повинні готувати «майбутнього фахівця», використовувати існуючі засоби та імплементацію таких технологій, організовувати навчальну діяльність за допомогою розроблених сервісів, формувати в них інформаційно-комунікаційні компетенції.

У зв'язку із складною ситуацією в світі, набрали шалених обертів нові та значно закріпили свої позиції існуючі провайдери хмарних інфраструктур, такі як: Tucha.ua (на ринку з 2012 р.); De Novo – провайдер хмарних сервісів IaaS (на ринку як центр обробки даних з 2008 р.); UCloud – провайдер хмарних сервісів IaaS (на ринку з 2011 р.); GigaCloud – провайдер, надає в оренду IaaS-послуги (на ринку з 2006 р. як GigaTrans, а з 2016 р. – була перейменована на GigaCloud), тощо.

Список використаних джерел

1. Global Cloud Computing Market in Education Sector 2017-2021. Research and Markets. [online]. Available: https://www.researchandmarkets.com/research/4z2jfn/global_cloud
2. J. Taylor, A. Harrison, "From P2P to Web services and grids: peers in a client/server world". Springer Science & Business Media, p. 148, 2005.
3. Литвинова С. Г. Проектування хмаро орієнтованого навчального середовища загальноосвітнього навчального закладу : монографія / С. Г. Литвинова - К .: Компринт, 2016. - 354 с.

Краснов Егор Валерійович
студент першого курсу другого (магістреського) рівня вищої освіти фізико-математичного факультету спеціальність 014.09 Середні освіта (Інформатика) Житомирського державного університету імені Івана Франка

РОЗРОБКА СПЕЦІАЛІЗОВАНОГО API ЗАСОБАМИ МОВИ ВИСОКОГО РІВНЯ PHP І ФРЕЙМВОРКА REST

Організація навчального процесу тісно пов'язана з обліком успішності, як з основним результатом навчання. Ефективна перевірка рівня знань та досягнень учнів і студентів є важливою частиною навчального процесу як у школі так і у закладах вищої освіти. Проте традиційні підходи до організації системи контролю вже не є настільки ефективними. Для забезпечення безперервного навчального процесу потрібна система, що дозволяє автоматизувати процес контролю успішності студентів. Використання систем керування навчальним процесом вимагає значних капіталовкладень, окрім цього

вони, більшою мірою, орієнтовані на організацію навчального процесу, а не на перевірку знань.

В роботі розглядається задача автоматизації тестування – забезпечення програмного продукту для контролю успішності з мобільних пристроїв. В умовах підвищення вимог до якості навчання, призначення стипендій в залежності від навчальних успіхів та необхідності створення умов для публічного та прозорого висвітлення навчальних досягнень студентів **актуальною проблемою є** розробка додатку для тестування з мобільного пристрою.

Аналіз останніх досліджень і публікацій. Дослідженню різних аспектів багатогранної проблеми створення та використання освітніх електронних ресурсів приділялась увага у працях Н. Р. Балик, О. М. Гончарової, Л. Е. Гризун, В. Б. Івасика, І. С. Іваськова, О.Г. Кузьмінської, Н.В. Морзе, В.П. Олексюка, С.А. Ракова, С.О. Семерікова, Ю.В. Триуса, О.І. Шиман; формування основ інформаційної культури розглядали М.І. Жалдак, О.А. Кузнецов, Г.О. Михалін, В.Ю. Мілітарєв, Ю.С. Рамський, Н.М. Розенберг, І.М. Яглом.

Розроблений мобільний додаток «InStudy» для смартфонів на платформі Android *засобами Java* містить потрібний функціонал для забезпечення ефективного контролю за якістю знань студентів вишів та учнів загальноосвітніх шкіл. Базова версія (basic version) включає такі можливості: створення завдань; додавання нагадувань; синхронізація з хмарним середовищем.

Документація генерується за допомогою пакета сторонніх бібліотек API Platform. Архітектура RESTful використовує кілька кінцевих точок для взаємодії з моделями. Далі додаються всі необхідні представлення, які будуть виводитися користувачеві в залежності від етапу виконання програми.

Можливість використання Android-додатку передбачає, що усі потрібні для його роботи таблиці уже створені та наповнені даними, а REST API при потребі переписано та адаптовано у відповідності до веб-продукту та СУБД. Єдиною вимогою при інтеграції додатку із веб-системою є використання рекомендованої структури бази даних, тобто забезпечення додатку усіма потрібними даними.

REST використовується для створення легких, простих у обслуговуванні і швидко масштабованих веб-сервісів [2]. Сервіс,

побудований на архітектурі, яка називається RESTful-сервісом. REST використовує HTTP як базовий мережевий протокол. Моделі описують доступну для клієнтських додатків інформацію. Для різних кінцевих точок моделі можуть змінюватися.

Як основну базу даних серверна частина додатку використовує СУБД MySQL [1]. Вона дозволяє забезпечити підтримку великої кількості типів таблиць. Зокрема користувачі можуть вибрати як таблиці типу MyISAM так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць.

Для всіх відношень автоматично створюються індекси по первинному ключі. Індексування значно пришвидшує пошук даних у БД. Проте надмірне використання індексування призводить до втрати продуктивності виконання запитів на поновлення та знищення.

Для написання веб-сервісів, які забезпечуватимуть доступ до глобальної бази даних, оптимальним вибором є використання мови PHP, яка має достатню кількість засобів, щоб швидко та якісно розв'язати проблему доступу до БД та має достатню кількість уже готових фреймворків для створення REST API та роботи з ним. Для написання REST API використано мікрофреймворк ApiPlatform. Основним завданням веб-сервісів, які надають додатку засоби для комунікацій із БД на сервері, є можливість отримання та надсилання даних через мережевий протокол HTTP. Для забезпечення зручної та швидкої роботи із даними, усі дані пересилаються у форматі JSON. Частина коду, яка наведена нижче демонструє частину API для роботи із базою даних та забезпечує можливість отримання викладачем списку студентів із глобальної БД. За допомогою методу POST додаток пересилає потрібні ідентифікаційні дані, після чого скрипт витягує потрібні дані із БД, форматує їх та надсилає додатку.

Для забезпечення стабільної роботи із проколом HTTP використовується спеціальна бібліотека NativeHTTP. Ця бібліотека підтримує усі сучасні засоби для роботи із протоколом HTTP та CURL. Для отримання даних API та коректного їх опрацювання у форматі JSON був розроблений спеціальний клас Persister, лістинг коду якого наведено нижче.

```
class ConnectionDataPersister implements ContextAwareDataPersisterInterface
{

    private $decorated;
    /**
     * @var TokenStorageInterface
     */
    private $tokenStorage;
    /**
     * @var EntityManagerInterface
     */
    private $entityManager;

    /**
     * PaymentDataPersister constructor.
     * @param DataPersisterInterface $decorated
     * @param TokenStorageInterface $tokenStorage
     * @param EntityManagerInterface $entityManager
     */
    public function __construct(
        DataPersisterInterface $decorated,
        TokenStorageInterface $tokenStorage,
        EntityManagerInterface $entityManager
    )
    {
        $this->decorated = $decorated;
        $this->tokenStorage = $tokenStorage;
        $this->entityManager = $entityManager;
    }
    /**
     * @inheritDoc
     */
    public function supports($data, array $context = []): bool
    {
        return $data instanceof Connection;
    }

    /**
     * @inheritDoc
     * @var Connection $data
     */
```

```
    public function persist($data, array $context = [])
    {
        $project = $this->tokenStorage->getToken()->getUser();
        if (!$project instanceof Project) {
            throw new BadCredentialsException();
        }
        $data->setProject($project);
        $this->entityManager->persist($data);
        $this->entityManager->flush();
        return $data;
    }

    /**
     * @inheritDoc
     */
    public function remove($data, array $context = [])
    {
        return $this->decorated->remove($data);
    }
}
```

Для того, щоб працювали push notification, які сповіщатимуть про нове повідомлення миттєво, використовується сервіс Firebase. Нижче відображено лістинг коду для сповіщення додатку про зміни в моделях API.

```
protected function execute(InputInterface $input, OutputInterface $output): int
{
    $io = new SymfonyStyle($input, $output);
    $arg1 = $input->getArgument('arg1');

    if ($arg1) {
        $io->note(sprintf('You passed an argument: %s', $arg1));
    }

    $apiKey = "AIzaSyCuumujVvGpGV3AkqC_s7NcI2XN0QotS74";
    $to = "dbTwQ8PrTPG6Ydrdx6r2jM:APA91bHmxmcav-huagU-
rpmH9P7zNzK2ieiBPZZJkP_so60cym5Ed6H3hT5eP-
nu0aqo8NjsAslX7csYfPmBRf0WNnBk_XU5dfGqmhJIBnWNRBI95d35bEdiIIhb
_70c1DjLYz2zBAZl";

    $client = new NativeHttpClient();
```

```
    $response = $client->request('POST', "https://fcm.googleapis.com/fcm/send", [
        'headers' => [
            'Content-Type' => 'application/json',
            'Authorization' => 'key='.$apiKey
        ],
        'json' => [
            "registration_ids" => [
                $to
            ],
            "data" => [
                "body" => "cym5Ed6H3",
                "title" => "Create new TEST"
            ],
            "notification" => [
                "title" => "Test_id_5d35bEdiI ",
                "body" => "status NEW",
            ]
        ]
    ]);

    return 0;
}
```

Розроблюваний додаток націлений на інтеграцію із різними платформами, зрозуміло що бізнес-правила, інструменти для роботи з БД та послідовність дій створення самої бази даних може бути різними в залежності від системи з якою додаток буде проводити інтеграцію. Виходячи з сукупності перерахованих вище фактів, вибір інших інструментів, інший тип СУБД та середовища її функціонування не вплине на роботу Android додатку, адже за зв'язок із БД на сервері відповідатимуть API та служби обробки кінцевих точок, які можна буде переписати потрібним чином, при цьому ми не повинні редагувати сам Android-додаток. Можливість використання Android-додатку передбачає, що усі потрібні для його роботи таблиці уже створені та наповнені даними, а REST API при потребі може бути переписано та адаптовано у відповідності до нового веб-продукту та СУБД.

Для отримання та запису даних отриманих від API Android-додаток використовуватиме кінцеві точки, які в свою чергу

отримуватимуть дані з сервера БД. Для роботи веб-служб із БД найкраще буде використовувати Entity.

Передбачаються наступні методи для забезпечення роботи API:

1) «ChangePasswordAction» – викликається після події change_password. Користувач отримує на поштову скриньку лист з посиланням на зміну паролю;
2) «ForgotPasswordAction» – надсилає тимчасовий пароль за пошту користувача для відновлення доступу;
3) «PostQuestionAction» – обробник кінцевої точки, викликаєтеся після створення нового запитання у тесті;
4) «PostUserAction» – створення нового користувача;
5) «UploadImageAction» – запит на зміну фотографії користувача у профілі, передається додатком у base64 форматі;
6) «PushNotify» – повідомляє додаток про зміни в моделях;
7) «JWTCreatedListener» – створює тимчасовий токен для авторизації;
8) «UserRepository» – отримує з БД статистичні дані про користувача, виконує запис та видалення з БД
9) «TestRepository» – отримує з БД статистичні дані про тест , виконує запис та видалення з БД;
10) «QuestionsRepository» – отримує з БД статистичні дані про питання , виконує запис та видалення з БД. Передбачений використання одного питання в декількох тестах;
11) «AnswerRepository» – отримує з БД статистичні дані про відповіді які пов'язані з питаннями за допомогою механізму індексації;
12) «ActiveTestRepository» – отримує з БД статистичні дані про активні тести, час завершення та початку тесту.

Наведенні методи є прикладами обробників кінцевих точок. Кожна кінцева точка має свій механізм реагування на події розроблений за допомогою шаблонів Споглядач, Фабрика та Будівник.

Розроблена система майже повністю автоматизує процес створення та редагування даних з використанням кінцевих точок. Також, програмна система надає засоби для інформування додатку змінами в моделях. За допомогою API platform розроблено ряд функціональних класів для обробки даних. Також системою передбачені функціональні класи слухачів для реагування на події пов'язані в редагуванням БД. Розроблені відношення між моделями

бази даних за допомогою ORM анотацій. Використано шаблон репозиторію для взаємодією з БД та реагування на кінцеві точки з механізмом фільтрування і пагінації.

Створено систему перевірки даних за допомогою анотації. Система передбачає два етапи перевірки. Перший етап, знаходження валідаційного класу який знаходиться в моделі. Другим етапом є виклик обробника анотації, виконання та перевірка умов зазначених у методі класу валідатора.

Представлено механізм реагування на події. Перехоплення запиту та обробка до або після основної функції. Це дозволяє більш детально налаштовувати життєвий цикл додатку та доповнювати моделі під час виконання головної частини коду.

Список використаних джерел

1. MySQL. URL: https://www.mysql.com/ (дата звернення: 05.02.2021).
2. REST API Tutorial. URL: https://restfulapi.net/ (дата звернення: 05.02.2021).

Дончак Леся Григорівна, к. е. н., доцент
Вінницький навчально-науковий інститут економіки Західноукраїнського національного університету, Вінниця
Шкварук Діна Григорівна, викладач
Вінницький навчально-науковий інститут економіки Західноукраїнського національного університету, Вінниця

СУЧАСНИЙ СТАН ТА ПЕРСПЕКТИВИ РОЗВИТКУ ДИСТАНЦІЙНОГО НАВЧАННЯ У ВИЩИХ НАВЧАЛЬНИХ ЗАКЛАДАХ

Якщо пару років назад ми говорили про дистанційне навчання як про новітній засіб отримання знань здобувачами вищих навчальних закладів, то вже сьогодні, з метою запобігання поширенню на території України коронавірусу COVID-19, дистанційну освіту маємо як обов'язковий та невід'ємний елемент організації навчального процесу. Звичайно ж, така освіта має свої як переваги, так і недоліки, проте, на даний час в нас немає інших альтернативних шляхів передачі студентам знань. Тому, на сьогоднішній день гостро стоїть питання ефективного впровадження дистанційного навчання в зв'язку з довготривалими карантиними обмеженнями.