені типові дані): `institution`, `department`, `city`, `country`, `lang`, `auth`, `timezone`. Існують також додаткові поля – поля, які також є необов'язковими. До них можна віднести: назву курсу (`course1`) – «коротка назва» (його наявність надасть можливість записати користувачів на цей курс; назву групи (`group1`) – надасть можливість додати користувачів до вказаної групи, при цьому потрібно звернути увагу на те, що вказана група вже повинна бути створена на курсі, групи повинні бути асоційовані з відповідним курсом, відповідно група в полі `group1` повинна належати до курсу `course1` тощо.

Цей метод зручний для випадку, коли необхідно занести до системи Moodle великий масив злагодженої інформації та може бути використаний після проведення приймальної кампанії та необхідно внести дані нових студентів до системи. Проте у період після цього, коли відбуваються зміни номеру груп у студентів, відрахування за несплату або невиконання навчального плану, поновлення студентів – оновлювати інформацію про це в Moodle через імпортування CSV-файлів є абсолютно незручним методом, бо потребує багато часу та обов'язково присутності людини-оператора, яка буде створювати цей CSV-файл та завантажувати його у Moodle.

Наразі для відпрацьовування цих змін використовується ручне введення через вебінтерфейс СДН Moodle, що відповідно також займає багато часу та не виключає можливість помилок.

Для обрання найкращого методу для оновлення даних студентів (а також в майбутньому викладачів, курсів, тощо) порівняємо існуючі методи взаємодії з Moodle (табл. 1.1).

Таблиця 1.1 – Порівняння методів взаємодії з СДН Moodle

	Витрати часу на дію	Вірогідність помилки	Універсальність	Необхідність додаткових витрат
Ручне введення даних	Високі	Висока	Універсальний	Немає
Імпорт даних з CSV-файлу	Високі	Середня	Обмежений	Немає
Використання інтеграційних плагінів	Малі	Маленька	Обмежений	В залежності від вартості плагіну
Використання міжнародного стандарту SCORM	Малі	Маленька	Універсальний (для обміну матеріалами курсів)	В залежності від засобів, що використовуються
Використання API веб-служб Moodle	Малі	Маленька	Універсальний	В залежності від засобів, що використовуються

Зважаючи, що методи ручного введення даних та імпорт даних з CSV-файлу мають високі витрати часу на виконання дій та велику вірогідність внесення помилкових даних, ці методи були виключені з розгляду. Взаємодія через стандарт SCORM не надає можливості автоматизації дій з користувачами, так само як і використання існуючих плагінів. Відповідно найкращім методом для автоматизації процесів керування обліковими записами користувачів СДН Moodle є розроблення власної інтеграційної системи з використанням API веб-служб Moodle.

Щодо вибору топології майбутньої системи, зважаючи на велику кількість програмних систем, які плануються інтегрувати з

СДН Moodle, взаємодія по типу «точка-точка» створить занадто багато зв'язків і буде вимагати узгодження інтерфейсів і форматів даних між взаємодіючими застосунками. Тому для створення інтеграційної системи було обрано топологію «хаб + спиці».

1.4 Висновки за розділом 1

Метою роботи визначено дослідження та програмну реалізацію методів інтеграції з системою дистанційного навчання для спрощення та автоматизації процесів взаємодії з СДН Moodle.

Для досягнення цієї мети поставлено такі задачі:

- дослідження методів інтеграції програмних систем та варіантів реалізації топології інтегрованих систем;
- порівняння методів взаємодії з СДН Moodle;
- виявлення вимог для реалізації методів інтеграції;
- розроблення методів для інтеграції з СДН Moodle;
- розробка програмного продукту;
- проведення тестування та експериментального дослідження;
- створення відповідної програмної документації;
- аналіз економіки та організації виробництва щодоекономічного обґрунтування доцільності розробки;
- аналіз стану охорони і умов праці об'єкта проєктування.

В розділі 1 було:

- розглянуто варіанти топології інтегрованих систем;
- виконано аналіз основних методів інтеграції програмних систем, наведено їх недоліки та переваги;
- виконано аналіз та порівняння існуючих методів інтеграції з СДН Moodle;

- обрано найкращім методом для автоматизації процесів керування обліковими записами користувачів СДН Moodle метод інтеграції через сервіси інтерфейсу прикладного програмування;
- поставлено завдання розробити власну інтеграційну систему з використанням API вебслужб Moodle.

2 АНАЛІЗ І СПЕЦИФІКАЦІЯ ВИМОГ

2.1 Специфікація системних вимог

Перевагою платформи e-learning Moodle є той факт, що почавши від її появи, тобто з 1999 року, вона неодноразово була модифікована і доповнена новими рішеннями та інструментами [5]. Програмне забезпечення платформи написано мовою PHP з використанням безкоштовних загальнодоступних БД (MySQL, PostgreSQL). Платформу Moodle можна встановити на будь-яку операційну систему (MS Windows, Unix, Linux).

Відповідно до цього, для розроблення інтеграційної системи з використанням API веб-служб Moodle найкращім рішенням буде використовувати мову PHP для розроблення скриптів для обробки вхідних запитів та викликів методів API, що надає Moodle. До того ж, створений WSDL (Web Services Description Language, мова опису веб-сервісів і доступу до них, заснований на мові XML) Moodle наразі не сумісний з JAVA або .NET, хоча у майбутньому планується реалізація створення JAVA-сумісного SOAP WSDL [6].

Використання БД для інтеграційної системи не планується, тому що усі виклики мають оброблюватися одразу після отримання, у зберіганні проміжної інформації у БД немає потреби.

2.2 Специфікація функціональних вимог

Розробники Moodle були дуже обережні при додаванні нових функцій, тому що підтримують політику не змінювати функції API після їх реалізації. Ті, які наразі представлені в Moodle, є найбільш часто запитуваними (для користувачів, курсів, груп та учнів).

Для розширення API на власному веб-сайті Moodle, необхідно створити власні функції API (вебслужбу). Найпростіший і рекомендований спосіб додати вебслужбу – це написати новий локальний плагін Moodle і

реалізувати там функції вебслужби. Після цього адміністратор Moodle має додати плагін в Moodle (скопійовати файли в /local/), і функції веб-служби будуть додані автоматично.

До розроблюваної інтеграційної системи висуваються наступні функціональні вимоги:

- поновлення студента (додавання нового користувача або активація існуючого акаунта);
- відрахування студента (деактивація користувача);
- видалення користувача;
- прикріплення студента до групи;
- зміна групи у студента;
- сповіщення користувача через email про зміни у його акаунті;
- видалення груп студентів.

2.3 Специфікація нефункціональних вимог

2.3.1 Вимоги до інтерфейсу

Інтерфейс програмного продукту повинен відповідати ергономічним засадам для зручної організації процесу інтеграції з іншими програмними системами. Він повинен містити правила введення даних та мати повідомлення у разі не отримання некоректних чи неповних даних, відсутності зв'язки з сервером тощо.

Графічний інтерфейс для розроблюваної системи не потрібен.

2.3.2 Вимоги до надійності

Надійне функціонування програмного виробу повинно бути забезпечене виконання сукупності організаційно-технічних заходів, перелік яких наведено нижче:

- організація безперебійного живлення технічних засобів;

- контроль вхідних даних, отриманих за запитом;
- контроль вихідних даних, які передають до СНД Moodle;
- наявність повідомлень для повернення до інтегрованої системи у разі не вірних введених даних та для обробки виняткових ситуацій.

Відмови програми можливі в наслідок некоректних звернень при взаємодії з системою, або надмірного навантаження на сервера програмного продукту.

2.3.3 Вимоги до захисту даних

Необхідно розробити, запровадити і підтримувати в дієздатному стані документовані методики керування конфігурацією програмного забезпечення (ПЗ) на всіх стадіях його життєвого циклу.

Вхідні дані до інтеграційної системи мають передаватися методом POST протоколу HTTP.

Для забезпечення передачі даних від веб-служби до Moodle необхідно проаналізувати та обрати один з протоколів, який використовується у СНД Moodle (XML-RPC, REST або SOAP).

3 МАТЕРІАЛИ І МЕТОДИ

Основним завданням для забезпечення надійної передачі даних між системами, що інтегруються, є вибір протоколу передачі даних. За допомогою API Moodle для додавання нових функцій до веб-служби та для зв'язку з іншими системами пропонується використання одного з ряду протоколів, таких як XML-RPC, REST або SOAP [7]. Вони відрізняються як контекстом, так і використанням. У світі XML існує два основних шляхи реалізації віддаленого виклику процедур: XML-RPC та SOAP. XML-RPC - протокол, котрий є прашуром SOAP і виділяється виключною простотою в застосуванні та використанні. SOAP - це протокол, в той час як REST - це архітектурний стиль. SOAP є добре продуманим протоколом, що використовується в веб індустрії і стандартизується World Wide Web Consortiumом (W3C). REST в свою чергу набирає популярності через свою простоту, масштабованість і архітектурну залежність від World Wide Web. Великі корпорації, як-то Google і Amazon, зокрема, використовують REST у переважній більшості своїх продуктів. Найголовнішою відмінністю між цими моделями є те, що SOAP є сильно зв'язаною системою, водночас REST – слабозв'язана система [8].

Протоколи, формати даних, які вони можуть повертати, та їх реалізація у СНД Moodle представлені у таблиці 3.1 [7]. Також вказано чи підтримує ця реалізація CORS (Cross-Origin Resource Sharing, укр. "Спільне використання ресурсів між різними джерелами") – це система, що складається з відправки HTTP заголовків, які визначають: заблокувати або виконати запит до обмеженого ресурсу на вебсторінці з іншого домену, відмінного від домену походження запитуваного ресурсу [9]. Правило обмеження домену за замовчуванням забороняє міждоменні запити, а CORS надає вебсерверам можливість контролювати міждоменні запити і дозволяє робити безпечний обмін даними між різними доменами.

Таблиця 3.1 – Реалізація протоколів для взаємодії з СДН Moodle

Назва протоколу	Доступно у версії Moodle	Опис	CORS
REST (повертає XML)	2.0	Дані XML, доступні через REST API (не restful)	Підтримується (3.0 і вище)
REST (повертає JSON)	2.2	Дані JSON, доступні через REST API (не restful)	Підтримується (3.0 і вище)
SOAP	2.0	Дуже важливий для взаємодії з програмами Java та .Net. Часткова реалізація була виконана в Moodle 2.0, у повній мірі у Moodle 3.0. Він повністю працює з клієнтами PHP	Не підтримується
XML-RPC	2.0	Виклик віддаленої процедури XML	Не підтримується

Розглянемо більш детально їх переваги та недоліки для вибору оптимального протоколу для передачі даних між інтегрованими системами.

3.1 Протокол XML-RPC

XML-RPC (скор. від англ. Extensible Markup Language Remote Procedure Call - XML-виклик віддалених процедур) - протокол виклику віддалених процедур, що використовує XML для кодування своїх повідомлень і HTTP в якості транспортного механізму [10]. Є прабатьком SOAP, відрізняється винятковою простотою в застосуванні. XML-RPC, як і будь-який інший інтерфейс Remote Procedure Call (RPC), визначає набір стандартних типів даних і команд, які програміст може використовувати для

доступу до функціональності іншої програми, що знаходиться на іншому комп'ютері в мережі.

Головна перевага XML-RPC – його простота, невелика кількість мережесих ресурсів (точок входу) і багато задіяних методів. Незважаючи на це, XML-RPC- це застарілий протокол, у якого ряд недоліків:

- не можна перевірити правильність XML-RPC повідомлення. Старий протокол створювався до того, як в XML були стандартизовані схеми (способи перевірки даних). Тобто сервер приймає запити, він повинен переконатися, що ці запити для нього, і що дані не суперечливі. У XML-RPC для цього типи даних декларуються, але це перевірка типу даних, а узгодженість даних не перевіряється (що було отримано структуру з усіма потрібними параметрами);
- не можна створювати комбіновані повідомлення;
- не можна розширювати повідомлення, тобто додавати додаткову інформацію.

Виходячи з усіх недоліків XML-RPC було створено протокол SOAP.

3.2 Протокол SOAP

SOAP (скор. від англ. Simple Object Access Protocol) - протокол доступу до об'єкта (до точки входу). Сьогодні це основний промисловий стандарт побудови розподілених додатків.

Він являє собою розширення мови XML-RPC. Тобто він побудований за принципом: 1 точка входу і будь-які методи. Сам протокол в плані транспортування даних (методів передачі даних) дає широкий вибір: SMTP, FTP, HTTP, MSMQ. SOAP лежить в основі реалізації XML вебсервісів (XML вебслужб).

Недолік SOAP – його складність у вивченні. Також використання SOAP для передачі повідомлень збільшує їх обсяг і знижує швидкість обробки. В системах, де швидкість важлива, частіше використовується

пересилання XML-документів через HTTP безпосередньо, де параметри запиту передаються як звичайні HTTP-параметри.

SOAP базується на обміні повідомленнями між клієнтом і сервером (синхронно і асинхронно). Кожне повідомлення несе інформацію про дані (які дані передаються-відправляються). SOAP заздалегідь описує всю структуру повідомлення за допомогою XML схем: що повинно бути в повідомленні, як воно буде передаватися. Це дає можливість, не знаючи деталей сервера, зрозуміти, що там відбувається, і дає можливість серверу перевірити, чи для нього це повідомлення.

Призначення методу SOAP – надатимінімальний транспортний функціонал, поверх якого можуть бути побудовані більш складні протоколи та процеси взаємодії.

Щоб бути незалежними від HTTP, SOAP має перевагу, а саме, елемент `attachment` — надсилання прикріплених файлів, що часто вважають найкращим рішенням для сервісів, що мають справу з прикріпленими файлами. Через свою надійність та рівень безпеки протоколи SOAP широко використовуються у банківських системах та корпоративних додатках[8].

3.3 Протокол REST

REST (скор. від англ. Representational state transfer) - це стиль архітектури програмного забезпечення для розподілених систем, таких як World Wide Web, який, як правило, використовується для побудови веб-служб [11]. Системи, що підтримують REST, називаються RESTful-системами.

У загальному випадку REST є дуже простим інтерфейсом управління інформацією без використання якихось додаткових внутрішніх прошарків. Кожна одиниця інформації однозначно визначається глобальним ідентифікатором, таким як URL. Кожна URL в свою чергу має строго заданий формат.

Головною ідеєю REST було скоріш використання добре розвиненого HTTP для передачі даних міжпристроями, ніж використання протоколу, побудованого поверх рівня HTTP, для передачі повідомлень. Додаток, розроблений слідуючи принципам даного архітектурного підходу, буде використовувати HTTP для виконання викликів між пристроями не спираючись на складні об'єктні механізми на кшталт CORBA (Common Object Request Broker Architecture), RPC чи SOAP. Відповідно, REST-додатки використовують функції HTTP запиту для надсилання, зчитування та видалення даних, використовуючи повну функціональність HTTP CRUD (Create, Read, Update і Delete) операцій. До того ж REST може використовувати HTTPS, надаючи безпечне передавання даних [8].

На відміну від SOAP, REST принципи легкі для розуміння та застосування розробнику, котрий знайомий з використанням HTTP [8].

Недоліки протоколу REST є продовженням його переваг:

- складно використовувати поза HTTP, тому що немає відповідних стандартів або існуючих реалізацій;
- складно використовувати в Enterprise рішеннях, де потрібна підтримка досить складного стану (наприклад, наявності сесій і ін.);
- повна залежність від транспорту в таких питаннях, як безпека, маршрутизація повідомлень тощо, що може негативно позначитися на роботі такого сервісу в гетерогенному середовищі.

3.4 Порівняння протоколів

Для обрання протоколу для передачі даних для інтеграції з СДН «Moodle» необхідно проаналізувати відповідність даних протоколів поставленій задачі (табл. 3.2).

Таблиця 3.2 – Аналіз відповідності протоколів задачі взаємодії з СДН Moodle

	XML-RPC	SOAP	REST
Простота реалізації	+	-	+
Швидкість передачі простих структур	+	-	+
Обмін складними структурами даних	-	+	-
Формалізація взаємодії клієнта та сервера	-	+	-
Підтримка безпеки та авторизації	Покладається на розробника	Вбудована функція протоколу	Покладається на розробника
Можливість кешування відповідей та різні формати відповідей на запити	-	+	+
Багаторазові транзакції	-	+	-

Зважаючи на необхідність надійного та захищеного каналу обміну даними між СДН Moodle та іншими системами ЗВО, яким надано доступ для інтеграції, а також для отримання можливості обміну складними структурами, було обрано для реалізації інтеграційної системи протокол передачі даних SOAP. Це дозволить формалізувати взаємодію клієнта та сервера, ввівши формальні корпоративні підходи, SOAP добре сумісний з популярними великою кількістю транспортних протоколів, що дозволяє реалізувати взаємодію типу точка-точка у мережі і надає можливість іншим пристроям у мережі «розуміти» веб-сервіс без втручання людини [8].

Також SOAP широко використовується у фінансових сервісах, платіжних системах та комунікаційних сервісах. Повсюдне використання даного протоколу у цих сферах дає впевненість у подальшій його популяризації та використанні у майбутньому [8].

Окрім протоколу SOAP було прийнято рішення також застосувати архітектурний стиль REST. Його перевагами є відносна легкість у реалізації, чітке розділення серверної та клієнтської реалізацій, можливість кешування відповідей та різні формати відповідей на запити, як-то JSON чи XML. Так як REST напряду залежить від HTTP, для передачі даних між клієнтом та сервером треба менше зусиль, порівняно з SOAP чи RPC–XML[8]. Переваги REST призвели до рішення використовувати цей протокол для взаємодії з СДН Moodle у частині отримання даних з Moodle, де захищеність каналу переходить на другий план у порівнянні з швидкістю взаємодії.

3.5 Мова програмування

Так як обраний для реалізації протокол SOAP повністю працює з клієнтами PHP, ця мова програмування була обрана для реалізації інтеграційної системи з СДН Moodle.

Найважливішими з вбудованих класів PHP для організації взаємодії між вебсерверами є класи SoapClient та SoapServer. Всі необхідні методи для прийому запитів (SoapServer) реалізовані на стороні СДН Moodle. Відповідно необхідно організувати відправлення даних на серверну частину за допомогою класу SoapClient та його методів (табл. 3.3).

Також було використано наступні класи:

- клас SoapHeader - спеціальний клас низького рівня для передачі або повернення заголовків SOAP. Це просто власник даних, і він не має ніяких спеціальних методів, крім свого конструктора. Його можна використовувати в методі SoapClient->__soapCall() для передачі заголовка SOAP або в обробнику заголовка SOAP для повернення заголовка у відповіді SOAP;
- клас SoapParam - спеціальний клас низького рівня для іменування параметрів та повернення значень не у WSDL форматі. Це просто власник даних, і він не має ніяких спеціальних методів, крім свого конструктора;

– SoapVar - це спеціальний клас низького рівня для кодування параметрів і повернення значень не у WSDL форматі. Це просто власник даних і не має ніяких спеціальних методів, крім конструктора. Використовується, коли необхідно встановити властивість type у запиті або відповіді SOAP.

Таблиця 3.3 – Методи класу SoapClient

Назва методу	Застосування
SoapClient->__construct()	Створює новий об'єкт SoapClient
SoapClient->__call()	Виклик функції SOAP (застаріла)
SoapClient->__doRequest()	Виконує запит SOAP
SoapClient->__getFunctions()	Повертає список функцій SOAP
SoapClient->__getLastRequest()	Повертає останній запит SOAP
SoapClient->__getLastRequestHeaders()	Повертає останні заголовки SOAP-запиту
SoapClient->__getLastResponse()	Повертає останню відповідь SOAP
SoapClient->__getLastResponseHeaders()	Повертає останні заголовки відповідей SOAP
SoapClient->__getTypes()	Повертає список типів SOAP
SoapClient->__setCookie()	Встановлює файл cookie, який буде надісланий із запитом SOAP
SoapClient->__soapCall()	Виклик функції SOAP

3.5 Висновки за розділом 3

У розділі були розглянуті протоколи XML-RPC, REST та SOAP, які API Moodle пропонує для додавання нових функцій до вебслужби та для зв'язку з іншими системами. Зважаючи на необхідність надійного та захищеного каналу обміну даними між СДН Moodle та іншими системами

ЗВО, яким надано доступ для інтеграції, а також для отримання можливості обміну складними структурами, було обрано для реалізації інтеграційної системи протокол передачі даних SOAP.

Також були розглянуті основні класи мови PHP для організації взаємодії між вебсерверами, яка була обрана для реалізації інтеграційної системи з СНД Moodle, тому що обраний для реалізації протокол SOAP повністю працює з клієнтами PHP.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

СДН Moodle має модульну структуру, що надає можливість інтегрувати в неї різноманітні модулі як розробникам програм, так і її користувачам. До основних типів модулів відносять: елементи курсу, звіти адміністратора, типи завдань, засоби аутентифікації, блоки, формати курсів, звіти стосовно курсів, поля бази даних, фільтри для курсів, звіти щодо оцінок, формати експорту та імпорту оцінок, типи запитань в тестах, імпорт та експорт тестів, звіти стосовно тестів, архів файлів, типи ресурсів, а також модуль мобільного навчання (MLE).

Після аналізу методів інтеграції з СДН Moodle було обрано використати для вирішення завдань інтеграції API-інтерфейс. Однак, для його використання необхідно вирішити проблему взаємодії внутрішніх і зовнішніх серверів університету. Наприклад, система з обліку контингенту встановлена наразі на локальному комп'ютері і доступна тільки з внутрішньої локальної мережі університету. Система дистанційного навчання встановлена на сервері і доступна тільки з зовнішньої мережі (доменну адресу <https://moodle.zp.edu.ua>), тому необхідно організувати пряму передачу даних між ними. Модель взаємодії розробленої інтеграційної системи з іншими інформаційними системами в інформаційній інфраструктурі закладу вищої освіти представлено на рисунку 4.1.

В процесі аналізу необхідного функціоналу для інтеграційної системи було вирішено, що немає необхідності реалізовувати функціонал для зарахування студентів на курс навчальної дисципліни. Студенти до своїх курсів-потоків записуватимуться через метод зарахування «Синхронізація гурту», і таким чином переведення студента до певного гурту автоматично синхронізує його з курсом-потокком та всіма навчальними курсами поточного семестру для цього потоку. В процесі синхронізації гурту з курсом-потокком необхідно прив'язати його до такої ж групи в цьому курсі. В результаті

працівники деканату отримують можливість переглядати успішність студентів за окремими групами [5].

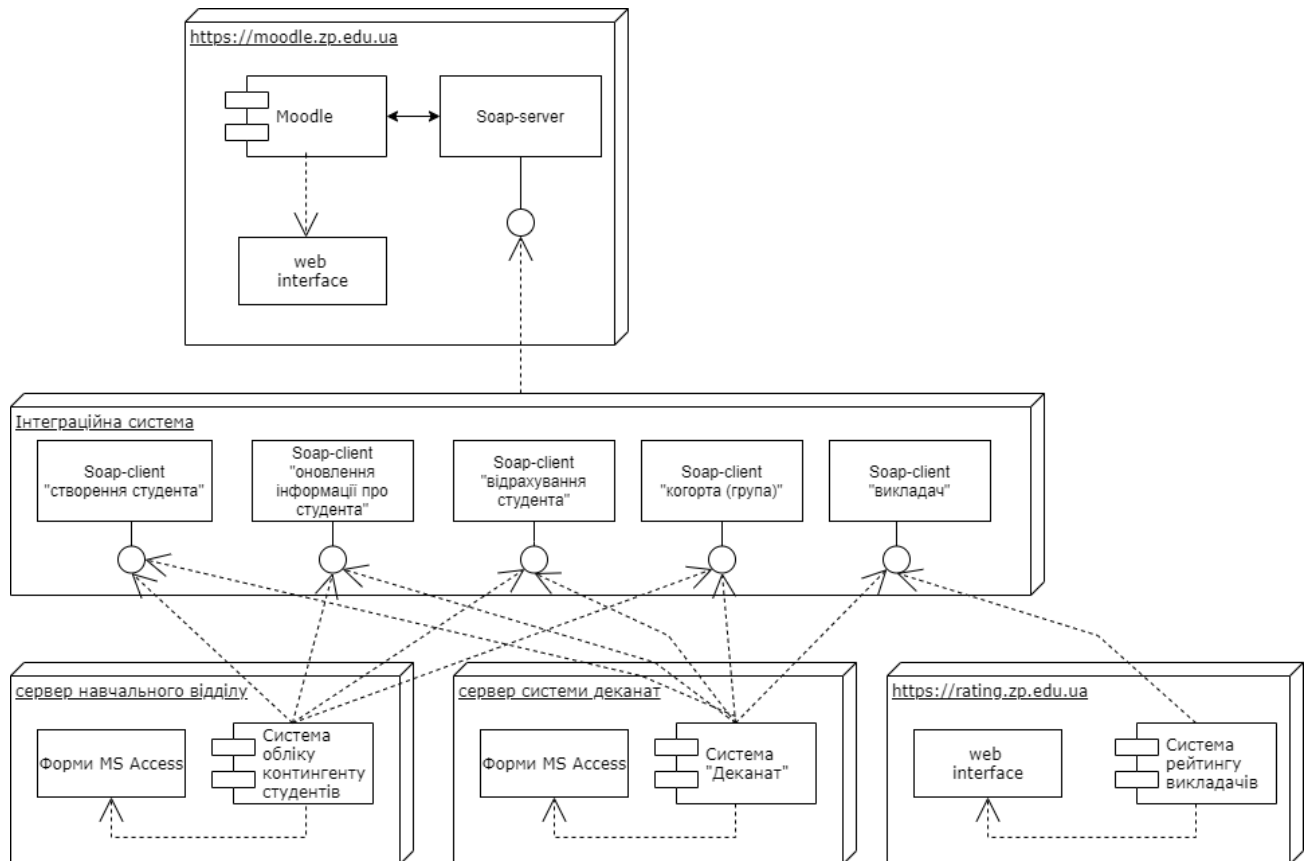


Рисунок 4.1 – Модель взаємодії розробленої інтеграційної системи з іншими інформаційними системами в інформаційній інфраструктурі закладу вищої освіти

В результаті реалізації такого підходу, буде отримана налагоджену систему зарахування студентів в електронні навчальні курси з розподілом по групах. При чому, якщо якогось студента зараховують на навчання на факультет пізніше, його достатньо ввести до відповідного гурту, далі він буде автоматично зарахованим до всіх потрібних курсів. Аналогічно, відрахованого студента буде виключено з гурту, і він перестане бути зарахованим на всіх курсах [5].

4.1 API-функції Moodle

Для реалізації інших функціональних вимог, описаних в розділі 2.2, була виконана інтеграція з API-функціями СНД Moodle [7], основні з яких наведено у табл. 4.1.

Таблиця 4.1 – Основні API-функції Moodle, обрані для інтеграції

Ім'я	Доступно з версії	Опис
1	2	3
auth_email_get_signup_settings	3.2	Отримання необхідних налаштувань для реєстрації та поля профілю
auth_email_signup_user	3.2	Додавання нового користувача (до підтвердження) на сайті
core_auth_confirm_user	3.2	Підтвердження облікового запису користувача
core_auth_request_password_reset	3.4	Запит на скидання пароля
core_auth_resend_confirmation_email	3.6	Запит на повторне підтвердження електронною поштою
core_auth_request_password_reset	3.4	Запит на скидання пароля
core_block_get_course_blocks	3.3	Отримання інформації про блоки для курсу
core_cohort_add_cohort_members	2.4	Додавання учасників до когорти
core_cohort_create_cohorts	2.4	Створення когорти
core_cohort_delete_cohort_members	2.4	Видалення учасників з когорти
core_cohort_delete_cohorts	2.4	Видалення когорти
core_cohort_get_cohort_members	2.4	Отримання членів когорти
core_cohort_get_cohorts	2.4	Отримання когорти
core_cohort_search_cohorts	3.0	Пошук когорт.
core_cohort_update_cohorts	2.4	Оновлення когорт
core_course_create_categories	2.3	Створення категорії
core_course_create_courses	2.0	Створення нового курсу
core_course_delete_categories	2.3	Видалення категорії
core_course_delete_courses	2.3	Видалення курсів
core_course_delete_modules	2.5	Видалення всіх вказаних екземплярів модуля
core_course_duplicate_course	2.3	Дублювання курсу
core_course_edit_module	3.3	Дія з модулем курсу (зміна видимості, дублювання, видалення тощо)
core_course_edit_section	3.3	Дія в розділі курсу (змінення видимості, встановлення маркера, видалення)
core_course_get_categories	2.3	Отримання категорії
core_course_get_contents	2.2	Отримання вмісту курсу (модулі + URL-адреси файлів веб-сервісу)

Продовження таблиці 4.1

1	2	3
core_course_get_course_module	3.0	Повернення інформації про модуль курсу.
core_course_get_courses_by_field	3.2	Отримання курсів, що відповідають певному полю (ідентифікатори, скорочене ім'я, ідентифікатор, категорія)
core_course_import_course	2.4	Імпорт даних курсу в інший курс (без жодних даних користувачів)
core_course_search_courses	3.0	Пошук курсів за ім'ям, модулем, блоком, тегом
core_enrol_get_enroll_users	2.1	Отримання зареєстрованих користувачів за ідентифікатором курсу
core_message_send_instant_messages	2.1	Відправлення миттєвого повідомлення
core_message_send_messages_to_conversation	3.6	Відправлення повідомлення до існуючої бесіди між користувачами
core_role_assign_roles	2.0	Розподіл ролей
core_role_unassign_roles	2.0	Видалення ролі у користувача
core_user_create_users	2.0	Створення користувачів - функція адміністратора
core_user_delete_users	2.0	Видалення користувачів - функція адміністратора
core_user_get_users	2.5	Пошук користувачів
core_user_get_users_by_field	2.5	Отримання інформації про користувачів для вказаного унікального поля
core_user_get_users_by_id	2.0	Отримання користувача за ідентифікатором
core_user_update_users	2.0	Оновлення користувачів
enrol_manual_enrol_users	2.0	Реєстрація користувачів вручну
enrol_manual_unenrol_users	3.0	Скасування реєстрацію користувачів вручну
gradereport_overview_get_course_grades	3.2	Отримання підсумкових оцінок даних курсів
tool_lp_search_cohorts	3.1	Пошук когорт
tool_lp_search_users	3.1	Пошук користувачів

4.2 Основні розроблені алгоритми

Під час розробки інтеграційної системи для спрощення та автоматизації процесів реєстрації та зміни інформації про студентів у СНД Moodle, актуалізації інформації з інших систем, що використовуються в університеті, зокрема, системи з обліку контингенту, було розроблено наступні алгоритми:

- реєстрація студента;
- поновлення студента;

- відрахування / надання академічної відпустки студенту;
- зарахування студента до групи;
- відрахування студентів після завершення навчання.

Розглянемо як приклад алгоритм поновлення студента:

- співробітники деканату обробляють заяву студента про поновлення, видають наказ;
- співробітник навчального відділу відпрацьовує цей наказ у системі з обліку контингенту;
- дані про поновленого студента передаються до інтеграційної системи (прізвище, ім'я, по батькові, група, email (за наявності)) з використанням методу передачі POST;
- інтеграційна система виконує пошук за наявними даними студента у СНД Moodle для перевірки на наявність облікового запису у системі з аналогічними даними (якщо студент навчався раніше, його обліковий запис вже може існувати) – для цього отримані через POST параметри з інформацією про студента компонуються до необхідного для передачі до API Moodle формату та викликається метод `core_user_get_users()`;
- Moodle отримує запит через SOAP протокол на виконання пошуку акаунту в системі з вказаними параметрами (прізвище, ім'я, email (за наявності)) та повертає масив `$returnedusers` з отриманими результатами;
- інтеграційна система отримує масив `$returnedusers`, якщо він містить інформацію про знайдені облікові записи, дані про це відправляються до системи з обліку контингенту (альтернативний потік), інакше – виконується перехід до наступного кроку;
- інтеграційна система формує запит на створення нового облікового запису в СНД Moodle, для чого передає у необхідному форматі масив даних про студента (прізвище, ім'я, по батькові, email (за наявності)) та викликається метод `API Moodle core_user_create_users()`;

- Moodle отримує запит через SOAPпротокол на створення акаунту в системі з вказаними параметрами (прізвище, ім'я, email (за наявності)) та повертає масив \$useridсз результатамиреєстрації;

- інтеграційна система відправляє електронний лист студенту з параметрами доступу до Moodle (якщо email студента було вказано) або на пошту деканату (якщо email студента не було вказано);

- інтеграційна система формує запит на додавання створеного облікового запису до групи (когорти) у СНД Moodle, для чого передає у необхідному форматі масив даних з ідакаунта студента та ідгрупи (для цього виконується пошук за назвою групи за допомогою методу APIMoodle core_cohort_search_cohorts()) та створення когорти, якщо відповідна група не була знайдена, за допомогою методу APIMoodle core_cohort_create_cohorts()) до методу APIMoodlecore_cohort_add_cohort_members());

- Moodle отримує запит через SOAPпротокол на додавання студенту до групи та повертає змінну \$resultсз результатамивиконання функції;

- інтеграційна система відправляє повідомлення до системи з обліку контингенту про внесення інформації про студента у СНД Moodle.

Альтернативний потік включає передачу до системи з обліку контингенту інформації про знайдений акаунт, працівник навчального відділу встановлює чи належить знайдений акаунт поновленому студенту. Якщо ні, працівник відправляє запит на створення нового акаунта. Якщо акаунт належить студенту або після створення нового акаунту, цей акаунт прив'язується до групи, в яку поновлюється студент.

Бізнес-діаграма за нотацією BPMN-2.0 [12] процесу поновлення студенту представлена на рисунку 4.2.

Алгоритм для процесів відрахування та надання академічної відпустки студенту було об'єднано, тому що було вирішено не видаляти акаунт студента одразу після відрахування, бо часто виникає необхідність поновити його до навчання впродовж року. Для деактивації акаунту було

вирішено видаляти студента з поточної групи та присвоювати статус «неактивний» акаунту (`suspended= true`).

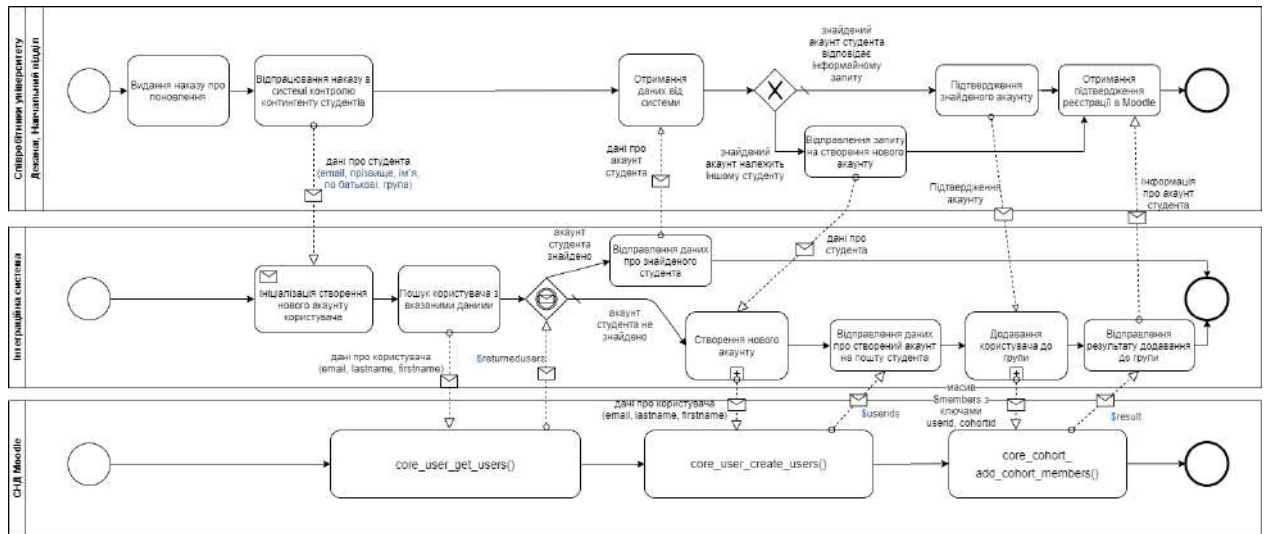


Рисунок 4.2 – Бізнес-діаграма процесу поновлення студенту

Алгоритм процесу відрахування потоку студентів:

- співробітник навчального відділу відпрацьовує в системі обліку студентів наказ про відрахування у зв'язку з завершенням навчання;
- назви груп, які відраховуються передаються до інтеграційної системи з використанням методу передачі POST;
- інтеграційна система виконує пошук за назвами груп (когорт) у СНД Moodle для отримання їх id– для цього отримані через POST назви груп компонуються до необхідного для передачі до API Moodle формату та у циклі викликається метод `search_cohorts()` для кожної групи;
- Moodle отримує запит через SOAP протокол на виконання пошуку id групи в системі за назвою групи та повертає масив `$cohorts`, який включає id груп, назви яких відповідають запиту;
- інтеграційна система отримує масив `$cohorts`, отримує з нього id група відправляє до СНД Moodle запит для отримання всіх членів цих груп за допомогою методу `core_cohort_get_cohort_members()`;

е) Moodle отримує запит через SOAP-протокол на виконання пошуку членів груп, ід яких передано до запиту, та повертає масив `$members`, який містить ід акаунтів студентів цих груп;

ж) інтеграційна система виконує деактивація всіх акаунтів отриманих студентів, для чого циклічно:

- 1) виконує пошук акаунту студента за ід – відправляється запит з ід студента до методу `APIget_user()` та отримується вся інформація про його акаунт (`$user`);

- 2) встановлює статус поля `suspended` у «true»;

- 3) виконує оновлення акаунту шляхом передачі змінної `$user` у вигляді масиву `$users` до методу `API update_users()`, у разі успішного оновлення акаунту студента метод повертає `null`;

з) інтеграційна система передає інформацію про деактивацію акаунтів студентів груп, які завершили навчання.

Бізнес-діаграма за нотацією BPMN-2.0 процесу відрахування потоку студентів (наприклад, після закінчення бакалаврату або магістратури) представлена на рисунку 4.3.

Усі дії інтеграційної системи направлено актуалізацію інформації в СНД Moodle, а також на надання додаткових можливостей, таких як автоматизація створення типових курсів, надання доступу до різних сервісів, використовуючи логін та пароль акаунту до Moodle, спрощення отримання інформації з Moodle для звітності та статистики.

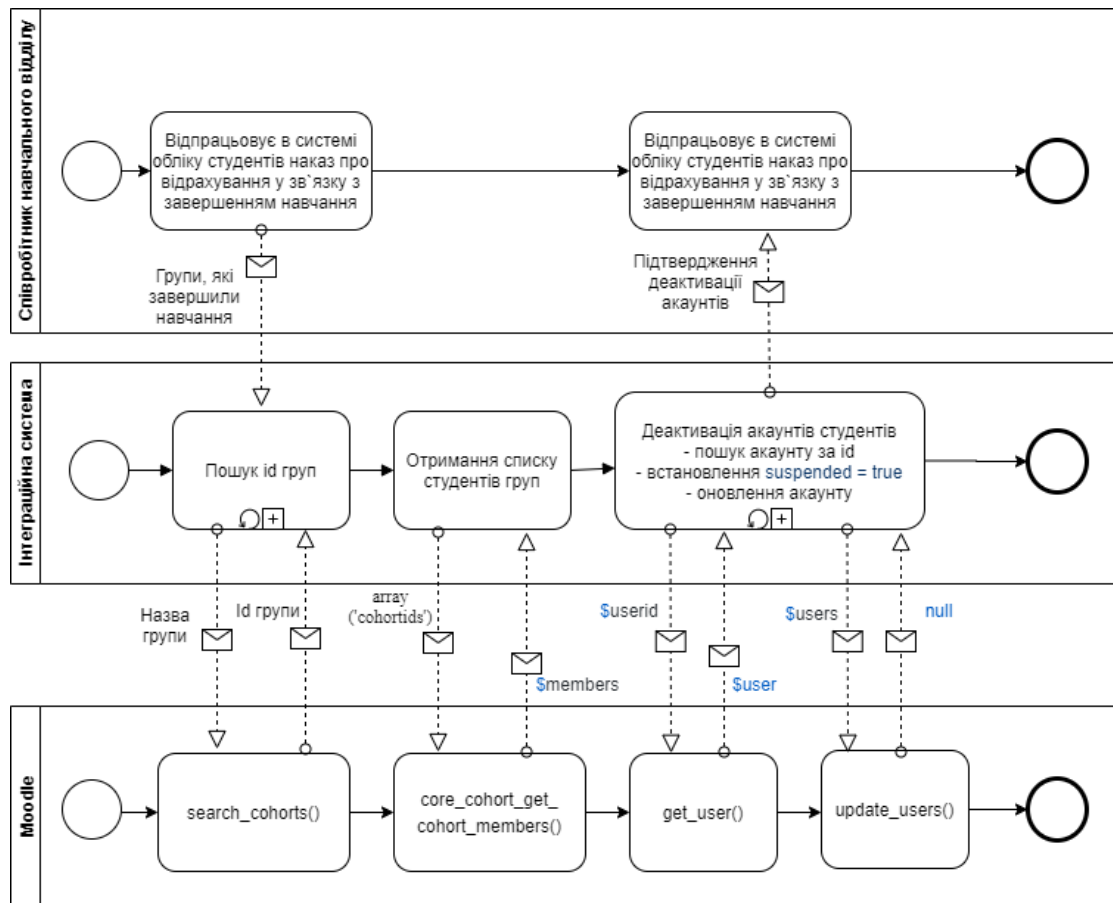


Рисунок 4.3 – Бізнес-діаграма процесу відрахування потоку студентів

4.3 Висновок за розділом 4

У цьому розділі було описано основні рішення щодо розробки програмного продукту, а саме було описано модель взаємодії розробленої інтеграційної системи з іншими інформаційними системами в інформаційній інфраструктурі університету, описані основні розроблені алгоритми та API - функції Moodle, використані під час розробки програмної системи для спрощення та автоматизації процесу взаємодії з СНД Moodle.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Порядок дій для підключення інтеграційної системи

Адміністратор Moodle має виконати підключення розробленої системи для надання їй доступу до APIметодів.

Алгоритм дій наступний:

а) увімкнення веб- сервісів:

- 1) перейти до розділу «Керування сайтом>Розширені функції»;
- 2) поставити прапорець «Дозволити Веб-сервіси» та натиснути «Зберегти зміни» (рис. 5.1);

б) увімкнення протоколів:

- 1) перейти до розділу «Керування сайтом>Модулі> Веб-служби> Керування протоколами»;
- 2) увімкнути протоколи (SOAP та REST) (рис. 5.2);

в) створення власної зовнішньої служби (рис. 5.3):

- 1) перейти до розділу «Керування сайтом>Модулі>Веб-служби>Зовнішні сервіси>Зовнішній сервіс»;
- 2) натиснути«Користувацькі сервіси>Додати», у переліку пунктів обрати:

– «Лише авторизовані користувачі» - ввімкнути та потім вибрати авторизованих користувачів вручну (в іншому випадку дозвіл отримають всі користувачі з відповідними дозволами);

– «Потрібні можливості» - ввімкнути, щоб будь-який користувач, який отримує доступ до веб-служби, перевірявся відповідно до вибраної можливості (додатковий необов'язковий рівень безпеки), в списку необхідно обрати доступні можливості (рис. 5.4);

3) ввести ім'я та встановити прапорець «Увімкнено»;

4) натиснути кнопку «Додати послугу»;

г) додавання функцій до служби:

- 1) натисніть на посилання «Додати функції»;
- 2) обрати функцію «створити групу» та натисніть кнопку «Додати функції»;
- 3) обрати необхідні можливості з переліку (рис. 5.5);
- д) надати відповідні дозволи:
 - 1) moodle/webservice:createtoken - дозволити користувачам генерувати ключ безпеки;
 - 2) webservice/rest:use, webservice/soap:use, webservice/xmlrpc:use, webservice/amf:use - для доступу до увімкнених протоколів.

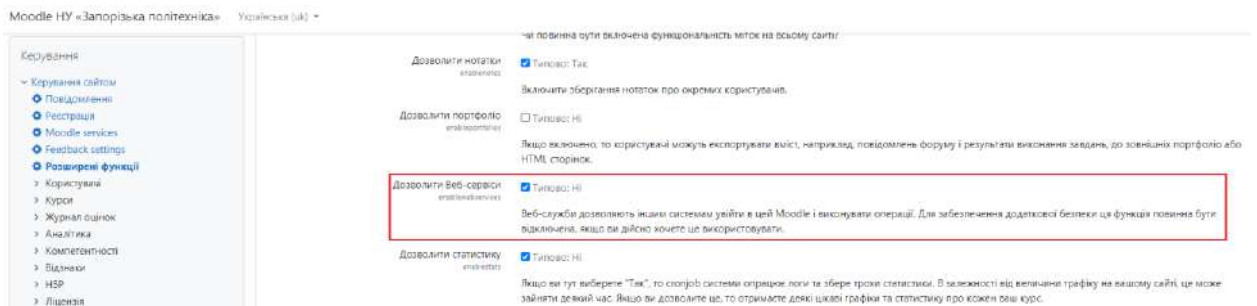


Рисунок 5.1 – Увімкнення веб-сервісів

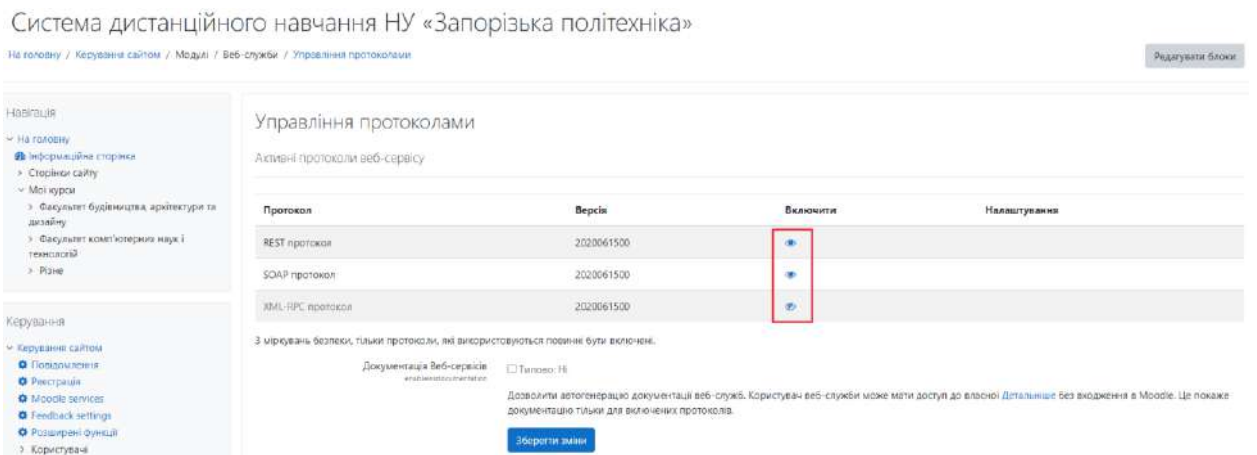


Рисунок 5.2 – Управління протоколами

Зовнішній сервіс

Назва

nuzp_users_synchro

Коротка назва

☒ Дозволено

☒ Тільки авторизовані користувачі ?

Показати додаткове...

Зберегти зміни

Скасувати

Обов'язкові поля форми помічені символом

Рисунок 5.3 – Створення зовнішнього сервісу

Веб-служби / Зовнішні сервіси / Функції / Додати функції

Редагувати блоки

Назва

Нічого не вибрано

Знайти

core_auth_is_age_digital_consent_verification_enabled:Checks if age digital consent verification is enabled.

core_auth_is_minior:Requests a check if a user is a digital minor.

core_auth_request_password_reset:Requests a password reset.

core_auth_resend_confirmation_email:Resend confirmation email.

core_backup_get_async_backup_links_backup:Gets the data to use when updating the status table row in the UI for when an async backup completes.

core_backup_get_async_backup_links_restore:Gets the data to use when updating the status table row in the UI for when an async restore completes.

core_backup_get_async_backup_progress:Get the progress of an Asynchronous backup.

core_backup_get_copy_progress:Gets the progress of course copy operations.

Обов'язкові поля форми помічені символом

Рисунок 5.4 – Надання доступу до вебслужби

Додати функцію до сервісу "nuzp_users_synchro"

Функція	Опис	Потрібні права	Редагувати
core_cohort_add_cohort_members	Adds cohort members.	moodle/cohortassign	Видалити
core_user_create_users	Create users.	moodle/user.create	Видалити

Рисунок 5.5 – Додавання функцій до вебслужби

Перелік необхідних для інтеграційної системи можливостей:

- auth_email_get_signup_settings;
- auth_email_signup_user;
- core_auth_confirm_user;
- core_auth_request_password_reset;

- core_auth_resend_confirmation_email;
- core_auth_request_password_reset;
- core_block_get_course_blocks;
- core_cohort_add_cohort_members;
- core_cohort_create_cohorts;
- core_cohort_delete_cohort_members;
- core_cohort_delete_cohorts;
- core_cohort_get_cohort_members;
- core_cohort_get_cohorts;
- core_cohort_search_cohorts;
- core_cohort_update_cohorts;
- core_course_create_categories;
- core_course_create_courses;
- core_course_delete_categories;
- core_course_delete_courses;
- core_course_delete_modules;
- core_course_duplicate_course;
- core_course_edit_module;
- core_course_edit_section;
- core_course_get_categories;
- core_course_get_contents;
- core_course_get_course_module;
- core_course_get_courses_by_field;
- core_course_import_course;
- core_course_search_courses;
- core_enrol_get_enroll_users;
- core_message_send_instant_messages;
- core_message_send_messages_to_conversation;
- core_role_assign_roles;
- core_role_unassign_roles;

- core_user_create_users;
- core_user_delete_users;
- core_user_get_users;
- core_user_get_users_by_field;
- core_user_get_users_by_id;
- core_user_update_users;
- enrol_manual_enrol_users;
- enrol_manual_unenrol_users;
- gradereport_overview_get_course_grades;
- tool_lp_search_cohorts;
- tool_lp_search_users.

5.2 Тестування та експериментальне дослідження

Для тестування розробленої інтеграційної системи було розроблено тестовий вебінтерфейс для виклику методів інтеграційної системи та передачі до них параметрів через POSTметод протоколу HTTP. Як приклад, було змодульовано процес створення нового акаунту в СНД Moodle. Через форму вводяться необхідні для реєстрації дані та відправляються до методу інтеграційної системи (рис. 5.5):

- логін користувача (username);
- пароль користувача password);
- прізвище студента (lastname);
- ім'я студента (firstname);
- прізвище студента (middlename);
- пошта студента (email);
- кафедра, до якої належить студент (department);
- група, до якої належить студент (cohort).

Назва закладу освіти (institution), місто та країна записується за замовченням (city, country).

Регістрація акаунту у системі дистанційної освіти Moodle

Логін	Логін
Пароль	Пароль
Прізвище (українською мовою за паспортом)	Прізвище
Ім'я (українською мовою за паспортом)	Ім'я
По батькові (українською мовою за паспортом)	По батькові
Мобільний номер (у форматі +380XXXXXXXX)	Мобільний номер
Пошта	Пошта
Кафедра:	Програмних засобів
Група	Група

[Зареєструватися](#)

Рисунок 5.5 – Тестовий вебінтерфейс для виклику методу інтеграційної системи «додавання нового користувача»

Інтеграційна система отримує POSTзапит, оброблює його та викликає методи API СНД Moodle:

- core_user_create_users();
- core_cohort_add_cohort_members().

Для відправлення параметрів до методів API створюється новий об'єкт класу SoapClient та викликається його метод __soapCall(). Для попередження виникнення виключних ситуацій застосовується обробник:

```
try {
    $resp = $client->__soapCall($functionname, array($params));
}
catch (SoapFault $e) {
    exit($e->faultstring);
}
```

У випадку виникнення помилки буде виконано аварійне завершення зі створенням об'єкту класу SoapFault для повідомлення про помилку.

Тестування надало можливість виявити всі можливі помилки під час взаємодії з СНД Moodle під час звернення до інтеграційної системи з інших

інформаційних систем університету, як наприклад, система з обліку контингенту (рис. 5.6).

The image shows two side-by-side screenshots of the 'Профіль Moodle' (Moodle Profile) window. Both windows have a title bar with 'Профіль Moodle' and a close button. They are divided into two main sections: 'База' (Base) and 'Профіль moodle' (Moodle Profile).

Left Screenshot (Activation):

- База:** id_stud: 485801, Прізвище: Астафієва, Ім'я: Ольга, По батькові: Олександрівна, Група: СНз-110, id_moodle: 9805.
- Профіль moodle:** Логін: studastafieva674, Пароль: [masked], Прізвище: Астафієва, Ім'я: Ольга, По батькові: Олександрівна, Гурт: 2889, Пошта: astafievaola@gmail.com, Активність: ☐ (button: Активувати), Замітка: Прийом 29.09.2020.
- Bottom:** Місто: Запоріжжя, Начальний заклад: НУ Запорізька політехніка.

Right Screenshot (Deactivation):

- База:** id_stud: 475922, Прізвище: Коротков, Ім'я: Єгор, По батькові: Сергійович, Група: М-119, id_moodle: 697.
- Профіль moodle:** Логін: studtest4060, Пароль: [masked], Прізвище: Коротков, Ім'я: Єгор, По батькові: Сергійович, Гурт: М-119, Пошта: studmoodle@zp.edu.ua, Активність: ☒ (button: Деактивувати), Замітка: Навчається.
- Bottom:** Місто: Запоріжжя, Начальний заклад: НУ Запорізька політехніка.

Рисунок 5.6 – Приклад інтерфейсу система з обліку контингенту для активації (створення) або деактивації акаунту студента

5.3 Обробка виключних ситуацій

В розробленій інтеграційній системі можливі наступні виключні ситуації, представлені в табл. 5.1, та описані дії для подолання цих помилок.

Таблиця 5.1 – Основні помилки під час інтеграції з СДН Moodle та дії для їх подолання

Помилка	Опис дій для її подолання
1	2
Не вдається автентифікувати користувача	Необхідно перейти до адміністративної частини Moodle за шляхом «Керування сайтом > Модулі > Веб-служби > Керування службами > Уповноважені користувачі», натиснути на повне ім'я користувача та відредагувати його дані
Термін дії користувача закінчився	
Не вдається автентифікувати користувача за токеном	Необхідно перейти до адміністративної частини Moodle за шляхом «Керування сайтом > Модулі > Веб-служби > Керування токенами» та перевірити дані токена
Токену користувача не існує	
Термін дії токена закінчився	

Продовження таблиці 5.1

1	2
Користувача немає в списку авторизованих	Необхідно перейти до адміністративної частини Moodle за шляхом «Керування сайтом>Модулі> Веб-служби> Керування службами> Уповноважені користувачі») та надати користувачу доступ
Користувач не має необхідних можливостей	Необхідно перейти до адміністративної частини Moodle за шляхом «Керування сайтом>Модулі> Веб-служби> Керування послугами> Редагувати послугу» та надати користувачу необхідні можливості
Функція вебслужби не включена до послуги	
Сайт в режимі обслуговування	
Дію акаунта користувача призупинено	
Акаунт користувача не підтверджений	Редагувати профіль користувача
Акаунт користувача видалено	
Для ідентифікації користувача встановлено значення nologin	
Вебслужба вимкнена	Необхідно перейти до адміністративної частини Moodle за шляхом «Керування сайтом>Розширені функції» та увімкнути вебслужбу
Користувач не має можливості використовувати використаний протокол вебсервісу	Надати доступ до протоколу (користувач повинен мати можливість 'webservice/use:xmlrpc'/'webservice/use:rest'/'..., в контексті системи)
Протокол вимкнено	Переконайтеся, що протокол REST та SOAP увімкнено («Керування сайтом> Модулі> Веб-служби> Керування протоколами»)
Недійсний параметр	Значення, які передаються, не відповідають стандарту типу параметра функції веб-служби (необхідно переконатися, що служба передає правильні значення, детально переглянувши параметри)
Недійсний доступ	Необхідно переконатись, що у користувача є доступ для запуску веб-служби (тому що вебслужба створена для авторизованих користувачів)
Не проходження всіх обов'язкових полів	Необов'язкові параметри можуть ігноруватися під час передачі, але обов'язкові поля повинні бути передані

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

У дипломній кваліфікаційній роботі магістра досліджено та виконана програмна реалізація методів інтеграції з СДН Moodle для спрощення та автоматизації процесів взаємодії.

Впровадження в експлуатацію розробленого ПЗ в закладі вищої освіти дозволить підвищити швидкість оброблення заявок до технічної підтримки, зменшити час працівників деканатів та навчального відділу на «ручну» (через вебінтерфейс) реєстрацію акаунтів студентів, актуалізацію інформації про них, прикріплення / зміну / видалення студентів з групи, тощо. Це дозволить скоротити 0,5 ставки одного інженера відділу технічної підтримки, що займається обробкою звернень щодо актуалізації даних у Moodle, заробітна плата якого складає 5000 гривень на місяць.

У розробці беруть участь програміст із заробітною платою 8000 грн. на місяць і консультант із заробітною платою 9000 грн. на місяць. На розробку проєкту було витрачено два місяці. Розробка ПЗ розпочалася першого вересня і має бути виконано до кінця жовтня 2020 року.

6.1 Планування розробки програмного продукту

Весь комплекс розробки програмного продукту можна розділити на етапи [13]. Для кожного етапу вказуються трудомісткість, кількість виконавців і тривалість робіт. Інформація з характеристики робіт по розробці програмного забезпечення зведена у таблицю 6.1.

Таблиця 6.1 – Характеристика робіт по розробці програми

Найменування робіт	Трудомісткість		Виконавці	Тривалість, днів
	люд-дн	%		
Аналіз ПЗ	5	7,5	Програміст	3
			Консультант	2
Визначення вимог до ПП	6	9	Програміст	3
			Консультант	3
Проектування структури програми	16	23,9	Програміст	9
			Консультант	7
Розробка схеми функціонування програми	5	7,5	Програміст	3
			Консультант	2
Створення програмного коду	18	26,9	Програміст	18
Тестування і налагодження програмного продукту	11	16,4	Програміст	6
			Консультант	5
Складання програмної документації	6	9	Програміст	3
			Консультант	3
Разом	67	100	Програміст	45
			Консультант	22

6.2 Визначення витрат на розробку програми

Для визначення витрат на розробку програми складають калькуляцію кошторисної вартості робіт, що включає статті:

– основна заробітна плата;

- додаткова заробітна плата;
- відрахування на єдиний соціальний внесок (ЄСВ);
- матеріали та комплектуючі вироби;
- витрати на спеціальне обладнання (на оплату машинного часу ЕОМ для написання і налагодження програмних засобів);
- накладні витрати.

Витрати за статтею «Основна заробітна плата» складаються з планового фонду зарплати всіх категорій працівників, зайнятих в розробці програми. Розрахунок зарплати на підставі даних про трудомісткість, представлений в таблиці 6.2.

Таблиця 6.2 – Основна заробітна плата

Посада виконавця	Чисельність, чол	місячний оклад	Кількість місяців роботи	Сума ЗП, грн
Програміст	1	8100	2	16200
Консультант	1	9100	1	9100
Разом	2			25300

Додаткову заробітну плату приймають рівною 10% від основної зарплати. Вона розраховується за формулою (6.1):

$$ЗП_{\text{дод}} = ЗП_{\text{осн}} \cdot 10\%. \quad (6.1)$$

Підставивши величину основної заробітної плати в дану формулу отримаємо:

$$ЗП_{\text{дод}} = 25300 \cdot 0.1 = 2530 \text{ грн.}$$

Відрахування на єдиний соціальний внесок (ЄСВ) визначають у відсотковому відношенні від сум основної та додаткової зарплати з урахуванням премій і доплат. Відрахування вираховуються за формулою (6.2):

$$\text{ЄСВ} = (\text{ЗП}_{\text{осн}} + \text{ЗП}_{\text{дод}}) \cdot 0,22, \quad (6.2)$$

$$\text{ЄСВ} = (25300 + 2530) \cdot 0,22 = 6122 \text{ грн.}$$

Використовується два найменування матеріалів: папір – 250 грн., картридж – 950 грн. Витрати на матеріали розраховують за формулою (6.3):

$$M = \sum_{i=1}^n (\text{Ц}_i \cdot N_i \cdot (1 + K_{\text{т.з}}) - \text{Ц}_{io} \cdot N_{io}), \quad (6.3)$$

де M – витрати на матеріали, покупні напівфабрикати і комплектуючі вироби, грн.;

$K_{\text{т.з}}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

Ц_i – ціна i -го найменування матеріалу, напівфабрикату і комплектуючого, грн.;

N_i – потреба в i -му матеріалі, напівфабрикаті і комплектуючому;

Ц_{io} – ціна зворотних відходів i -го найменування матеріалу, грн.;

N_{io} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів і комплектуючих.

$$\text{Ц}_{io} = 0; N_{io} = 0; K_{\text{т.з}} = 0,05,$$

$$M = (1 + 0,05) \cdot (250 + 950) = 1260 \text{ грн.}$$

Таким чином, разом витрати на матеріали та комплектуючі вироби становлять 1260 гривень.

У статтю «Витрати на спеціальне обладнання» входять витрати на придбання, транспортування, монтаж і налагодження нестандартного

обладнання. Практично, в даному випадку, в цій статті враховуються витрати на оплату машинного часу ЕОМ для написання і налагодження програмних засобів. Для цього необхідно скласти кошторис «витрат на утримання і експлуатацію устаткування», виходячи з якого визначиться вартість одного машино-години роботи ПК, після множення якої на машинний час пішло на написання і налагодження програми отримаємо витрати на оплату машинного часу.

Амортизаційні відрахування визначають за формулою (6.4):

$$A = БВ \cdot \frac{H_a}{100}, \quad (6.4)$$

де БВ – балансова вартість обчислювальної техніки, грн;

H_a – норма амортизаційних відрахувань на повне оновлення обчислювальної техніки,%. Приймаємо, що техніка повинна працювати 4 роки, тому $H_a = 25\%$.

Балансова вартість обчислювальної техніки становить 31200 грн. Тому амортизаційні відрахування становлять:

$$A = 31200 \cdot 0,25 = 7800 \text{ грн.}$$

Статтю "Експлуатація обладнання" розраховують підсумовуванням витрат на силову електроенергію і допоміжні матеріали. Витрати на електроенергію визначають за формулою (6.5):

$$C_{\Sigma} = N_H \cdot \Phi_{\text{еф}} \cdot K_{\text{зв}} \cdot K_{\text{зм}} \cdot C_e, \quad (6.5)$$

де N_H – номінальна потужність ЕОМ, кВт;

$\Phi_{\text{еф}}$ – річний ефективний фонд часу роботи ЕОМ, машино–год;

$K_{\text{зв}}$ – середній коефіцієнт завантаження за часом;

$K_{зм}$ – коефіцієнт завантаження по потужності;

Ц_ϵ – ціна одного кВт · год електроенергії, грн / (кВт · год).

Номінальна Потужність ЕОМ – 0,2 кВт. Річний ефективний фонд часу роботи ЕОМ ставити 1800 годин. Середні коефіцієнти завантаження за часом та за потужністю рівні відповідно 0,9 та 0,6. Ціна однієї кіловат-години електроенергії ставить 2,11 грн. Отримуємо:

$$C_e = 0,2 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,11 = 410,18 \text{ грн.}$$

Приймаємо, що $C_e = 410$ грн.

Витрати за статтею "Заробітна плата обслуговуючих робітників та відрахування на соціальне страхування в інші фонди" визначають за формулою (6.6):

$$ЗП_{\text{обсл}} = \Phi ЗП_{\Gamma} \cdot (1 + K_{\text{відрах.}}) \cdot \frac{t_{\text{обсл}}}{\Phi_{\text{еф.обсл}}}, \quad (6.6)$$

де $\Phi ЗП_{\Gamma}$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робітників, грн.;

$K_{\text{відрах}}$ – коефіцієнт, що враховує відрахування на соціальне страхування і в інші фонди;

$t_{\text{обсл}}$ – час протягом року, необхідне технічне обслуговування ЕОМ, годин / рік. Приймаємо $t_{\text{обсл}} = 36$ год.;

$\Phi_{\text{еф.обсл}}$ – річний ефективний фонд часу обслуговуючого персоналу, годин / рік. Приймаємо, що річний ефективний фонд часу обслуговуючого персоналу становить 1800 годин

Заробітна плата обслуговуючого персоналу дорівнює 6000 грн/місяць.

Розрахуємо:

$$\Phi ЗП_{\Gamma} = 6000 \cdot 12 = 7200 \text{ грн.,}$$

$$ЗП_{\text{обсл}} = \frac{72000 \cdot (1 + 0,22) \cdot 36}{1800} = 1757 \text{ грн.}$$

Суму витрат за статтею "Поточний ремонт обладнання" обчислимо як 3% від балансової вартості обладнання, оскільки неможливо заздалегідь визначити, скільки коштів необхідно витратити на придбання запасних частин наявного обладнання. Балансова вартість обчислювальної техніки становить 31200 грн. Тому сума витрат за статтею "Поточний ремонт обладнання" становить: $31200 \cdot 0,03 = 936$ грн.

Інших витрат за проектом не передбачено, тому сума витрат за цією статтею дорівнює нуль гривень.

Кошторис витрат на утримання і експлуатацію устаткування наведений в таблиці 6.3.

Таблиця 6.3 – Кошторис витрат на утримання і експлуатацію устаткування

Найменування статей витрат	Сума, грн.
Амортизація обладнання	7800
Заробітна плата основна і додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	1757
Експлуатація обладнання (крім витрат на поточний ремонт)	410
Поточний ремонт обладнання	936
Інші витрати	0
Разом	10903

Витрати на оплату машинного часу ЕОМ для написання і налагодження програмних засобів визначаються за формулою (6.7):

$$C_{\text{мо}} = P_{\text{екс}} \cdot t_{\text{мо}}, \quad (6.7)$$

де $C_{\text{мо}}$ – витрати на оплату машинного часу, грн;

$P_{\text{екс}}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн / машино–год.;

$t_{\text{мо}}$ – машинний час цифрової ЕОМ для написання і налагодження даного програмного продукту, машино–год.

$$P_{\text{екс}} = \frac{10903}{1800} = 6,06 \text{ грн/машин – год.}$$

Для написання і налагодження даного програмного продукту ЕОМ експлуатується протягом 45 днів в одну зміну по 8 годин, що складає в сумі 360 годин. Таким чином отримуємо суму за статтею «Витрати на спеціальне обладнання»:

$$C_{\text{мо}} = 6,06 \cdot 360 = 2181 \text{ грн.}$$

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і т.д.), утримання та експлуатацію будівель. Накладні витрати включаються до вартості розробки програми непрямым шляхом – у відсотках до основної заробітної плати розробників. Для обчислення цієї статті видатків у дипломному проєкті приймемо, що накладні видатки становлять 10% від основної заробітної плати розробників, що складає 25300 грн. Тому сума накладних видатків становить: $25300 \cdot 0,1 = 2530$ грн.

Калькуляція кошторисної вартості робіт з розробки програми наведена в таблиці 6.4.

Таблиця 6.4 – Калькуляція кошторисної вартості робіт з розробки програми

Найменування статей витрат	Сума, грн.
Основна заробітна плата	25300
Додаткова заробітна плата	2530
ЄСВ	6122
Матеріали і комплектуючі	1260
Витрати на спеціальне обладнання	2181
Накладні витрати	2530
Разом	39923

6.3 Розрахунок економічної ефективності програмного продукту

Річну економію від впровадження програми, що дозволяє знизити витрати машинного часу ЕОМ для рішення певного завдання в порівнянні із програмою, що застосовувалася раніше, визначають за формулою (6.8):

$$E = (T_{M_1} - T_{M_2}) \cdot B_{\text{екс.КТЗ}} + E_{\text{н}}^a \cdot \frac{\Phi_{\text{б.КТЗ}} \cdot (T_{M_1} - T_{M_2})}{\Phi_{\text{еф.КТЗ}}} - \frac{S_{\text{рп}}}{T_{\text{с}}} + E_{\text{зп}}, \quad (6.8)$$

де T_{M_1}, T_{M_2} – машинний час, необхідний для розв’язання поставлених завдань, відповідно в старому та у новому варіантах, машино–год./рік;

$B_{\text{екс}}$ – експлуатаційні витрати, що доводяться на 1 год машинного часу ЕОМ, грн/ машино–год.;

$E_{\text{н}}^a$ – нормативний коефіцієнт ефективності капітальних вкладень у засоби автоматизації;

$\Phi_{\text{б.КТЗ}}$ – балансова вартість ЕОМ (КТЗ), грн.;

$\Phi_{\text{еф.КТЗ}}$ – річний ефективний фонд часу роботи ЕОМ, машино–год.;

$S_{\text{рп}}$ – сумарні витрати на розробку програми, грн.;

T_c – термін служби впроваджуваної програми до її морального зношування, років. Приймаємо цей термін 5 років;

$E_{\text{зп}}$ – економія на заробітній платі.

До впровадження даного ПЗ дані функції виконувались співробітником, що відповідає за обробку рукописних символів, 1800 годин на рік, а після 900 годин на рік (економія складає 0,5 ставки одного інженера відділу технічної підтримки, з заробітною платою 6100 гривень на місяць).

$$E = (1800 - 900) \cdot 6,06 + 0,2 \cdot (31200 \cdot (1800 - 900)) / 1800 - 39923/5 + \\ + 6100 \cdot 0,5(1 + 0,22) \cdot 12 = 45241 \text{ грн.}$$

Коефіцієнт ефективності капітальних вкладень E и строк їхньої окупності $T_{\text{ок}}$ розраховуються за формулами (6.9) та (6.10):

$$E\phi = \frac{E}{S_{\text{рп}}}, \quad (6.9)$$

$$T_{\text{ок}} = \frac{S_{\text{рп}}}{E}, \quad (6.10)$$

$$E\phi = 45241 / 39923 = 1,13, \\ T_{\text{ок}} = 39923 / 45241 = 0,88 \text{ року.}$$

Зведемо отримані техніко–економічні показники в таблицю 6.5.

Таблиця 6.5 – Техніко-економічні показники

Показники	Сума	Одиниця виміру
Витрати на розробку програми	39923	грн.
Річна економія	45241	грн.
Строк окупності програми	0,88	рік

Таким чином, впровадження в експлуатацію розробленого програмного забезпечення на підприємстві дозволить скоротити 0,5 ставки одного інженера відділу технічної підтримки, що займається обробкою звернень щодо актуалізації даних у Moodle, заробітна платня якого складає 6100 гривень на місяць. Економічні розрахунки показали, що строк окупності програми склав 0,88 року, що значно менше часу її нормативного строку окупності (5 років). Крім того, річна економія при використанні програми становить 45241 грн. Виходячи з виконаних розрахунків, можна зробити висновок, що розробка даного програмного продукту є економічно доцільною.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

Дослідження та програмна реалізація методів інтеграції з системою дистанційного навчання Moodle передбачає роботу в приміщенні обчислювального центру.

Це передбачає використання обчислювальної техніки (персональної ЕОМ, периферійних пристроїв, тощо) та, як і будь-яка виробнича діяльність, пов'язане з впливом небезпечних і шкідливих виробничих чинників. Небезпечні і шкідливі чинники визначені в ДСанПіН «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу» №0472-14 [13].

Усі потенційні небезпеки, що виникають при роботі в обчислювальному центрі можна поділити на такі групи:

- потенційні небезпеки, фізичного характеру;
- потенційні небезпеки психофізіологічного характеру;
- потенційні небезпеки санітарно-гігієнічного характеру;
- потенційні небезпеки, пов'язані з порушеннями правил пожежної безпеки;
- потенційні небезпеки, пов'язані з проявом наслідків надзвичайних ситуацій.

До потенційних небезпек фізичного характеру належать механічні травмування, що можуть статися в наслідок невідповідного планування робочого простору виробничого приміщення (розташування електричних або комунікаційних дротів на підлозі в зонах з високою прохідністю, несправні офісні меблі, погано закріплені полиці та шафи, тощо), можливі ураження електричним струмом під час роботи за комп'ютером або використанні периферійної комп'ютерної техніки (в наслідок порушення правил безпеки під час роботи або використанні несправного обладнання), тощо. Відомо, що

робота за комп'ютером супроводжується значним статичним навантаженням на опорно-руховий апарат та органи зору. Професійним захворюванням фахівців, чия праця пов'язана з довготривалим використанням ЕОМ (інженери-програмісти, оператори ЕОМ, тощо) є синдром зап'ястного каналу, або тунельний синдром зап'ястя, який може бути наслідком хронічної травми. Причиною цього захворювання є тривала робота з мишею: постійна напруга і здавлювання приводить до мікротравм, здавлювання нерва прилеглими оточуючими тканинами, через що виникає набряк.

Потенційні небезпеки психофізіологічного характеру пов'язані з незадовільною організацією робочого місця (недостатнє врахуванням вимог ергономіки робочого місця, зокрема нераціональне розташування комп'ютерного та периферійного обладнання, неправильно підібрані столи та стільці відносно параметрів людини) або виробничого процесу (надлишкові комунікації під час робіт, що потребують зосередженості та значних розумових зусиль, тощо).

До потенційних небезпек санітарно-гігієнічного характеру можна віднести ті, що виникають в наслідок порушення вимог санітарно-гігієнічних норм до приміщення, в якому безпосередньо здійснюється виробничий процес.

До групи небезпек, що пов'язані з порушенням правил пожежної безпеки, можна віднести потенційні небезпеки, що пов'язані з недотриманням або неналежним дотриманням правил пожежної безпеки при плануванні виробничого приміщення, виробничого процесу або під час використання пожежонебезпечного обладнання чи устаткування (комп'ютерної техніки, периферійних пристроїв, тощо).

Нарешті, до потенційних небезпек, що пов'язані з проявами наслідків надзвичайних ситуацій, слід віднести ті, що пов'язані з невідповідністю персоналу в умовах надзвичайних ситуацій (наприклад, під час пожежі або стихійного лиха).

7.2 Заходи із забезпечення безпеки

Для запобігання виникненню потенційних небезпек фізичного, психофізіологічного, санітарно-гігієнічного характеру та небезпек, що пов'язані з порушеннями правил пожежної безпеки, та з проявом наслідків надзвичайних ситуацій, і мінімізації впливу шкідливих виробничих факторів передбачено низку заходів по забезпеченню техніки безпеки.

Передбачені заходи можна поділити на організаційні та технічні.

З метою запобігання ураження, під час виконання службових обов'язків, персоналу електричним струмом передбачено низку заходів технічного характеру згідно Правил улаштування електроустановок «ПУЕ-2017» [14], ДСТУ Б В.2.5-82:2016 Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом [15], відповідно яких захисне заземлення наявного електроустаткування не перевищує 4 Ом.

З організаційних заходів, передбачене проведення навчання з правил електробезпеки, перевірка знань та атестація персоналу на кваліфікаційну групу з електробезпеки, згідно НПАОП 0.00-4.12-05 «Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці» [13], з метою попередження можливих травмувань, проведення періодичних інструктажів з правил електробезпеки.

Для профілактики синдрому зап'ястного каналу у операторів ОЕМ, що використовують розроблений під час виконання проекту програмний додаток проведено відповідний інструктаж з персоналом і внесено відповідний пункт до правил внутрішньої трудової дисципліни, а саме необхідність перерв протягом декількох хвилин після кожних 45 хвилин безперервної роботи за комп'ютером.

Для запобігання впливу факторів, що обумовлюють психофізіологічні небезпеки впроваджено низку заходів організаційного характеру, а саме:

- оптимізація і обмеження системи комунікацій (вертикальних і горизонтальних) у відповідності до потреб існуючих бізнес процесів організації;
- введення системи тайм-менеджменту;
- проведення корпоративних заходів розважального характеру для співробітників організації;
- введення системи контролю за виконанням вимог внутрішнього трудового розпорядку.

7.3 Заходи з виробничої санітарії та гігієни праці

До санітарно-гігієнічних умов відносять:

- мікрокліматичні умови та їх забезпечення (відповідно до вимог ДСН 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень» [16], та ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування» [17]);
- освітленість робочої зони та її забезпечення (відповідно до вимог ДБН В.2.5-28:2018 «Природне та штучне освітлення» [18]);
- рівень шуму в робочій зоні (відповідно до вимог ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» [19]);
- виробничі вібрації (відповідно до ДСН 3.3.6.039-99 «Державні санітарні норми виробничої загальної локальної вібрації»);
- електромагнітні поля та випромінювання (відповідно до НПАОП 0.00-7.15-18 Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроям [20];
- ергономіка робочого місця (відповідно до ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги» [21].

При організації роботи операторів ЕОМ було враховано вимоги НПАОП 0.00-7.15-18. Було розроблено і узгоджено інструкції щодо режиму

роботи співробітників організації з урахуванням специфіки роботи, що регламентують час початку і час закінчення робочого дня, тривалість обідньої перерви, наявність і тривалість додаткових перерв, тощо.

Відповідно до ДСН 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень» робота з розробки програмного забезпечення відноситься до легкої категорії фізичних робіт (категорія Ia). Тож норми мікроклімату відповідають параметрам визначеним в ДСН 3.3.6-042-99 [16], (табл. 7.1).

Для забезпечення оптимального рівня параметрів повітряного середовища у приміщенні влаштована система водяного опалення приміщення, що забезпечує необхідну температуру повітря в холодний період року відповідно до ДБН В.2.5-67:2013 [17].

Для забезпечення оптимального температурного режиму приміщення у теплий період року встановлено систему кондиціонування повітря.

Таблиця 7.1 – Рівень параметрів повітряного середовища виробничого приміщення

Період року	Температура, °C	Відносна вологість, %	Швидкість руху, м/с
Холодний	22-24	40-60	0,1-0,3
Теплий	23-25	40-60	0,1-0,4

Необхідний повітрообмін у приміщенні визначається за формулою (7.1):

$$L = \frac{Q}{c \cdot \gamma \cdot (t - t_{np})}, \quad (7.1)$$

де Q – сумарна кількість теплоти, що утворюється в приміщенні, кДж/год;

C – питома теплоємність повітря, що дорівнює $C = 1 \text{ кДж}/(\text{кг} \cdot ^\circ\text{C})$;

γ – густина зовнішнього повітря, що дорівнює $\gamma = \frac{353}{(273 + t_{np})}$, $\text{кг}/\text{м}^3$;

t – температура повітря, що видаляється, °C;

t_{np} – температура приточного повітря.

Сумарна кількість теплоти, що утворюється у приміщенні за годину, очікується на рівні 5500 кДж, а температура повітря, що видаляється, 27 °C. Встановлена система кондиціювання повітря забезпечує температуру приточного повітря на рівні 19 °C.

Таким чином, необхідний повітрообмін у приміщенні дорівнюватиме:

$$L = \frac{5500}{1 \cdot \frac{353}{(273+19)} \cdot (27-19)} = 568,8 \text{ м}^3/\text{год}.$$

Отже для асиміляції надлишкового тепла у приміщенні в теплий період року необхідне встановлення системи кондиціювання повітря, що забезпечить повітрообмін на рівні 600 м³ за годину. На виконання цієї вимоги було встановлено мультизональний кондиціонер Mitsubishi Heavy FDTС45 КХЕ6 з касетним внутрішнім блоком і потужністю (на охолодження) 4500 Вт.

Штучне освітлення в приміщеннях з робочими місцями, обладнаними ВДТ ЕОМ і ПЕОМ здійснюється системою загального рівномірного освітлення. Освітленість на поверхні столу в зоні розміщення документів згідно розряду зорового навантаження від 5мм клас VI освітлення складає 200-300 лк згідно ДБН В.2.5-28:2018 «Природне і штучне освітлення». Якщо ці значення освітленості неможливо забезпечити системою загального освітлення, допускається використати місцеві освітлення. При цьому місцеве освітлення встановлювати так, щоб не створювати відблисків на поверхні екрану, а освітленість екрану більше 250 лк.

Передбачається, що в приміщенні не буде використовуватися устаткування, під час роботи якого, за умови справного стану, можливе перевищення допустимих норм шуму (згідно ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку»). Враховуючи те, що

потенційним джерелом шуму може виявитись друкувальний пристрій (принтер), його місце знаходження обрано таким чином, щоб він знаходився якомога далі від робочих місць персоналу.

Враховуючи наявність в приміщенні встановленого захисного заземлення наявного електроустаткування і відповідність комп'ютерної техніки сучасним стандартам безпеки і енергосбереження, не передбачається перевищення норм електромагнітного випромінювання зазначених у ДСанПІН 3.3.2.007 – 98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [22].

Усі комп'ютерні монітори на робочих місцях відповідають стандарту безпеки і енергозбереження ТСО'99 і, відповідно, не викликають перевищення рівнів електромагнітного і іонізуючого випромінювання (згідно ДСанПІН 3.3.2.007 – 98).

Для забезпечення ергономіки робочого місця передбачено проведення інструктажу по ДСанПіН 3.3.2.007-98 [22].

Робоче місце має наступні показники:

- відстань між бічними поверхнями сусідніх дисплеїв – 1.2 м;
- відстань від тильної поверхні одного дисплея до екрану іншого дисплея – 2.5 м;
- прохід між рядами робочих місць має бути не менше 1 м;
- висота робочої поверхні робочого столу – 680-800 мм;
- ширина робочої поверхні робочого столу – 600-1400 мм;
- глибина робочої поверхні робочого столу – 800-1000 мм;
- простір для ніг: заввишки – 600 мм, ширина – 500 мм, глибина – 450 мм;

Конструкція робочого місця користувача ВДТ повинна забезпечувати підтримку оптимальної робочої пози з такими ергономічними характеристиками:

- ступні ніг - на підставці для ніг або на підлозі;

- стегна - в горизонтальній площині;
- передпліччя - вертикально;
- лікті - під кутом 70-90 градусів до вертикальної площини;
- зап'ястки - зігнуті під кутом не більше 20 градусів відносно горизонтальної площини;
- нахил голови – 15-20 градусів відносно вертикальної площини.

7.4 Заходи з забезпечення безпеки у надзвичайних ситуаціях

7.4.1 Заходи з пожежної безпеки

Розробка програмного забезпечення супроводжується використанням не лише комп'ютерної техніки та необхідних меблів, а також веденням відповідної документації для чого в приміщенні додатково встановлюється необхідне електрообладнання (принтери) та організовується зберігання паперів. Також за необхідності може бути встановлене і додаткове електрообладнання (проектори, кухонні прилади тощо). Вся техніка, що може знаходитися в приміщенні, меблі, а також супутні матеріали, при неправильній організації пожежної безпеки можуть спричинити пожежу. Для мінімізації ризику виникнення пожежі в обчислювальному центрі маємо необхідність впровадження правил та заходів з пожежної безпеки.

Планування та впровадження заходів пожежної безпеки здійснено у відповідності до Закону України Про пожежну безпеку, Правил пожежної безпеки в Україні та вимогам відповідних нормативних актів.

Згідно НАПБ.03.001-2004 "Типові норми належності вогнегасників" тип і необхідну кількість вогнегасників визначають згідно з наступними параметрами:

- граничну захисну площу;
- клас пожежі горючих речовин і матеріалів;
- категорію приміщень і будівель за вибухопожежною і пожежною небезпекам.

При плануванні заходів з пожежної безпеки враховано, що обчислювальний центр, згідно ДСТУ Б В.1.1-36:2016 «Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою відноситься до категорії «Д» [23], а клас можливої пожежі, відповідно до ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги», визначається як «А» (горіння твердих речовин) та «Е» (горіння електроустановок, що перебувають під напругою).

Враховуючи площу приміщення, що складає 42 м², згідно з ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» [24], приміщення обладнане двома порошковими вогнегасниками ВП-5.

Розроблено графік перевірки вогнегасників і призначено відповідальну особу. Вогнегасники в приміщенні розташовані таким чином, щоб відстань від найвіддаленішого місця приміщення до найближчого вогнегасника не перевищувала 20 метрів.

Окрім цього, згідно вимог ДБН В.2.5-56:2014 «Системи протипожежного захисту», приміщення обладнане пожежною сигналізацією, що використовує комбіновані теплові мультисенсорні датчики та димові іонізаційні датчики, підключені до сигналізаційної димової пожежної установки типу СДПУ-1 [25].

З метою запобігання пожеж в обчислювальному центрі обладнаному ПК з ВДТ передбачені наступні протипожежні заходи. На силовому обладнанні, силових та освітлювальних колах, згідно вимог пункту 3.1 «ПУЕ», встановлені захисні пристрої, що вимикають джерело живлення від ділянки електричного кола, у якій виникло коротке замикання.

Меблі та устаткування розташоване таким чином, щоб забезпечити вільний прохід у разі термінової евакуації завширшки не менш 1 метру.

Для обробки стін, підлоги та стелі приміщення використано негорючі полімерні матеріали, що зменшує ризики швидкого розповсюдження вогню у разі виникнення пожежі.

З організаційних заходів щодо забезпечення пожежної безпеки відповідно до вимог ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» розроблено план евакуації персоналу в разі виникнення пожежі і розміщено в доступному для перегляду місці, проведено інструктаж з персоналом щодо дотримання правил пожежної безпеки у приміщенні і вміння користуватися наявними засобами пожежогасіння, розроблено план навчань персоналу (з використанням учбових пожежних тривог).

7.4.2 Заходи з цивільного захисту. Єдина державна система цивільного захисту, її складові та режими функціонування

Єдина державна система цивільного захисту – це сукупність органів управління, сил і засобів центральних та місцевих органів виконавчої влади, Ради міністрів Автономної Республіки Крим, виконавчих органів рад, підприємств, установ та організацій, які забезпечують реалізацію державної політики у сфері цивільного захисту.

Відповідно до Кодексу Цивільного захисту України цивільний захист забезпечується з урахуванням особливостей, визначених Законом України «Про основи національної безпеки України», суб'єктами, уповноваженими захищати населення, території, навколишнє природне середовище і майно, згідно з вимогами цього Кодексу – у мирний час, а також в особливий період – у межах реалізації заходів держави щодо оборони України.

Координацію діяльності органів виконавчої влади у сфері цивільного захисту у межах своїх повноважень здійснюють:

- Рада національної безпеки і оборони України;
- Кабінет Міністрів України.

Для координації діяльності центральних і місцевих органів виконавчої влади, підприємств, установ та організацій, пов'язаної з техногенно-

екологічною безпекою, захистом населення і територій, запобіганням і реагуванням на надзвичайні ситуації:

- Кабінетом Міністрів України утворюється Державна комісія з питань техногенно-екологічної безпеки та надзвичайних ситуацій;
- районними державними адміністраціями, виконавчими органами міських рад, районними у містах та селищними радами утворюються місцеві комісії з питань техногенно-екологічної безпеки та надзвичайних ситуацій;
- керівними органами підприємств, установ та організацій утворюються комісії з питань надзвичайних ситуацій.

Для координації робіт з ліквідації конкретної надзвичайної ситуації та її наслідків на державному, регіональному, місцевому та об'єктовому рівнях утворюються спеціальні комісії з ліквідації надзвичайної ситуації.

Основні завдання єдиної державної системи цивільного захисту:

- забезпечення готовності міністерств та інших центральних та місцевих органів виконавчої влади, органів місцевого самоврядування, підпорядкованих їм сил і засобів до дій, спрямованих на запобігання і реагування на надзвичайні ситуації;
- забезпечення реалізації заходів щодо запобігання виникненню надзвичайних ситуацій;
- навчання населення щодо поведінки та дій у разі виникнення надзвичайної ситуації;
- виконання державних цільових програм, спрямованих на запобігання надзвичайним ситуаціям, забезпечення сталого функціонування підприємств, установ та організацій, зменшення можливих матеріальних втрат;
- опрацювання інформації про надзвичайні ситуації, видання інформаційних матеріалів з питань захисту населення і територій від наслідків надзвичайних ситуацій;

- прогнозування і оцінка соціально-економічних наслідків надзвичайних ситуацій, визначення на основі прогнозу потреби в силах, засобах, матеріальних та фінансових ресурсах;
- створення, раціональне збереження і використання резерву матеріальних та фінансових ресурсів, необхідних для запобігання і реагування на надзвичайні ситуації;
- оповіщення населення про загрозу та виникнення надзвичайних ситуацій, своєчасне та достовірне інформування про фактичну обстановку і вжиті заходи;
- захист населення у разі виникнення надзвичайних ситуацій;
- проведення рятувальних та інших невідкладних робіт щодо ліквідації наслідків надзвичайних ситуацій, організація життєзабезпечення постраждалого населення;
- пом'якшення можливих наслідків надзвичайних ситуацій у разі їх виникнення;
- здійснення заходів щодо соціального захисту постраждалого населення;
- реалізація визначених законом прав у сфері захисту населення від наслідків надзвичайних ситуацій, в тому числі осіб (чи їх сімей), що брали безпосередню участь у ліквідації цих ситуацій;
- інші завдання, визначені законом.

Єдина державна система цивільного захисту залежно від масштабів і особливостей надзвичайної ситуації, що прогнозується або виникла, може функціонувати в наступних режимах:

- повсякденного функціонування;
- підвищеної готовності;
- надзвичайної ситуації;
- надзвичайного стану.

Режим повсякденного функціонування— встановлюється за умов нормальної виробничо-промислової, радіаційної, хімічної, сейсмічної,

гідрогеологічної, гідрометеорологічної, техногенної та пожежної обстановки та за відсутності епідемій, епізоотій, епіфітотій.

Режим підвищеної готовності– встановлюється тимчасово, у разі загрози виникнення надзвичайної ситуації, у повному обсязі або частково для окремих територіальних підсистем (ОТП) ЄДСЦЗ (єдиної державної системи цивільного захисту) за рішенням відповідно Кабінету Міністрів України, обласних, Київської міської державної адміністрації.

Режим надзвичайної ситуації– встановлюється тимчасово, у разі виникнення надзвичайної ситуації, у повному обсязі або частково для ОТП ЄДСЦЗ (окремих територіальних підсистем єдиної державної системи цивільного захисту) за рішенням відповідно Кабінету Міністрів України, обласних, Київських міських державних адміністрацій.

Режим надзвичайного стану– встановлюється тимчасово, у повному обсязі або частково, у межах території, на якій введено правовий режим надзвичайного стану для ОТП ЄДСЦЗ відповідно до Закону України «Про правовий режим надзвичайного стану».

ВИСНОВКИ

В роботі було проведено дослідження та програмна реалізація методів інтеграції з системою дистанційного навчання Moodle для спрощення та автоматизації процесів взаємодії. Для досягнення цієї мети було досліджено методи інтеграції програмних систем та варіанти реалізації топології інтегрованих систем, виконано порівняння методів взаємодії з СДН Moodle, виявлено вимоги для реалізації методів інтеграції та на їх основі розроблено методи для інтеграції СДН Moodle з іншими інформаційними системами ЗВО.

В пояснювальній записці описані варіанти топології інтегрованих систем, виконано аналіз основних методів інтеграції програмних систем, наведено їх недоліки та переваги; виконано аналіз та порівняння існуючих методів інтеграції з СДН Moodle; обрано найкращім методом для автоматизації процесів керування обліковими записами користувачів СДН Moodle метод інтеграції через сервіси інтерфейсу прикладного програмування; описані основні рішення щодо розробки програмного продукту, створена модель взаємодії розробленої інтеграційної системи з іншими інформаційними системами в інформаційній інфраструктурі університету, описані основні розроблені алгоритми та API -функції Moodle.

Для розроблення інтеграційної системи були розглянуті протоколи XML-RPC, REST та SOAP, які API Moodle пропонує для додавання нових функцій до вебслужби та для зв'язку з іншими системами, було обрано для реалізації інтеграційної системи протокол передачі даних SOAP.

Після розроблення програмного продукту було проведено тестування та експериментальне дослідження створеної інтеграційної системи. Виявлені можливі помилки було класифіковано та описано дії для їх подолання.

Інтеграції розробленої системи з системою обліку контингенту і СДН Moodle дозволяє:

- здійснювати автоматичну реєстрацію студентів в СДН Moodle, раніше внесених в систему з обліку контингенту;

- автоматично формувати глобальні групи в СДН Moodle у відповідності з академічними групами в системі з обліку контингенту;
- автоматизувати управління обліковими записами студентів в СДН Moodle при виконанні наказів по студентському контингенту в системі з обліку контингенту: видалення облікового запису при відрахуванні, блокування на період академічної відпустки, відновлення, переміщення / виключення студента з глобальної групи;
- організувати автоматизовану передачу результатів внутрісесеместрових атестацій з СДН Moodle.

Практична цінність розробленої інтеграційної системи полягає в тому, що впровадження в експлуатацію розробленого ПЗ в закладі вищої освіти дозволить підвищити швидкість оброблення заявок до технічної підтримки, зменшити час працівників деканатів та навчального відділу на «ручну» (через вебінтерфейс) реєстрацію акаунтів студентів, актуалізацію інформації про них, прикріплення / зміну / видалення студентів з групи, тощо.

Наукова новизна роботи полягає в розробленій моделі взаємодії інтеграційної системи з іншими інформаційними системами в інформаційній інфраструктурі закладу вищої освіти, розроблених алгоритмів взаємодії з API - функціями СДН Moodle.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Триус, Ю.В. Система електронного навчання ВНЗ на базі MOODLE / Ю.В. Триус, І.В. Герасименко, В.М. Франчук. – Черкаси: ЧДТУ, 2012. – 222 с.
2. Система дистанційного навчання НУ «Запорізька політехніка» [Електрон. ресурс]. – Режим доступу: <https://moodle.zp.edu.ua>
3. Топології комп'ютерної мережі [Електрон. ресурс]. – Режим доступу https://comp-net.at.ua/index/topologija_komp_39_juternikh_merezh/0-6
4. Система openmeetings 2.x та її інтеграція з Moodle 2.X [Електрон. ресурс]. – Режим доступу: <http://2013.moodlemoot.in.ua/course/view.php?id=44>
5. Інтеграція Moodle в інформаційно-освітнє середовище університету [Електрон. ресурс]. – Режим доступу: <http://2016.moodlemoot.in.ua/>
6. Using web services [Electronic resource]. – Access mode: https://docs.moodle.org/310/en/Using_web_services
7. Web service API functions [Electronic resource]. – Access mode: https://docs.moodle.org/dev/Web_service_API_functions
8. Кравчук, Є.С. Аналіз протоколів взаємодії у клієнт-серверній архітектурі / Є.С. Кравчук // Міжнародний науковий журнал. – 2016. – № 6(1). – С. 44 – 48. – Режим доступу: [http://nbuv.gov.ua/UJRN/mnj_2016_6\(1\)__11](http://nbuv.gov.ua/UJRN/mnj_2016_6(1)__11)
9. CORS [Электрон. ресурс]. – Режим доступа: <https://developer.mozilla.org/ru/docs/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/CORS>
10. What is XML-RPC? [Electronic resource]. – Access mode: <http://xmlrpc.com/>
11. Архитектура REST [Электрон. ресурс]. – Режим доступа: <https://habr.com/ru/post/38730/>
12. Business Process Model and Notation (BPMN),

Version 2.0 [Electronic resource]. – Access mode: <http://www.bpmn.org/>

13. Жидецький, В.Ц. Основи охорони праці : підручник / В.Ц. Жидецький. – 5-те вид., доп. – К. : Знання, 2014. – 373 с.
14. Правила улаштування електроустановок [Текст]. – Міненерговугілля України, 2017. – 617с.
15. ДБН В.2.5-27-2006 Електробезпека в будівлях і спорудах. Загальні вимоги [Текст]. – [Чинний від 2017-04-01]. – К: «УкрНДНЦ», 2016. – 109 с.
16. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень [Текст]. – [Чинний від 1999-12-01]. – Харків : Форт, 2000. 24 с.
17. ДБН В.2.5-67:2013 Опалення, вентиляція та кондиціонування [Текст]. – Київ, Мінрегіон України, 2013. – 240 с.
18. ДБН В.2.5-28:2018. Природне та штучне освітлення : посібник [Текст] : проблеми природного і штучного освітлення. – К. : Етін, 2019. – 180 с.
19. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку [Текст]. – К. : МОЗ України, 1999. – 24 с.
20. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроям. НПАОП 0.00-7.15-18. – [Чинний від 2018-05-18]. – К. : Мінсоцполітики України, 2018. – 6 с.
21. ДСТУ 8604:2015 Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги [Текст]. – Наказ від 21.12.2015
22. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Електрон. ресурс] : ДСанПіН 3.3.2.007-98.– Чинний від 1998-12-10. – К. : МОЗ України, 1998. – Режим доступу: <http://mozdocs.kiev.ua/view.php?id=2445>.
23. Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою [Текст] : ДСТУ Б

В.1.1-36:2016. – На заміну НАПБ Б.03.002-2007 ; чинний від 2017-01-01. – К. : Мінрегіонбуд України, 2016. – 66 с.

24. ДБН В.1.1-7:2016 Пожежна безпека об'єктів будівництва. Загальні вимоги [Текст].– Київ: Міністерство регіонального розвитку, будівництва та житлово-комунального господарства України, 2017. – 39 с

25. ДБН В.2.5-56:2014 [Текст]. Системи житлово-комунального господарства України, 2015. – 134с.

26. В.О., Гончарук В.Є., Качан С.І., Мохняк С.М. Основи цивільного захисту [Текст]: навч. посібник / Васійчук. – Львів, 2010.- 384 с.

ДОДАТОК А
Текст програми

create_users.php

```

<?php
$functionname = 'core_user_create_users';

$return_arr = array();

if (isset($_POST['username']) &&
    isset($_POST['password']) &&
    isset($_POST['email']) &&
    isset($_POST['lastname']) &&
    isset($_POST['firstname']) &&
    isset($_POST['groupname'])
)
{
    try {
        $user1 = new stdClass();
        $user1->username = cleanName($_POST['username']);
        $user1->password = cleanName($_POST['password']);
        $user1->lastname = cleanName($_POST['lastname']);
        $user1->firstname = cleanName($_POST['firstname']);
        $user1->middlename = cleanName($_POST['midname']);
        if (!$user1->middlename) {
            $user1->middlename = "";
        }
        $user1->alternatename = $user1->lastname & " " & $user1->
        >firstname & " " & $user1->middlename;
        $user1->email = cleanName($_POST['email']);
        $user1->department = cleanName($_POST['fac']);
        $user1->institution = 'НУ Запорізька політехніка';
        $user1->city = 'Запоріжжя/Zaporizhzhia';
        $user1->country = 'UA';
        $user1->auth = 'manual';
        $params = array($user1);

        // SOAP CALL
        $serverurl =
        'https://moodle.zp.edu.ua/webservice/soap/server.php?wsdl=1&wstoken=';
        //Do the main soap call
        $client = new SoapClient($serverurl);
        try {
            $resp = $client->__soapCall($functionname,
            array($params));
        }
        catch (SoapFault $e) {
            exit($e->faultstring);
        }

        var_dump($resp);

        //add user to group
        $userid = $resp[0]['id'];
        $groupname = cleanName($_POST['groupname']);
        $functionname = 'core_cohort_add_cohort_members';
        $cohortadd = new stdClass();
        $cohortadd->cohorttype =
        array('type'=>'idnumber', 'value'=>$groupname);
    }
}

```

```

        $cohortadd->usertype =
array('type'=>'id','value'=>$userid);
        $params = array($cohortadd);
        $serverurl =
'https://moodle.zp.edu.ua/webservice/soap/server.php?wsdl=1&wstoken=';
        ///Do the main soap call
        $client = new SoapClient($serverurl);
        try {
            $resp = $client->__soapCall($functionname,
array($params));
        }
        catch (SoapFault $e) {
            exit($e->faultstring);
        }
        var_dump($resp);
        $return_arr['status'] = "success";
        $return_arr['message'] = $resp;

        } catch(PDOException $e) {
            $return_arr['status'] = "error";
            $return_arr['message'] = $e->getMessage();
        }
    } else {
        $return_arr['status'] = "error";
        $return_arr['message'] = "There are should be next POST
params: username, password, lastname, firstname, groupname, email";
    }
    var_dump($return_arr);
    //return $return_arr;
?>

```

add_cohort_members.php

```

<?php
$functionname = 'core_cohorts_add_cohort_members';

$return_arr = array();

if (isset($_POST['cohort_id']) && isset($_POST['user_id']))
{
    try {

        $foundcohort = $DB->get_record('cohort', array('id' =>
$_POST['cohort_id']));

        $roleid = $this->assignUserCapability('moodle/cohort:assign', $contextid);

        // Call the external function.
        $addcohortmembers =
core_cohort_external::add_cohort_members(array($foundcohort));
        $addcohortmembers =
external_api::clean_returnvalue(core_cohort_external::add_cohort_membe
rs_returns(), $addcohortmembers);

        $this->assertEquals(1, count($addcohortmembers));
    }
}

```

```

        foreach ($addcohortmembers as $addcohortmember) {
            $dbcohort = $DB->get_record('cohort_members',
array('cohortid' => $foundcohort->id));
            $this->assertEquals($dbcohort->cohortid,
$foundcohort['cohorttype']['value']);
            $this->assertEquals($dbcohort->userid,
$_POST['user_id']);
        }

        // Call without required capability.
        $foundcohort = array(
            'cohorttype' => array('type' => 'id', 'value' =>
$foundcohort->id),
            'usertype' => array('type' => 'id', 'value' => '2')
        );
        $this->unassignUserCapability('moodle/cohort:assign',
$contextid, $roleid);
        $this->
>expectException(required_capability_exception::class);
        $addcohortmembers =
core_cohort_external::add_cohort_members(array($foundcohort));

        var_dump($addcohortmembers);
        $return_arr['status'] = "success";
        $return_arr['message'] = $addcohortmembers;

    } catch(PDOException $e) {
        $return_arr['status'] = "error";
        $return_arr['message'] = $e->getMessage();
    }
} else {
    $return_arr['status'] = "error";
    $return_arr['message'] = "There are should be next POST
params: cohort_id, user_id";
}

return $return_arr;
?>

```

create_cohort.php

```

<?php
$functionname = 'core_cohorts_create_cohorts';

$return_arr = array();

if (isset($_POST['cohort_name']))
{
    try {

        $category = $this->getDataGenerator()-
>create_category();

        $new_cohort = array(
            'categorytype' => array('type' => 'id', 'value' =>
$category->id),
            'name' => cleanName($_POST['cohort_name']),

```

```

        'idnumber' => cleanName($_POST['cohort_name']),
        'description' => '',
        'theme' => 'classic'
    );

    // Call the external function.
    $this->setCurrentTimeStart();
    $createdcohorts =
core_cohort_external::create_cohorts(array($new_cohort));
    $createdcohorts =
external_api::clean_returnvalue(core_cohort_external::create_cohorts_r
eturns(), $createdcohorts);

    // Check we retrieve the good total number of created
cohorts + no error on capability.
    $this->assertEquals(2, count($createdcohorts));

    foreach ($createdcohorts as $createdcohort) {
        $dbcohort = $DB->get_record('cohort', array('id' =>
$createdcohort['id']));
        if ($createdcohort['idnumber'] ==
$new_cohort['idnumber']) {
            $sconid = $DB->get_field('context', 'id',
array('instanceid' => $new_cohort['categorytype']['value'],
'contextlevel' => CONTEXT_COURSECAT));
            $this->assertEquals($dbcohort->contextid,
$sconid);
            $this->assertEquals($dbcohort->name,
$new_cohort['name']);
            $this->assertEquals($dbcohort->description,
$new_cohort['description']);
            $this->assertEquals($dbcohort->visible, 1);
            $this->assertEquals($dbcohort->theme,
$new_cohort['theme']);
        } else {
            $this->fail('Unrecognised cohort found');
        }
        $this->assertTimeCurrent($dbcohort->timecreated);
        $this->assertTimeCurrent($dbcohort->timemodified);
    }

    // Call when $CFG->allowcohortthemes is disabled.
    set_config('allowcohortthemes', 0);
    $createdcohorts =
core_cohort_external::create_cohorts(array($new_cohort));
    $createdcohorts =
external_api::clean_returnvalue(core_cohort_external::create_cohorts_r
eturns(), $createdcohorts);
    foreach ($createdcohorts as $createdcohort) {
        $dbcohort = $DB->get_record('cohort', array('id' =>
$createdcohort['id']));
        if ($createdcohort['idnumber'] ==
$new_cohort['idnumber']) {
            $sconid = $DB->get_field('context', 'id',
array('instanceid' => $new_cohort['categorytype']['value'],
'contextlevel' => CONTEXT_COURSECAT));
            $this->assertEquals($dbcohort->contextid,
$sconid);

```

```

        $this->assertEquals($dbcohort->name,
$new_cohort['name']);
        $this->assertEquals($dbcohort->description,
$new_cohort['description']);
        $this->assertEquals($dbcohort->visible, 1);
        $this->assertEquals($dbcohort->theme, '');
    }
}

// Call without required capability.
$this->unassignUserCapability('moodle/cohort:manage',
$contextid, $roleid);
$this-
>expectException(required_capability_exception::class);
$createdcohorts
core_cohort_external::create_cohorts(array($new_cohort));

var_dump($createdcohorts);
$return_arr['status'] = "success";
$return_arr['message'] = $createdcohorts;

} catch(PDOException $e) {
    $return_arr['status'] = "error";
    $return_arr['message'] = $e->getMessage();
}
} else {
    $return_arr['status'] = "error";
    $return_arr['message'] = "There are should be next POST
params: cohort_name";
}

return $return_arr;
?>

```

delete_cohort_members.php

```

<?php
$functionname = 'core_cohorts_delete_cohort_members';

$return_arr = array();

if (isset($_POST['cohort_id']) && isset($_POST['user_id']))
{
    try {

        $foundcohort = $DB->get_record('cohort', array('id' =>
$_POST['cohort_id']));

        $context = context_system::instance();
        $roleid = $this-
>assignUserCapability('moodle/cohort:assign', $context->id);

        $cohortaddmember1 = array(
            'cohorttype' => array('type' => 'id', 'value' =>
$foundcohort->id),
            'usertype' => array('type' => 'id', 'value' =>
$_POST['user_id'])

```

```

        );
        $cohortmembers1 =
core_cohort_external::add_cohort_members(array($cohortaddmember1));
        $cohortmembers1 =
external_api::clean_returnvalue(core_cohort_external::add_cohort_membe
rs_returns(), $cohortmembers1);

        // Check
        $this->assertEquals(2, $DB-
>count_records_select('cohort_members', ' ((cohortid = :idcohort1 AND
userid = :iduser1)
        OR (cohortid = :idcohort2 AND userid = :iduser2))',
        array('idcohort1' => $foundcohort->id, 'iduser1' =>
$_POST['user_id'], 'idcohort2' => $foundcohort->id, 'iduser2' =>
$_POST['user_id'])));

        // Call the external function.
        $cohortdell = array(
            'cohortid' => $foundcohort->id,
            'userid' => $_POST['user_id']
        );

core_cohort_external::delete_cohort_members(array($cohortdell));

        // Check we retrieve no cohorts + no error on
capability.
        $this->assertEquals(0, $DB-
>count_records_select('cohort_members', ' ((cohortid = :idcohort1 AND
userid = :iduser1)',
        array('idcohort1' => $foundcohort->id, 'iduser1' =>
$user1->id)));

        // Call without required capability.
        $this->unassignUserCapability('moodle/cohort:assign',
        $context->id, $roleid);
        $this-
>expectException(required_capability_exception::class);

        $return_arr['status'] = "success";
        $return_arr['message'] = "deleted";

        } catch(PDOException $e) {
            $return_arr['status'] = "error";
            $return_arr['message'] = $e->getMessage();
        }
    } else {
        $return_arr['status'] = "error";
        $return_arr['message'] = "There are should be next POST
params: cohort_id, user_id";
    }

    return $return_arr;
?>

```

delete_cohorts.php

```

<?php
$functionname = 'core_cohorts_delete_cohorts';

$return_arr = array();

if (isset($_POST['cohort_name']))
{
    try {

        $foundcohort = core_cohort_external::search_cohorts($_POST['cohort_name'],
        $coursecontext, 'all');

        $contextid = $foundcohort->contextid;
        $roleid = $this->assignUserCapability('moodle/cohort:manage', $contextid);

        // Call the external function.
        core_cohort_external::delete_cohorts(array($foundcohort->id));

        $this->unassignUserCapability('moodle/cohort:manage',
        $contextid, $roleid);
        $this->expectException(required_capability_exception::class);
        core_cohort_external::delete_cohorts(array($foundcohort->id));

        $return_arr['status'] = "success";
        $return_arr['message'] = "deleted";

    } catch(PDOException $e) {
        $return_arr['status'] = "error";
        $return_arr['message'] = $e->getMessage();
    }
} else {
    $return_arr['status'] = "error";
    $return_arr['message'] = "There are should be next POST
params: cohort_name";
}

return $return_arr;
?>

```

delete_users.php

```

<?php
$functionname = 'core_user_delete_users';

$return_arr = array();

if (isset($_POST['email']) &&
    isset($_POST['lastname']) &&
    isset($_POST['firstname']))

```

```

    )
    {
        try {

            $searchparams = array(
                array('key' => 'lastname', 'value' =>
$_POST['lastname']),
                array('key' => 'email', 'value' => $_POST['email']),
                array('key' => 'firstname', 'value' =>
$_POST['firstname']));

            // Call the external function.
            $result = core_user_external::get_users($searchparams);

            $result =
external_api::clean_returnvalue(core_user_external::get_users_returns(
), $result);
            $expectedreturnedusers = 1;
            $returnedusers = $result['users'];
            $this->assertEquals($expectedreturnedusers,
count($returnedusers));

            foreach($returnedusers as $returneduser) {
                try {
                    $user1 = ($returneduser['id'] == $USER->id) ?
                        $USER :
$foundusers[$returneduser['id']];

                    $context = context_system::instance();
                    $roleid = $this-
>assignUserCapability('moodle/user:delete', $context->id);

                    // Call the external function.
                    core_user_external::delete_users(array($user1-
>id));

                    // Call without required capability.
                    $this-
>unassignUserCapability('moodle/user:delete', $context->id, $roleid);
                    $this-
>expectException('required_capability_exception');
                    core_user_external::delete_users(array($user1-
>id));

                } catch (SoapFault $e) {
                    exit($e->faultstring);
                }
                var_dump($result);
                $return_arr['status'] = "success";
                $return_arr['message'] = $result;

            } catch(PDOException $e) {
                $return_arr['status'] = "error";
                $return_arr['message'] = $e->getMessage();
            }
        } else {
            $return_arr['status'] = "error";

```

```

        $return_arr['message'] = "There are should be next POST
params: lastname, firstname, email";
    }

    return $return_arr;
?>

```

get_cohorts.php

```

<?php
$functionname = 'core_cohorts_get_cohorts';

$return_arr = array();

if (isset($_POST['cohort_id']))
{
    try {

        $context = context_system::instance();
        $roleid = $this->assignUserCapability('moodle/cohort:view', $context->id);

        // Call the external function.
        $returnedcohorts = core_cohort_external::get_cohorts($_POST['cohort_id']);
        $returnedcohorts = external_api::clean_returnvalue(core_cohort_external::get_cohorts_retu
rns(), $returnedcohorts);

        // Check we retrieve the good total number of enrolled
cohorts + no error on capability.
        $this->assertEquals(2, count($returnedcohorts));

        foreach ($returnedcohorts as $enrolledcohort) {
            if ($enrolledcohort['idnumber'] == $foundcohort->idnumber) {
                $this->assertEquals($foundcohort->name,
                $enrolledcohort['name']);
                $this->assertEquals($foundcohort->description,
                $enrolledcohort['description']);
                $this->assertEquals($foundcohort->visible,
                $enrolledcohort['visible']);
                $this->assertEquals($foundcohort->theme,
                $enrolledcohort['theme']);
            }
        }

        // Check that a user with cohort:manage can see the
cohort.
        $this->unassignUserCapability('moodle/cohort:view',
        $context->id, $roleid);
        $roleid = $this->assignUserCapability('moodle/cohort:manage', $context->id, $roleid);
        // Call the external function.
        $returnedcohorts = core_cohort_external::get_cohorts(array(
            $foundcohort->id));
    }
}

```

```

        $returnedcohorts =
external_api::clean_returnvalue(core_cohort_external::get_cohorts_retu
rns(), $returnedcohorts);

        // Check we retrieve the good total number of enrolled
cohorts + no error on capability.
        $this->assertEquals(2, count($returnedcohorts));

        // Check when allowcohortstheme is disabled, theme is
not returned.
        set_config('allowcohortthemes', 0);
        $returnedcohorts =
core_cohort_external::get_cohorts(array(
            $foundcohort->id));
        $returnedcohorts =
external_api::clean_returnvalue(core_cohort_external::get_cohorts_retu
rns(), $returnedcohorts);
        foreach ($returnedcohorts as $enrolledcohort) {
            if ($enrolledcohort['idnumber'] == $foundcohort-
>idnumber) {
                $this->assertNull($enrolledcohort['theme']);
            }
        }

        var_dump($returnedcohorts);
        $return_arr['status'] = "success";
        $return_arr['message'] = $returnedcohorts;

    } catch(PDOException $e) {
        $return_arr['status'] = "error";
        $return_arr['message'] = $e->getMessage();
    }
} else {
    $return_arr['status'] = "error";
    $return_arr['message'] = "There are should be next POST
params: cohort_id";
}

return $return_arr;
?>

```

get_users.php

```

<?php
$functionname = 'core_user_get_users';

$return_arr = array();

if (isset($_POST['email']) &&
    isset($_POST['lastname']) &&
    isset($_POST['firstname'])
)
{
    try {

        $searchparams = array(

```

```

        array('key'      =>      'lastname',      'value'      =>
$_POST['lastname']),
        array('key' => 'email', 'value' => $_POST['email']),
        array('key'      =>      'firstname',      'value'      =>
$_POST['firstname']));

        // Call the external function.
        $result = core_user_external::get_users($searchparams);

        // We need to execute the return values cleaning process
to simulate the web service server
        $result =
external_api::clean_returnvalue(core_user_external::get_users_returns(
), $result);

        // Check we retrieve the good total number of enrolled
users + no error on capability.
        $expectedreturnedusers = 1;
        $returnedusers = $result['users'];
        $this->assertEquals($expectedreturnedusers,
count($returnedusers));

        foreach($returnedusers as $returneduser) {
            $founduser = ($returneduser['id'] == $USER->id) ?
                                $USER                                :
$foundusers[$returneduser['id']];
            $this->assertEquals($founduser->username,
$returneduser['username']);
            if (!empty($founduser->idnumber)) {
                $this->assertEquals($founduser->idnumber,
$returneduser['idnumber']);
            }
            $this->assertEquals($founduser->firstname,
$returneduser['firstname']);
            $this->assertEquals($founduser->lastname,
$returneduser['lastname']);
            if ($founduser->email != $USER->email) { // Don't
check the tmp modified $USER email.
                $this->assertEquals($founduser->email,
$returneduser['email']);
            }
            if (!empty($founduser->address)) {
                $this->assertEquals($founduser->address,
$returneduser['address']);
            }
            if (!empty($founduser->phone1)) {
                $this->assertEquals($founduser->phone1,
$returneduser['phone1']);
            }
            if (!empty($founduser->department)) {
                $this->assertEquals($founduser->department,
$returneduser['department']);
            }
            if (!empty($founduser->institution)) {
                $this->assertEquals($founduser->institution,
$returneduser['institution']);
            }
            if (!empty($founduser->description)) {

```

```

        $this->assertEquals($founduser->description,
$returneduser['description']);
    }
    if (!empty($founduser->city)) {
        $this->assertEquals($founduser->city,
$returneduser['city']);
    }
    if (!empty($founduser->country)) {
        $this->assertEquals($founduser->country,
$returneduser['country']);
    }
    if (!empty($founduser->url)) {
        $this->assertEquals($founduser->url,
$returneduser['url']);
    }
}

// Test the invalid key warning.
$warnings = $result['warnings'];
$this->assertEquals(count($warnings), 1);
$warning = array_pop($warnings);
$this->assertEquals($warning['item'], 'invalidkey');
$this->assertEquals($warning['warningcode'],
'invalidfieldparameter');

// Test sending twice the same search field.
try {
    $searchparams = array(
        array('key' => 'lastname', 'value' =>
$_POST['lastname']),
        array('key' => 'email', 'value' => $_POST['email']),
        array('key' => 'firstname', 'value' =>
$_POST['firstname']));

    // Call the external function.
    $result =
core_user_external::get_users($searchparams);
    $this->fail('Expecting \'keyalreadyset\'
moodle_exception to be thrown.');
```

```

    } catch (SoapFault $e) {
        exit($e->faultstring);
    }
    var_dump($result);
    $return_arr['status'] = "success";
    $return_arr['message'] = $result;

    } catch(PDOException $e) {
        $return_arr['status'] = "error";
        $return_arr['message'] = $e->getMessage();
    }
} else {
    $return_arr['status'] = "error";
    $return_arr['message'] = "There are should be next POST
params: lastname, firstname, email";
}

return $return_arr;
?>

```

get_users_by_field.php

```

<?php
$functionname = 'core_user_get_users_by_field';

$return_arr = array();

if (isset($_POST['email']) &&
    isset($_POST['lastname']) &&
    isset($_POST['firstname'])
)
{
    try {

        $searchparams = array(
            array('key' => 'lastname', 'value' =>
$_POST['lastname']),
            array('key' => 'email', 'value' => $_POST['email']),
            array('key' => 'firstname', 'value' =>
$_POST['firstname']));

        $fieldstosearch = array('id', 'idnumber',
'username', 'email');

        foreach ($fieldstosearch as $fieldtosearch) {

            // Call the external function.
            $returnedusers =
core_user_external::get_users_by_field($fieldtosearch,
            array($USER->{$fieldtosearch},
$user1->{$fieldtosearch}, $user2->{$fieldtosearch}));
            $returnedusers =
external_api::clean_returnvalue(core_user_external::get_users_by_field
_returns(), $returnedusers);

            if ($fieldtosearch == 'idnumber') {
                $expectedreturnedusers = 2;
            } else {
                $expectedreturnedusers = 3;
            }

            // Check we retrieve the good total number of
enrolled users + no error on capability.
            $this->assertEquals($expectedreturnedusers,
count($returnedusers));

            foreach($returnedusers as $returneduser) {
                $founduser = ($returneduser['id'] == $USER-
>id) ?
                    $USER
                    :
                    $foundusers[$returneduser['id']];
                $this->assertEquals($founduser->username,
$returneduser['username']);
                if (!empty($founduser->idnumber)) {
                    $this->assertEquals($founduser-
>idnumber, $returneduser['idnumber']);
                }
            }
        }
    }
}

```

```

        $this->assertEquals($founduser->firstname,
$returneduser['firstname']);
        $this->assertEquals($founduser->lastname,
$returneduser['lastname']);
        if ($founduser->email != $USER->email) {
//don't check the tmp modified $USER email
        $this->assertEquals($founduser->email,
$returneduser['email']);
        }
        if (!empty($founduser->address)) {
        $this->assertEquals($founduser->address,
$returneduser['address']);
        }
        if (!empty($founduser->phone1)) {
        $this->assertEquals($founduser->phone1,
$returneduser['phone1']);
        }
        if (!empty($founduser->department)) {
        $this->assertEquals($founduser->
>department, $returneduser['department']);
        }
        if (!empty($founduser->institution)) {
        $this->assertEquals($founduser->
>institution, $returneduser['institution']);
        }
        if (!empty($founduser->description)) {
        $this->assertEquals($founduser->
>description, $returneduser['description']);
        }
        if (!empty($founduser->descriptionformat)
and isset($returneduser['descriptionformat'])) {
        $this->assertEquals($founduser->
>descriptionformat, $returneduser['descriptionformat']);
        }
        if (!empty($founduser->city)) {
        $this->assertEquals($founduser->city,
$returneduser['city']);
        }
        if (!empty($founduser->country)) {
        $this->assertEquals($founduser->country,
$returneduser['country']);
        }
        // Default language and no theme were used
for the user.
        $this->assertEquals($CFG->lang,
$returneduser['lang']);
        $this->assertEmpty($returneduser['theme']);
    }
}

// Test that no result are returned for search by
username if we are not admin
$this->setGuestUser();

// Call the external function.
$returnedusers =
core_user_external::get_users_by_field('username',

```

```

        array($USER->username, $user1->username,
$user2->username));
        $returnedusers =
external_api::clean_returnvalue(core_user_external::get_users_by_field
_returns(), $returnedusers);

        // Only the own $USER username should be returned
        $this->assertEquals(1, count($returnedusers));

        // Call the external function.
        $returnedusers =
core_user_external::get_users_by_field('username',
        array($USER->username, $user1->username, $user2-
>username));
        $returnedusers =
external_api::clean_returnvalue(core_user_external::get_users_by_field
_returns(), $returnedusers);

        // Only the own $USER username should be returned
still.
        $this->assertEquals(1, count($returnedusers));
    } catch (SoapFault $e) {
        exit($e->faultstring);
    }
    var_dump($result);

    $return_arr['status'] = "success";
    $return_arr['message'] = $result;
    } catch(PDOException $e) {user2->username));
    $returnedusers =
external_api::clean_returnvalue(core_user_external::get_users_by_field
_returns(), $returnedusers);
    $return_arr['status'] = "error";
    $return_arr['message'] = $e->getMessage();
    }
    } else {
        $return_arr['status'] = "error";
        $return_arr['message'] = "There are should be next POST
params: lastname, firstname, email";
    }

    return $return_arr;
?>

```

search_cohorts.php

```

<?php
$functionname = 'core_cohorts_search_cohorts';

$return_arr = array();

if (isset($_POST['cohort_name']))
{
    try {

        $category = $this->getDataGenerator()-
>create_category();
    }
}

```

```

        $result
        core_cohort_external::search_cohorts($_POST['cohort_name'],
        $coursecontext, 'all');
        $this->assertEquals(1, count($result['cohorts']));
        $this->assertEquals($_POST['cohort_name'],
        $result['cohorts'][$foundcohort->id]->name);

        // Detect invalid parameter $includes.
        $this->setUser($creator);
        try {
            $result
            core_cohort_external::search_cohorts($_POST['cohort_name'],
            $syscontext, 'invalid');
            $this->fail('Invalid parameter includes');
        } catch (coding_exception $e) {
            // All good.
        }

        var_dump($result);
        $return_arr['status'] = "success";
        $return_arr['message'] = $result;

    } catch(PDOException $e) {
        $return_arr['status'] = "error";
        $return_arr['message'] = $e->getMessage();
    }
} else {
    $return_arr['status'] = "error";
    $return_arr['message'] = "There are should be next POST
params: cohort_name";
}

return $return_arr;
?>

```

update_cohorts.php

```

<?php
$functionname = 'core_cohorts_update_cohorts';

$return_arr = array();

if (isset($_POST['cohort_id']))
{
    try {

        $context = context_system::instance();
        $roleid = $this->assignUserCapability('moodle/cohort:manage', $context->id);

        $dbcohort = $DB->get_record('cohort', array('id' =>
        $_POST['cohort_id']));
        $contextid = $DB->get_field('context', 'id',
        array('instanceid' => $foundcohort['categorytype']['value'],
        'contextlevel' => CONTEXT_COURSECAT));
        $this->assertEquals($dbcohort->contextid, $contextid);
    }
}

```

```

        $this->assertEquals($dbcohort->name,
$foundcohort['name']);
        $this->assertEquals($dbcohort->idnumber,
$foundcohort['idnumber']);
        $this->assertEquals($dbcohort->description,
$foundcohort['description']);
        $this->assertEquals($dbcohort->visible, 0);
        $this->assertEmpty($dbcohort->theme);

        core_cohort_external::update_cohorts(array($foundcohort
+ array('visible' => 1)));
        $dbcohort = $DB->get_record('cohort', array('id' =>
$foundcohort['id']));
        $this->assertEquals(1, $dbcohort->visible);

core_cohort_external::update_cohorts(array($foundcohort));
        $dbcohort = $DB->get_record('cohort', array('id' =>
$foundcohort['id']));
        $this->assertEquals(1, $dbcohort->visible);

        set_config('allowcohortthemes', 1);
        core_cohort_external::update_cohorts(array($foundcohort
+ array('theme' => 'classic')));
        $dbcohort = $DB->get_record('cohort', array('id' =>
$foundcohort['id']));
        $this->assertEquals('classic', $dbcohort->theme);

        set_config('allowcohortthemes', 0);
        core_cohort_external::update_cohorts(array($foundcohort
+ array('theme' => 'boost')));
        $dbcohort = $DB->get_record('cohort', array('id' =>
$foundcohort['id']));
        $this->assertEquals('classic', $dbcohort->theme);

        $this->unassignUserCapability('moodle/cohort:manage',
$context->id, $roleid);
        $this-
>expectException(required_capability_exception::class);

core_cohort_external::update_cohorts(array($foundcohort));

        var_dump($foundcohort);
        $return_arr['status'] = "success";
        $return_arr['message'] = $foundcohort;

    } catch(PDOException $e) {
        $return_arr['status'] = "error";
        $return_arr['message'] = $e->getMessage();
    }
} else {
    $return_arr['status'] = "error";
    $return_arr['message'] = "There are should be next POST
params: cohort_id";
}

return $return_arr;
?>

```

update_users.php

```

<?php
$functionname = 'core_user_update_users';

$return_arr = array();

if (isset($_POST['email']) &&
    isset($_POST['lastname']) &&
    isset($_POST['firstname'])
)
{
    try {

        $searchparams = array(
            array('key' => 'lastname', 'value' =>
$_POST['lastname']),
            array('key' => 'email', 'value' => $_POST['email']),
            array('key' => 'firstname', 'value' =>
$_POST['firstname']));

        // Call the external function.
        $result = core_user_external::get_users($searchparams);

        // We need to execute the return values cleaning process
to simulate the web service server
        $result =
external_api::clean_returnvalue(core_user_external::get_users_returns(
), $result);

        // Check we retrieve the good total number of enrolled
users + no error on capability.
        $expectedreturnedusers = 1;
        $returnedusers = $result['users'];
        $this->assertEquals($expectedreturnedusers,
count($returnedusers));

        foreach($returnedusers as $returneduser) {
            $founduser = $returneduser;
            $roleid = $this-
>assignUserCapability('moodle/user:update', $context->id);
            $this-
>assignUserCapability('moodle/user:editprofile', $context->id,
$roleid);

            // Call the external function.
            core_user_external::update_users(array($founduser));

            $dbuser = $DB->get_record('user', array('id' =>
$founduser['id']));
            $this->assertEquals($dbuser->username,
$founduser['username']);
            $this->assertEquals($dbuser->idnumber,
$founduser['idnumber']);
            $this->assertEquals($dbuser->firstname,
$founduser['firstname']);

```

```

        $this->assertEquals($dbuser->lastname,
$founduser['lastname']);
        $this->assertEquals($dbuser->email,
$founduser['email']);
        $this->assertEquals($dbuser->description,
$founduser['description']);
        $this->assertEquals($dbuser->city,
$founduser['city']);
        $this->assertEquals($dbuser->country,
$founduser['country']);
        $this->assertEquals($dbuser->department,
$founduser['department']);
        $this->assertEquals($dbuser->institution,
$founduser['institution']);
        $this->assertEquals($dbuser->phone1,
$founduser['phone1']);
        $this->assertEquals($dbuser->maildisplay,
$founduser['maildisplay']);
        $this->assertEquals('atto',
get_user_preferences('htmleditor', null, $dbuser));
        $this->assertEquals(null,
get_user_preferences('invalidpreference', null, $dbuser));

        // Confirm user interests have been saved.
        $interests =
core_tag_tag::get_item_tags_array('core', 'user', $founduser['id'],
core_tag_tag::BOTH_STANDARD_AND_NOT, 0, false);
        // There should be 3 user interests.
        $this->assertCount(3, $interests);

        // Confirm no picture change when parameter is not
        supplied.

        unset($founduser['userpicture']);
        core_user_external::update_users(array($founduser));
        $dbusernopic = $DB->get_record('user', array('id' =>
$founduser['id']));
        $this->assertEquals($dbuser->picture, $dbusernopic-
>picture, 'Picture not change without the parameter.');
```

```

        // Confirm delete of picture deletes the picture
        from the user record.
        $founduser['userpicture'] = 0;
        core_user_external::update_users(array($founduser));
        $dbuserdelpic = $DB->get_record('user', array('id'
=> $founduser['id']));
        $this->assertEquals(0, $dbuserdelpic->picture,
'Picture must be deleted when sent as 0.');
```

```

        // Call without required capability.
        $this->unassignUserCapability('moodle/user:update',
$context->id, $roleid);
        $this-
>expectException('required_capability_exception');
        core_user_external::update_users(array($founduser));
    }
    var_dump($result);
    $return_arr['status'] = "success";

```

```

        $return_arr['message'] = $result;

        } catch(PDOException $e) {
            $return_arr['status'] = "error";
            $return_arr['message'] = $e->getMessage();
        }
    } else {
        $return_arr['status'] = "error";
        $return_arr['message'] = "There are should be next POST
params: lastname, firstname, email";
    }

    return $return_arr;
?>

```

test_add_user.php

```

<!DOCTYPE html>
<html><head><meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0, maximum-scale=1.0, user-scalable=no">
    <link href="./index_files/main.css" rel="stylesheet">
    <link rel="stylesheet" href="./index_files/font-
awesome.min.css">
    <!-- theme bootstrap stylesheets -->
    <link href="./index_files/bootstrap.css" rel="stylesheet">

</head>

<body style="">
<header>
<nav class="header-top-all header-inner" role="navigation"
style="position:relative;">
    <div class="container">
        <div class="navbar-header">
            <a href="http://pkentry.pp.ua/index.php">
                
            </a>
        </div>
        <div id="navbar" class="navbar-collapse text-center blue">
            <h1> Тестуємо інтеграційну систему до СНД Moodle Запорізької
політехніки </h1>
        </div>
    </div>
</nav>
</header>
<section id="wrapper" class="container">
    <section id="content" class="content">
        <!-- app body -->

        <section id="content-main" class="content-main">
            <div class="container">
                <div class="row">
                    <div class="app-body col-md-12">

```

```

        <form class="form-horizontal" role="form" action="user_add.php"
method="post"          id="vendorForm"          enctype="multipart/form-data"
novalidate="novalidate">
        <div class="well bs-component row">
        <legend>
        <h3 class="text-center blue">Реєстрація акаунту у системі
дистанційної освіти Moodle</h3>
        </legend>
        <fieldset>

                <div class="form-group row">
                <label class="col-lg-4 control-label"
for="username">Лорін</label>
                <div class="col-lg-8">
                <input type="text" class="form-control typeahead top10 uniform-
input text" id="username" placeholder="Лорін" name="username">
                </div>
                </div>
                <div class="form-group row">
                <label class="col-lg-4 control-label"
for="password">Пароль</label>
                <div class="col-lg-8">
                <input type="text" class="form-control typeahead top10 uniform-
input text" id="password" placeholder="Пароль" name="password">
                </div>
                </div>

                <div class="form-group row">
                <label class="col-lg-4 control-label" for="lastname">Прізвище
(українською мовою за паспортом)</label>
                <div class="col-lg-8">
                <input type="text" class="form-control typeahead top10 uniform-
input text" id="lastname" placeholder="Прізвище" name="lastname">
                </div>
                </div>
                <div class="form-group row">
                <label class="col-lg-4 control-label" for="firstname">Ім'я
(українською мовою за паспортом)</label>
                <div class="col-lg-8">
                <input type="text" class="form-control typeahead top10 uniform-
input text" id="firstname" placeholder="Ім'я" name="firstname">
                </div>
                </div>

                <div class="form-group row">
                <label class="col-lg-4 control-label" for="midname">По батькові
(українською мовою за паспортом)</label>
                <div class="col-lg-8">
                <input type="text" class="form-control typeahead top10 uniform-
input text" id="midname" placeholder="По батькові" name="midname">
                </div>
                </div>

                <div class="form-group row">
                <label class="col-lg-4 control-label " for="email">Пошта</label>
                <div class="col-lg-8">

```

```

<input type="text" class="form-control typeahead uniform-input
text" id="email" placeholder="Почта" name="email">
</div>
</div>

<div class="form-group row">
<label class="col-lg-4 control-label"
for="fac">Факультет:</label>
<div class="col-lg-8">
<select class="form-control valid" id="fac" name="fac">
<option value="ФКХТ">ФКХТ</option>
</select>
</div>
</div>
<div class="form-group row">
<label class="col-lg-4 control-label"
for="groupname">Група</label>
<div class="col-lg-8">
<input type="text" class="form-control typeahead uniform-input
text" id="groupname" placeholder="Група" name="groupname">
</div>
</div>

<div class="form-group row">
<div class="col-lg-offset-2 col-lg-10">
<button type="button" class="btn btn-primary pull-right"
id="submitBtn" onclick="return submitVendorForm();"
name="updateBtn">Запеструватися</button>
</div>
</div><!-- /form-group -->
</fieldset><!-- /fieldset -->
</div><!-- /box -->
</form>
</div><!-- /app body -->
</div><!-- /content app -->
<div class="row">
<div class="col-md-12">
</div>
</div>
</div><!-- /content app -->

</section><!-- /content main -->
</section><!-- /content -->
</section><!-- /wrapper -->
<footer>
<div class="container">
<div class="row">
<div class="col-md-6">
<h5 class="top10" style="text-align:justify;">
</h5>
</div>
<div class="col-md-6 text-center top20">
© Copyright 2020
</div>
</div>
</div>
</footer>

```

```

<!-- jQuery, theme required for theme -->
<script src="./index_files/jquery.js "></script>
<script src="./index_files/bootstrap.min.js "></script>

<script type="text/javascript" src="./index_files/jquery.min.js
"></script>
<script type="text/javascript"
src="./index_files/jquery.validate.min.js "></script>
<script type="text/javascript"
src="./index_files/jquery.uniform.min.js"></script>

<script type="text/javascript" src="./index_files/bootstrap-
multiselect.js "></script>
<link rel="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-
multiselect/0.9.13/css/bootstrap-multiselect.css" type="text/css">

<script type="text/javascript">

    function submitVendorForm()
    {
        $("form#vendorForm").submit();
        return true;
    }
</script>

<script type="text/javascript">
    // document ready function
    $(document).ready(function() {

        //fixing style for multiselect
        $('.btn-group').css('width','100%');

        jQuery.extend(jQuery.validator.messages, {
            accept: "Будь ласка, введіть коректну пошту"
        });

        $("input, textarea, select").not('.nostyle').uniform();
        $("#vendorForm").validate({
            rules: {
                lastname: {
                    required: true,
                    minlength: 2
                },
                firstname: {
                    required: true,
                    minlength: 2
                },
                email: {
                    required: true,
                    email: true
                }
            },
            messages: {
                lastname: {
                    required: "Будь ласка, введіть своє
    прізвище",

```

```

                                minlength: "Будь ласка, введіть своє
прізвище"
                                },
                                firstname: {
                                    required: "Будь ласка, введіть своє ім'я",
                                    minlength: "Будь ласка, введіть своє ім'я"
                                },
                                email: {
                                    required: "Будь ласка, введіть свою пошту",
                                    email: "Будь ласка, введіть коректну поштову
адресу"
                                }
                            }
                        });
                    });
</script>

<div id="wt-sky-root"></div></body></html>

```

ДОДАТОК Б
Слайди презентації



Рисунок Б.1 – Слайд 1

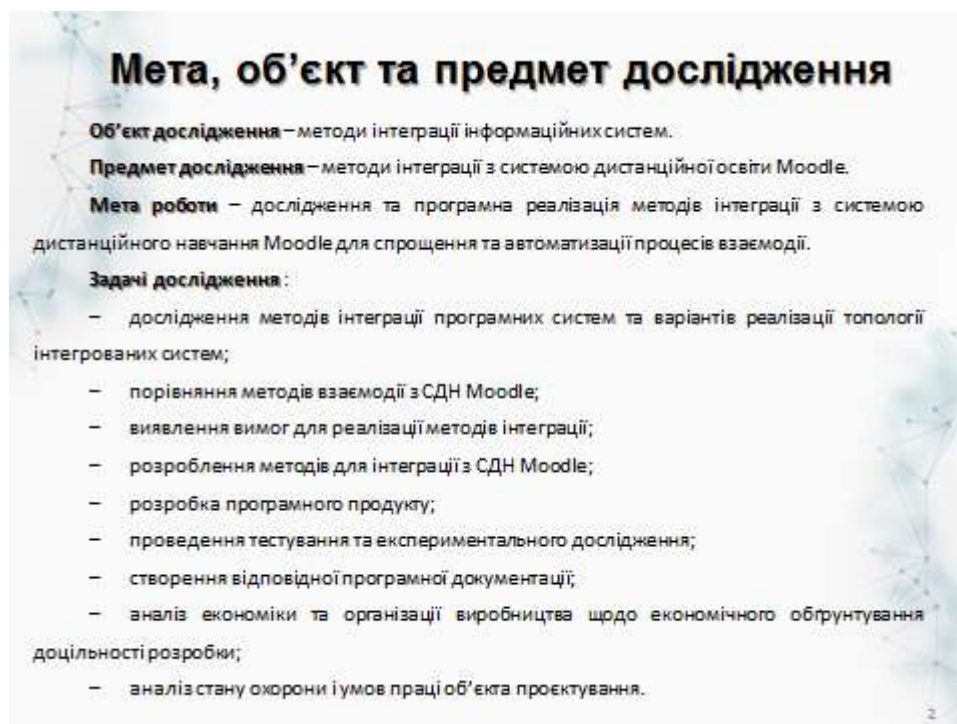


Рисунок Б.2 – Слайд 2

Аналіз предметної області

Середні і великі університети зазвичай експлуатують як мінімум десяток багатокористувацьких систем. У цих системах часто обробляються однакові дані, починаючи з довідників і класифікаторів.

Наслідками відсутності належного вирішення проблеми інтеграції є:

- повторне ручне введення даних (довідники, дані про студентів, викладачів, фінансові транзакції і т.п.);
- багаторазові і нескінченні «звірки та коригування», які не виключають помилок;
- непомірні витрати на формування зведеної звітності;
- завищені терміни і собівартість виконання навіть звичайних завдань.

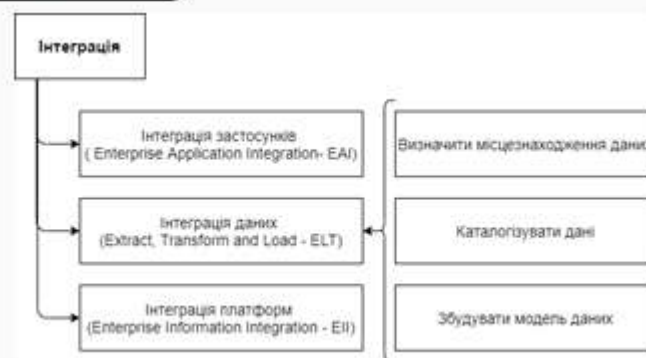
Перераховане вище визначає необхідність інтеграції застосунків університету, зокрема СДН Moodle та системи з обліку контингенту, які містять загальну інформацію про студентів та їх групи.

3

Рисунок Б.3 – Слайд 3

Основні методи інтеграції програмних систем

- 1 Обмін файлами
- 2 Спільна база даних
- 3 Віддалений виклик
- 4 Асинхронний обмін повідомленнями
- 5 Інтеграції через сервіси API



4

Рисунок Б.4 – Слайд 4

Існуючі методи інтеграції з СДН Moodle				
	Витрати часу на дію	Вірогідність помилки	Універсальність	Необхідність додаткових витрат
Ручне введення даних	Високі	Висока	Універсальний	Немає
Імпорт даних з CSV-файлу	Високі	Середня	Обмежений	Немає
Використання інтеграційних плагінів	Малі	Маленька	Обмежений	В залежності від вартості плагіну
Використання міжнародного стандарту SCORM	Малі	Маленька	Універсальний (для обміну матеріалами курсів)	В залежності від засобів, що використовуються
Використання API веб-служб Moodle	Малі	Маленька	Універсальний	В залежності від засобів, що використовуються

Рисунок Б.5 – Слайд 5

Аналіз і специфікація вимог

Функціональні вимоги:

- поновлення студента (додавання нового користувача або активація існуючого акаунта);
- відрядження студента (деактивація користувача);
- видалення користувача;
- прикріплення студента до групи;
- зміна групи у студента;
- сповіщення користувача через email про зміни у його акаунті;
- видалення груп студентів.

Вимоги до інтерфейсу

Інтерфейс програмного продукту повинен відповідати ергономічним засадам для зручної організації процесу інтеграції з іншими програмними системами. Він повинен містити правила введення даних та мати повідомлення у разі не отримання некоректних чи неповних даних, відсутності зв'язка з сервером тощо.

Графічний інтерфейс для розроблюваної системи не потрібен.

Системні вимоги

Використання мови PHP для розроблення скриптів для обробки вхідних запитів та викликів методів API, що надає Moodle. Використання бази даних для інтеграційної системи не планується, тому що усі виклики мають оброблюватися одразу після отримання, у зберіганні проміжної інформації у базі даних немає потреби.

Рисунок Б.6 – Слайд 6

Реалізація протоколів для взаємодії з СДН Moodle

Назва протоколу	Доступно у версії Moodle	Опис	CORS
REST (повертає XML)	2.0	Дані XML, доступні через REST API (не restful)	Підтримується (3.0 і вище)
REST (повертає JSON)	2.2	Дані JSON, доступні через REST API (не restful)	Підтримується (3.0 і вище)
SOAP	2.0	Дуже важливий для взаємодії з програмами Java та .Net. Часткова реалізація була виконана в Moodle 2.0, у повній мірі у Moodle 3.0. Він повністю працює з клієнтами PHP	Не підтримується
XML-RPC	2.0	Виклик віддаленої процедури XML	Не підтримується

Рисунок Б.7 – Слайд 7

Аналіз відповідності протоколів задачі взаємодії з СДН Moodle

	XML-RPC	SOAP	REST
Простота реалізації	+	-	+
Швидкість передачі простих структур	+	-	+
Обмін складними структурами даних	-	+	-
Формалізація взаємодії клієнта та сервера	-	+	-
Підтримка безпеки та авторизації	Покладається на розробника	Вбудована функція протоколу	Покладається на розробника
Можливість кешування відповідей та різні формати відповідей на запити	-	+	+
Багаторазові транзакції	-	+	-

Рисунок Б.8 – Слайд 8



Рисунок Б.9 – Слайд 9

Основні API -функції Moodle, обрані для інтеграції

Ім'я	Вер.	Опис	Ім'я	Вер.	Опис
auth_email_get_signup_settings	3.2	Отримання необхідних налаштувань для реєстрації та поля профілю	core_course_get_course_module	3.0	Повернення інформації про модуль курсу.
auth_email_signup_user	3.2	Додавання нового користувача	core_course_get_courses_by_field	3.2	Отримання курсів, що відповідають певному полю
core_auth_confirm_user	3.2	Підтвердження облікового запису користувача	core_course_import_course	2.4	Імпорт даних курсу в інший курс
core_auth_request_password_reset	3.4	Запит на скидання пароля	core_course_search_courses	3.0	Пошук курсів за ім'ям, модулем, блоком, темою
core_auth_request_password_reset	3.4	Запит на скидання пароля	core_enrol_get_enroll_users	2.1	Отримання зареєстрованих користувачів за ідентифікатором курсу
core_cohort_add_cohort_members	2.4	Додавання учасників до когорти	core_message_send_instant_messages	2.1	Відправлення миттєвого повідомлення
core_cohort_create_cohorts	2.4	Створення когорти	core_role_assign_roles	2.0	Розподіл ролей
core_cohort_delete_cohort_members	2.4	Видалення учасників з когорти	core_role_unassign_roles	2.0	Видалення ролі у користувача
core_cohort_delete_cohorts	2.4	Видалення когорти	core_user_create_users	2.0	Створення користувачів
core_cohort_get_cohort_members	2.4	Отримання членів когорти	core_user_delete_users	2.0	Видалення користувачів
core_cohort_get_cohorts	2.4	Отримання когорти	core_user_get_users	2.5	Пошук користувачів
core_cohort_search_cohorts	3.0	Пошук когорти	core_user_get_users_by_field	2.5	Отримання інформації про користувачів для вказаного унікального поля
core_cohort_update_cohorts	2.4	Оновлення когорти	core_user_get_users_by_id	2.0	Отримання користувача за ідентифікатором
core_course_create_categories	2.3	Створення категорії	core_user_update_users	2.0	Оновлення користувачів
core_course_create_courses	2.0	Створення нового курсу	enrol_manual_enrol_users	2.0	Реєстрація користувачів вручну
core_course_delete_categories	2.3	Видалення категорії	enrol_manual_unenrol_users	3.0	Скасування реєстрацію користувачів вручну
core_course_delete_courses	2.3	Видалення курсів	gradereport_overview_get_course_grades	3.2	Отримання підсумкових оцінок даних курсів
core_course_delete_modules	2.5	Видалення екземплярів модуля	tool_lp_search_cohorts	3.1	Пошук когорти
core_course_duplicate_course	2.3	Дублювання курсу	tool_lp_search_users	3.1	Пошук користувачів
core_course_edit_module	3.3	Дія з модулем курсу			
core_course_edit_section	3.3	Дія в розділі курсу			
core_course_get_categories	2.3	Отримання категорії			
core_course_get_contents	2.2	Отримання вмісту курсу			

Рисунок Б.10 – Слайд 10

Порядок дій для підключення інтеграційної системи

Ваш сервіс / Інформація про сервіс / Зовнішній сервіс

Зовнішній сервіс

Назва: my_app_service

Джерело коду:

Тип зовнішнього сервісу: Додати новий

Пов: 1024 MB

Порт: 8080

URL: http://localhost:8080

API ключ:

Додати новий

Створення нового зовнішнього сервісу

Рисунок 3 – Створення зовнішнього сервісу

Ваш сервіс / Інформація про сервіс / Функції / Додати функцію

Додати функцію

Назва: my_app_service

Функція: my_app_service

Джерело коду: my_app_service

Тип зовнішнього сервісу: Додати новий

Пов: 1024 MB

Порт: 8080

URL: http://localhost:8080

API ключ:

Додати новий

Створення нової функції

Рисунок 4 – Надання доступу до вебслужби

Додати функцію до сервісу 'my_app_service'

Функція	Назва	Пов'язаний сервіс	Статус
my_app_service_my_app_service	my_app_service	my_app_service	Активний
my_app_service_my_app_service	my_app_service	my_app_service	Активний

Рисунок 5 – Додавання функцій до вебслужби

Рисунок Б.14 – Слайд 14

Тестовий вебінтерфейс для виклику методу інтеграційної системи «додавання нового користувача»

Ресурси акаунту у системі дистанційної освіти Moodle

Логін	
Пароль	
Прізвище (українською мовою та латиницею)	
Ім'я (українською мовою та латиницею)	
По батькові (українською мовою та латиницею)	
Мобільний номер (у форматі +380XXXXXXXX)	
Пошта	
Кафедра	
Група	

15

Рисунок Б.15 – Слайд 15

Приклад інтерфейсу системи з обліку контингенту для активації (створення) або деактивації акаунту студента

Профіль Moodle

Базис:

Прізвище:

Ім'я:

По батькові:

Група:

Місто:

Навчальний заклад:

Логін:

Пароль:

Прізвище:

Ім'я:

По батькові:

Група:

Пошта:

Активність: ☒

Закінка:

Профіль Moodle

Базис:

Прізвище:

Ім'я:

По батькові:

Група:

Місто:

Навчальний заклад:

Логін:

Пароль:

Прізвище:

Ім'я:

По батькові:

Група:

Пошта:

Активність: ☒

Закінка:

16

Рисунок Б.16 – Слайд 16

Основні помилки під час інтеграції з СДН Moodle та дії для їх подолання	
Помилка	Опис дій для її подолання
не вдається аутентифікувати користувача	необхідно перейти до адміністративної частини Moodle за шляхом «Керування сайтом» > Модуль Веб-служби > Керування службами > Уповноважені користувачі», натиснути на повне ім'я користувача та відредагувати його дані
термін дії користувача закінчився	необхідно перейти до адміністративної частини Moodle за шляхом «Керування сайтом» > Модуль Веб-служби > Керування токенами» та перевірити дані токена
не вдається аутентифікувати користувача за токеном	необхідно перейти до адміністративної частини Moodle за шляхом «Керування сайтом» > Модуль Веб-служби > Керування службами > Уповноважені користувачі» та надати користувачу доступ
токену користувача не існує	необхідно перейти до адміністративної частини Moodle за шляхом «Керування сайтом» > Модуль Веб-служби > Керування службами > Уповноважені користувачі» та надати користувачу доступ
термін дії токена закінчився	необхідно перейти до адміністративної частини Moodle за шляхом «Керування сайтом» > Модуль Веб-служби > Керування службами > Уповноважені користувачі» та надати користувачу доступ
користувача немає в списку авторизованих користувачів	необхідно перейти до адміністративної частини Moodle за шляхом «Керування сайтом» > Модуль Веб-служби > Керування службами > Уповноважені користувачі» та надати користувачу доступ
користувач не має необхідних можливостей	необхідно перейти до адміністративної частини Moodle за шляхом «Керування сайтом» > Модуль Веб-служби > Керування службами > Уповноважені користувачі» та надати користувачу необхідні можливості
функція вебслужби не включена до послуги	перевірити цей факт у налаштуваннях Moodle
сайт в режимі обслуговування	необхідно перейти до адміністративної частини Moodle за шляхом «Керування сайтом» > Розширені функції» та увімкнути вебслужбу
вебслужба викинута	надати доступ до протоколу (користувач повинен мати можливість 'webservice/usexmlrpc/'webservice/use:rest/'... в контексті системи)
користувач не має можливості використовувати використаний протокол вебсервісу	переконатися, що протокол REST та SOAP увімкнено («Керування сайтом» Модуль Веб-служби > Керування протоколами)
протокол викинута	значення, які передаються, не відповідають стандарту типу параметра функції веб-служби (необхідно переконатися, що служба передає правильні значення, детально переглянувши параметри)
недійсний параметр	необхідно переконатися, що у користувача є доступ для запуску веб-служби (тому що вебслужба створена для авторизованих користувачів)
недійсний доступ	необхідно переконатися, що у користувача є доступ для запуску веб-служби (тому що вебслужба створена для авторизованих користувачів)
не проходження всіх обов'язкових полів	необов'язкові параметри можуть ігноруватися під час передачі, але обов'язкові поля повинні бути передані

Рисунок Б.17 – Слайд 17

ВИСНОВКИ

В роботі було проведено дослідження та програмна реалізація методів інтеграції з системою дистанційного навчання Moodle для спрощення та автоматизації процесів взаємодії. Для досягнення цієї мети було досліджено методи інтеграції програмних систем та варіанти реалізації топології інтегрованих систем, виконано порівняння методів взаємодії з СДН Moodle, виявлено вимоги для реалізації методів інтеграції та на їх основі розроблено методи для інтеграції СДН Moodle з іншими інформаційними системами ЗВО.

Для розроблення інтеграційної системи були розглянуті протоколи XML-RPC, REST та SOAP, які API Moodle пропонує для додавання нових функцій до вебслужби та для зв'язку з іншими системами, було обрано для реалізації інтеграційної системи протокол передачі даних SOAP.

Практична цінність розробленої інтеграційної системи полягає в тому, що впровадження в експлуатацію розробленого ПЗ в закладі вищої освіти дозволить підвищити швидкість оброблення заявок до технічної підтримки, зменшити час працівників деканатів та навчального відділу на «ручну» (через вебінтерфейс) реєстрацію акаунтів студентів, актуалізацію інформації про них, прикріплення / зміну / видалення студентів з групи, тощо.

Наукова новизна роботи полягає в розробленій моделі взаємодії інтеграційної системи з іншими інформаційними системами в інформаційній інфраструктурі закладу вищої освіти, розроблених алгоритмів взаємодії з API - функціями СДН Moodle.

Рисунок Б.18 – Слайд 18