МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки
Факультет Комп'ютерних наук
Кафедра Програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)

(рівень вищої освіти)

Дослідження методів інтеграції даних з різнорідних веб-систем. Агрегація та уніфікація даних

Виконав:
студент 2 курсу групи ІПЗм-20-3

Кошовий М.Ю.

(прізвище, ініціали)

Спеціальність	121 – Інженерія програмного забезпечення
Тип програми	Освітньо-наукова
Керівник	доц. Каук В.І.

(посада, прізвище, ініціали)

Допускається до захисту
Зав. Кафедри З.В. Дудар

2022 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121– Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«__»_____202_ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Кошового Максима Юрійовича
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів інтеграції даних з різнорідних веб-систем. Агрегація та уніфікація даних»

затверджена наказом університету від «__»_____202___р. №____

2. Термін подання студентом роботи до екзаменаційної комісії «__» _____202__р.

3. Вихідні дані до роботи *інтеграція даних, Google Calendar, Android Calendar, Moodle, пояснювальна записка.*

4. Перелік питань, що потрібно опрацювати в роботі *аналіз предметної галузі та постановка задачі, аналіз способів інтеграції додатків, веб-скрейпінг, реалізація інтеграції з Google Calendar, Android Calendar та Moodle.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі та постановка задачі	17.01.2022	виконано
2	Аналіз існуючих методів	31.01.2022	виконано
3	Проектування та розробка програмної системи	21.02.2022	виконано
4	Підготовка пояснювальної записки	14.03.2022	виконано
5	Підготовка презентації та доповіді		
6	Перевірка на академічний плагіат		
7	Нормоконтроль		
8	Рецензування		
9	Занесення диплома в електронний архів		
10	Попередній захист		
11	Допуск до захисту		

Дата видачі завдання 17 січня 2022 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Каук В.І.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 67 с., 25 рис., 1 табл., 11 джер.

ІНТЕГРАЦІЯ, ANDROID CALENDAR, API, GOOGLE CALENDAR, MOODLE, REST

Об’єктом дослідження є методи інтеграції з зовнішніми системами, а також системи для інтеграції ЦИСТ, Google Calendar, Android Calendar та Moodle.

Метою роботи є дослідження методів інтеграції даних з різнорідних веб-систем.

Результатом кваліфікаційної роботи є реалізовані методи інтеграції даних з ЦИСТ, Google Calendar, Android Calendar та Moodle.

INTEGRATION, ANDROID CALENDAR, API, GOOGLE CALENDAR, MOODLE, REST

The object of research is methods of integration with external systems, as well as systems for integration Google Calendar, Android Calendar and Moodle.

The aim of the work is to explore methods of integrating data from disparate web systems.

The result of the course work is implemented methods of data integration with Google Calendar, Android Calendar and Moodle.

Я, Кошовий Максим Юрійович, студент групи ІПЗм-20-3, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів інтеграції даних з різнорідних веб-систем. Агрегація та уніфікація даних», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві

відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Нині затребуваність дистанційної форми навчання неухильно зростає. Актуальність дистанційної освіти раптово зросла на тлі пандемії. В умовах локдауну це справді вихід і, як показує практика, один із найефективніших. Дистанційна форма навчання гнучка, зручна і доступна, передбачає широку варіативність та диференціацію у виборі змісту та форм здобуття освіти. Дистанційне навчання є засобом забезпечення доступної якісної освіти [1].

Сьогодні бізнес стикається з величезною конкуренцією у зв'язку з розвитком технологій. Організації повинні мати можливість використовувати інформацію та створювати нові знання, щоб вести бізнес. Інновації у технології та взаємопов'язаності речей щодня генерують величезну кількість даних. Інтеграція даних допомагає переміщати дані із джерела в ціль.

Сучасні додатки та сайти не можуть бути ізольовані. Через зовнішню інтеграцію налаштовується авторизація через соцмережі, онлайн оплата, замовлення доставки на постійній основі. Для оптимізації бізнес-процесу сайт та інтегровані послуги повинні автоматично обмінюватися даними в режимі реального часу.

Інтеграція додатків використовується для управління та підтримки всіх додатків у актуальному стані, одночасно усуваючи дублювання та надмірність даних. Створюючи мережу інтеграції додатків, яка дозволяє програмам взаємодіяти один з одним, бізнес та робочі процеси можуть виконуватися більш ефективно та результативно. Інтеграція програм додатково допомагає пом'якшити проблему інформаційних сховищ.

Необхідність інтеграції програм пов'язана з необхідністю переміщення даних між програмами. Це включає передачу даних з одного додатка в інший запланованим, поточним способом або для переміщення даних один раз з застарілої системи в щось нове.

Завдяки інтеграції програм можна вводити дані один раз і підключати їх до кількох програм замість того, щоб вводити їх стільки разів, скільки є програм. При додаванні нових даних до програми, яка була інтегрована з іншими програмами, дані будуть автоматично розподілені по всіх підключених програмах. Це зменшує кількість людських помилок, необхідність ручного втручання та загалом забезпечує узгодженість на платформах.

Наслідками відсутності вирішення проблеми інтеграції є:

– повторне ручне введення даних (довідники, дані про відвантаження, фінансові транзакції тощо);

– багаторазові та нескінченні звіряння та коригування, що не виключають помилок;

– непомірні витрати на формування зведеної звітності;

– неприйнятні терміни і собівартість виконання навіть звичайних завдань.

Це підкреслює актуальність тематики даної дослідницької роботи та визначає цілі інтеграції додатків. Загальні цілі інтеграції додатків можна сформулювати так:

– зменшити вартість експлуатації сукупності додатків підприємства;

– збільшити швидкість виконання типових завдань чи гарантувати терміни їх виконання;

– підняти якість виконання завдань за рахунок формалізації процесів та мінімізації людського фактора, як основного джерела помилок.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної галузі

Корпоративні програми отримують, обробляють і передають дані. Найчастіше для виконання одного бізнес-процесу компанія використовує кілька інформаційних систем, і між ними відбувається обмін даними. Одна система отримує інформацію від користувача і передає її через канали інтеграції. Наприклад, для авторизації на сервісному порталі користувач не створює новий обліковий запис: система отримує дані з Active Directory, а програма контролю доступу завантажує довідник співробітників з бази даних ERP SAP.

Інтеграція прискорює вирішення завдань, підвищує якість, крім людського фактора, знижує вартість володіння інформаційних систем без посередників і зменшує витрати.

Найпростіші способи інтеграції – це обмін файлами та повідомленнями або звернення до загальної бази даних [2]. Ці способи мають безліч недоліків, особливо в епоху поширення веб-додатків. Формати файлів можуть відрізнятися, а вивантаження, завантаження та конвертація – це додатковий вплив людського фактора та праці. Надавати всім доступ до однієї бази даних і контролювати правильність її використання різними програмами – серйозний ризик для цілісності та безпеки зберігання даних.

Оптимальний спосіб інтеграції – це API (application program interface) або програмний інтерфейс програми. Люди звикли, що куплений квиток у кіно автоматично додається до календаря, а миттєво авторизувавшись через google-аккаунт можна залишити коментар. Саме API з'єднує сайт із зовнішнім світом і дозволяє зробити необхідну дію – реєстрацію, покупку, підписку, не йдучи з сайту.

Можна з нуля розробити внутрішню інтеграцію, наприклад, інтегрувати CRM із сервісом поштових розсилок та імпортувати всі адреси клієнтів.

А найпопулярніші послуги розробляють свої API, роблять їх громадськими і одночасно складають документацію, що описує процес інтеграції. Таким чином

можна підключити сервіси бронювання, голосовий пошук та інші програми для зручності користувачів.

Розробник сайту зможе звернутися до існуючого API у своєму коді, а далі все залежатиме лише від функціональностей програми. Наприклад, авіакомпанії передають на сайт-агрегатор інформацію про свої пропозиції, що уможливлює бронювання авіаквитка в кілька кліків.

Незважаючи на те, що рішення необхідно створювати під кожен вид інтеграції та кожне бізнес-завдання, це спрощує роботу програміста та розробку продукту.

Інтеграція збільшує функціональність і використовується для того, щоб вирішувати такі завдання:

– приймати онлайн-заявки, обробляти реєстраційну інформацію від клієнта, знайомити покупців із товарами, розміщувати дані про асортимент;

– автоматизувати транспортування замовлень, відстежувати доставку товарів, моніторити виконання заявок;

– відстежувати кількість товарних одиниць на складі та синхронізувати обмін актуальними даними з інтернет-магазинами (маркетплейсами);

– синхронізувати оплату, складання рахунків, формування документів у автоматичному режимі;

– налаштувати технічну підтримку клієнтів у чаті на сайті, в офіційних групах у соцмережах, підключити телефонію;

– зробити більш простими старт рекламних кампаній та оцінку їхньої ефективності;

– зменшити витрати на запуск та реалізацію маркетингових стратегій, отримувати статистичні дані щодо них.

Все це дає можливість оптимізувати бізнес-процеси, вимкнути зайві ланки на етапах формування замовлення, підвищити привабливість сайту в очах користувачів, зацікавлених у товарах та послугах компанії, знизити витрати на обробку інформації вручну, отримувати звіти про продаж онлайн. Інтеграція

порталу зі сторонніми сервісами розширює функціонал сайту, підвищує його ефективність.

Можна додати таким чином наступні функціональності:

– чат: Slack, Facebook Messenger, WhatsApp;

– авторизація та доступи: LastPass, OneLogin, One Identity;

– рекомендації: Foursquare, Yelp;

– опитування: Typeform, Form.io;

– онлайн-оплата: Mastercard API, PayPal;

– публікації медіаконтенту: Youtube, Flickr, Last.fm, Vimeo;

– безпека: PhotoCaptcha, Key Captcha;

– аналітика: Google Analytics Management.

А також сервіси для швидкого відправлення електронної пошти, використання електронного підпису, карти Google Maps, стрімінгові платформи або Wikipedia.

Серед автоматичних режимів обміну даними поширені такі формати:

– текстові файли/CSV. Формат призначений для представлення даних як таблиць. Кожен рядок у них відповідає рядку тексту з одним або декількома полями, які розділені комами;

– XML/JSON [3]. Текстовий формат обміну даними. В основі його роботи лежить JavaScript. JSON – читабельний формат, застосовується у веб-додатках для обміну інформацією між браузером та сервером (AJAX). Також використовується для надсилання даних між серверами (HTTP-сполучення);

– проміжна база даних (MySQL/MS SQL). Підходить для малих та середніх додатків. MySQL/MS SQL використовується як сервер, до якого звертаються клієнти. За допомогою бібліотеки внутрішнього сервера проміжна база даних може включатися до автономних програм;

– web-сервіси (SOAP/REST). Протокол обміну структурованими повідомленнями використовується для передачі інформації у форматі XML. SOAP – розширена версія XML-RPC. Рішення призначене для роботи з протоколами прикладного рівня, такими як HTTP, HTTPS, SMTP та іншими;

– NoSQL-рішення. Призначені для створення масштабованого сховища даних. БД може використовуватися як система обробки інформації, яка надходить у режимі реального часу. Рішення забезпечує високу швидкість обміну даними між порталом та інтегрованою системою. Користувач може додавати нові вузли, масштабувати систему для збільшення її продуктивності.

За допомогою зовнішньої інтеграції можна значно розширити можливості сайту, покращити UX, автоматизувати обмін даними. Крім того, інтеграція із зовнішнім сервісом може стати елементом партнерського маркетингу.

1.2 Аналіз проблеми та постановка задачі

Календарі в програмах стали настільки звичними, що користувачі майже не помічають їх. Вони розглядають календар як невід'ємну частину програми. Однак додати календар до програми не так просто, як здається, з технічних причин.

При проектуванні додатку електронного розкладу ХНУРЕ виникла потреба у створенні унікального календаря розкладу, оскільки стандартний календар не підтримує багато функцій, наприклад повторення подій через один тиждень, що необхідно для двотижневого розкладу [4].

На основі цього виникла проблема інтеграції програми з різними системами, такими як Google Calendar, Android Calendar, Moodle.

Інтеграція розрізнених програмних рішень – одна з найпоширеніших проблем, із якими нині стикаються компанії. Більшість додатків не пов'язані один з одним, у результаті залишаються невеликі острівці даних.

На щастя, існує широкий спектр можливих рішень, коли справа доходить до кращої інтеграції інструментів, додатків та даних компанії.

Було сформовано завдання:

– проаналізувати предметну область;

– визначити цілі інтеграції додатків;

– проаналізувати проблему;

– провести аналіз способів інтеграції додатків;

– дослідити відмінності існуючих методів інтеграції;

– обрати технологію;

– дослідити способи інтеграції з Google Calendar;

– дослідити способи інтеграції з Android Calendar;

– дослідити способи інтеграції з Moodle;

– проаналізувати та реалізувати способи підключення обраних систем.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ

2.1 Аналіз методів інтеграції

На зміну архаїчним та незручним способам інтеграції прийшли сучасні технології, що використовують API (інтерфейси прикладного програмування) для зв'язку веб-додатків [5]. Розробники створюють свої інформаційні системи з API-інтерфейсами, щоб програми могли взаємодіяти та передавати дані один одному.

Application Programming Interface є пакетом функцій або правил, які дозволяють користувачеві посилатися і взаємодіяти з частинами програми для своїх власних цілей. Це забезпечує контроль і гнучкість для кінцевого користувача, водночас дозволяючи початковим розробникам підтримувати програму без перешкод. Інтеграція API дозволяє керувати процесами в багатьох секторах та рівнях організації, забезпечуючи синхронізацію даних, підвищуючи продуктивність та збільшуючи прибуток.

API – це інтерфейс, який є невід'ємною частиною практично всього в цифровому світі. Незалежно від бізнесу та розміру підприємства, API-інтерфейси забезпечують безперебійну роботу та продуктивність додатків та веб-систем.

Усередині API знаходяться протоколи, що дозволяють здійснювати інтеграцію додатків.

Хоча API в наш час найчастіше посилаються на веб-API, API вже давно використовуються з інтеграцією локальних програмних систем. Вони відіграли значну роль у SOA (сервіс-орієнтована архітектура), яка в основному проводилася локально. Це означає, що інтеграція додатків справді могла відбуватися лише зсередини. За допомогою Інтернету та розробки хмарних програм інтеграція програм найчастіше є зовнішнім процесом. Необов'язково мати програму, що зберігається на комп'ютері, щоб мати можливість доступу до програми API.

API для веб-застосунків стали дуже популярними і необхідними для різних завдань через збільшення кількості мобільних програм і хмарного програмного забезпечення.

Веб-API відрізняються з наступних причин:

– вони виставляються через Інтернет за допомогою протоколу HTTP або HTTPS;

– загальнодоступні веб-API доступні через Інтернет, як правило, із супровідною документацією;

– вони найчастіше будуються за допомогою JSON або XML.

Незважаючи на те, що веб-API мають спільні узгодженості, більшість веб-API будуть унікальними у тому сенсі, як можна їх використовувати. Ці відмінності залежать від того, як розробники програми вирішили створити відповідний API. Отже, конкретна інформація, необхідна для повного використання певного API, значною мірою залежить від документації.

Можливість підключення до API допомагає програмам обмінюватися даними та взаємодіяти один з одним без втручання людини. Забезпечується зв'язок між двома веб-інструментами або програмами через їх API. Це дозволяє організаціям автоматизувати системи, покращити безперешкодний обмін даними та інтегрувати поточні програми.

Підприємства неспроможні ігнорувати важливість інтеграції API у світі. Після стрімкого зростання кількості хмарних продуктів та додатків організаціям необхідно створити підключену систему, в якій дані автоматично передаватимуться між різними програмними інструментами. Інтеграція API робить це можливим, оскільки дозволяє обмінюватися даними процесів та підприємства між програмами в даній екосистемі.

Це відкриває новий рівень гнучкості надання інформації та послуг. Це також спрощує вбудовування контенту з різних сайтів та програм. API діє як інтерфейс, що дозволяє інтегрувати дві програми.

Існує кілька помітних переваг інтеграції API. До найпримітніших відносяться:

– автоматизація. Інтеграція API дозволяє автоматично передавати інформацію та дані від однієї програми до іншої. Успішна автоматизація допомагає

усунути ручний (людський) компонент, що заощаджує час і значно знижує кількість помилок;

– масштабованість. Інтеграція API дозволяє компаніям зростати, оскільки їм не потрібно починати з нуля під час створення підключених систем та додатків;

– оптимізована прозорість/комунікація/звітність. Інтеграція API забезпечує повну видимість всіх систем та процесів для покращення взаємодії та звітності. Завдяки оптимізованому підходу можна ефективно відстежувати та контролювати дані, створюючи надійні звіти на основі конкретних та всеосяжних наборів даних;

– зменшує кількість помилок. Інтеграція API дозволяє передавати складні та об'ємні дані з меншою кількістю помилок та невідповідностей.

Існують два основних стилі API – SOAP і REST, вони мають різні архітектури, але здебільшого використовують загальний транспорт – HTTP-протокол.

SOAP та REST вирішують однакову задачу: дозволяють розробникам за допомогою API налаштовувати обмін даними між додатками. Але якщо SOAP є протоколом обміну інформацією, то REST – це стиль або набір рекомендацій, яким повинен слідувати розробник для надання веб-сервісу – RESTful, тобто розробленого з урахуванням вимог REST і не порушуючого обмежень, які він накладає.

SOAP (Simple Object Access Protocol) - відмінно стандартизований протокол, що давно використовується [6]. Це одна з причин, через яку його вибирають як API корпоративних додатків. Він працює поверх протоколів HTTP, SMTP, TCP або UDP, але передає дані лише у форматі XML. Для застарілих систем і тих, які виробляють складні транзакції, а також висувають високі вимоги до безпеки, SOAP все ще хороший варіант. Він широко застосовується у банківських та інших фінансових додатках, CRM, комунальними службами та при наданні телекомунікаційних послуг. Там, де важлива стабільність і цілісність даних, використовують SOAP, наприклад, робота світлофорів, каналізації та електропостачання міста має завжди виконуватися безвідмовно та передбачувано.

Можливість асинхронної передачі даних SMTP робить цей протокол незамінним для інтеграції в системах з нестабільним каналом зв'язку.

На рисунку 2.1 зображено схему роботи SOAP та REST технологій.

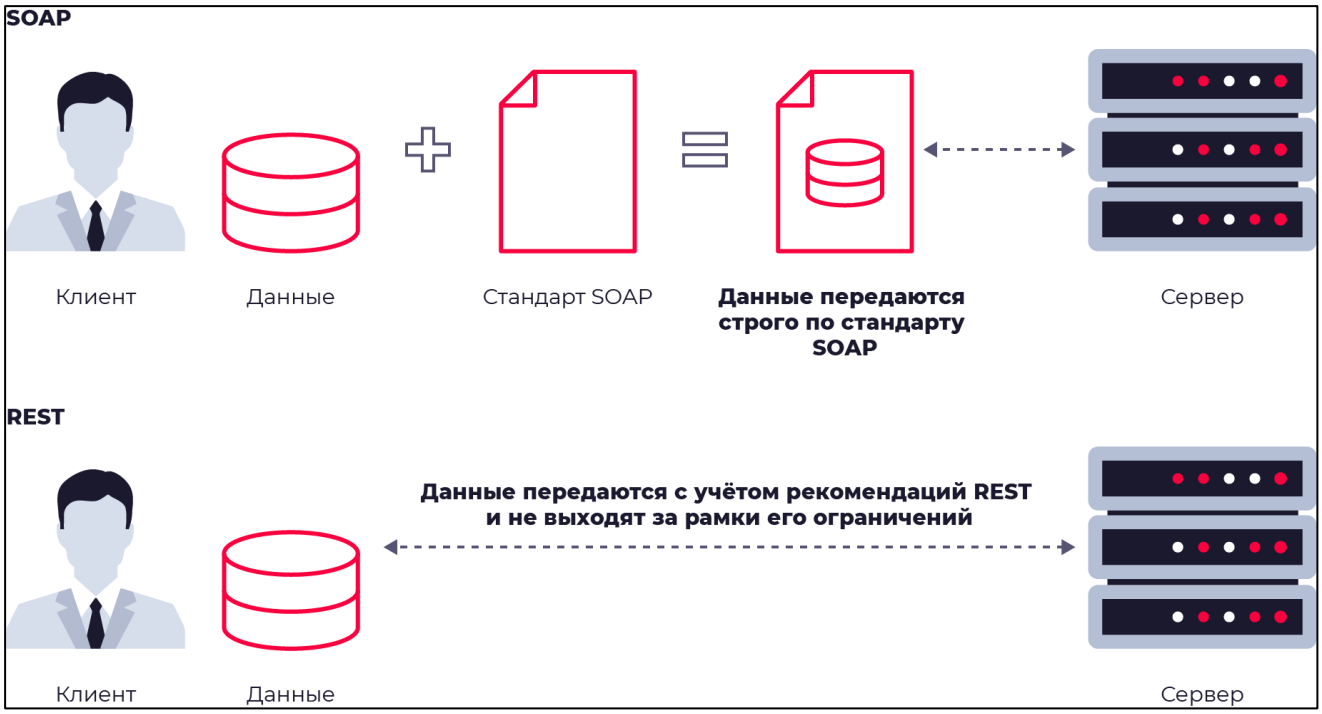

Рисунок 2.1 – SOAP – протокол, REST – архітектурний стиль

REST (REpresentational State Transfer) – досить молодий, але дуже популярний архітектурний стиль для створення інтеграційних API [7]. Він набув популярності у розробників у 2018 році, і зараз більшість інтернет-сервісів його використовують як загальнодоступний API-інтерфейс. Twitter, WordPress, Google Maps та інші відомі сервіси мають REST API для взаємодії з іншими веб-сервісами та сайтами користувача.

Для обміну даними REST використовує лише HTTP як транспортний протокол, але формати повідомлень можуть бути будь-якими – HTML, JSON, XML, YAML або простий текст. Універсальний формат JSON (JavaScript Object Notation): його легко аналізувати, у нього простий синтаксис і він не залежить від мови програмування. У JSON використовується менше слів, його простіше писати та

читати, такі повідомлення мають меншу вагу, тому швидкість їх передачі вища, ніж із XML.

Нижче наведено таблицю порівняння веб-сервісів REST та SOAP.

Таблиця 2.1 – Порівняння веб-сервісів REST та SOAP

	Веб-сервіси REST	Веб-сервіси SOAP
Скорочення	Representational State Transfer – передача репрезентативного стану	Simple Object Access Protocol – простий протокол доступу до об'єктів
Стандарт	Немає стандарту	Декларативний (використовується стандартний WSDL)
Служба підтримки	Численні типи контенту	Тільки XML
Надійність	Специфічний додаток	WS – Надійний обмін повідомленнями
Кешування	Операції Get можуть бути кешовані за бажанням	Немає
Розмір повідомлення	Полегшений	Порівняно тяжкий
Реалізація	Проста	Складна
Області застосування	Обмежена пропускна спроможність та ресурси Повністю операції без стану Кешування ситуацій	Асинхронна обробка та виклик Формальні контракти Операції зі станом
Вид розробника	Ресурсно-орієнтований	Об'єктно-орієнтований
Підтримка транспортного протоколу	HTTP	HTTP, SMTP, JMS

REST – простий, зручний та універсальний спосіб інтеграції корпоративних програм, в більшості випадків веб-сервіси RESTful можуть взаємодіяти з будь-якими іншими сервісами.

Щоб створити REST API, необхідно дотримуватися шести архітектурних обмежень:

– уніфікований інтерфейс – запити від різних клієнтів мають виглядати однаково, наприклад, той самий ресурс не повинен мати більше одного URI;

– розділення клієнт-сервер – клієнт та сервер повинні діяти незалежно один від одного. Вони повинні взаємодіяти один з одним лише через запити та відповіді;

– безгромадянство – не повинно бути жодних сеансів на стороні сервера. Кожен запит повинен мати всю інформацію, яку сервер повинен знати;

– Кешировані ресурси – відповіді сервера повинні містити інформацію про те, чи кешуються дані, що надсилаються ними, чи ні. Кешировані ресурси повинні надходити з номером версії, щоб клієнт міг не вимагати ті самі дані більше одного разу;

– багаторівнева система. Між клієнтом та сервером, який повертає відповідь, може бути кілька рівнів серверів. Це не повинно впливати ні на запит, ні на відповідь;

– код за запитом – якщо це необхідно, відповідь може містити код, що виконується (наприклад, JavaScript у відповіді HTML), який може виконати клієнт.

Оскільки SOAP є офіційним протоколом, він поставляється зі строгими правилами та розширеними функціями безпеки, такими як вбудована сумісність з ACID та авторизація. Вища складність потребує більшої пропускної спроможності та ресурсів, що може призвести до зниження часу завантаження сторінки. REST був створений для вирішення проблем SOAP. Тому в нього гнучкіша архітектура. Він складається лише з простих рекомендацій та дозволяє розробникам реалізовувати рекомендації по-своєму. Він дозволяє різні формати повідомлень, такі як HTML, JSON, XML і простий текст, в той час як SOAP дозволяє тільки XML. REST також є легшою архітектурою, тому веб-сервіси RESTful мають більш високу продуктивність. Через це він став неймовірно популярним в епоху мобільних

пристроїв, де навіть кілька секунд мають велике значення (як за час завантаження сторінки, так і за доходами).

REST працює швидше, а технологія RESTful-сервісів простіше. SOAP взаємодіє з операціями, тому найкраще підходить для реалізації транзакцій та складної логіки. Крім того, SOAP може працювати з будь-яким протоколом транспортного рівня замість HTTP і використовується у більшості застарілих інформаційних систем, з якими може знадобитися інтеграція.

2.2 Способи інтеграції з Google Calendar

Google Calendar – онлайн-сервіс для планування зустрічей та подій, організації робочого часу. Google Calendar допомагає командам організувати робочі процеси та планувати майбутні заходи. Користуватися календарем можна у браузері та з мобільних пристроїв на iOS та Android. Календар Google має синхронізацію з іншими сервісами корпорації.

Особливості Google Календаря:

– створення та відстеження подій;

– декілька видів перегляду (рік, місяць, тиждень, день);

– імпорт файлів календаря з Microsoft Outlook;

– публікація календарів на сайті;

– загальні календарі;

– інтеграція з хмарними сервісами;

– вкладення файлів та посилань у події.

Компанії та окремі команди для спільної роботи можуть створювати групові календарі, де відображатимуться спільні події та завдання. За допомогою інтеграції Календаря Google та інших сервісів компанії можна оптимізувати робочі процеси. Наприклад, можна створювати події з листів Gmail або з календаря переходити в

призначену конференцію в Google Meet. Також Google Календар має інтеграцію з великою кількістю сервісів, які можуть збільшити його функціональність.

API Календаря Google дозволяє розробникам додавати повні дані календаря та функції у свою програму за допомогою інтерфейсу REST або однієї з клієнтських бібліотек, які Google пропонує для таких мов, таких як Java, Python, PHP, JavaScript та інші.

2.2.1 Підключення облікового запису Google до програми

Перш ніж отримати доступ до облікового запису Календаря Google, необхідно автентифікувати обліковий запис з відповідними дозволами. Всі API Google використовують OAuth 2.0 для автентифікації та авторизації облікового запису, який встановлює процес входу в систему, в якому програма узгоджує з платформою Google Identity Platform токен доступу для облікових записів користувачів. Цей токен надає обмежений доступ до ресурсів користувача, на які користувач погодився під час процесу аутентифікації.

2.2.2 Календарі та події Google

Основними організаційними компонентами Google Calendar API є календарі та події.

Календарі – це набір пов'язаних подій, які включають відповідні метадані, такі як ім'я, часовий пояс тощо. Цей календар не може бути видалений користувачем, але він може бути доступний іншим користувачам. Користувачі також можуть створювати будь-яку кількість додаткових календарів, і вони можуть

бути змінені, видалені та надані іншим користувачам. Приклади додаткових календарів:

– особистий та робочий календарі;

– календарі, що належать іншим людям в організації Workspace for EDU користувача;

– загальні календарі, у подіях яких можуть брати участь кілька осіб із однієї команди чи компанії;

– користувальницькі календарі, створені користувачем для відстеження певних типів подій.

API Календаря Google також включає об'єкт CalendarList: набір усіх календарів, на які підписан користувач. Необхідно використовувати кінцеву точку CalendarList для отримання доступу до властивостей календаря, що залежать від користувача, таких як нагадування за замовчуванням та кольори фону/переднього плану. Необхідно використовувати об'єкт Calendar, коли потрібно створювати та видаляти календарі, або встановлювати глобальні властивості, які є спільними для всіх користувачів з доступом до календаря, наприклад, заголовок та часовий пояс за замовчуванням.

Події – це об'єкти, які пов'язані з певним часом або діапазоном дат і включають таку інформацію, як заголовок, опис, місцезнаходження, вкладення та учасники, і вони можуть бути налаштовані на певний період часу або на цілий день. Кожна подія в Календарі Google пов'язана з одним або кількома календарями, і всі події мають організатора з доступом до первинної копії події та будь-якої кількості учасників, які мають власні вторинні копії події. Кожна з цих копій подій матиме власні унікальні ідентифікатори для кожного календаря, в якому вона відображається, але всі вони будуть мати однаковий ідентифікатор iCal .

2.2.3 Аналіз події за допомогою API Календаря Google

API Календаря Google дозволяє аналізувати події лише з одного календаря за один раз. Щоб проаналізувати всі події в певному календарі, необхідно використовувати функцію events.list(), вказавши ідентифікатор календаря, з якого необхідно повернути події. Щоб отримати детальну інформацію про конкретну подію, необхідно використовувати метод events.get(), вказавши як календар, так і ідентифікатори події.

2.2.4 Створення та редагування події

Перед створенням події за допомогою Google Calendar API необхідно переконатися, що обліковий запис має відповідні області OAuth для редагування подій календаря. Також необхідно переконатися, що користувач має право на запис у календар, в який необхідно додати подію. Для цього необхідно перевірити поле accessRole для календаря за допомогою calendarList.get().

Щоб створити нову подію, необхідно використовувати метод events.insert(), передаючи щонайменше відповідний ідентифікатор календаря, а також час початку та закінчення. Також можна передати будь-яку кількість інших полів, які надає ресурс подій, заголовок, опис, учасників, розташування, ресурси і т.д.

Якщо потрібно змінити подію, необхідно скористатися методом events.update(), передавши відповідний ідентифікатор події. За допомогою цього методу можна змінити усі поля, крім ідентифікатора.

2.2.5 Виявлення вільного/зайнятого часу

Кінцева точка Freebusy API Календаря Google дозволяє бачити, коли користувач вільний або зайнятий, залежно від того, чи існують події протягом зазначеного періоду часу. Це особлива форма доступу для читання, яку можуть мати облікові записи Google, яка показує лише час початку та закінчення подій, коли користувач позначений як зайнятий, і не надає жодних інших даних про події. Одна з основних переваг виявлення вільного/зайнятого часу – це можливість порівнювати кілька календарів та визначати часові інтервали, в яких обидва календарі доступні, без розкриття будь-якої інформації про події у календарях.

2.2.6 Обробка повторюваних подій за допомогою Google Calendar API

Календар Google має два типи подій: одиничні та повторювані. Поодинокі події відбуваються один раз, а повторювані можуть повторюватися безліччю різних способів. При управлінні подіями, що повторюються, важливо знати, як вони представлені всередині. При створенні повторення створюються два різні типи подій. Перший – це подія, яка представляє серію і зберігає як поле повторення, як визначено RFC 5545, так і вихідний час початку і закінчення. Поле повторення визначає такі речі, як частота, з якою подія буде повторюватися (щодня, щотижня, щомісяця і т. д.), інтервал (щодня, раз на два тижні і т. д.), кількість разів, коли подія повинна повторюватися, та дні тижня, для яких подія має повторюватися.

2.2.7 Доступ до подій, що повторюються

За умовчанням API Календаря Google повертає не всі екземпляри події, що повторюється, а тільки головну подію. Якщо необхідно повернути всі екземпляри, що повторюються, необхідно встановити для параметра singleEvents значення true. Якщо необхідно отримати всі екземпляри подій, що повторюються для певної основної події, необхідно використовувати метод events.instances(), надаючи ідентифікатор основної події.

2.2.8 Створення і редагування повторюваної події

Процес створення та редагування повторюваної події за допомогою Google Calendar API ідентичний процесу створення будь-якої іншої події. Винятком є те, що також необхідно встановити рядок RRULE у полі повторення. У Google Calendar можна створити не більше 730 екземплярів повторюваної події.

2.2.9 Події як об'єкти першого класу

Можливо робити запити GET до ресурсів подій з API Календаря Google лише як об’єкти другого класу, при цьому ідентифікатор календаря є об’єктом першого класу. Це означає, що необхідно завжди знати, в якому календарі існує подія, перш ніж вимагати її, і не можна шукати події в декількох календарях. API Календаря Google компенсує це, дозволяючи отримати доступ до основного календаря користувача, передавши "первинний" як ідентифікатор.

2.2.10 Різниця між оновленням та редагуванням

Метод оновлення для подій використовує HTTP PUT, тобто він повністю замінює наявну подію на нову подію, яка має відповідні деталі, за винятком ідентифікатора події та значень, які ви вказуєте в методі оновлення. Якщо потрібно відслідковувати подію після оновлення за допомогою цього методу, необхідно враховувати той факт, що ідентифікатор події зміниться. Якщо необхідно зберегти існуючий ідентифікатор події, потрібно замість цього використовувати метод patch, який відповідає угодам HTTP PATCH.

2.2.11 Переміщення подій між календарями

Import та Move методи пов'язані з відправкою події з одного календаря в інший, але вони роблять це по-різному. Імпорт більш точно слід називати дублікатом, оскільки він не створює зв'язку між вихідною подією та скопійованою подією. Import очікує, що ідентифікатор календаря буде передано як параметр шляху, а відомості про подію будуть передані в тілі.

З іншого боку, Move змінює організатора події, що фактично видаляє подію з вихідного календаря. Для цього потрібно передати ідентифікатор календаря та ідентифікатор події як параметри шляху, але також потрібно передати ідентифікатор календаря призначення як параметр запиту призначення.

2.3 Способи інтеграції з Android Calendar

2.3.1 Запит про дозвіл на використання календаря

Нижче наведено програмну реалізацію додавання дозволу:

```
     <uses-permission android:name="android.permission.READ_CALENDAR">
</uses-permission>
     <uses-permission android:name="android.permission.WRITE_CALENDAR">
</uses-permission>
```

2.3.2 Уніфікований ідентифікатор ресурсу

Для користування календарем Android необхідно звернутися до його контенту провайдера за певним URI. URI відрізняється у різних версіях ОС.

Для Froyo [2.2] – content://com.android.calendar/calendars.

Для версій <2.2 – content://calendar/calendars.

Нижче наведено програмну реалізацію визначення версії системи та використання відповідного URI:

```
     // Calendar URI before 2.2
     Uri calendars = Uri.parse("content://calendar/calendars");
     Cursor managedCursor = this.managedQuery(calendars, new String[] {
"_id", "name" }, null, null, null);
     if(managedCursor == null || managedCursor.getCount() == 0)
     {
          // 2.2 (Froyo)
     }
     else
     {
          // < 2.2
     }
```

2.3.3 Отримання списку календарів

Загальний календар складається з одного або декількох дрібніших календарів, які можуть відрізнятися за кольором маркерів. Зазвичай вони мають назви "особистий", "робота", "навчання" тощо. Створювати такі календарі можна тільки на сайті Google Calendar, в Android не можна це зробити стандартними засобами.

Для додавання подій потрібно знати ID календаря. Нижче наведено програмну реалізацію використання конструкції з попереднього пункту:

```
     String[] projection = new String[] { "_id", "name" };
     Cursor managedCursor = this.managedQuery(URI, projection, null, null,
null);
     if (managedCursor != null && managedCursor.moveToFirst())
     {
         String calName;
         String calID;
         int nameColumn = managedCursor.getColumnIndex("name");
         int idColumn = managedCursor.getColumnIndex("_id");
         do
         {
             calName = managedCursor.getString(nameColumn);
             calID = managedCursor.getString(idColumn);
             if (calName != null) // … UI
         } while (managedCursor.moveToNext());
         managedCursor.close();
      }
```

Отриманий набір імен можна вивести в інтерфейс та дозволити користувачеві вибрати.

2.3.4 Додавання події

Робота з календарем відбувається, як і з будь-яким іншим контентом провайдером.

Нижче наведено програмну реалізацію роботи з календарем:

```
ContentValues event = new ContentValues();
event.put("calendar_id", calID);
event.put("title", title);
event.put("description", desc);
event.put("eventLocation", loc);
event.put("dtstart", start);
event.put("dtend", end);
eventsURI=Uri.parse(URI + "/events");
this.getContentResolver().insert(eventsURI, event);
```

Поле опису слід використовувати для індикації події. Використовуючи це поле, можна буде видалити всі події, створені програмою. Поля назви, опису та місця проведення є рядковими. Поля часу мають тип long. Нижче наведено програмну реалізацію генерації значень цих полів:

```
     public static long pickDate(int year, int month, int day, int hour,
int minute)
     {
           Calendar rightNow = Calendar.getInstance();
           int timeShift = rightNow.get(Calendar.ZONE_OFFSET);
           return Date.UTC(year - 1900, month, day, hour - (timeShift /
3600000), minute, 0);
     }
```

2.3.5 Видалення події

Щоб видалити повідомлення за ID, необхідно додати його до URI. Нижче наведено програмну реалізацію видалення повідомлення за ID:

```
//eventsURI = URI + events/
uri = ContentUris.withAppendedId(eventsURI, EVENT_ID);
this.getContentResolver().delete(uri, null, null);
```

Для видалення за іншими полями необхідно курсором вибрати всі події, а потім видалити вибірково за ID.

2.4 Порівняння способів інтеграції з календарями

Відсутність інтеграції. Цей метод припускає, що користувачі переглядатимуть події лише у даному застосунку. Це надає максимальну гнучкість у відображанні подій та логіці взаємодії з ними. Недоліками є: необхідність повторної реалізації функціоналу, що вже є в інших календарях, та незручність для користувача, бо йому потрібно вивчати новий застосунок, та дивитися свої події у різних місцях.

Інтеграція з Android Calendar. Цей метод припускає зберігання подій у локальному календарі на телефоні (або планшеті). Це дозволяє користувачу зберігати та переглядати усі свої події в одному місці, та розробнику потрібно буде реалізовувати менше дублюючого функціоналу у додатку (наприклад, повідомлення про початок події). Також зручно, що події будуть зберігатися у окремому календарі, тому їх буде легко відокремити від усіх інших. Недоліками є: відсутність синхронізації цих подій з іншими пристроями користувача, наприклад, з комп’ютером.

Інтеграція з Google Calendar. Цей метод майже такий самий як попередній, за вийнятком, що дозволяє синхронізувати події, але вони не будуть відокремлені від інших.

У кожного разглянутого метода є унікальні преваги та недоліки, тож не можна обрати лише один з них.

Оптімальним рішенням буде сумістити усі підходи використавши сильны сторони кожного з них.

2.5 Способи інтеграції з Moodle

Moodle – це система управління навчанням з відкритим вихідним кодом, розроблена для навчання на основі соціальних конструкційних теорій, яка розміщується на власному хостингу. Moodle є найбільш широко використовуваною системою управління навчанням в освіті [8].

Викладачі, адміністратори та студенти можуть скористатися цією модульною системою PHP, яка спрямована на персоналізацію навчальних середовищ. Для інтеграції в програми LMS Moodle пропонує основні API, такі як API для доступу, керування даними та навігації, а також понад 20 загальних API, які включають API календаря, реєстрації та медіа. Moodle підтримує інтеграцію через свої основні API та стандарт LTI.

Як і інші LMS, Moodle має API, до яких можна отримати доступ через інтегрований додаток сторонніх розробників. Однак доступ до основних API Moodle можна отримати лише за допомогою встановлення плагіна.

Moodle вимагає від розробників програм написати плагін, який дозволить програмі взаємодіяти з середовищем Moodle. Таким чином, плагін повинен бути написаний на PHP і вручну встановлений адміністратором Moodle в екземплярі Moodle.

Якщо адміністратор не здійснює регулярні оновлення, це може вплинути на сумісність програми з LMS. Це може викликати проблеми, якщо встановлене середовище Moodle не підтримує найновішу версію LTI або набір основних API.

Інтеграція LMS зазвичай не така проста, як може здатися на перший погляд. Насправді, потрібно враховувати відмінності та особливості кожної платформи, перш ніж створювати власні інтеграції.

Інфраструктура веб-сервісів:

– клієнт надсилає ім'я користувача та пароль сценарію входу до веб-служби;

– сценарій повертає токен для цього облікового запису користувача;

– клієнт викликає певну функцію веб-служби на сервері протоколу, включаючи токен;

– сервер протоколу використовує токен, щоб перевірити, чи може викликати функцію;

– сервер протоколу викликає відповідну зовнішню функцію, розташовану в просторі імен \component\external (раніше у файлі externallib.php всередині відповідного модуля);

– зовнішня функція перевіряє, чи поточний користувач має можливість виконати цю операцію;

– зовнішня функція викликає відповідну базову функцію Moodle (зазвичай lib.php);

– основна функція може повертати результати зовнішньої функції;

– зовнішня функція поверне результат до сервера протоколу;

– сервер протоколу повертає результат клієнту.

2.6 Методи веб-скрейпінгу

Часто у вебмайстра, маркетолога або SEO-фахівця виникає необхідність отримати дані зі сторінок сайтів і відобразити їх у зручному вигляді для подальшої обробки. Це може бути парсинг цін в інтернет-магазині, отримання кількості лайків або вилучення вмісту відгуків з ресурсів.

За замовчуванням більшість програм технічного аудиту сайтів збирають лише вміст заголовків H1 і H2, проте, якщо, наприклад, потрібно зібрати заголовки H5, то їх вже потрібно буде отримувати окремо. І щоб уникнути рутинної ручної роботи з парсингу та вилучення даних з HTML-коду сторінок – зазвичай використовують веб-скрапери.

Веб-скрейпінг – це автоматизований процес вилучення даних з сторінок сайту, що цікавлять, за певними правилами [9].

На рисунку 2.3 зображено технологію веб-скрейпінгу.

Рисунок 2.3 – Технологія веб-скрейпінгу

Можливі сфери застосування веб-скрейпінгу:

– відстеження ціни товарів в інтернет-магазинах;

– вилучення описів товарів та послуг, отримання кількості товарів та картинок у лістингу;

– вилучення контактної інформації (адреси електронної пошти, телефони тощо);

– збір даних для маркетингових досліджень (лайки, шери, оцінки у рейтингах);

– вилучення специфічних даних із коду HTML-сторінок (пошук систем аналітики, перевірка наявності мікророзмітки);

– моніторинг оголошень.

Основними способами веб-скрейпінгу є методи розбирання даних, використовуючи XPath, CSS-селектори, XQuery, RegExp та HTML templates.

XPath є спеціальною мовою запитів до елементів документа формату XML/XHTML [10]. Для доступу до елементів XPath використовує навігацію по DOM, описуючи шлях до потрібного елемента на сторінці. З його допомогою можна отримати значення елемента за його порядковим номером у документі, вийняти текстовий вміст або внутрішній код, перевірити наявність певного елемента на сторінці.

CSS-селектори використовуються для пошуку елемента його частини (атрибут). CSS синтаксично схожий на XPath, при цьому в деяких випадках CSS-

локатори працюють швидше і описуються наочно і коротко. Мінусом CSS є те, що він працює лише в одному напрямку – углиб документа. XPath працює в обидві сторони (наприклад, можна шукати батьківський елемент по дочірньому).

XQuery має в якості основи мову XPath. XQuery імітує XML, що дозволяє створювати вкладені вирази таким чином, який неможливий в XSLT.

RegExp – формальна мова пошуку для отримання значень з безлічі текстових рядків, що відповідають необхідним умовам (регулярному виразу) [11].

HTML templates – мова вилучення даних з HTML документів, яка є комбінацією HTML-розмітки для опису шаблону пошуку потрібного фрагмента плюс функції та операції для вилучення та перетворення даних.

Зазвичай за допомогою веб-скрейпінгу вирішуються завдання, з якими важко впоратися вручну. Це може бути веб-скрейпінг описів товарів при створенні нового інтернет-магазину, скрейпінг у маркетингових дослідженнях для моніторингу цін або моніторингу оголошень (наприклад, з продажу квартир). Для завдань SEO-оптимізації зазвичай використовуються вузько спеціалізовані інструменти, в яких вже вбудовані веб-скрейпінги з усіма необхідними налаштуваннями вилучення основних параметрів SEO.

Незважаючи на те, що метод веб-скрейпінгу все ще розвивається, він віддає перевагу більш практичним рішенням, які базуються на вже існуючих програмах і технологіях, на відміну від його більш амбітних аналогів, які вимагають більш складних проривів і знань для роботи.

Існують такі методи веб-скрейпінгу:

– COPY-PASTING. Під час ручного скрейпінгу людина копіює та вставляє веб-вміст. Це займає багато часу, потребує великої кількості повторень і вимагає більш ефективних засобів веб-скрейпінгу. Однак він дуже ефективний, оскільки захист веб-сайту націлений на автоматичний скрейпінг, а не на ручні методи скрейпінгу. Навіть маючи цю перевагу, ручний скрейпінг майже не виконується, оскільки це займає багато часу, а автоматичний скрейпінг швидший та дешевший.

– HTML PARSING. HTML parsing виконується за допомогою JavaScript і націлений на лінійні або вкладені сторінки HTML. Це швидкий і надійний метод,

який використовується для вилучення тексту, скрейпінгу екрана та вилучення ресурсів серед інших.

– DOM PARSING. DOM – це скорочення від Document Object Model – об’єктної моделі документа, яке визначає стильову структуру та вміст файлів XML. Скрепери використовують парсери DOM, щоб отримати поглиблене уявлення про структуру веб-сторінки. Вони також можуть використовувати парсер DOM, щоб отримати вузли, що містять інформацію, а потім використовувати такий інструмент, як XPath, для очищення веб-сторінок. Браузери Internet Explorer або Firefox можуть бути вбудовані для вилучення всієї веб-сторінки або окремих її частин.

– VERTICAL AGGREGATION. Платформи вертикальної агрегації створюються компаніями, які мають доступ до великомасштабної обчислювальної потужності для націлювання на конкретні вертикалі. Деякі компанії використовують платформи в хмарі. Створення та моніторинг ботів для певних вертикалей здійснюються цими платформами без будь-якого втручання людини. Якість ботів вимірюється на основі якості даних, які вони отримують, оскільки вони створюються на основі бази знань для конкретної вертикалі.

– XPATH. XML Path Language – це мова запитів, яка використовується з документами XML. XPath можна використовувати для навігації по XML-документах через їх деревоподібну структуру, вибираючи вузли на основі різних параметрів. XPath можна використовувати разом із аналізом DOM, щоб очистити всю веб-сторінку.

– GOOGLE SHEETS. Таблиці Google – це інструмент для веб-скрейпінгу, який досить популярний серед веб-скреперів. Усередині таблиць скрепер може використовувати функцію IMPORT XML (,), щоб отримати необхідну кількість даних з веб-сайтів. Цей метод корисний лише тоді, коли на веб-сайті потрібні конкретні дані або шаблони. Також можна використовувати цю команду, щоб перевірити, чи веб-сайт захищений від скрейпінгу.

– TEXT PATTERN MATCHING. Це метод узгодження, який передбачає використання команди UNIX grep і використовується з популярними мовами програмування, такими як Perl або Python.

Техніка веб-скрейпінгу передбачає використання інструментів і послуг, які можна легко отримати в Інтернеті. Для того, щоб володіти веб-скрейпінгом, потрібно знати всі техніки. Інструменти й послуги автоматичного веб-скрейпінгу включають limeproxies, cURL, Wget, HTTrack, Import.io, Node.js та список інших. Для цілей скрейпінгу скрепери зазвичай використовують браузери, такі як Phantom.js, Slimmer.js і Casper.js.

Технології веб-скрейпінгу:

– SELENIUM. Selenium – це інструмент автоматизації веб-браузера, який дозволяє робити багато попередньо налаштованих речей, як і з використанням бота. Використання Selenium допоможе дізнатися, як працюють веб-сайти. Його використання може імітувати звичайне відвідування людиною сторінки за допомогою звичайного веб-браузера, і це дозволить отримати точні дані. Часто він також використовується для емуляції викликів ajax у веб-скрейпінгу.

Будучи потужним інструментом автоматизації, він може дати більше, ніж просто можливість виконувати веб-скрейпінг. Також можна тестувати веб-сайти та автоматизувати будь-які тривалі дії в Інтернеті.

– BOILERPIPE. Коли потрібно витягти текст разом із пов’язаними заголовками, можна використовувати Boilerpipe. Boilerpipe – це бібліотека Java, яка створена для вилучення даних з Інтернету, будь то структуровані чи неструктуровані дані. Він усуває небажані теги HTML та інші шуми на веб-сторінках, залишаючи чистий текст. Приваблива особливість цієї технології полягає в тому, що вона виконує швидкий скрейпінгу веб-сторінок з невеликим внеском з боку користувача. Швидкість не впливає на точність даних, оскільки вона має високу точність, і все це робить його одним із найпростіших доступних інструментів для скрейпінгу.

– NUTCH. Коли згадується тема «золотого стандарту» технології веб-скрейпінгу, Nutch є одним із найкращих варіантів, які будуть представлені. Це веб-

сканер з відкритим кодом, який витягує дані з веб-сторінок з блискавичною швидкістю. Nutch сканує, витягує та зберігає дані, коли вони були запрограмовані. Завдяки його потужному алгоритму він виділяється як один із найкращих доступних інструментів для веб-скрейпінгу.

Для того щоб виконувати скрейпінг з Nutch, веб-сторінки мають бути закодовані в Nutch вручну. Як тільки це буде зроблено, він просканує сторінки, отримає необхідні дані та збереже їх на сервері.

– WATIR. Watir – це сімейство бібліотек Ruby з відкритим вихідним кодом, яке є хорошим вибором для автоматизації веб-браузера, оскільки він простий у використанні та дуже гнучкий. Однією з причин, чому він ідеально підходить для веб-скрейпінгу, є те, що він взаємодіє з веб-браузерами так само, як і люди.

Його можна використовувати для натискання посилань, заповнення форм, натискання кнопок і всього, що може уявити людина на веб-сторінці. Ruby робить використання Watir дуже приємним і простим, оскільки Ruby, як і інші мови програмування, дає можливість читати файли даних, експортувати XML, підключатися до баз даних і писати електронні таблиці.

– CELERITY. Це обгортка JRuby, створена навколо HtmlUnit. Його API простий у використанні і дозволяє легко переміщатися по веб-програмам. Він працює з дуже вражаючою швидкістю, оскільки не вимагає багато часу для візуалізації графічного інтерфейсу або непотрібних завантажень. Оскільки він є масштабованим і ненав’язливим, він може працювати у фоновому режимі без звуку після початкового налаштування. Celerity – це хороший інструмент для автоматизації браузера, який можна використовувати для ефективного скрейпінгу.

Веб-скрейпінг – це практика, яка з кожним днем стає все популярнішою, оскільки вона дає доступ до даних, корисних для власників бізнесу та інших подібних. Це непросте завдання, тому щоб ефективно виконати його та отримати надійні дані, потрібно використовувати найкращі методи та інструменти веб-скрейпінгу.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

3.1 Взаємодія з університетською системою автоматизації або ЦІСТ

Проаналізувавши різноманітні сервіси, що надають дані учням та викладачам ХНУРЕ, особливості взаємодії з ними, потреби учнів та викладачів, та існуючі застосунки, що групують ці дані, було розроблено новий застосунок. Були додані інтеграції з новими сервісами та нові функції, які задовольняють вимоги користувачів та роблять користування додатком зручним.

Під час розробки додатку API для отримання розкладу було відключене через перенавантаження, що призвело до припинення роботи усіх додатків-аналогів. Щоб уникнути такої залежності від API та забезпечити коректну роботу додатку навіть при великому навантажені на ЦІСТ, був розроблений алгоритм для отримання списку груп та викладачів з HTML сторінок сайту cist.nure.ua та розкладу завдяки комбінуванню даних, що надаються у CSV та XLS виглядах.

На рисунку 3.1 зображено доступні альтернативні методи отримання розкладу.

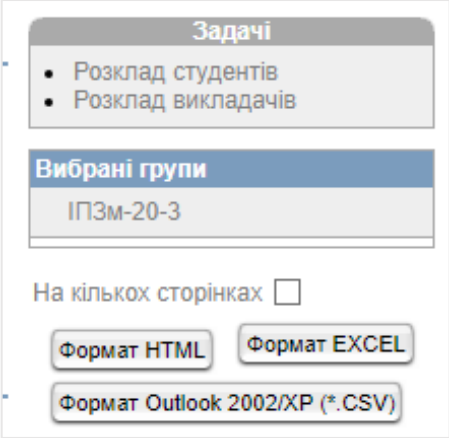

Рисунок 3.1 – Альтернативні методи отримання розкладу

Нижче наведено частину програмної реалізації алгоритму розпізнавання інформації про заняття групи з csv файлу, отриманого з ЦІСТ, та конвертування її

у вигляд, що буде зрозумілий для додатку. Завдяки цьому отримується інформація про назву предмета, аудиторію, тип заняття, його дату та час проведення.

```
string[] rawEvent = eventStr
    .Split(new string[] { "\",\"" })
    .Select(str => str.Trim('"'))
    .ToArray();

string[] concurrentEventsList = rawEvent[0].Split(new[] { "; " });

string groupName = defaultKey;
if (isManyGroups)
{
    groupName =
concurrentEventsList[0].Remove(concurrentEventsList[0].IndexOf(' '));
    concurrentEventsList[0] =
concurrentEventsList[0][(concurrentEventsList[0].IndexOf(" - ") + 3)..];
}

// Checking for lessons with spaces
int typeIndex = -1;
for (int i = eventDescription.Count - 1; i >= 0; i--)
{
    if
(KnownEventTypes.Values.Contains(eventDescription[i].ToLower()))
    {
        typeIndex = i;
        break;
    }
}
while (typeIndex > 1)
{
    eventDescription[0] += $" {eventDescription[1]}";
    eventDescription.RemoveAt(1);
    typeIndex--;
}

// Checking for multiple rooms
while (eventDescription[2].EndsWith(",") && eventDescription.Count >
2)
{
    eventDescription[2] += $" {eventDescription[3]}";
    eventDescription.RemoveAt(3);
}
```

Такий підхід виявився досить успішним, але у процесі дослідження виявилося, що він не надає таких даних як ім’я викладача та повну назву предмета (наявна лише коротка). Для вирішення цієї проблеми був розроблений алгоритм

для розпізнавання цих даних з розкладу у xlsx форматі (де вони наявні), та об’єднання їх з даними, отриманими з csv. Нижче наведено реалізацію алгоритму.

```
    List<string> timetableInfoRaw = cistXlsTimetableData
        .Split(new string[] { @"ss:Type=""String"">" })
        .Skip(1).ToList();

    List<string> infoRaw = timetableInfoRaw[i + 1]
        .Remove(timetableInfoRaw[i + 1].IndexOf("</Data>"))
        .Split(new string[] { ": ", "\n" }).ToList();

    string lessonShortName =
timetableInfoRaw[i].Remove(timetableInfoRaw[i].IndexOf("</Data>"));
    string lessonLongName = infoRaw[0].Trim();
    foreach (string groupName in groupsLessons.Keys)
    {
        groupsLessons[groupName].Add(new LessonInfo
        {
            ShortName = lessonShortName,
            LongName = lessonLongName
        });
    }

    List<string> eventTypeInfo = eventTypeInfoRaw
        .Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries)
        .ToList();
    if (eventTypeInfo.Count <= 4)
    {
        // Event type doesn't have teacher
        continue;
    }

    // Checking for groups with spaces
    while (!eventTypeInfo[3].EndsWith(",") && eventTypeInfo.Count > 3)
    {
        eventTypeInfo[3] += $" {eventTypeInfo[4]}";
        eventTypeInfo.RemoveAt(4);
    }
```

Завдяки об’єднанню даних з цих джерел вдалося досягти інформативності розкладу на рівні використання API, але повністю незалежно від нього.

Залишалася проблема отримання списку груп та викладачів, щоб користувачі могли шукати та обирати розклад для перегляду. Така інформація недоступна для завантаження (окрім як через API), але користувачам вона відображається на сторінці сайту.

На рисунку 3.2 зображений вигляд списку груп на сайті.

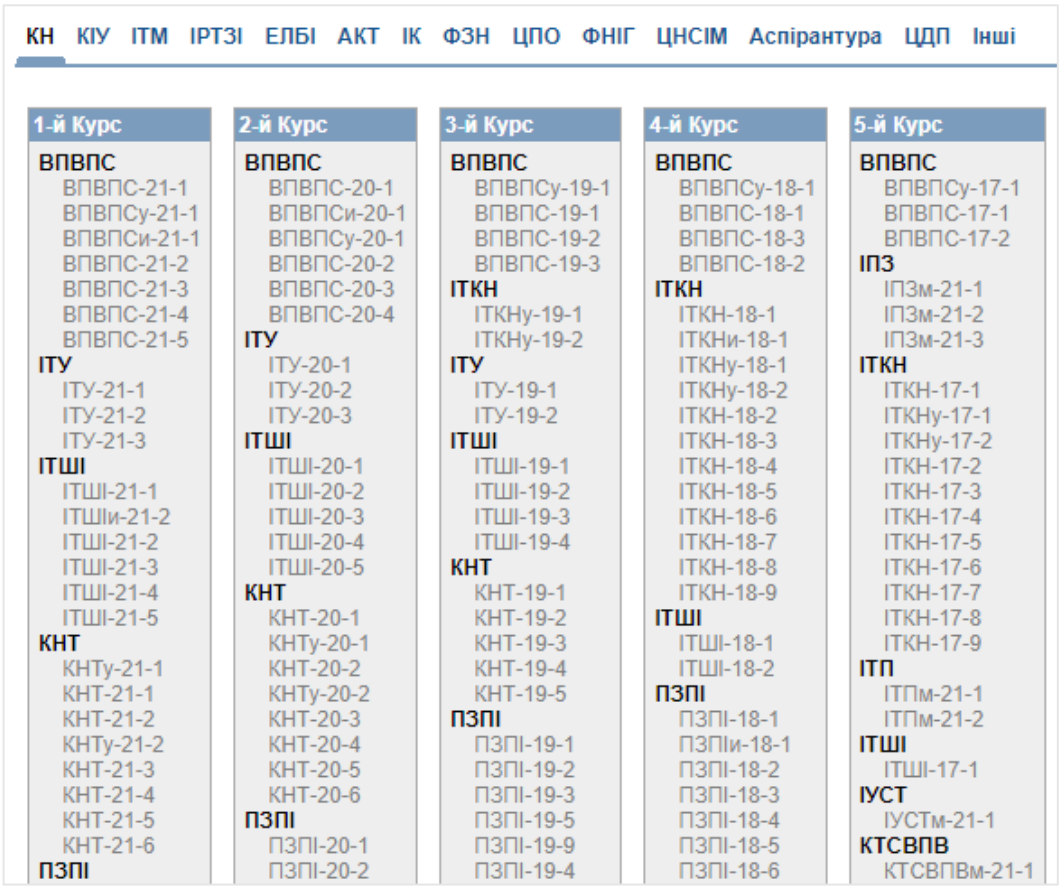

Рисунок 3.2 – Список груп на сторінці сайту

Нижче наведена частина алгоритму, що завантажує та створює список груп ХНУРЕ засновуючись на інформації зі сторінки зображеної на рисунку 3.2.

```
      faculty.Directions = new List<Cist::Direction> { new
Cist::Direction() };

      uri = Urls.CistSiteAllGroups(faculty.Id);
      string branchGroupsPage = await uri.GetStringOrWebExceptionAsync();
      foreach (string part in branchGroupsPage.Split(new[] {
"IAS_ADD_Group_in_List(" }, StringSplitOptions.RemoveEmptyEntries).Skip(1))
      {
          string[] groupInfo = part
              .Remove(part.IndexOf(")\">"))
              .Split(new[] { ',', '\'' });

          if (groupInfo.Length != 2 || !int.TryParse(groupInfo[1], out int
groupID))
          {
              continue;
```

```
        }

        string groupName = groupInfo[0];
        faculty.Directions[0].Groups.Add(new()
        {
            Id = groupID,
            Name = groupName
        });
    }
```

На цьому мінімальна функціональність, потрібна для функціонування додатку була завершена.

Наступним кроком була інтеграція з API, яку надає cist. Нижче наведено мапінг даних, що повертає API на структуру даних, що використовується додатком.

```
    timetable.Events = cistTimetable.Events
        .Select(ev =>
        {
            Local::Event localEvent = MapConfig.Map<Cist::Event,
Local::Event>(ev);
            localEvent.Lesson = MapConfig.Map<Cist::Lesson,
Local::Lesson>(cistTimetable.Lessons.First(l => l.Id == ev.LessonId));
            var cistType = cistTimetable.EventTypes.FirstOrDefault(et =>
et.Id == ev.TypeId);
            if (cistType != null)
            {
                localEvent.Type = MapConfig.Map<Cist::EventType,
Local::EventType>(cistType);
            }
            localEvent.Teachers = cistTimetable.Teachers
                .Where(t => ev.TeacherIds.Contains(t.Id))
                .DistinctBy(t => t.ShortName.Replace('ї', 'i'))
                .Select(t => MapConfig.Map<Cist::Teacher,
Local::Teacher>(t))
                .ToList();
            localEvent.Groups = cistTimetable.Groups
                .Where(g => ev.GroupIds.Contains(g.Id))
                .Select(g => MapConfig.Map<Cist::Group, Local::Group>(g))
                .ToList();
            return localEvent;
        })
        .Distinct()
        .ToList();
```

Наразі ця більшість даних отримується саме за допомогою API, але зрідка бувають збої (такі як некоректний JSON), тому запасні шляхи отримання

інформації, що були описані вище, грають значну роль у стабільності роботи додатку.

3.2 Взаємодія з Moodle

Під час пандемії університет перейшов на дистанційне навчання, що призвело до більшого використання платформи DL Nure (що збудована за допомогою Moodle).

На рисунку 3.3 зображена сторінка логіну.

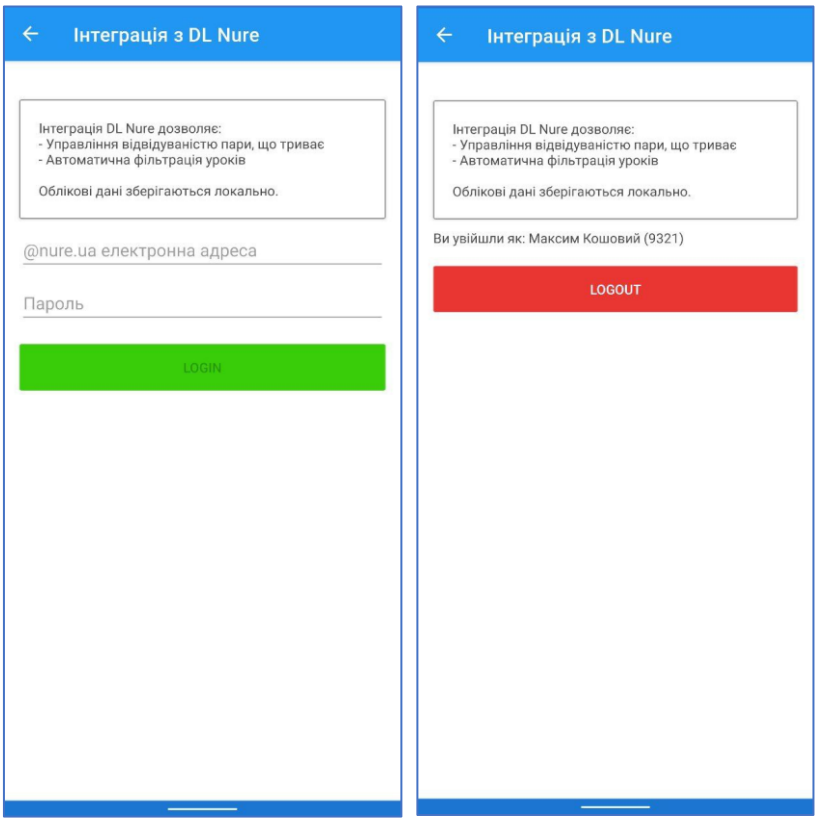

Рисунок 3.3 – Інтеграція з Moodle

Moodle – це навчальна платформа, призначена для об'єднання педагогів, адміністраторів і учнів (студентів) в одну надійну, безпечну та інтегровану систему для створення персоналізованого навчального середовища.

Для інтеграції з цією платформою структуру даних застосунка було розширено, що дозволило зберігати ідентифікатори предметів та посилання на сторінки відмічання присутності з DL Nure.

Також було розроблено сервіс для комунікації. Нижче наведено метод для отримання вмісту курсу (сторінки предмету).

```
    public async Task<List<FullCourse>> GetEnrolledCoursesAsync(int?
userId = null, bool returnUserCount = false)
    {
        var query = SetFunction("core_enrol_get_users_courses")
            .SetQueryParams(new
            {
                userid = userId ?? User!.Id,
                returnusercount = returnUserCount ? 1 : 0
            });
        return await ExecuteActionAsync<List<FullCourse>>(query);
    }

    public async Task<List<CourseSection>> GetCourseContentsAsync(int
courseId, Dictionary<GetCourseContentsOption, object>? Options = null)
    {
        Options ??= new();

        var arguments = Options
            .SelectMany((item, index) => new KeyValuePair<string,
object>[]
            {
                new($"options[{index}][name]",
item.Key.ToString().ToLowerInvariant()),
                new($"options[{index}][value]", item.Value.ToString()!)
            })
            .ToList();
        arguments.Add(new("courseid", courseId));

        var query =
SetFunction("core_course_get_contents").SetQueryParams(arguments);
        return await ExecuteActionAsync<List<CourseSection>>(query);
    }
```

Сервіс включає у себе функціональність відновлення токену доступу, тож користувачу не потрібно хвилюватися про закінчення сесії.

3.3. Взаємодія з Google Calender

Багатьом користувачам зручніше, коли всі їх події зберігаються в одному місті, яким зазвичай є Google Calender, також він має зручні сповіщення за декілька хвилин до обраної події. Саме тому додаток може не тільки агрегувати дані, а й експортувати їх.

Нижче наведено метод для формування події у Google Calender.

```
     public static CalendarEvent GenerateCalendarEvent(Event ev, int
eventNumber, int eventsCount, string? notes)
     {
         CalendarEvent calendarEvent = new()
         {
             Start = ev.StartUtc,
             End = ev.EndUtc,
             Name = $"{ev.Lesson.ShortName} - {ev.Type.ShortName}
({eventNumber}/{eventsCount})",
             Description = $"{string.Format(LN.EventClassroom,
ev.RoomName)}\n" +
                 $"{string.Format(LN.EventTeachers, string.Join(", ",
ev.Teachers.Select(t => t.Name)))}\n" +
                 $"{string.Format(LN.EventGroups, string.Join(", ",
ev.Groups.Select(t => t.Name).GroupBasedOnLastPart()))}\n",
             Location = $"KHNURE -\"{ev.RoomName}\"",
             Reminders = new List<CalendarEventReminder>()
         };
         if (notes != null)
         {
             calendarEvent.Description += $"\n{notes}\n";
         }
         if (SettingsRepository.Settings.TimeBeforeEventReminder != null)
         {
             calendarEvent.Reminders.Add(new()
             {
                 Method = CalendarReminderMethod.Alert,
                 TimeBefore =
SettingsRepository.Settings.TimeBeforeEventReminder.Value
             });
         }

         return calendarEvent;
     }
```

До події додається уся наявна інформація про подію – як отримана з сторонніх джерел, так і згенерована користувачем (наприклад, нотатки).

На рисунку 3.4 зображено процес додавання події у календар.

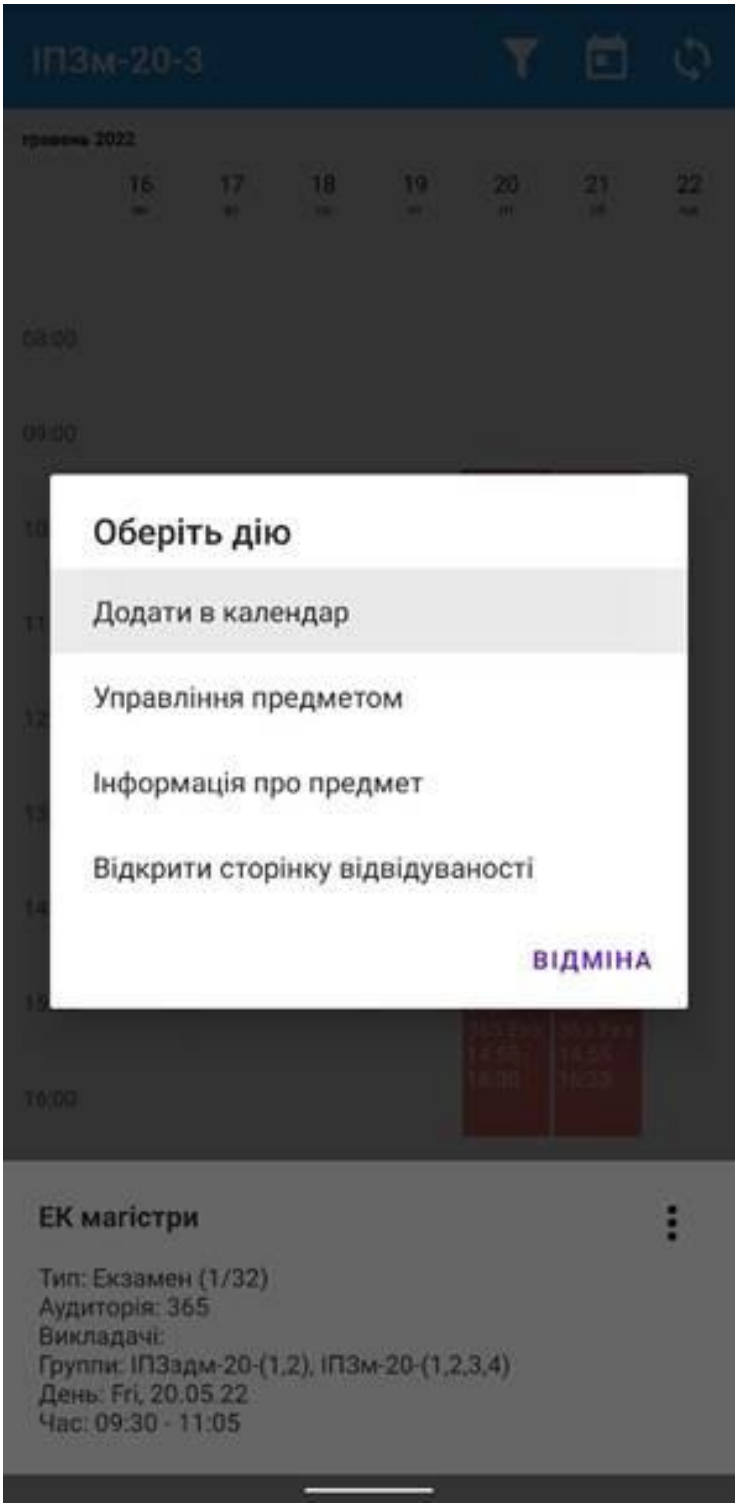

Рисунок 3.4 – Експорт події

Також враховуються, задані користувачем, налаштування, такі як календар за замовчуванням та за скільки хвилин виставляти налаштування до пари.

На рисунку 3.5 зображені налаштування експорту.

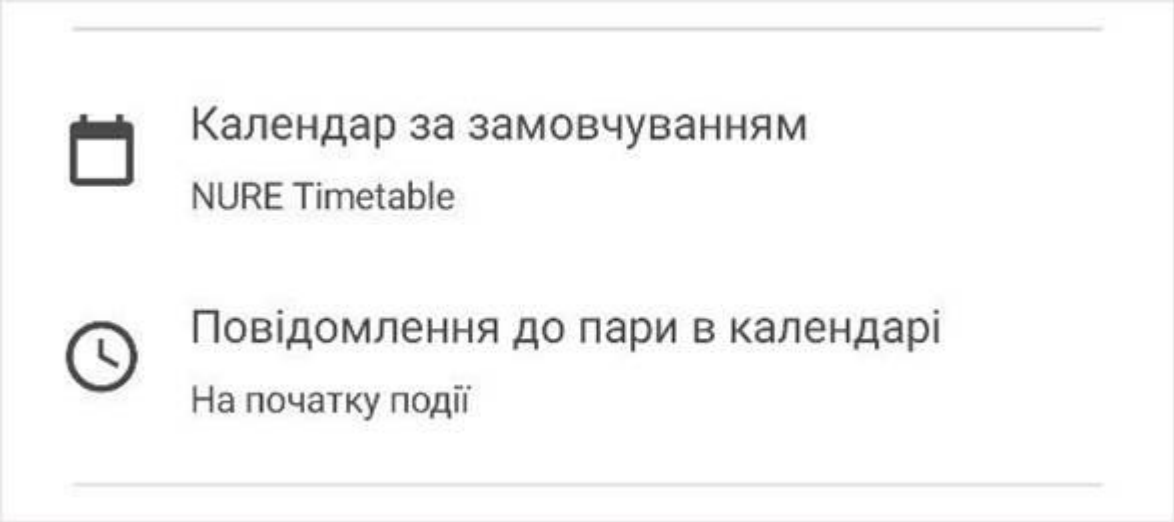

Рисунок 3.5 – Налаштування експорту

Якщо не обрати календар за замовчуванням, його потрібно буде вказувати при експорті кожної події. Також додаток створює власний календар під назвою «Nure Timetable» для того, щоб можна було відокремити події університету та усі інші.

Щодо повідомлень, то за замовчуванням вони виставляються за 30 хвилин до події, але є можливість вставляти повідомлення за 10 та 0 хвилин до події, або взагалі вимкнути їх.

3.4. Користувацькі дані

Окрім даних розкладу, що отримуються з зовнішніх джерел, додаток також інкапсулює дані, що створюються користувачами.

Програма дозволяє користувачам налаштовувати розклад під себе. Спочатку в розкладі відображаються всі предмети як обов'язкові, так і альтернативи. Але

користувач може вимкнути відображення непотрібних йому предметів, наприклад, альтернатив, які він не вибирав.

На рисунку 3.6 зображено налаштування предметів та нотатки.

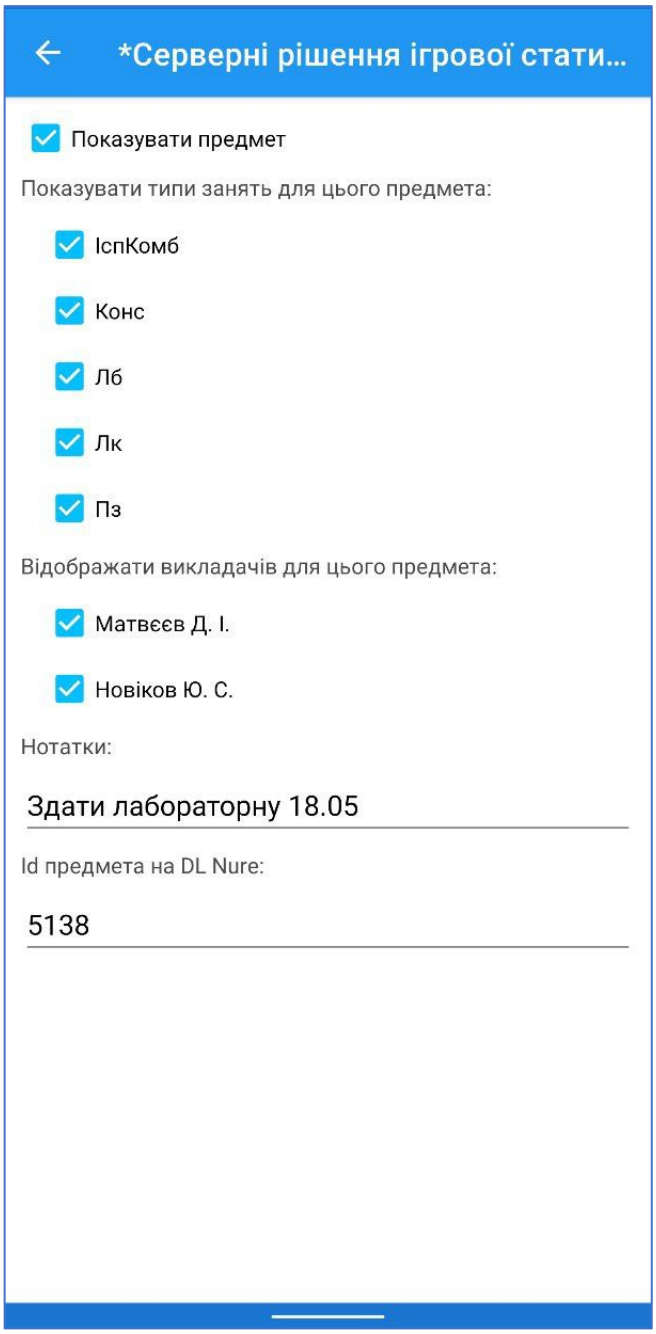

Рисунок 3.6 – Налаштування предметів та нотатки

Також користувач може вимкнути відображення певного типу занять кожного предмета. Крім цього, є можливість додати нотатки до предмета, які відображатимуться в розкладі.

Нижче наведено алгоритм побудови розкладу з урахуванням користувацьких налаштувань.

```
     public static TimetableInfoList Build(List<TimetableInfo>?
timetableInfos, bool applyHiddingSettings)
     {
         timetableInfos ??= new();

         if (applyHiddingSettings)
         {
             timetableInfos.ForEach(tt => tt.ApplyLessonSettings());
         }
         TimetableInfoList timetableInfoList = new
         (
             timetables: timetableInfos.AsReadOnly(),
             events: timetableInfos.SelectMany(tt => tt.Events)
                 .GroupBy(e => (e.Type, e.Lesson, e.Start, e.RoomName))
                 .Select(group =>
                 {
                     Event? combinedEvent = null;
                     foreach (var e in group)
                     {
                         if (combinedEvent == null)
                         {
                             combinedEvent = e;
                             continue;
                         }

combinedEvent.Groups.AddRange(e.Groups.Except(combinedEvent.Groups));

combinedEvent.Teachers.AddRange(e.Teachers.Except(combinedEvent.Teachers));
                     }
                     return combinedEvent!;
                 })
                 .ToList()
                 .AsReadOnly()
         );
         return timetableInfoList;
     }
```

Наведений алгоритм також реалізує об’єднання декількох розкладів в один. У режимі відображення декількох розкладів, події, що співпадають за назвою предмета та часом проведення – видаляються (лишаючи лише один екземпляр), нотатки ігноруються, а налаштування приховування подій застосовуються окремо для кожного розкладу перед групуванням.

Нижче наведено алгоритм застосування налаштувань по приховуванню подій.

```
public void ApplyLessonSettings()
{
    foreach (var lInfo in LessonsInfo.Where(ls =>
ls.Settings.IsSomeSettingsApplied))
    {
        // Hiding settings
        if (lInfo.Settings.Hiding.ShowLesson == false)
        {
            Events.RemoveAll(ev => ev.Lesson == lInfo.Lesson);
        }
        else if (lInfo.Settings.Hiding.ShowLesson == null)
        {
            Events.RemoveAll(ev => ev.Lesson == lInfo.Lesson &&

(lInfo.Settings.Hiding.EventTypesToHide.Contains(ev.Type.ID) ||

lInfo.Settings.Hiding.TeachersToHide.Intersect(ev.Teachers.Select(t =>
t.ID)).Any())
            );
        }
    }
}
```

Завдяки цьому користувач може бачити лише актуальні для нього події з наявного розкладу.

ВИСНОВКИ

В рамках написання кваліфікаційної роботи магістра було створено додаток, який дозволяє заповнити розкладом Android календар, а також інтегрувати систему з Google Calendar та Moodle.

В ході виконання атестаційної роботи були виконані настіпні кроки:

– визначена актуальність теми;

– визначено цілі інтеграції додатків;

– проведено аналіз предметної галузі інтеграції додатків із сторонніми системами, а також інтеграції даних;

– проведено аналіз проблеми;

– визначено наслідки відсутності вирішення проблеми інтеграції;

– поставлено завдання;

– проведено аналіз способів інтеграції додатків;

– досліджено відмінності існуючих методів інтеграції: найпростіших – обмін файлами та повідомленнями або звернення до загальної бази даних; та сучасних технології, що використовують API – SOAP і REST;

– на основі зібраної інформації було обрано технологію REST. REST API – це надійний та зручний інтерфейс веб-служб. Переваги даного інтерфейсу полягають у зручності інтеграції та розгортання, тому він ідеально підходить для використання у веб-проектах та програмах для мобільних пристроїв;

– досліджено способи інтеграції з Google Calendar;

– досліджено способи інтеграції з Android Calendar;

– досліджено способи інтеграції з Moodle;

– досліджено методи веб-скрейпінгу;

– проаналізовано та реалізовано способи підключення обраних систем.

Розроблену в ході виконання атестаційної роботи систему доцільно застосовувати для перегляду розкладів груп, викладачів та аудиторій ХНУРЕ.

Подальший розвиток системи може бути пов'язаний з додаванням нових функцій, а також з розширенням бази університетів.

За результатами дослідження були опубліковані тези в рамках дванадцятої міжнародної науково-технічної конференції [12].

ПЕРЕЛІК ПОСИЛАНЬ

1. Каук В., Гребенюк В., Пуголовок К., Водяницький Д. Виклики, які надають нові можливості // Екстренне дистанційне навчання в Україні: Монографія. 2020. С. 223-232.

2. Хоп Г. Шаблоны интеграции корпоративных приложений. М.: Диалектика, 2019. – 672с.

3. Patni S. Pro RESTful APIs: Design, Build and Integrate with REST, JSON, XML and JAX-RS. N.: Apress, 2017. – 143p.

4. Аитов В. Интеграция информационной системы вуза с системой e-learning. М.: Синергия, 2015. – 7с.

5. Меджуи М. Непрерывное развитие API. Правильные решения в изменчивом технологическом ландшафте. П.: Питер Пресс, 2020. – 272с.

6. Лоре А. Проектирование веб-API. М.: ДМК Пресс, 2020. – 440с.

7. Richardson L. RESTful Web APIs. S.: O'Reilly Media, 2013. – 406c.

8. Rice W. Moodle E-Learning Course Development: A complete guide to successful learning using Moodle. B.: Packt Publishing, 2006. – 256p.

9. Smith V. Go Web Scraping Quick Start Guide: Implement the power of Go to scrape and crawl data from the web. B.: Packt Publishing, 2019. – 132p.

10. Gardner J. XSLT and Xpath: A Guide to XML Transformations. H.: Prentice Hall, 2001. – 592p.

11. Goyvaerts J. Regular Expressions Cookbook: Detailed Solutions in Eight Programming Languages. S.: O'Reilly Media, 2012. – 612p.

12. Кошовий М. Методи інтеграції даних з різнорідних веб-систем. // Дванадцята міжнародна науково-технічна конференція. 2022. С. 156.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ ЗА НАУКОВИМИ НАПРЯМАМИ НАУКОВЦІВ КАФЕДРИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

1. Каук В., Гребенюк В., Пуголовок К., Водяницький Д. Виклики, які надають нові можливості // Екстренне дистанційне навчання в Україні: Монографія. 2020. С. 223-232.