Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

СТВОРЕННЯ ПРЕДМЕТНО-ОРІЄНТОВАНИХ КОМПОНЕНТІВ СИСТЕМ ЕЛЕКСТРОННОГО НАВЧАННЯ

Текстова частина до курсової роботи
за спеціальністю „Інженерія програмного забезпечення”

Керівник курсової роботи
к.ф-м.н., доц. Бублик В. В.

_____
(підпис)
“____” _____ 2021 р.

Виконав студент Лелюк О.О.
“____” _____ 2021 р.

Київ 2021

Зміст

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри мультимедійних систем,

доцент, к.ф.-м.н.

_____ О. П. Жежерун

(підпис)

“____”_____ 2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Лелюку Олекснадру

4-го курсу факультету інформатики

ТЕМА: Створення предметно-орієнтованих компонентів систем електронного навчання

Вихідні дані:

- Плагін для системи контролю навчання Moodle

Зміст ТЧ до курсової роботи:

Вступ
Анотація
1. Аналіз предметної області та наявних проблем і задач
2. Теоретичні засади розробки компонент системи керування навчанням “Moodle”
3. Реалізація практичної частини роботи
Висновки
Список джерел
Додатки (за необхідністю)

Дата видачі “___” _____ 2021 р.

Керівник _____ Завдання отримано _____

Календарний план виконання курсової роботи

Тема: Створення предметно-орієнтованих компонентів систем електронного навчання

№ п/п	Назва етапу курсового проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	січень 2021 р.	
2.	Огляд предметної області та літератури за темою роботи	лютий-березень 2021 р.	
3.	Огляд технологій потрібних для реалізації практичної частини	лютий-березень 2021 р.	
4.	Створення практичної частини роботи	березень-квітень 2021 р.	
5.	Написання пояснювальної роботи	березень-квітень 2021 р.	
6.	Створення слайдів для доповіді та написання доповіді	квітень 2021 р.	
7.	Надання роботи керівнику для перевірки	квітень 2021 р.	
8.	Корегування роботи за результатами перевірки керівником	квітень 2021 р.	
9.	Остаточне оформлення пояснювальної роботи та слайдів	квітень 2021 р.	
10.	Захист курсової роботи	квітень 2021 р.	

Студент Лелюк О.О. _____

Керівник Бублик В.В. _____

“_____” _____ _____ р.

Вступ

Завдяки швидкому прогресу в галузі електронних технологій та розвитку навчальних систем таке поняття як електронне навчання набуло великої популярності. Стандартизація освітнього процесу, покращена комунікація та обмін досвідом, масштабованість, розширюваність, ефективність – одні з головних переваг, що надає електронне навчання студентам та викладачам. Завдяки цим перевагам електронне навчання довело та продовжує доводити свою ефективність не тільки в навчальних закладах, а й в корпоративному секторі.

Усі ці переваги досягаються завдяки концепції “система управління навчання”, яка є головною частиною в досягненні ефективної інтеграції електронного навчання в освітній процес. Реалізація цієї концепції дає змогу повністю контролювати освітній процес та в перспективі має повністю задовольняти потреби викладачів та студентів.

На жаль, у більшості випадків, реалізації систем управління навчання не можуть задовільнити потреби всіх учасників освітнього процесу. Через це власники цих реалізацій надають функціонал, завдяки якому можна розширити систему електронного навчання таким чином, щоб задовільнити ці потреби. Ця робота присвячена системам управління електронним навчанням та, зокрема, проблемі їх розширюваності.

Анотація

Роботу присвячено дослідженню систем контролю електронного навчання та, зокрема, їх підходу до розширюваності. На основі результатів дослідження було розроблено плагін для системи “Moodle”, який демонструє процес реалізації та інтеграції нового функціоналу в систему відповідно до нових потреб користувачів.

1. Аналіз предметної області та наявних проблем і задач

1.1 Електронне навчання

Впродовж всього існування людства однією з головних потреб людини було здобуття нових знань та навичок. Головними характеристиками є ефективність здобуття та якість знань. Задля їх покращення людство постійно розробляє нові підходи до навчання. Одним з таких є електронне навчання, який згодом довів свою ефективність практично.

Електронне навчання – це такий тип навчання, який базується на формалізованому викладанні за допомогою електронних ресурсів.

Виходячи з цього визначення, електронним навчанням є будь-яке навчання, в якому використовуються електронні ресурси і не зрозуміло, яким чином використання електронних ресурсів покращить якість навчання. Справді, використання електронних ресурсів саме по собі сильно не впливає на ефективність навчання. Значний ефект можна побачити тільки тоді, коли в процесі навчання дотримуються певних принципів, що описані в теорії електронного навчання.

Теорія електронного навчання – концепція, яка описує когнітивні наукові принципи ефективного мультимедійного навчання. Ця теорія формує набір таких принципів:

- Мультимедійність – поєднання тексту та відповідних візуальних елементів.
- Модальність – подача матеріалу не у вигляді простого тексту, а у формі графічної інформації у супроводі з звуковим поясненням.
- Узгодженість – подача матеріалів синхронізовано з графічним супроводом.
- Суміжність – логічно пов’язана інформація знаходиться поруч.
- Сегментація – вміст розбивається на невеликі частини.
- Значущість – використання зорових, слухових та інших сигналів, які мають привернути уваги до найбільш значущих моментів.

- Контроль – контроль швидкості опановування знань учнями.
- Персоналізація – соціальна присутність у процесі навчання.
- Попередня підготовка: поняття, процеси та інші об’єкти, що згадуються під час навчання, супроводжуються визначеннями або поясненнями.
- Надлишковість – графічні елементи пояснюються лише голосом, а не голосом та текстом на екрані.
- Спеціальні знання – надлишкове використання вище вказаних принципів, може призвести до покращення ефективності навчання учнів з низьким рівнем знань, але не вплинути або навіть знизити ефективність навчання учнів з високим рівнем знань.

Варто зазначити, що через складність дослідження досі не вдалося визначити точний вплив цих принципів на ефективність навчання поза рамками лабораторних умов.

У будь-якому разі в процесі інтеграції ефективного електронного навчання в освітній процес дотримання цих принципів не є важливим. Важливо, щоб після самого процесу інтеграції була створена можливість дотримуватись цих принципів.

1.2 Система управління навчанням

Під час інтеграції ефективного електронного навчання в навчальний процес виникає багато проблем. Наприклад, потрібно адмініструвати процес навчання, реалізовувати проведення навчальних програм, курсів і інших аналогічних задач. Задля цього можна, наприклад, створити застосунки для вирішення кожної окремої задачі, або ж створити один застосунок, який буде одразу вирішувати усі задачі. Не залежно від обраної архітектури цей застосунок буде являти собою систему управління навчанням.

Система управління навчанням – це програмне забезпечення, яке керує усіма аспектами навчального процесу, що охоплює електронне навчання. Прикладами таких аспектів є адміністрування, звітність, автоматизація, проведення навчальних курсів, програм навчання та розвитку.

Насправді, чітких відповідальностей у систем управління навчанням не існує. Вони залежать від специфіки навчального закладу, його потреб та об’єму навчального процесу, який має охоплювати електронне навчання. У цій роботі наводяться три головних відповідальності, які, в більшості випадків, потребує будь-який навчальний заклад:

- Можливість управління курсами, користувачами та їх ролями.
- Оцінювання та відстеження відвідуваності студентів.
- Отримування відгуків як від студентів, так і від викладачів.

У більшості випадків системи управління навчанням реалізуються у вигляді веб-застосунків. Таким чином усі учасники освітнього процесу зможуть легко працювати з системою без встановлення додаткового програмного забезпечення з будь-якої зручної для них платформи.

Система управління електронного навчання може бути реалізована самим навчальним закладом або може бути використана вже існуюча. У цій роботі розглядається саме другий варіант.

У випадку другого варіанту інтеграція системи електронного навчання в навчальний процес складається з таких кроків:

- Збір даних від фокус груп, які прийматимуть участь у електронному навчанні.
- Опрацювання зібраних даних.
- Вибір відповідної системи контролю навчання та початок інтеграції електронного навчання в навчальний процес за допомогою обраної системи.
- Підтримка та адаптація системи під потреби учасників освітнього процесу.

До того ж, останній етап може не завершуватися протягом усього використання обраної системи, оскільки освітній процес із плином часу може змінюватись, а відповідно вимагати модифікацію системи контролю навчання.

1.3 Види систем контролю навчання

Системи контролю навчання не поділяють на види залежно від функціоналу, які вони надають, оскільки, як вже було описано вище, ці системи не мають чітко визначених відповідальностей. Однак, деякі джерела виділяють термін “система управління змістом навчання”.

Система управління змістом навчання (СУЗН) – це програмне забезпечення, яке керує усіма аспектами навчального процесу та надає інструменти для створення змісту, який буде використовуватись у електронному навчанні. З цього визначення можна зробити висновок, що СУЗН являє собою систему контролю навчання у більш конкретизованому вигляді відносно роботи з інформацією, яка використовуватиметься у електронному навчанні. Назва цього терміну є не дуже вдалою, бо може здатися, що система контролю навчання та СУЗН мають різні області відповідальностей, які зовсім не перетинаються. Назва “система управління навчанням та змістом” більш вдало описувала б такий тип програмного забезпечення.

1.4 Проблеми розробки систем контролю навчання

Для того, щоб визначити, які характеристики повинні мати системи контролю навчання, спочатку потрібно обрати стандарт, який описує якість програмного забезпечення. У цій роботі буде використаний стандарт ISO/IEC 9126.

Обраний стандарт визначає такі якісні характеристики програмного забезпечення:

- Функціональність.
- Надійність.
- Зручність використання.
- Продуктивність.
- Можливість підтримки.
- Портативність.

Варто зазначити, що реалізація системи контролю навчання прямо залежить від потреб навчального закладу та специфіки освітнього процесу, тому не можливо визначити, які саме характеристики будуть найбільш важливими для кожного окремого навчального закладу.

Саме характеристики, які обрані найбільш важливими, визначають проблеми, з якими зіштовхнуться розробники системи контролю навчання. Наприклад, якщо були обрані такі характеристики як функціональність, зручність використання та можливість підтримки, то найголовнішими проблемами, які постануть перед розробниками будуть:

- Відповідність реалізованого функціоналу специфіці навчального процесу та заявленим відповідальностям системи.
- Захищеність даних, якими оперує система.
- Зрозумілість.
- Легкість опанування системи.
- Стабільність системи.
- Можливість змінювати систему під нові потреби та відповідальності.

Варто додати, що реалізована система контролю навчанням відіграє роль підтримки у навчальному процесі. Вона не має обмежувати ні викладачів ні студентів. В іншому випадку, використання такої системи не покращить ефективність навчального процесу і може навіть знизити її.

1.5 Проблеми підтримки систем контролю навчання

Навчальний процес може змінюватись. Наприклад:

- Викладач після підвищення своєї кваліфікації хоче змінити формат свого курсу.
- Студенти хочуть зробити навчальний процес більш різноманітним, що потребує введення нових типів активностей у рамках освітнього процесу.
- Навчальний заклад вирішує збільшити покриття освітнього процесу форматом електронного навчання.

Можливість адаптуватись під нові потреби та підтримувати свою актуальність для навчального процесу є дуже важливою задачею для систем контролю навчання. Нездатність системи вирішити такі проблеми може призвести до відмови від неї, оскільки замість підтримки освітнього процесу така система починає його обмежувати.

Існують такі підходи до вирішення зазначеної проблеми:

- Відкриття вихідного коду системи. Гнучкість можливих змін прямо залежить від гнучкості системи. Цей підхід потребує глибоких знань внутрішньої роботи системи та може бути небезпечним. Наприклад, через недостатню кваліфікацію можуть бути утворені вразивості в безпеці. До того ж, процес оновлення системи може значно ускладнитися.
- Надання функціоналу для створення розширень системи. Цей підхід потребує помірних знань внутрішньої роботи системи, є досить безпечним та може ніяк не впливати на процес оновлення системи. Гнучкість можливих змін системи прямо залежить від розробників системи.

- Надання відкритого API[1]. Із використанням відкритого API можна створювати застосунки, які розширюватимуть функціонал системи і в той самий час будуть синхронізовані з існуючою системою. Цей підхід не потребує знань внутрішньої роботи системи та є найбільш безпечним підходом, бо система не модифікується зсередини. Гнучкість можливих змін прямо залежить від гнучкості API.
- Виділення спеціальної команди розробників, яка надаватиме послуги для розширення системи. Гнучкість можливих змін прямо залежить від гнучкості створеної системи. Цей підхід є найбільш надійним, але потребує економічних витрат зі сторони власників системи або навчального закладу.

Варто зазначити, що в цій роботі зазначені лише найбільш поширені способи вирішення визначеної проблеми. Усі вище зазначені підходи можуть використовуватись окремо або спільно, щоб нівелювати недоліки один одного.

[1] API (application programming interface) – набір операцій, за допомогою яких можна обмінюватись інформацією з програмою.

2. Засоби розробки компонент системи керування навчанням “Moodle”

2.1 Анотація

Як було зазначено у розділі 1.2, через специфіку предметної області не існує єдиної архітектури для систем контролю навчання. Через цю обставину метою цього розділу не може бути опис засобів розробки компонент для будь-якої системи керування навчанням.

Натомість, цей розділ описує засоби розробки компонент для однієї з найрозповсюдженіших систем контролю навчання під назвою “Moodle”.

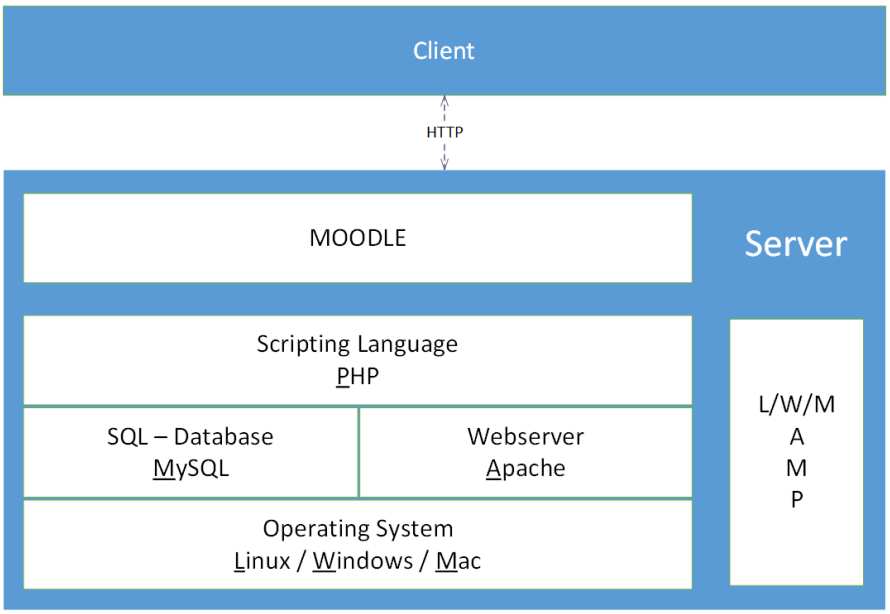
Moodle – це безкоштовна система керування навчанням з відкритим кодом, що написана мовою PHP і розповсюджується під загальною публічною ліцензією. Moodle надає функціонал, якого достатньо середньостатистичному навчальному закладу, а також надає багато можливостей для модифікації системи під специфіку окремих навчальних закладів.

2.2 Архітектура системи

Перед тим, як досліджувати процес розробки компонент для системи контролю навчання, варто дослідити її архітектуру.

Розгорнута інсталяція Moodle складається з таких частин:

- PHP-код, що виконується на веб-сервері, що підтримує PHP.
- База даних, керована реляційною системою управління базою даних.
- Сховище для завантажених та створених файлів.



Зображення 1 Діаграма архітектури інсталяції Moodle

Усі ці частини можуть працювати на більш ніж одному сервері. Виконуваний код може бути розділений між декількома веб-серверами, база даних – між кластером баз даних, сховище даних може знаходитись на окремому файловому сервері.

2.2.1 Організація коду

Moodle проектувався як модульна система, тому структурно Moodle складається з основного модулю, який надає API для роботи з системою, та набору модулів, які, використовуючи основний модуль, надають певний функціонал.

Завдяки такому підходу Moodle має високу здатність до розширення та налаштування, не вдаючись при цьому до модифікації коду в основному або інших модулях.

Модулі в Moodle діляться на типи відповідно до свого функціоналу. Більш детальна інформація про типи модулів знаходиться в розділі 2.5. Фізично ж, модуль являє собою папку, в якій знаходяться PHP, JavaScript, CSS та інші файли за потреби. Основний модуль взаємодіє з іншими модулями, використовуючи точки входу, що визначені в них. Наприклад, деякі точки входу можуть бути визначені у файлі lib.php.

2.2.2 Організація бази даних

База даних Moodle може містити багато таблиць, бо вся база даних є сукупністю таблиць, що відносяться до кожного модулю. Структура бази даних Moodle визначається у файлах install.xml всередині папки під назвою db кожного модулю.

2.3 Середовище розробки

Moodle являє собою веб-застосунок. Як було описано в розділі 2.2, його серверна частина написана мовою PHP і використовує реляційну базу даних для зберігання даних. Щоб мати можливість ефективно розробляти новий функціонал для системи, потрібно організувати локальне середовище розробки.

2.3.1 Підготовка

Головною частиною середовища розробки є інсталяція системи Moodle на локальній машині. Ця інсталяція потребує:

- Робочий веб-сервер, який підтримує PHP.
- Сервер бази даних.

Варто зазначити, що для коректної роботи Moodle може вимагати конкретні версії PHP та серверу бази даних, тому перед інсталяцією варто ознайомитися з примітками щодо версій на сторінці https://docs.moodle.org/dev/Releases.

Як альтернативу, замість встановлення системи локально, можна використати Docker контейнер, але ця робота не описує налаштування середовища розробки з використанням Docker.

2.3.2 Інсталяція

Після того, як всі потреби виконані, потрібно отримати вихідний код системи та скопіювати його у папку зі статичними ресурсами серверу. Вихідний код можна отримати з відкритого репозиторію на GitHub https://github.com/moodle/moodle або з офіційного сайту системи https://download.moodle.org/. Обидва варіанти надають можливість отримати потрібну версію.

Moodle надає можливість провести процес інсталяції, використовуючи веб-браузер. Для цього, після копіювання вихідного коду у папку зі статичними ресурсами, потрібно в браузері перейти по URL, який вказує на місцезнаходження Moodle на локальному сервері. Далі почнеться процес інсталяції Moodle. У рамках цього процесу потрібно підтвердити угоду про

авторські права та надати інформацію, що описує деталі облікового запису адміністратора та деталі сайту. Веб-інтерфейс продемонструє всі таблиці та додаткові модулі, що будуть створені. Після закінчення інсталяції має відкритися головна сторінка системи.

2.3.3 Фінальні кроки

Конфігурація Moodle за замовчуванням не оптимізована для розробки. Для її оптимізації можна змінити потрібні налаштування, використовуючи інтерфейс “Site Administration”. Більш зручним способом є визначення налаштувань у файлі config.php, що знаходиться в кореневій директорії вихідного коду системи. Нижче наведені налаштування, які мають підходити для вирішення більшої кількості задач:

```
// other system properties

$CFG->debug = E_ALL;
$CFG->debugdisplay = 1;
$CFG->langstringcache = 0;
$CFG->cachetemplates = 0;
$CFG->cachejs = 0;
$CFG->perfdebug = 15;
$CFG->debugpageinfo = 1;
```

Така конфігурація вмикає найвищий рівень логування повідомлень, відлагодження та виведення інформації, потрібної для відлагодження, в користувацький інтерфейс. До того ж, вимикається збереження даних в кеш, завдяки чому зміни, що вносяться у вихідний код, можна одразу побачити після перевантаження сторінки.

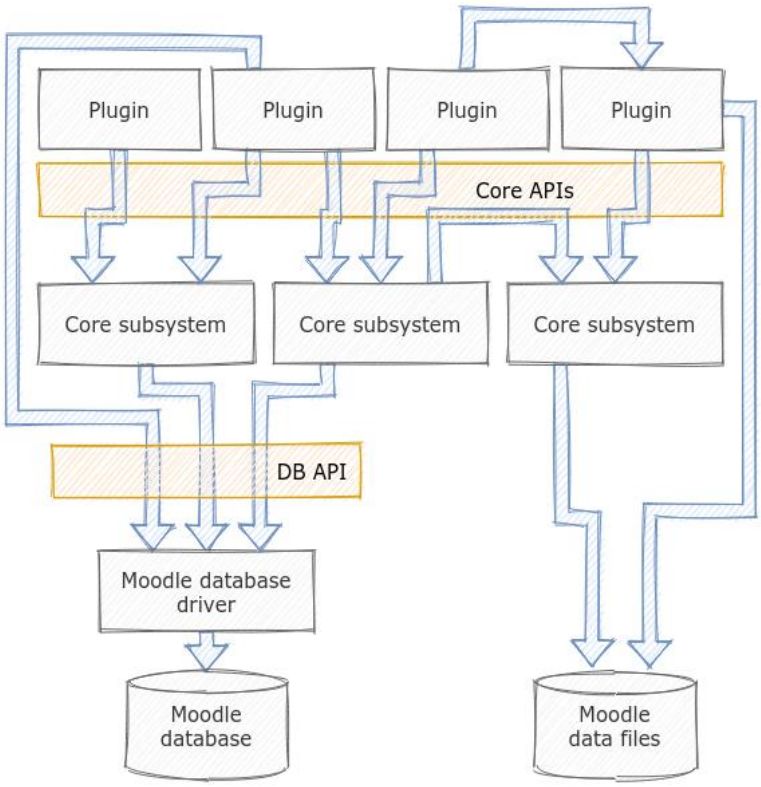
2.4 Поняття “плагін”

Більшість функцій, з якими користувачі взаємодіють, реалізовані за допомогою незалежних модулів, які в Moodle називають плагінами.

За замовчуванням Moodle має значну кількість стандартних плагінів, які є частиною інсталяції Moodle за замовчуванням. Якщо функціоналу, що надають ці плагіни, не вистачає, то можна знайти або створити додаткові плагіни, які надаватимуть потрібний функціонал. Існує офіційне сховище, де знаходяться додаткові плагіни під назвою “Moodle Plugins Directory”. Це сховище знаходиться за адресою https://moodle.org/plugins/.

Стандартні плагіни займають майже половину системи контролю навчання у файловій системі. Друга половина складається з підсистем, що відносяться до основного модулю. Ці підсистеми надають базове API, які використовують плагіни. Більше інформації про базове API знаходиться в розділі 2.5.

Ця схема відображає поведінку плагінів в описаній архітектурі:



Зображення 2

2.5 Базове API

Перед тим, як більш детально переходити до дослідження плагінів, варто описати базове API Moodle, якими можуть користуватися плагіни.

Базове API Moodle – набір інструментів, які мають використовувати плагіни для реалізації свого функціоналу. Дуже важливо знати ці інструменти, бо невиправдане написання своєї реалізації може призвести до некоректної роботи системи та створення проблем із її захищеністю.

Варто зазначити, що ця робота ставить собі за мету описати найбільш використовану частину базового API Moodle для кращого розуміння наступних розділів. Більш детальну інформацію можна знайти в офіційній документації Moodle за адресою https://docs.moodle.org/dev.

Базове API Moodle складається з:

- Access API – набір функцій, завдяки яким можна визначати права користувачів у системі. Також являє собою модуль, який контролює розширення можливостей системи за допомогою плагінів.
- Data manipulation API – набір функцій, завдяки яким можна виконувати операції відносно бази даних.
- File API – набір функцій, завдяки яким можна контролювати зберігання файлів у файловій системі.
- Form API – набір функцій, завдяки яким можна створювати веб-форми для обміну інформацією з користувачем.
- Logging API – набір функцій, завдяки яким можна логувати інформацію та контролювати збереження інформації, що логується.
- Navigation API – набір функцій, завдяки яким можна модифікувати навігаційне дерево системи.
- Page API – набір функцій, завдяки яким можна контролювати візуальне відображення веб-сторінки.

- Output API – набір функцій, завдяки яким можна контролювати видачу HTML сторінки користувачеві.
- String API – функціонал, що відповідає за локалізацію.
- Upgrade API – функціонал, що відповідає за встановлення та оновлення модулів системи.
- Moodlelib API – набір функцій, що являють собою утиліти, що можуть бути використані будь-яким плагіном. Наприклад, отримання параметрів з HTTP запиту описане саме в цьому API.

2.6 Типи плагінів

У Moodle плагіни відносяться до різних типів, кожен з яких фокусується на певній області функціональності. Важливо розуміти, як обрати відповідний тип для реалізації потрібного функціоналу. Це важливо, бо плагіни в Moodle можуть мати тільки один тип і якщо спочатку обрати тип невірно, то процес розробки може значно ускладнитись.

Директорії в кореневій частини вихідного коду відділяють плагіни різних типів один від одного. Наприклад, плагіни, що знаходяться в директорії /mod/, належать до типу “модулі діяльності”.

У Moodle існує дуже велика кількість типів плагінів і ця робота не ставить собі за мету описати їх всі. Ця робота буде описувати тільки деяку кількість з них.

2.6.1 Activity modules (Модулі діяльності)

Модулі діяльності – це тип плагінів, які надають функціонал, за допомогою якого можна використовувати певну діяльність у веденні курсу. Знаходяться плагіни цього типу в папці /mod.

Діяльність в Moodle – екземпляр модулю діяльності, що використовується у курсі. Наприклад, конкретні завдання, форуми, опитувальники і тому подібне.

Деякі модулі діяльності такі як “книга”, “URL” називають ресурсами, бо самі по собі ці модулі не мають поведінки.

Особливості даного типу плагінів:

- Можливість використання “Gradebook API”, за допомогою якого може здійснюватися оцінювання.
- Можливість використання “Activity Completion API”, завдяки якому можна контролювати життєвий цикл діяльності.
- Можливість використання “Backup API”, за допомогою якого активності можна включити до резервної копії курсу.
- Можливість використання “Availability API”, завдяки якому можна контролювати доступ до діяльності.

2.6.2 Blocks (Блоки)

Блоки – це тип плагінів, які надають функціонал, за допомогою якого можна створювати елементи сторінки в Moodle. Екземпляри блоків можна створювати тільки в спеціальних регіонах, які контролюються макетом обраної теми. Знаходяться плагіни цього типу в папці /blocks.

Екземпляри блоків, зазвичай, використовуються для відображення деякої інформації або для її вводу. Цей тип плагінів є досить схожим на модулі діяльності, але на відміну від цього типу майже не має поведінки. Через це блок не можна назвати діяльністю.

Особливості даного типу плагінів:

- Екземпляри блоків можуть бути створенні в будь-якій частині системи.
- Екземпляри блоків, як і модулів поведінки, можуть бути додані в резервну копію курсу.
- Екземпляр блоку можна конфігурувати не впливаючи на інші екземпляри.

2.6.3 Authentication (Аутентифікаційні)

Аутентифікаційні плагіни – тип плагінів, які модифікують процес аутентифікації користувачів. Наприклад, окрім аутентифікації за допомогою локального профілю, можна надати можливість робити це за використовуючи профілі із зовнішніх сервісів за допомогою протоколів LDAP, OAuth та інших. Знаходяться плагіни цього типу в папці /auth.

2.6.4 Local (Локальні)

Локальні плагіни – тип плагінів, які надають функціонал потрібний для конкретних установ, що використовують Moodle. Це тип плагінів, що використовується для всіх додаткових локальних налаштувань. Знаходяться плагіни цього типу в папці /local.

У більшості випадків цей тип використовуються для додаткових локальних налаштувань Moodle або для реалізації функцій, потреби яких не вписуються в інші типи плагінів Moodle.

Особливості даного типу плагінів:

- Конфігурація навігаційних елементів ресурсу.
- Можливість створення заголовної сторінки ресурсу.
- Можливість розширювати публічне API. Це корисна особливість, якщо потрібно адаптувати публічне API системи для використання конкретними зовнішніми сервісами.

2.6.5 Admin tools (Адміністративні утиліти)

Адміністративні утиліти – тип плагінів, які надають функціонал, потрібний для адміністрування та обслуговування системи. Зазвичай вони доступні у меню адміністрування системи. Знаходяться плагіни цього типу в папці /admin/tool.

Насправді, плагіни такого типу можуть використовуватися не тільки адміністраторами системи. Наприклад, функціонал навчальних планів надає плагін, який є адміністративною утилітою.

2.5.6 Інші типи плагінів

Як вже було описано раніше, ця робота не описує всі існуючі типи плагінів. Інформацію про всі типи плагінів можна знайти в офіційній документації Moodle за адресою https://docs.moodle.org/dev/.

2.7 Файлова структура плагінів

Незважаючи на те, що існує багато різних типів плагінів, усі вони мають деяку спільну файлову структуру. Варто зазначити, що всі файлові шляхи описані в цьому розділі є відносними до директорії плагіну.

Файлова структура для будь-якого плагіну складається з:

- /version.php – цей файл визначає метадані про сам плагін. Наприклад, номер версії або зовнішні залежності.
- /lang/{language}/{plugin_type}_{plugin_name}.php – набір файлів, у яких визначаються строки потрібні для локалізації. Параметр шляху language визначає мову, для якої надаються строки, а plugin_type та plugin_name визначають тип та назву плагіну відповідно. Якщо плагін відноситься до модулів діяльності, то тип плагіну не вказується.
- /lib.php – файл, в якому знаходяться функції, що являють собою інтерфейс між основним модулем Moodle та визначеним плагіном. Вміст залежить від типу плагіну. Зазвичай основний модуль системи підвантажує ці файли з кожного плагіну, тому, з міркувань продуктивності, рекомендується не робити ці файли великими.
- /locallib.php – файл, в якому зберігається внутрішня логіка плагіну. Використання цього файлу є застарілим, бо їх використання “забруднює” глобальний простір імен та погіршує розуміння логіки плагіну. Замість таких файлів рекомендується використання файлів у директорії /classes/ із використанням об’єктно-орієнтованого підходу.
- /db/install.xml – файл, що визначає схему бази даних плагіна. Цей файл генерується редактором XMLDB, що надається системою.
- /db/upgrade.php – файл, що визначає кроки оновлення схеми бази даних та інших даних плагіну. Може бути згенерований XMLDB редактором.

- /db/access.php – файл, що визначає відповідальності плагіну. Цей файл використовує Access API системи.
- /db/install.php – файл, що визначає PHP-код, який має виконуватися після встановлення схеми бази даних плагіну.
- /db/uninstall.php – файл, що визначає PHP-код, який має виконуватися після видалення схеми бази даних плагіну.
- /db/events.php – файл, що визначає обробники подій плагіну. У цьому файлі перераховуються всі події, за яким плагін спостерігатиме. Обробники подій визначаються у асоціативному масиві $observers. Наприклад, обробник події оновлення курсу визначається таким чином:

```
$observers = [
    [
        'eventname' => '\core\event\course_updated',
        'callback' => '\local_course_utils\course_event_observer::course_updated'
    ]
];
```

  Параметр “eventname” визначає назву події. Параметр “callback” визначає статичний метод класу, який відіграватиме роль обробника події.
- /db/messages.php – файл, що описує можливості в обміні повідомленнями плагіном.
- /db/services.php – файл, що описує зовнішні функції, що надаються плагіном.
- /db/renamedclasses.php – описує класи, які були перейменовані для підтримки автоматичного завантаження.
- /classes/ – набір файлів, які являють собою класи, в яких зберігається внутрішня логіка плагіну.
- /cli/ – набір файлів, що являють собою скрипти, що мають виконуватись із використанням командної строки.

- /settings.php – файл, що описує адміністративні налаштування плагіну. Moodle надає можливість описати налаштування в цьому файлі таким чином, щоб їх можна було змінювати з користувацького інтерфейсу.
- /amd/ – набір JavaScript файлів плагіну, що написані у форматі AMD або ES6.
- /yui/ – набір файлів, які описують спосіб додавання JavaScript коду в логіку плагіну.
- /jquery/ – набір файлів, що описують модулі jQuery, що використовуються у плагіні.
- /styles.css – файл, що описує CSS плагіну.
- /pix/icon.svg – зображення, що використовуватиметься як значок плагіну.
- /thirdpartylibs.xml – файл, що описує зовнішні бібліотеки, які використовуються у плагіні.
- /readme_moodle.txt – файл, що містить докладні вказівки щодо імпортування сторонніх бібліотек в систему.
- /environment.xml – файл, що описує додаткові вимоги до середовища для коректної роботи плагіну.
- /README.md або /README.txt – файл, що описує плагін та його роботу.
- /CHANGES.md або /CHANGES.txt – файл, що описує зміни у версії плагіну.

Варто зазначити, що ця робота ставить собі за мету описати файлову структуру плагіну. Майже кожен з вище описаних файлів має свої особливості в оформленні та взаємодії з системою. Ці особливості описані в офіційній документації Moodle за адресою https://docs.moodle.org/dev/.

2.8 Встановлення та оновлення плагінів

Кожен плагін має визначену версію. Вона визначається у файлі плагіна version.php як $plugin->version. Наприклад:

```
<?php

$plugin = new stdClass();
$plugin->version = 2021033100;
```

У даному випадку версія плагіну дорівнює 2021040800. Цей номер є важливим, бо він відіграє значну роль при встановленні та оновленні плагіну.

Версія плагіну являє собою номер і представляється у форматі “YYYYMMDDXX”, де:

- YYYY – рік.
- MM – місяць.
- DD – день.
- XX – порядковий номер.

Таким чином, розробники визначають версію свого плагіну, базуючись на даті, під час якої відбувається розробка. Коли плагін встановлюється, система аналізує файл version.php та зберігає версію плагіну. Таким чином, коли версія плагіну збільшиться, система буде про це проінформована і почнеться процес оновлення. Зазвичай версія плагіну використовується файлом /db/upgrade.php плагіну, щоб коректно виконати процес оновлення схеми бази даних. Moodle не підтримує оновлення до більш старої версії. У такому випадку потрібно видалити плагін та встановити потрібну версію.

2.9 Робота з базою даних

Moodle підтримує більшість реляційних баз даних. На жаль, різні реалізації реляційних баз даних мають відмінності у синтаксисі, тому використання чистих запитів DDL[2] та DML [3]може призвести до прямої залежності від конкретної реалізації бази даних.

Щоб уникнути цього, Moodle пропонує такі підходи для уникнення цієї проблеми:

- Схема бази даних зберігається у форматі XML. Потім дані з XML формату перетворюються на запити DDL.
- Використання Data Manipulation API, якщо його функціоналу вистачає. Якщо все ж таки потрібно написати SQL [4]запит, до цього процесу потрібно підійти дуже відповідально і впевнитися, що запит не використовує функціонал властивий певній реалізації бази даних.

2.9.1 Створення та редагування схеми бази даних

Раніше схема бази даних прописувалась у форматі SQL запитів для кожної з можливих реалізацій баз даних. Для вирішення цієї проблеми в Moodle було впроваджено визначення схеми бази даних за допомогою формату XML.

Щоб розробникам не доводилося власноруч редагувати XML файли був створений інструмент під назвою “XMLDB editor”, який вбудовано у саму систему. Його можна знайти за шляхом:

Site Administration > Development > XMLDB editor

Завдяки цьому інструменту можна з легкістю маніпулювати таблицями, ключами, полями та індексами. До того ж, цей інструмент може допомогти розробити коректний файл upgrade.php, щоб при оновленні плагіну схема бази даних була коректною у відношенні до встановленої версії плагіну.

[2] DDL (Data Definition Language) – мова, що використовується для визначення структур даних.
[3] DML (Data Manipulation Language) – мова, що використовується для додавання (вставки), видалення та модифікації (оновлення) даних у базі даних.
[4] SQL (Structured Query Language) – мова, яка призначена для управління даними, що зберігаються в реляційній СУБД.

Отже, в результаті роботи з XMLDB editor можуть бути створені такі файли:

- /db/index.xml – файл, який зберігає у собі схему бази даних, яка має бути створена для коректної роботи плагіна.
- /db/upgrade.php – файл, що описує оновлення схеми бази даних для коректної роботи обраної версії плагіну та інші процедури, які мають бути виконані для коректної роботи оновленого плагіну.

2.9.2 Запити до бази даних

Доступ до Data Manipulation API, яке потрібне для виконання запитів до бази даних, відбувається за допомогою глобального об’єкту $DB, який є екземпляром класу moodle_database.

Це API надає такі можливості:

- Виконання CRUD [5]операцій відносно бази даних.
- Використання логічних імен таблиць. Зазвичай, Moodle створює таблиці з префіксом mdl_. Завдяки Data Manipulation API розробнику не потрібно думати про це. Достатньо написати ім’я таблиці без суфікса.
- Можливість використання параметрів в SQL запитах із захистом від SQL ін’єкцій.
- Можливість визначення поведінки залежно від наявності чи відсутності результату після виконання запиту.
- Можливість працювати з кількістю ресурсів, що виділяється для виконання запитів.
- Можливість працювати з транзакціями.
- Можливість виконувати власні SQL запити.

Варто зазначити, що використання всіх вище описаних можливостей, окрім останньої, надає повну незалежність бази даних, бо фактично не вимагає від розробника написання SQL коду.

[5]CRUD (create, read, update, delete) – набір базових операцій для маніпулювання даними в базі даних.

Більш детальну інформацію про Data Manipulation API можна знайти за адресою https://docs.moodle.org/dev/Data_manipulation_API.

2.10 Локалізація

У Moodle англійська є мовою за замовчуванням, тому не залежно від того, чи розроблюється плагін для публічного використання, чи для використання в конкретній установі, якщо мова цієї установи не англійська, усі строкові значення, що бачить користувач, мають бути локалізовані.

Для виконання цієї цілі Moodle надає Localization API. Усі локалізовані дані знаходяться за шляхом /lang/{language}/{plugin_type}_{plugin_name}.php, де параметр шляху language визначає мову, для якої надаються строки, а plugin_type та plugin_name визначають тип та назву плагіну відповідно.

Локалізаційні строки визначаються за допомогою асоціативного масиву під назвою $string. Наприклад:

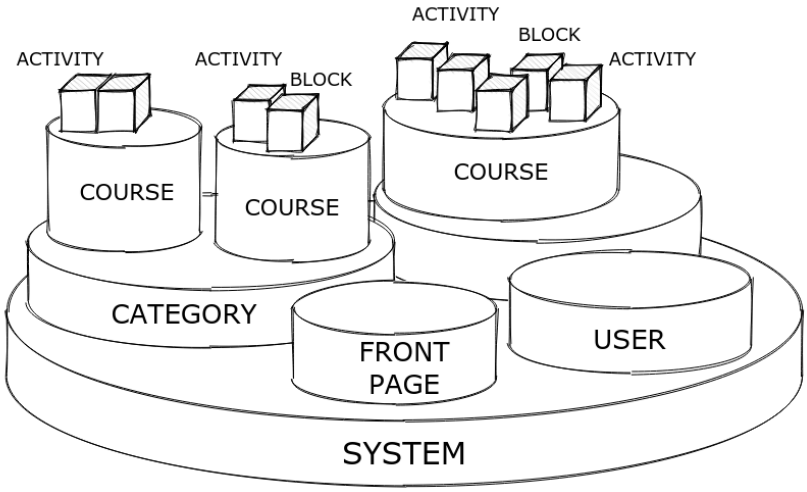
```
$string[‘pluginname’] = 'Localised plugin name';
```

Отримання локалізованих строк відбувається за допомогою глобального методу get_string({key}, {pluginname}), де параметр key являє собою ключ, якому відповідає локалізована строка, а pluginname назва плагіну, в якому знаходиться масив з локалізованими значеннями. Варто зазначити, що Localization API також надає функціонал для локалізації дат та чисел.

2.11 Контроль доступу

У системі контролю навчання наявні різні діючі сторони. Наприклад, викладачі, студенти, адміністратори та інші, залежно від специфіки установи. У Moodle наявний функціонал для контролю роботи системи залежно від типів користувачів.

Усі елементи системи належать до ієрархії областей, що називаються контекстами. Користувачі, у свою чергу, мають свої ролі в цих контекстах, відповідно до яких відображається інформація у різному вигляді. Наприклад, користувачі з ролями викладача та студента можуть бачити наповнення курсу по-різному, оскільки таким чином налаштовано контекст.



Зображення 3 Приблизна ієрархія контекстів в системі

Завдяки такому підходу стає можливим створення комплексних освітніх середовищ та використання елементів, що стосуються освітнього процесу, різними способами.

Система Moodle вже має перелік визначених ролей, яких достатньо для підтримки стандартного навчального процесу. Серед таких ролей є “студент”, “викладач”, “адміністратор” та інші. Якщо цих ролей недостатньо, то можна створити додаткові ролі.

Кожна роль має пов’язані з нею дозволи. Щоб визначити, чи може користувач отримати доступ до певної інформації, контекст отримує всі дозволи відповідно до ролей користувача, далі залежно від цих дозволів приймає остаточне рішення. Наприклад, користувач в контексті деякого

форуму хоче створити обговорення. Контекст, до якого належить форум, враховуючи ролі користувача, отримує його дозволи і, якщо серед них є дозвіл, що відповідає створенню обговорення, відображає кнопку створення.

Для контролю доступу в описаній моделі Moodle надає Access API. Воно надає такі можливості для розробників плагінів:

- Визначення дозволів, потрібних для плагіна. Уся подібна інформація знаходитиметься в файлі /db/access.php плагіну.
- Визначення наявності потрібних дозволів у користувача. Ця можливість досягається за допомогою використання глобальних методів has_capability та require_capability.

Однією з важливих задач розробника плагіну є приділення уваги контролю доступу до функцій свого плагіну. Недостатня увага в цьому напрямі може призвести до того, що користувачі зможуть працювати з тими частинами системи, до яких вони не повинні мати доступу.

2.12 Підтримка захищеності системи

На жаль, гнучкість у реалізації функціоналу для Moodle може призвести до погіршення захищеності системи, якщо розробник не приділяє цьому увагу.

Для того, щоб система була захищена, не достатньо тільки слідкувати за контролем доступу. Ще варто приділити увагу таким аспектам:

- Перевірка аутентифікації користувача. За замовчуванням, Moodle не вимагає аутентифікації користувача для доступу до веб-сторінок плагінів. Для цього, перед відпрацюванням потрібних скриптів, потрібно додати виклик глобальної функції require_login. Таким чином, якщо користувач не аутентифікований, то скрипт не виконається, а користувач отримає відповідне повідомлення.
- Перевірка наявності потрібних дозволів. Як вже було сказано у минулому розділі, недостатня увага до наявних дозволів у користувача може призвести до неуповноваженого доступу до частини системи.
- Дані, отримані від користувача, мають бути коректно оброблені перед роботою з ними за допомогою глобальних функцій format_text та format_string. Без подібної процедури користувач може надіслати дані в специфічному форматі, наприклад, HTML форматі, що може призвести до неочікуваних наслідків.
- Дані, що відправляються користувачеві, теж мають бути коректно оброблені, як описано в попередньому пункті.
- В SQL запитах мають використовуватись заповнювачі параметрів, щоб уникнути SQL ін’єкції.

До того ж, варто ознайомитися з вразливостями, що описує OWASP[6], щоб мати можливість вчасно побачити та виправити існуючі вразливості в реалізації.

[6] OWASP (Open Web Application Security Project) – некомерційна проект, який працює над підвищенням безпеки програмного забезпечення.

3. Реалізація практичної частини

3.1 Опис плагіну

У результаті аналізу предметної області, дослідження засобів розробки та наявних недоліків у поточній реалізації Moodle було реалізовано плагін для системи Moodle, що спрощує роботу викладачів з датами у своїх курсах.

Наприклад, викладач проводить свій курс кожен рік. Він не хоче кожного року створювати новий курс та заповнювати його із самого початку. Також він не хоче переносити дати, пов’язані з курсом та його активностями, вручну.

Головною функцією плагіну є автоматична зміна дат для всіх активностей, що належать до курсу, при зміні дати початку курсу. Таким чином створений плагін вирішує описану проблему, і складність перенесення курсу в інший проміжок часу зводиться до зміни лише однієї дати.

Нові дати для активностей визначаються за формулою, де newDate – дата активності, що вираховується, date – поточна дата активності, newStartDate – модифікована дата початку курсу, oldStartDate – дата початку курсу до модифікації:

$$newDate = date + newStartDate - oldStartDate$$

Плагін написано мовою PHP із використанням API, що надає основний модуль Moodle та бази даних, що надається системою. До того ж, розробка велась відповідно до правил вказаних у офіційній документації Moodle, щоб цей плагін міг використовуватися будь-якою організацією, що використовує Moodle як систему контролю навчання.

3.2 Обґрунтування обраного підходу

Реалізувати описаний плагін можливо багатьма способами. Щоб обрати найбільш коректний спосіб реалізації під час процесу розробки плагіну, були встановлені такі критерії:

1. Розроблений плагін має бути сумісним з будь-якою збіркою Moodle з версією 3.1 та вище.
2. Плагін не має модифікувати структуру та поведінку визначених процедур системи.
3. Його інсталяція та видалення не має залишати після себе побічних ефектів.
4. Взаємодія користувача з плагіном має бути мінімальною.
5. Плагін має виконувати свої обов’язки тільки після явного підтвердження користувачем.

Головною задачею під час розробки було визначення моменту, коли плагін має починати виконувати свою головну задачу. Наприклад, цим моментом міг би бути процес, який модифікує дані курсу, або можна було б створити кнопку у самому курсі, при натиску на яку, починався б процес автоматичної зміни дат. Ці варіанти були відкинуті, бо порушують критерії під номером 2 та 4 відповідно. Натомість було вирішено використати Events API для відслідковування подій модифікації курсів. Таким чином плагін автоматично починає перенесення дат після модифікації курсу, ніяким чином не втручаючись в основні процеси роботи системи.

Цей підхід ускладнив реалізацію задачі, оскільки сама подія не несе у собі інформацію про дату початку курсу до модифікації, яка потрібна для вирахування нової дати для активності. Для цього було вирішено зберігати дати початків курсів в окремій таблиці та синхронізувати дані у цій таблиці відповідно до всіх курсів у системі. Для цього, плагіну потрібно відслідковувати створення, видалення та модифікацію курсів. Відслідковування та опрацювання цих подій було реалізовано за аналогічним підходом, описаним у попередньому абзаці.

Наступною задачею було визначення підходу, за яким користувач сигналізуватиме, що дати активностей в курсі мають автоматично оновлюватись. Насправді, можна було б зробити так, що дати просто оновлювалися у всіх курсах, але цей підхід порушує критерій під номером 5. Було вирішено, що до курсу буде додано параметр, який виконуватиме функцію індикатору для плагіну. Для реалізації даного підходу було використано Custom Fields API. Завдяки цьому API обраний підхід було реалізовано, до того ж, модель даних курсів ніяк не змінилася.

Останньою задачею було визначення підходу до оновлення дат активностей. На жаль, Moodle не вимагає реалізовувати функціонал із використанням об’єктно-орієнтованого підходу. Через це всі види активностей, кожний з яких належить до окремого плагіну, спроектовані та реалізовані за різними підходами і являють собою повністю різні сутності, хоча, з точки зору предметної області, мають багато спільного. Спочатку, було вирішено реалізувати окремий алгоритм оновлення дат для кожного типу активності, але цей підхід невиправдано ускладнював процес розробки. Замість цього було вирішено, що плагін зберігатиме метадані про види активностей і, коли потрібно, передаватиме активність та метадані про неї алгоритму, який використовуючи ці дані виконував би оновлення всіх потрібних дат.

Додатково реалізацію задачі оновлення дат ускладнило те, що дати активностей та відповідні дати у календарі Moodle зберігаються окремо. Через це довелося реалізувати загальний для всіх видів активностей алгоритм, що синхронізує дати в календарі з відповідними датами активності.

Було вирішено використовувати об’єктно-орієнтований підхід для реалізації функціоналу плагіну задля того, щоб зробити код більш структурованим та зрозумілим та полегшити процес розширення функціоналу плагіну у майбутньому.

3.3 Опис розробки програми

3.3.1 Точка виконання

Точкою виконання головної задачі плагіну є статичний метод course_updated класу course_event_observer, який автоматично викликається системою, коли створюється подія, що сигналізує модифікацію курсу.

Для автоматичного виклику потрібна конфігурація у файлі events.php:

```php
<?php

defined('MOODLE_INTERNAL') || die();

$observers = [
    [
        'eventname' => '\core\event\course_updated',
        'callback' => '\local_course_utils\course_event_observer::course_updated'
    ],
// other code
```

Сам метод у класі course_event_observer має такий вигляд:

```php
// header of the PHP file
class course_event_observer {

  // class fields

  public static function course_updated(course_updated $updatedEvent): void {
    // logic of the method
  }

  // other methods
```

3.3.2 Збереження та синхронізація даних потрібних для оновлення дат

Дані, потрібні для коректної роботи плагіну зберігаються у таблиці під назвою local_course_utils_managed. Схема цієї таблиці описується в XML форматі у файлі install.xml:

```
<?xml version="1.0" encoding="UTF-8" ?>
<XMLDB PATH="local/course_utils/db" VERSION="20210330" COMMENT="XMLDB file for Moodle
local/course_utils"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../../../lib/xmldb/xmldb.xsd"
>
 <TABLES>
  <TABLE NAME="local_course_utils_managed" COMMENT="Plugin managed courses">
   <FIELDS>
    <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" SEQUENCE="true"/>
    <FIELD NAME="courseid" TYPE="int" LENGTH="10" NOTNULL="true" SEQUENCE="false"/>
    <FIELD NAME="laststartdate" TYPE="int" LENGTH="10" NOTNULL="true" SEQUENCE="false"/>
   </FIELDS>
   <KEYS>
    <KEY NAME="local_course_utils_pk" TYPE="primary" FIELDS="id"/>
    <KEY NAME="local_course_utils_course_fk" TYPE="foreign" FIELDS="courseid"
REFTABLE="mdl_course" REFFIELDS="id"/>
   </KEYS>
  </TABLE>
 </TABLES>
</XMLDB>

}
```

Синхронізація даних у цій таблиці із даними всіх курсів відбувається таким чином:

- Під час встановлення плагіну потрібні дані курсів переносяться в таблицю плагіну через виклик методу xmldb_local_course_utils_install у файлі install.php:

```php
// header of the PHP file
function make_existing_courses_managed_by_plugin(): void {
  global $DB;
  $managedCoursesService = managed_courses_service::get_instance();
  $coursesDO = $DB->get_records('course', array(), '', 'id, startdate');
  $managedCoursesService->make_courses_managed($coursesDO);
}

// other code

function xmldb_local_course_utils_install() {
  make_existing_courses_managed_by_plugin();
  // other code
}
```

- Плагін відслідковує події створення та видалення курсів і відповідно оновлює дані:

```php
<?php

defined('MOODLE_INTERNAL') || die();

$observers = [
    // other entries
    [
        'eventname' => '\core\event\course_created',
        'callback' => '\local_course_utils\course_event_observer::course_created'
    ],
    [
        'eventname' => '\core\event\course_deleted',
        'callback' => '\local_course_utils\course_event_observer::course_deleted'
    ]
];
```

```php
// header of the PHP file
class course_event_observer {

  // other methods

  public static function course_created(course_created $createdEvent): void {
    $courseDO = $createdEvent->get_record_snapshot('course', $createdEvent->objectid);
    $managedCoursesService = managed_courses_service::get_instance();
    $managedCoursesService->make_course_managed($courseDO);
  }

  public static function course_deleted(course_deleted $deletedEvent): void {
    $managedCoursesService = managed_courses_service::get_instance();
    $managedCoursesService->make_course_unmanaged($deletedEvent->objectid);
  }
}
```

Уся логіка, що відноситься до збереження та синхронізації даних, що відноситься до курсів, знаходиться в класі managed_courses_service:

```php
// header of the PHP file
class managed_courses_service {

  // instantiation specific code

  public function make_courses_managed(array $coursesDO): void {
    global $DB;
    $existingCourseIds = array_keys($DB->get_records(
        db_constants::MANAGED_COURSES_TABLE, array(), '',
db_constants::MANAGED_COURSES_COLUMN_COURSE_ID
    ));
    $notInsertedDO = array_filter($coursesDO,
        function($record) use ($existingCourseIds) {
          return !in_array($record->id, $existingCourseIds);
        }
    );
    $toInsertDOs = array_map(
        function($courseDO) {
          $insertDO = new stdClass();
          $insertDO->courseid = $courseDO->id;
          $insertDO->laststartdate = $courseDO->startdate;
          return $insertDO;
        },
        $notInsertedDO);
    $DB->insert_records(db_constants::MANAGED_COURSES_TABLE, $toInsertDOs);
  }

  public function make_course_managed(stdClass $courseDO): void {
    $this->make_courses_managed([$courseDO]);
  }

  public function make_course_unmanaged(int $courseId): void {
    global $DB;
    $DB->delete_records(db_constants::MANAGED_COURSES_TABLE, ['courseid' => $courseId]);
  }

// other methods and instantiation specific code
}
```

3.3.3 Додавання додаткових параметрів до моделі курсу

Нові параметри для курсу додаються під час встановлення плагіну. Фактично, плагін просто додає дані у таблиці, що зберігають додаткові параметри для курсів. Уся логіка, що виконує цю задачу, знаходиться в класі custom_field_service:

```php
// header of PHP file
class custom_field_service {

// instantiation specific code

  public function create_field_category(string $name, ?string $description): stdClass {
    global $DB;
    $fieldCategoryDO = self::create_field_category_do($name, $description);
    $categoryId = $DB->insert_record(custom_category::TABLE, $fieldCategoryDO);
    $fieldCategoryDO->id = $categoryId;
    return $fieldCategoryDO;
  }

  public function create_custom_field(int $categoryId, string $name, string $shortname, ?string
$description): stdClass {
    global $DB;
    $customFieldDO = self::create_custom_field_do($categoryId, $name, $shortname, $description);
    $customFieldId = $DB->insert_record(custom_field::TABLE, $customFieldDO);
    $customFieldDO->id = $customFieldId;
    return $customFieldDO;
  }

// instantiation specific code

  private static function create_field_category_do(string $name, ?string $description): stdClass {
    $customFieldCategory = new stdClass();
    $customFieldCategory->name = $name;
    $customFieldCategory->description = $description;
    $customFieldCategory->descriptionformat = 0;
    $customFieldCategory->sortorder = 0;
    $curTime = time();
    $customFieldCategory->timecreated = $curTime;
    $customFieldCategory->timemodified = $curTime;
    $customFieldCategory->component = course::COMPONENT;
    $customFieldCategory->area = 'course';
    $customFieldCategory->itemid = 0;
    $customFieldCategory->contextid = 1;
    return $customFieldCategory;
  }

  private static function create_custom_field_do(
      int $categoryId, string $name, string $shortname, ?string $description): stdClass {
    $customField = new stdClass();
    $customField->shortname = $shortname;
    $customField->name = $name;
    $customField->type = 'checkbox';
    $customField->description = $description;
    $customField->descriptionformat = 1;
    $customField->sortorder = 0;
    $customField->categoryid = $categoryId;
    $customField->configdata =
'{"required":"0","uniquevalues":"0","checkbydefault":"0","locked":"0","visibility":"1"}';
    $curTime = time();
    $customField->timecreated = $curTime;
    $customField->timemodified = $curTime;
    return $customField;
  }
}
```

У процесі встановлення плагін дістає зі своєї конфігурації назву категорії додаткових параметрів і назву самого додаткового параметру та передає ці дані у відповідні методи екземпляру класу custom_field_service.

3.3.4 Алгоритм оновлення дат активностей

Уся логіка, що відноситься до оновлення дат активностей та календарних дат, знаходиться в класі date_service:

```
// header of the PHP file
class date_service {

  // instantiation specific code

  public function shift_events_date(cm_info $courseModule, int $shiftAmount, array $eventTypes = []):
void {
    $courseModuleEvents = self::get_events_for($courseModule);
    if (empty($eventTypes)) {
      $filteredEvents = $courseModuleEvents;
    } else {
      $filteredEvents = array_filter($courseModuleEvents,
          function($record) use ($eventTypes) {
            return in_array($record->eventtype, $eventTypes);
          }
      );
    }
    foreach ($filteredEvents as $event) {
      if (date_utils::shift_date_property_if_possible($event, 'timestart', $shiftAmount)) {
        $event = new calendar_event($event);
        $event->update(new stdClass());
      }
    }
  }

  public function shift_course_module_dates(cm_info $cm, array $dateProperties, int $shiftAmount) {
    global $DB;
    if (!empty($dateProperties)) {
      $instanceDO = $DB->get_record($cm->modname, ['id' => $cm->instance], '*', MUST_EXIST);
      foreach ($dateProperties as $dateProperty) {
        date_utils::shift_date_property_if_possible($instanceDO, $dateProperty, $shiftAmount);
      }
      $DB->update_record($cm->modname, $instanceDO);
    }
  }

  private static function get_events_for(cm_info $cm): array {
    global $DB;
    return $DB->get_records('event', [
        'modulename' => $cm->modname,
        'instance' => $cm->instance,
    ]);
  }

  // instantiation specific code
}
```

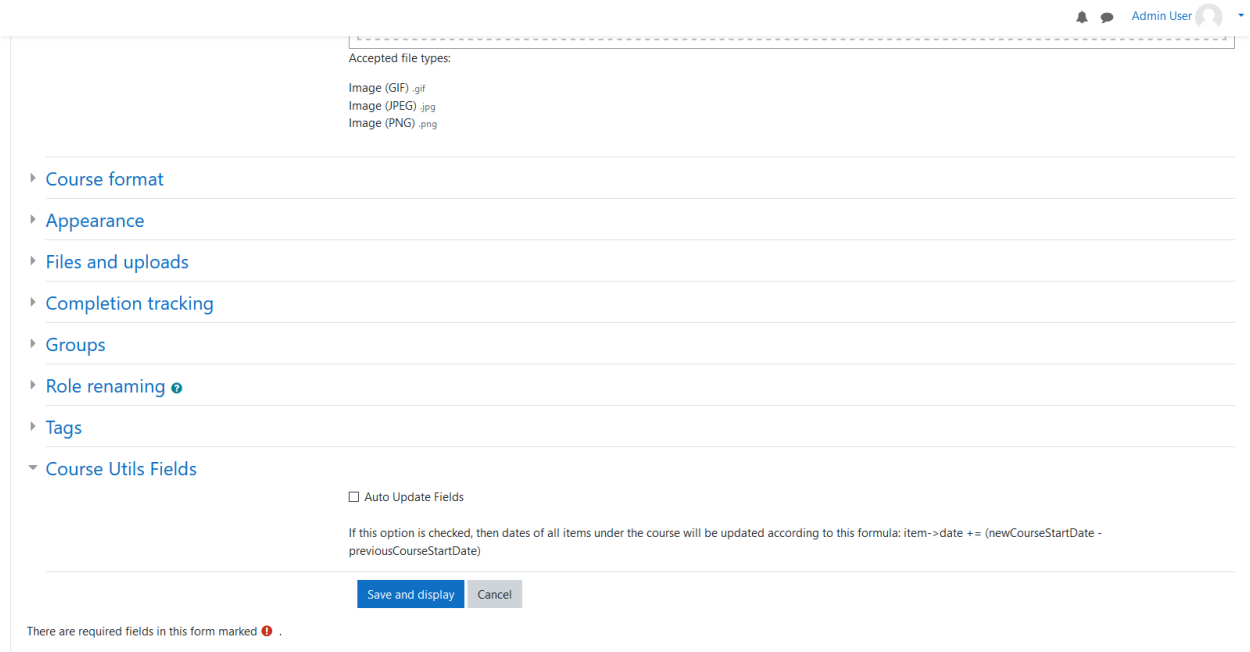
Коли курс модифіковано, система автоматично викликає метод course_event_observer::course_updade плагіну. У цьому методі визначається, чи потрібне автоматичне оновлення дат, і якщо потрібне, то для всіх активностей, що знаходяться в курсі, викликаються відповідні методи класу date_service. Код, що викликається після модифікації курсу:

```
$managedCoursesService = managed_courses_service::get_instance();
$courseId = $updatedEvent->objectid;
$courseDO = $updatedEvent->get_record_snapshot('course', $courseId);

$courseManagedInfo = null;
if ($managedCoursesService->can_course_auto_update($courseDO)) {
  $dateService = date_service::get_instance();
  $courseManagedInfo = $managedCoursesService->get_course_managed_info($courseId);

  $shiftAmount = $courseDO->startdate - $courseManagedInfo->laststartdate;
  if ($shiftAmount !== 0) {
    $modInfo = get_fast_modinfo($courseDO);
    $courseModules = $modInfo->get_cms();
    foreach ($courseModules as $courseModule) {
      $modName = $courseModule->modname;
      if (array_key_exists($modName, self::MOD_NAME_TO_DATE_FIELDS)) {
        $dateParameters = self::MOD_NAME_TO_DATE_FIELDS[$modName];
        $dateService->shift_course_module_dates($courseModule, $dateParameters, $shiftAmount);
      }
      $dateService->shift_events_date($courseModule, $shiftAmount);
    }
  }
}

if (is_null($courseManagedInfo)) {
  $managedCoursesService->update_managed_course_info_by_course_id($updatedEvent->objectid,
$courseDO->startdate);
} else {
  $courseManagedInfo->laststartdate = $courseDO->startdate;
  $managedCoursesService->update_managed_course_info_with_do($courseManagedInfo);
}
```

3.4 Опис файлів даних та інтерфейсу програми

Плагін не зберігає дані у файлах.

Плагін додає секцію в інтерфейс модифікації курсу. У цій секції наявний параметр під назвою “Auto Update Dates”. Якщо користувач зробить його активним, то автоматичне оновлення дат буде ввімкнено для курсу.



Зображення 4 Секція, що додається до інтерфейсу модифікації курсу

Висновки

У цій роботі було досліджено предметну область, до якої відноситься електронне навчання, проблеми, що ця область ставить, та засоби, які потрібні для розробки компонент систем контролю навчання, на прикладі Moodle. У результаті було реалізовано плагін, що спрощує роботу викладачів з датами у своїх курсах. Завдяки дотриманню визначених критеріїв розроблений плагін може використовуватись не тільки в конкретному навчальному закладі, для якого він розроблявся, а й у будь-якій установі, що використовує систему Moodle. При належному покращенні й розвитку плагіну він може стати якісним набором утиліт, що полегшують роботу з системою для всіх учасників освітнього процесу.

Майбутнє вдосконалення плагіну варто почати з реалізації тестування для нього, без якого розширення функціоналу плагіну може бути неякісним. Більше того, більшість процесів цього плагіну відбуваються непомітно для користувача, тому це також може призвести до невчасно помічених багів. У результаті це може вплинути на освітній процес, чого не варто допускати. Після реалізації тестування можна зібрати зворотній зв’язок щодо роботи в системі Moodle від усіх учасників освітнього процесу і почати реалізовувати функціонал, що буде вирішувати найбільш критичні із вказаних проблем у системі.

Варто зазначити про вдосконалення системи Moodle, що зараз використовується в НаУКМА. Плагін, що був створений із метою покращення колаборативності навчання, під назвою “Review” потребує розширення свого функціоналу. Наприклад, цей плагін не дає можливості призначення рецензентів, якщо робіт, які можна рецензувати, менше ніж самих рецензентів, не вистачає можливості групового рецензування, а також функціоналу автоматичного призначення рецензентів. Систему можна розвивати в напрямку підвищення інтерактивності навчання. Наприклад, використання коду у вигляді інтерактивних вікон замість зображень або тексту. Ці та інші вдосконалення плануються в рамках наступних досліджень.

Список Джерел

1. Розділ “До питання електронного навчання програмуванню” з роботи “Наукові записки НаУКМА т.151. Комп’ютерні науки”. Автор – Бублик В. В. Рік видання – 2013.
2. Розділ “Особливості впровадження навчальної групової розробки програмних систем” з роботи “Наукові записки НаУКМА Т.86. Комп’ютерні науки”. Автори – Бублик В. В., Афонін А. О., Борозенний С. О. Рік видання – 2009 р.
3. “Multimedia learning”. Автор – Річард Е. Майєрю Рік видання – 2001.
4. "An Argument for Clarity: What are Learning Management Systems, What are They Not, and What Should They Become?". Автор – Вільям Р. Уотсон. Рік видання – 2007.
5. "History and Trends of Learning Management System (Infographic)". Автор – ресурс Oxagile. Дата публікації – 2016 р.
6. “Integrating a Learning Management System into the Instructional Program and Learning Process: Challenges and Advantages”. Автори – Ожужан Бозоглу, Селін Армаган, Каглаян Ердонмез. Рік видання – 2016.
7. “System and software quality model ISO/IEC 25010:2011”. Рік публікації – 2011.
   https://www.iso.org/standard/35733.html
8. Курс “Moodle Plugin Development Basics”.
   https://learn.moodle.org/course/view.php?id=26428
9. Офіційна користувацька документація Moodle.
   https://docs.moodle.org/310/en/Main_page
10. Офіційна документація Moodle для розробників.
    https://docs.moodle.org/dev/Main_Page
11. Офіційний сайт Moodle.
    https://moodle.org/