УДК 004.5 **doi: 10.26906/SUNZ.2025.2.154-160**

М. В. Ліпчанський, О. В. Ліпчанська, Г. І. Молчанов, К. А. Христофоров

Національний технічний університет «Харківський політехнічний інститут», Україна, Харків

РОЗРОБКА API ДЛЯ АДАПТИВНОГО ТЕСТУВАННЯ В СИСТЕМАХ УПРАВЛІННЯ НАВЧАЛЬНИМИ КУРСАМИ

Анотація. Стаття присвячена розробці API для системи управління навчальними курсами з інтегрованою підтримкою адаптивного тестування. Актуальність теми зумовлена зростаючою потребою в індивідуалізації освітнього процесу та підвищенні ефективності оцінювання знань у сучасних умовах цифрової трансформації освіти. Метою статті є дослідження принципів та обґрунтування підходів до створення ефективного, захищеного та гнучкого API для управління навчальними курсами. Особливістю API є підтримка адаптивного тестування з урахуванням рівня знань користувача. У рамках дослідження проведено аналітичний огляд систем діагностики знань, адаптивних моделей та стратегій тестування. На основі отриманих результатів спроєктовано архітектуру системи, обрано сучасні технології, зокрема Spring Boot та PostgreSQL, реалізовано бізнес-логіку. Розроблено програмний код, що демонструє роботу алгоритмів адаптивного тестування, включаючи механізм вибору тестових завдань на основі попередніх відповідей користувача. API забезпечує можливість інтеграції адаптивного тестування в різноманітні освітні платформи, що підвищує ефективність та індивідуалізацію навчального процесу. Впровадження API відкриває нові можливості для персоналізації навчання та вдосконалення якості освітніх послуг.

Ключові слова: API, REST, IRT, база даних, адаптивне тестування, навчальний курс.

Вступ

У сучасних умовах цифрової трансформації освіти зростає значення технологій, які сприяють індивідуалізації навчального процесу та підвищенню ефективності оцінювання знань. Однією з таких технологій є адаптивне тестування, яке дає змогу створювати персоналізовані тести, що враховують рівень підготовки користувача.

Таким чином, у сучасному цифровому часі сфера освіти зазнає значних змін під впливом технологічного прогресу. Зростає потреба в ефективних інструментах, які б дозволяли не лише надавати знання, а й об'єктивно оцінювати рівень їх засвоєння кожним окремим студентом.

Постановка проблеми. Традиційні методи оцінювання, такі як іспити та контрольні роботи, часто не здатні повною мірою врахувати індивідуальні особливості навчального процесу та можуть бути суб'єктивними.

Адаптивне тестування є однією з інноваційних технологій, що дозволяє вирішити зазначені проблеми. Воно передбачає індивідуалізацію процесу тестування, коли складність завдань підлаштовується під рівень знань кожного конкретного користувача, що призводить до більш точних оцінок знань, зменшує час тестування та підвищує мотивацію студентів. Проте, для ефективного впровадження адаптивного тестування в освітній процес необхідні відповідні програмні рішення. Одним з ключових елементів таких рішень є API (Application Programming Interface) – інтерфейс, що дозволяє різним системам та платформам взаємодіяти між собою. Розробка якісного API для системи управління навчальними курсами з підтримкою адаптивного тестування є складною задачею, що вимагає врахування багатьох факторів.

Таким чином, існує нагальна потреба у розробці ефективного, масштабованого, безпечного та зручного в інтеграції API для систем управління навчальними курсами з підтримкою адаптивного тестування. Вирішення цієї проблеми сприятиме підвищенню якості освіти та впровадженню сучасних технологій в навчальний процес.

Відповідно, **метою статті** є вивчення принципів та обґрунтування методології побудови API, який буде ефективним, безпечним і гнучким. Це дозволить керувати навчальними курсами з підтримкою адаптивного тестування для врахування індивідуального рівня знань студента.

Аналіз останніх досліджень і публікацій. Сучасні реформи у вищій освіті спрямовані на створення сприятливих умов для розвитку та реалізації інтелектуального потенціалу суспільства. В [1] підкреслюється важливість контролю якості освіти та оцінювання навчальних досягнень студентів вищих навчальних закладів для забезпечення якісного навчального процесу та як ключового елемента розвитку інтелектуального потенціалу суспільства. Також підкреслюється значущість розробки та впровадження ефективних систем оцінювання у навчальний процес та необхідність постійного вдосконалення методів контролю.

У [2] особлива увага приділяється історичному розвитку контролю знань у навчальному процесі. Автор проводить аналіз еволюції форм і методів оцінювання, підкреслюючи, що головна мета оцінювання – визначення рівня засвоєння навчального матеріалу – залишається незмінною протягом історії. З цього можна зробити висновок про те, що розуміння історичного контексту є важливим для глибшого осмислення сучасних викликів у сфері оцінювання.

У [3] детально аналізується поняття "тест" у педагогічному контексті, приділяючи увагу його практичному застосуванню як методу та інструменту вимірювання. Автори підкреслюють необхідність розробки та використання тестів з дотриманням на-

укових стандартів. Вони пропонують усвідомлений підхід до тестування, що базується на знанні його теоретичних основ. Ці пропозиції є корисними для розробників та користувачів тестів, які прагнуть забезпечити якість та ефективність процесу оцінювання, зокрема в контексті адаптивного тестування, де розуміння сутності та методології тестування є критично важливим для створення валідних та надійних інструментів оцінювання.

У роботі [4] розглядаються перші наукові спроби вимірювання індивідуальних відмінностей, здійснені Френсісом Гальтоном. Автор аналізує методи, які Гальтон використовував для дослідження чутливості та інших характеристик. При цьому виділяються три ключові принципи тестування, сформульовані Гальтоном, які зберігають свою актуальність і в сучасній практиці. Підкреслюється фундаментальний внесок Гальтона у становлення теорії тестів та важливість статистичного підходу в цій галузі.

У контексті історичного розвитку педагогічного тестування, у [5] приділяється увага значній ролі Альфреда Біне в сфері діагностики інтелектуальних здібностей. Праці Біне є важливою відправною точкою для подальшого прогресу в галузі вимірювань у психології та освіті. У зв'язку з цим, пропонується застосовувати пропозиції Біне при розробці сучасних інструментів оцінювання.

Таким чином, у сучасній освіті контроль знань студентів є ключовим для управління навчальним процесом та досягнення позитивних результатів. Проведений аналіз літератури показав, що контроль знань є важливою складовою освітнього процесу, а адаптивне тестування має значний потенціал для його вдосконалення. Однак, недостатньо дослідженою залишається проблема розробки ефективних API для інтеграції адаптивного тестування в існуючі системи управління навчанням. Це дослідження спрямоване на розв'язання цієї проблеми шляхом розробки API, що забезпечує гнучкість, масштабованість та безпеку при впровадженні адаптивного тестування. Серед різноманітних технологій контролю знань особливу увагу привертає адаптивне комп'ютерне тестування, яке належить до найсучасніших підходів у цій сфері.

Виклад основного матеріалу

Розробка API для адаптивного тестування в системі управління навчальними курсами відбувається в декілька етапів.

Аналіз вимог та проєктування системи. На першому етапі визначаються функціональні і нефункціональні вимоги до системи, відбувається проєктування архітектури системи, розробляються моделі даних та проєктування бази даних, аналіз існуючих систем діагностики знань та сучасних підходів до адаптивного тестування.

Під час визначення функціональних і нефункціональних вимог до системи проведено аналіз потреб користувачів та визначені основні функції, які повинна виконувати система. Функціональні вимоги включають:

- реєстрація та автентифікація користувачів;
- керування навчальними курсами (створення, редагування, видалення);
- проведення адаптивних тестування;
- реалізація алгоритму адаптивного вибору тестових завдань;
- обробка результатів тестування.

Нефункціональні вимоги визначають:

- продуктивність (час відгуку API на запити);
- безпека (захист даних користувачів та системи);
- масштабованість (здатність системи обробляти велику кількість користувачів та даних);
- надійність (стабільність роботи системи та мінімізація збоїв);
- зручність використання (простота інтеграції API з іншими системами).

Під час проєктування архітектури системи було визначено ключові компоненти системи, принципи їхньої взаємодії, а також технології, які були використані для реалізації розробки. У рамках даного дослідження було прийнято рішення використовувати наступні технології та підходи:

- Spring Boot для розробки серверної частини;
- RESTful API для забезпечення взаємодії між клієнтом та сервером;
- JWT-аутентифікація для захисту API.

Spring Boot обраний як основний фреймворк для розробки серверної частини застосунку. Spring Boot спрощує процес створення веб-застосунків на Java, надаючи готовий до використання набір інструментів та конфігурацій. Це дозволяє зосередитися на реалізації бізнес-логіки, а не на налаштуванні інфраструктури, що відповідає вимогам швидкої розробки та масштабованості.

Для організації взаємодії між клієнтською та серверною частинами системи використано RESTful API. RESTful API є архітектурним стилем, який забезпечує стандартизований обмін даними між системами через HTTP. Цей підхід забезпечив гнучкість, масштабованість та простоту інтеграції з різними клієнтськими застосунками.

Для захисту API та забезпечення безпеки обміну даними використано JWT (JSON Web Token) аутентифікацію. JWT є стандартом для безпечної передачі інформації між сторонами у вигляді JSON-об'єктів. Цей механізм дозволив перевіряти справжність запитів та надавати доступ до ресурсів лише авторизованим користувачам.

Такий вибір технологій та архітектурних рішень, як Spring Boot, RESTful API та JWT, обумовлений необхідністю створення ефективної, масштабованої та безпечної системи, яка відповідає сучасним вимогам веброзробки та забезпечує простоту інтеграції з іншими системами.

Наступним кроком у розробці системи є визначення моделі даних.

Розробка моделі даних та проєктування бази даних є важливим етапом у створенні будь-якої системи, що працює з інформацією. На цьому етапі визначені сутності, які будуть зберігатися в базі даних, зв'язки між ними та їх атрибути.

При розробці API для системи управління навчальними курсами з підтримкою адаптивного тестування визначенні наступні сутності:

- користувачі (users): інформація про користувачів системи, включаючи їхні облікові дані та ролі (викладач, студент);
- навчальні курси (courses): дані про курси, які пропонуються в системі;
- теми (topics): розділи або теми, на які поділяються навчальні курси;
- тестові завдання (questions): запитання для тестування, включаючи їхній текст, рівень складності та інші параметри, необхідні для адаптивного тестування;
- варіанти відповідей (answer_options): можливі відповіді на тестові завдання з позначенням правильної відповіді;
- результати тестування (test_sessions): інформація про проходження тестів користувачами, включаючи їхні відповіді, час проходження та результати.

Для кожної сутності визначені атрибути, що характеризують її (наприклад, для сутності users це: id, email, password, role), а також атрибути мають тип даних (наприклад, текст, число, дата), обмеження (наприклад, унікальність, обов'язковість) та інші властивості (рис. 1).

Визначені зв'язки між сутностями. Наприклад, "один-до-багатьох" між courses і topics (один курс може містити багато тем), "багато-до-багатьох" між users і courses (студенти можуть вивчати багато курсів, і курс можуть вивчати багато студентів) та інші.

```
CREATE TYPE USER_ROLES AS ENUM ('USER',
'TEACHER');

CREATE TABLE users
(
    id       BIGSERIAL PRIMARY KEY,
    email    TEXT UNIQUE NOT NULL,
    password TEXT        NOT NULL,
    role     USER_ROLES          NOT NULL
);
```

Рис. 1. Сутність users в базі даних

Для реалізації бази даних обрано систему управління базами даних (СУБД) PostgreSQL, як потужну, надійну та розширювану реляційну СУБД з відкритим вихідним кодом, яка підтримує широкий спектр типів даних, включаючи стандартні типи SQL, а також розширені типи, такі як JSON, XML та інші. Вибір PostgreSQL обумовлений її надійністю, продуктивністю, підтримкою розширених функцій та активною спільнотою розробників. Застосування PostgreSQL забезпечило високий рівень цілісності даних, підтримання транзакцій, ACID-властивостей та багато інших функцій, важливих для сучасних вебзастосунків.

Традиційні методи оцінювання, такі як іспити та контрольні роботи, мають низку недоліків, зокрема суб’єктивність оцінювання з боку викладача та ризик недостовірності результатів. На відміну від них, адаптивне тестування являє собою діалогову тестову систему, яка динамічно змінює порядок подання завдань залежно від успішності виконання попередніх. Існують різні стратегії адаптивного тестування, зокрема двокрокові та багатокрокові підходи, кожен із яких характеризується власними особливостями. До основних методів адаптивного тестування належать пірамідальне тестування, тестування з будь-якого рівня та тестування на основі банку тестових завдань, реалізація яких описана на відповідних етапах дослідження.

Реалізація базової функціональності користувачів є наступним етапом у реалізації завдання для забезпечення безпеки процесу реєстрації та авторизації, щоб захистити дані користувачів від несанкціонованого доступу. На цьому етапі виконується розробка API для реєстрації та авторизації користувачів, реалізація механізмів валідації вхідних даних, хешування паролів та управління сесіями, створення сутності User та відповідного функціоналу для взаємодії з базою даних.

Розробка API для реєстрації та авторизації користувачів включає в себе:

- розробка ендпоінтів API (створення URL-адрес, за якими клієнтські застосунки можуть надсилати запити на реєстрацію та авторизацію; визначення HTTP-методів для реєстрації або авторизації, які будуть використовуватись для цих операцій);
- обробка запитів на реєстрацію (отримання даних користувача, надісланих клієнтським застосунком; перевірка валідності цих даних; перевірка, чи не існує вже користувача з таким самим ім'ям або електронною поштою);
- створення облікових записів користувачів (збереження даних користувача в базі даних; забезпечення безпечного зберігання паролів);
- перевірка облікових даних при вході в систему (отримання облікових даних, надісланих клієнським застосунком; пошук облікового запису користувача в базі даних; перевірка правильності введеного пароля);
- повернення токенів аутентифікації (у разі успішної авторизації – генерація унікального токена, який буде використовуватись для подальших запитів до API; відправлення цього токена клієнтському застосунку; визначення терміну дії токена) (рис. 2).

Реалізація механізмів валідації вхідних даних, хешування паролів та управління сесіями є критично важливим етапом для забезпечення безпеки та надійності будь-якої вебсистеми, особливо тієї, яка обробляє конфіденційні дані користувачів.

Валідація вхідних даних запобігає введенню некоректних, шкідливих або небезпечних даних, які можуть призвести до помилок у роботі системи, витоку інформації або навіть атак. В розробці реалізовані наступні елементи валідації:

- перевірка формату електронної пошти;
- перевірка довжини пароля;
- перевірка наявності обов'язкових полів;
- перевірка типу даних.

Реалізація валідації здійснюється як на клієнтській стороні за допомогою JavaScript, так і на серверній стороні за допомогою Java (рис. 3).

```
@Override
public String generateToken(User user) {
    ApplicationProperties.JWTInfo jwtInfo =
properties.getJwt();
    String secret = jwtInfo.secret();
    Date now = new
Date(System.currentTimeMillis());
    Date expiryDate = new Date(now.getTime() +
jwtInfo.expirationTime());
    Claims claims = Jwts.claims()
            .subject(user.getEmail())
            .add("userId", user.getId() + "")
            .add("role", user.getRole())
            .build();

    return Jwts.builder()
            .claims(claims)
            .issuedAt(now)
            .expiration(expiryDate)
            .signWith(getSignInKey(secret))
            .compact();
}
```

Рис. 2. Метод для генерації токена

```
@Override
public boolean validateToken(String token) {
    String secret =
properties.getJwt().secret();
    try {
        Jwts.parser()

.verifyWith(getSignInKey(secret))
                .build()
                .parseSignedClaims(token)
                .getPayload();
        return true;
    } catch (JwtException ex) {
        log.error(ex.getMessage());
        return false;
    }
}
```

Рис. 3. Метод для перевірки токена

Під час розробки API реалізовано механізм хешування паролів користувачів. При реєстрації система обчислює хеш пароля, який зберігається в базі даних. Під час авторизації обчислюється хеш введеного користувачем пароля, який порівнюється із хешем, збереженим у базі даних. Використання хешування замість зберігання паролів у відкритому вигляді значно підвищує безпеку системи.

Також задіяний механізм управління сесіями, який дозволяє зберігати інформацію про користувача між його запитами до сервера. Це важливо для реалізації таких функцій, як авторизація користувача, збереження даних поточного тесту, персоналізація контенту. Сервер генерує JWT-токен (JSON Web Token), який містить інформацію про користувача, і передає його клієнту. Клієнт передає цей токен серверу при кожному запиті у заголовку Authorization. Також використовуються refresh tokens для отримання нових JWT-токенів без необхідності повторної авторизації користувача. Це підвищує зручність використання системи та безпеку, оскільки JWT-токени мають обмежений термін дії.

Розробка функціональності управління навчальними курсами та тестовими завданнями включає в себе наступні складові: створення сутностей Topic та Question для представлення тем та тестових завдань; реалізація CRUD-операцій для курсів (створення, отримання списку, оновлення, видалення курсів) забезпечення зв'язку між курсами та тестовими завданнями (рис. 4).

Створення сутностей Topic та Question для зберігання інформації про теми тестування та самі тестові запитання відбувається на базі таблиць topics (поля id, name) та questions (поля id, text, difficulty, guessing, topic_id).

Застосування зовнішнього ключа topic_id в таблиці questions дозволяє пов'язати кожне тестове завдання з конкретною темою, що є складовою частиною навчального курсу.

Рис. 4. CRUD-операції та методи REST API.

Реалізація CRUD-операцій для курсів (Create, Read, Update, Delete) забезпечує стандартний набір функцій для роботи з даними у базі даних та підтримує зв’язок між курсами і тестовими завданнями. Автоматичне створення CRUD-методів відбувається в результаті застосування концепція репозиторіїв Spring Data JPA (рис. 5).

```
public interface TopicRepository extends
JpaRepository<Topic, Long> {
    Optional<Topic> findByName(String name);
}
```

Рис. 5. Репозиторій для сутності Topic

Реалізація алгоритму адаптивного тестування передбачає застосування обраної моделі адаптивного тестування. Були розглянуті наступні моделі:

- багатоступінчасте тестування на основі моделей лінійного програмування, що використовує критерій максимуму інформації та дозволяє враховувати різноманітні обмеження для різних тестових структур;
- пірамідальне тестування як один з видів багатоступінчастого підходу, де траєкторія відповідей на запитання візуалізується у вигляді піраміди;
- постійна адаптація, у якій зміна порядку пред'явлення тестового завдання відбувається на кожному кроці тестування, тобто після кожної відповіді;
- блокова адаптація, яка передбачає, що рішення про зміну порядку проходження завдань приймається на основі аналізу результатів відповідей випробуваного на спеціальному блоці завдань. При цьому завдання згруповані в блоки, а порядок їх подання всередині кожного блоку залишається фіксованим.

Зазвичай, для адаптивного тестування використовуються одновимірні моделі IRT (Item Response Theory), де рівень знань студента може бути описаний одним параметром theta. Це параметр, що відображає рівень здібностей, знань або навичок особи, є одновимірним континуумом і зазвичай має стан-

дартизовану шкалу, чим вище значення theta, тим вищий рівень здібностей.

Перед початком тестування, theta студента ініціалізується початковим середнім значенням здібностей студента (в даній реалізації запропоновано 0.5).

Процес тестування починається зі створення сутності TestSession для збереження інформації про тестові сесії, розробку та реалізацію алгоритму адаптивного вибору тестових завдань на основі відповідей користувача, реалізацію механізму обчислення результатів тестування з урахуванням складності завдань, збереження історії відповідей користувача.

Реалізована структура даних TestSession, призначена для зберігання всієї необхідної інформації, пов'язаної з кожною окремою сесією тестування. Ця сутність включає атрибути: ідентифікатор сесії, ідентифікатор користувача, час початку та закінчення тесту, поточний стан тесту, поточний номер питання, список питань у тесті, відповіді користувача на кожне питання, оцінка за тест. Використання цієї сутності дозволяє системі коректно відстежувати прогрес кожного користувача під час тестування, зберігати результати та забезпечувати можливість продовження перерваних сесій (рис. 6).

```
@Entity
@Table(name = "test_sessions")
@Data
@NoArgsConstructor
public class TestSession {
    @Id
    @GeneratedValue(strategy =
GenerationType.UUID)
    private UUID id;
    @ManyToOne
    @JoinColumn(name = "user_id")
    private User user;
    @ManyToOne
    @JoinColumn(name = "topic_id")
    private Topic topic;
    @Column(nullable = false, precision = 4,
scale = 2)
    private BigDecimal score = BigDecimal.ZERO;
    @Column(nullable = false)
    private BigDecimal currentTheta =
BigDecimal.valueOf(0.5);
    @OneToMany(mappedBy = "session", cascade =
CascadeType.ALL)
    private List<UserAnswer> answers;

    public TestSession(User user, Topic topic)
{
        this.user = user;
        this.topic = topic;
    }
}
```

Рис. 6. Сутність TestSession в Java

Адаптивне тестування базується на динамічному виборі питань залежно від відповідей користувача. Основна мета цієї логіки - поступово оцінювати рівень знань користувача, використовуючи параметри складності запитань та алгоритм оновлення рівня знань. Запропоновано метод selectNextQuestion(), який обирає наступне питання для тесту, керуючись рівнем знань користувача. Питання підбираються таким чином, щоб їх складність мінімально відрізнялася від поточного рівня користувача (рис. 7).

```
@Override
public QuestionResponseDto
selectNextQuestion(Long userId, UUID sessionId)
{
    TestSession session =
testSessionService.getSessionByIdAndUserId(sess
ionId, userId);
    if (session.getAnswers().size() == 10) {
        throw new
MaxQuestionsAnsweredException(String.format("Us
er %d answered all questions", userId));
    }

    BigDecimal theta =
session.getCurrentTheta();
    return
QuestionMapper.toResponseDto(questionService.ge
tWithMinDifficultyDifference(theta));
}
```

Рис. 7. Метод для отримання нового питання

У процесі адаптивного тестування створюється модель суб'єкта навчання, яка використовується для вибору наступних завдань, що відповідають його рівню підготовки. Таке тестування має задовольняти вимоги щодо регулювання пропорцій легких, середніх і важких завдань, тематичних розділів та рівнів складності, а також передбачати адаптивний перехід на вищі рівні. Процес тестування стає формалізованим і гнучким, що дозволяє проводити адекватну оцінку знань, а можливість регулювання складності завдань базується на використанні статистичних методів для забезпечення валідної апроксимації успішності.

Проведення тестування розробленого API виконується для перевірки його працездатності та функціонування системи управління навчальними курсами в цілому. На цьому етапі здійснюється комплексна перевірка, спрямована на забезпечення коректної роботи API та його відповідності заданим вимогам. Це включає в себе перевірку функціональності окремих компонентів, таких як реєстрація, авторизація, CRUD-операції з курсами, адаптивне тестування, а також інтеграційне тестування для перевірки взаємодії між ними. Особлива увага приділяється тестуванню алгоритму адаптивного вибору тестових завдань та обробці помилок.

На рис. 8 показано, як змінюється поточна оцінка theta в залежності від відповідей студента.

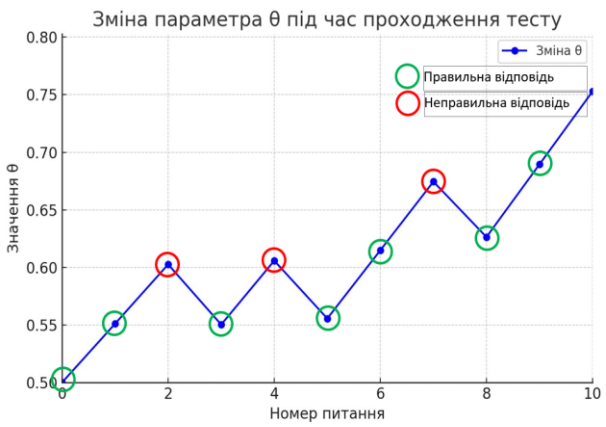

Рис. 8. Зміна оцінки theta в процесі тестування

На основі оцінки theta система обирає наступне завдання, складність якого максимально наближена до поточної оцінки theta студента. Студент відповідає на обране завдання (правильно або неправильно), що враховується у оновленому значенні оцінки theta. Якщо студент відповідає правильно на завдання, складність якого була близькою до його поточної theta, то оцінка theta зростає. Якщо студент відповідає неправильно на завдання, складність якого була близькою до його поточної theta, то оцінка theta зменшується. Масштаб зміни theta залежить від параметрів завдання. Критерієм зупинки тестування є досягнення максимальної кількості завдань.

Таким чином, theta постійно адаптується, прагнучи якомога точніше оцінити реальний рівень знань студента, на невеликій кількісті заданих питань. Це дозволяє системі швидко сходитися до достовірної оцінки, не переобтяжуючи студента надто легкими або надто складними завданнями.

Також перевірено процес взаємодії користувача з системою, починаючи з реєстрації. Розглянуті сценарії успішної та неуспішної реєстрації, а також перевірка наявності облікового запису в базі даних, обробка помилок при введенні невірних даних. Перевірені генерація токенів доступу та їх оновлення.

Головна увага приділена процесу адаптивного тестування.

Детально розглянуто кроки взаємодії, отримання питань, надання відповідей, зміну складності питань залежно від відповідей, обмеження на кількість питань, обчислення та збереження результатів (рис. 9).

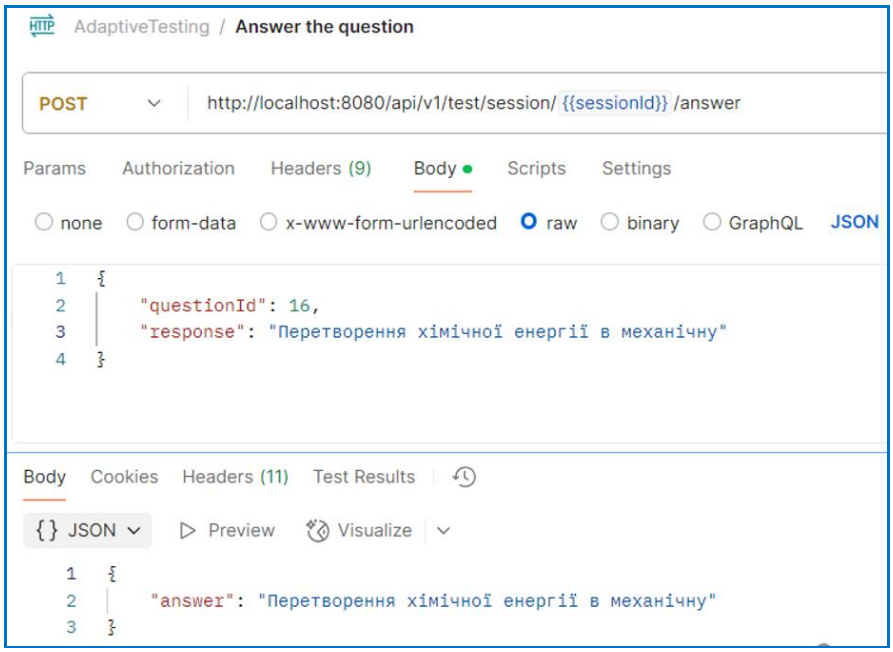

Рис. 9. Процес тестування

Висновки

У результаті проведеного дослідження обґрунтовано і реалізовано API системи управління навчальними курсами з використанням адаптивного тестування, а саме:

- здійснено аналіз сучасних підходів до комп'ютерного тестування, зокрема адаптивних методів, які забезпечують індивідуалізований підхід до оцінювання знань користувачів;
- спроєктовано архітектуру системи та обрано сучасні технології (Spring Boot, REST API, PostgreSQL);
- реалізовано бізнес-логіку відповідно до функціональних вимог;
- розроблено програмний код, що демонструє роботу алгоритмів адаптивного тестування, включаючи вибір завдань на основі попередніх відповідей користувача.

Розроблене програмне забезпечення надає можливість інтеграції адаптивного тестування в освітні системи та платформи дистанційного навчання, що сприяє індивідуалізації навчального процесу та підвищенню ефективності оцінювання знань.

Представлене дослідження має певні обмеження.

По-перше, розроблений API був зосереджений на реалізації базової функціональності адаптивного тестування, і деякі розширені можливості, такі як підтримка різних форматів тестових завдань (наприклад, аудіо- чи відеозапитання), не були включені, API був протестований на обмеженій вибірці даних, і подальша перевірка на більших обсягах може виявити додаткові аспекти, які потребують оптимізації.

По-друге, хоча в роботі розглянуто декілька моделей адаптивного тестування, реалізовано лише одну з них, що обмежує порівняльний аналіз їх ефективності.

Подальші дослідження можуть бути спрямовані на розширення функціональності API, включення підтримки інших моделей адаптивного тестування та проведення масштабніших експериментів для оцінки продуктивності та масштабованості системи.

СПИСОК ЛІТЕРАТУРИ

1. Ревуцька Н.М. "Оцінювання якості знань студентів педагогічних спеціальностей засобами тестового контролю" URL: http://www.narodnaosvita.kiev.ua/?page_id=3381 (дата звернення 30.04.2025)
2. Галузяк В. М. Педагогічна діагностика : курс лекцій / В. М. Галузяк, І. Л. Холковська. – Вінниця : Нілан ЛТД, 2015. – 155 с.
3. Baker, Frank B.; Kim, Seock-Ho. "Item Response Theory: Parameter Estimation Techniques". - Marcel Dekker. - 2004. – 495 p. URL: https://openlibrary.org/books/OL3383823M/Item_response_theory (дата звернення 30.04.2025)
4. Michael Bulmer. "Francis Galton—Pioneer of Heredity and Biometry". Baltimore and London: The Johns Hopkins University Press, 2003. 357 p.
5. Біографічний довідник "Видатні психологи". Київ – 2005. 120 с.
6. Ковальчук Ю.О. "Теорія освітніх вимірювань". – Ніжин 2012. – 200 с.
7. Король Т. Г. "Адаптивний тестовий контроль англомовної компетентності в читанні фахової літератури студентами немовних спеціальностей". Педагогічні науки 126’ 2015 с. 86 – 94.
8. Радкевич О. "Адаптивне тестування в контексті використання електронних засобів навчання: суть, розроблення та оцінювання". Професійна педагогіка 1(26). 2023, С. 58-73.
9. О. Ф. Шевчук, А. А. Яровий, Ю. М. Паночишин, С. І. Петришин, О. А. Козловський. "Моделювання адаптивного тестування знань: поріг ефективності, рівень складності та час виконання завдань" Вісник ВПІ, вип. 1, 2025. С. 104–112.
10. Patni, S., Pro RESTful APIs: Design, Build and Integrate with REST, JSON, XML and JAX-RS, 1st ed., Apress, Santa Clara, California, USA, 2017. 126 p.
11. "What is API: Definition, Types, Specifications, Documentation". 2019. URL: https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/

Received (Надійшла) 17.04.2025
Accepted for publication (Прийнята до друку) 11.06.2025

ВІДОМОСТІ ПРО АВТОРІВ/ ABOUT THE AUTHORS

Ліпчанський Максим Валентинович – кандидат технічних наук, доцент, доцент кафедри комп’ютерної інженерії та програмування, Національний технічний університет "Харківський політехнічний інститут", Харків, Україна;
Maksym Lipchanskyi – Candidate of Technical Sciences, Associate Professor, Associate Professor of the Department of Computer Engineering and Programming, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: Maksym.Lipchanskyi@khpi.edu.ua; ORCID Author ID: https://orcid.org/0000-0003-2837-0444;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57218514897.

Ліпчанська Оксана Валентинівна – кандидат технічних наук, доцент кафедри комп’ютерної інженерії та програмування, Національний технічний університет "Харківський політехнічний інститут", Харків, Україна;
Oksana Lipchanska – Candidate of Technical Sciences, Associate Professor of the Department of Computer Engineering and Programming, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: Oksana.Lipchanska@khpi.edu.ua; ORCID Author ID: https://orcid.org/0000-0003-4173-699X;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57218516877.

Молчанов Георгій Ігоревич - старший викладач кафедри комп’ютерної інженерії та програмування, Національний технічний університет "Харківський політехнічний інститут", Харків, Україна;
Heorhii Molchanov – Senior Lecturer of the Department of Computer Engineering and Programming, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: Heorhii_Molchanov@epam.com; ORCID Author ID: https://orcid.org/0000-0001-9299-461X;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=59921154600.

Христофоров Кирило Андрійович – магістр кафедри комп’ютерної інженерії та програмування, Національний технічний університет "Харківський політехнічний інститут", Харків, Україна;
Kyrylo Khristoforov – Master of the Department of Computer Engineering and Programming, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: Kyrylo.Khrystoforov@cit.khpi.edu.ua; ORCID Author ID: https://orcid.org/0009-0008-3198-8329.

Developing an API for adaptive testing in a learning management system

Maksym Lipchanskyi, Oksana Lipchanska, Heorhii Molchanov, Kyrylo Khrystoforov

Abstract. The article is devoted to the development of an API for a learning course management system with integrated adaptive testing functionality. The relevance of the topic is due to the increasing need for individualization of the educational process and improving the efficiency of knowledge assessment in the current context of the digital transformation of education. The aim of the article is to study the principles and substantiate approaches to creating an effective, secure, and flexible API for managing learning courses. A feature of the API is its support for adaptive testing, taking into account the user's knowledge level. Within the study, an analytical review of knowledge diagnostic systems, adaptive models, and testing strategies was conducted. Based on the results obtained, the system architecture was designed, modern technologies, including Spring Boot and PostgreSQL, were chosen, and the business logic was implemented. Program code demonstrating the operation of adaptive testing algorithms, including the mechanism for selecting test tasks based on the user's previous answers, was developed. The API enables the integration of adaptive testing into various educational platforms, which increases the efficiency and individualization of the learning process. The implementation of the API opens up new opportunities for personalizing learning and improving the quality of educational services.

Keywords: API, REST, IRT, database, adaptive testing, learning course.