УДК 004.75
DOI https://doi.org/10.32782/2521-6643-2024-1-67.7

Олексійчук Ю. Ф., кандидат фізико-математичних наук,
доцент кафедри комп'ютерних наук та інформаційних технологій
Полтавського університету економіки і торгівлі
ORCID: 0000-0002-0585-3307

Ольховський Д. М., кандидат фізико-математичних наук,
доцент кафедри комп'ютерних наук та інформаційних технологій
Полтавського університету економіки і торгівлі
ORCID: 0000-0003-0313-6977

Ольховська О. В., кандидат фізико-математичних наук,
завідувач кафедри комп'ютерних наук
та інформаційних технологій
Полтавського університету економіки і торгівлі
ORCID: 0000-0001-5366-5995

Андрушків О. М., магістр за спеціальністю «Комп'ютерні науки»
Полтавського університету економіки і торгівлі
ORCID: 0009-0000-0919-0152

ПРОЄКТУВАННЯ, РОЗРОБКА ТА ТЕСТУВАННЯ WEB-СЕРВІСУ ДЛЯ ВИБОРУ ТЕМ ДИПЛОМНИХ РОБІТ

У роботі розглядається проєктування, розробка та тестування web-сервісу для вибору тем студентських дипломних робіт. Цей застосунок може бути корисний як викладачам, так і студентам. Web-сервіс реалізує серверну частину програмного забезпечення, яке надає можливості по створенню, збереженню та розподілу тем серед студентів ВНЗ.

REST API – це архітектурний стиль для реалізації веб-сервісів, що ґрунтується на простих та уніфікованих принципах, таких як використання стандартних HTTP-методів для взаємодії з ресурсами через URI, підтримка безстанової комунікації між клієнтом і сервером, підтримка передачі даних у форматах XML і JSON, розділення обов'язків для масштабованості та стійкості систем. Використання REST API спрощує розробку, розгортання та супровід веб-застосунків, роблячи їх більш масштабованими та ефективними.

Web-сервіс написаний на мові програмування Java та використовує сімейство фреймворків Spring: Spring Boot, Spring Data JPA, Spring Web, Spring Security.

Структура проєкту побудована на основі патерну проєктування Controller-Service-Repository, що включає відповідні шари для впорядкування функціональності. У шарі контролера визначається зовнішній інтерфейс сервісу, він відповідає за надання даних клієнтам. Шар сховища відповідає за зберігання та отримання даних. Сервісний шар є місцем, де розташовується вся бізнес-логіка. Якщо бізнес-логіка потребує отримання або збереження даних, вона взаємодіє з репозиторієм. Для отримання доступу до бізнес-логіки клієнти викликають класи з шару контролера. Як сховище даних використовується СКБД PostgreSQL.

Тестування є важливою складовою розробки програмного забезпечення. Для тестування web-сервісу використовуються юніт-тести, що перевіряють окремі компоненти, інтеграційні тести, що перевіряють взаємодію між компонентами, та тести API, які перевіряють працездатність API сервісу. Для тестування REST API використовуються Swagger та Postman. Бібліотека Testcontainers дозволяє створювати тести з використанням реальних залежностей за допомогою контейнерів Docker, що полегшує тестування з використанням реальних сервісів.

Web-сервіс дозволяє автоматизувати процес створення та обрання тем дипломних робіт для викладачів і студентів відповідно.

Ключові слова: *web-сервіс, java, spring, REST API.*

Oleksiichuk Yu. F., Olkhovsky D. M., Olkhovska O. V., Andrushkiv O. M. Designing, developing and testing a web service for selecting thesis topics

The paper discusses the design, development, and testing of a web service for selecting topics for student theses. This application serves both educators and students by facilitating the creation, storage, and distribution of thesis topics among university students.

REST API is an architectural style used to implement web services. It relies on simple and standardized principles, such as using standard HTTP methods to interact with resources via URIs, supporting stateless communication between clients and servers, facilitating data transmission in XML and JSON formats, and dividing responsibilities for scalability and system robustness. Utilizing REST API simplifies the development, deployment, and maintenance of web applications, enhancing their scalability and efficiency.

The web service is developed using the Java programming language and uses the Spring framework, including Spring Boot, Spring Data JPA, Spring Web, and Spring Security. The project structure follows the Controller-Service-Repository design pattern, which organizes functionality into corresponding layers. The controller layer defines the external interface of the service and handles data provision to clients. The repository layer is responsible for data storage and retrieval, while the service layer contains all business logic. If the business logic requires data retrieval or storage, it interacts with the repository. Clients access the business logic by invoking classes from the controller layer. PostgreSQL is used as the data storage.

Testing is a crucial component of software development. The web service will undergo unit tests to verify individual components, integration tests to examine component interaction, and API tests to ensure the functionality of the service's API. Swagger and Postman are used for testing REST API functionality. The Testcontainers library facilitates the creation of tests using real dependencies with Docker containers, simplifying testing with real services.

The web service automates the process of creating and selecting thesis topics for educators and students, respectively.

Key words: *web service, java, spring, REST API.*

Постановка проблеми. Автоматизація різних освітніх процесів є актуальною задачею в сучасному світі. Вибір студентами тем бакалаврських, магістерських робіт та наукових керівників займає багато часу та потребує суттєвих організаційних зусиль. Тому хоча б часткова автоматизація цього процесу є актуальною задачею.

В університеті вже впроваджено ряд програмних систем, що автоматизують складання розкладу [1], розподіл та облік навантаження викладачів, планування навчального процесу [2], вибір студентами індивідуальної освітньої траєкторії тощо. Реалізації системи для вибору тем дипломних робіт у вигляді web-сервісу дозволить просто інтегрувати її у програмні продукти, що вже існують або будуть створюватися.

Розглянемо основні вимоги до системи. Сервіс має підтримувати:

1. Різні функціональні ролі. Супер-адміністратор, адміністратор, куратор, викладач та студент мають різні функціональні можливості та обов'язки.

2. Модель ролей. Вибір між однією роллю на користувача або кількома ролями забезпечить гнучкість та логічність управління доступом.

3. Керування групами. Система включає можливість реєстрації та управління групами студентів, встановленням активного та архівного статусів груп.

4. Ліміти вибору тем. Викладач має можливість задавати ліміти на кількість тем для кожної групи.

5. Реєстрацію користувачів. Реєстрація користувачів з можливістю надавати їм ролі адміністраторів, викладачів чи студентів. Реєстрація може бути як одного користувача (вручну) так і багатьох (шляхом імпорту з файлу).

6. Управління темами. Викладачі можуть створювати, редагувати та видаляти теми дипломних робіт.

7. Статуси тем. Теми можуть мати різні статуси, що відображають їх поточний стан (редагується, невибрана, забронювана, вибрана, затверджена, виконана, невиконана).

8. Кабінети користувачів. Сервіс може бути інтегрований з особистими кабінетами студентів та викладачів з можливістю перегляду та взаємодії з темами.

9. Затвердження тем та генерацію наказу. Механізм затвердження тем куратором або адміністратором, а також автоматична генерація наказу для університетської документації.

10. Інформаційне сповіщення. Можливості для сповіщення викладачів про вибір тем та інші події через email або інші зручні канали.

Аналіз останніх досліджень та публікацій. В останні роки для розробки програмних продуктів з API найчастіше використовується технологія REST (Representational State Transfer) [3-7].

REST API або RESTful API – це архітектурний стиль для реалізації веб-сервісів, який ґрунтується на простих та уніфікованих принципах, таких як використання стандартних HTTP методів для взаємодії з ресурсами через URI, підтримка безстанової комунікації між клієнтом і сервером, підтримка передачі даних у форматах XML і JSON, розділення обов'язків, що сприяє масштабованості та стійкості систем. Використання REST API спрощує розробку, розгортання та супроводження веб-додатків, роблячи їх більш масштабованими та ефективними [8-9].

Можливість простої інтеграції в інші програмні продукти сприяє тому, що web-сервіси широко використовуються для вирішення різноманітних задач [9-10]. Зокрема, web-сервіси є важливими компонентами мікросервісної архітектури [11].

Задача автоматизації розподілу тем дипломних робіт актуальна для кожного навчального закладу, але враховуючи різну архітектуру наявного програмного забезпечення, вирішуватися вона може по різному [12-14]. В [12] розглядається розробка серверної частини застосунку для розподілу тем та керування процесом написання дипломних робіт. В системі є наступні ролі: навчальний персонал, викладач, студент, голова

кафедри. Кожен із учасників виконує власні функції: заповнення навантаження викладачів, заповнення графіку виконання дипломних робіт, створення та редагування тем дипломних робіт, вибір тем, схвалення тем тощо. В [13] описується проєктування та розробка iOS-застосунку для вибору тем робіт. Застосунок має серверну частину, в якій виділяються, зокрема, сутності викладач, студент, дипломна робота. Розробка модуля системи для управлінням процесу написання дипломних робіт розглядається в [14]. В системі виділяються сутності викладач, студент та додатковий персонал.

Метою статті є проєктування, розробка та тестування web-сервісу для управління розподілом тем дипломних робіт. Web-сервіс має підтримувати:

а) можливість реєстрації та авторизації користувачів;

б) різні ролі користувачів з відповідними правами;

в) можливість викладачам пропонувати теми дипломних робіт для вибору;

г) можливість студентам робити вибір;

д) можливість інтеграції з іншими програмними продуктами.

Виклад основного матеріалу. Для реалізації Web-сервісу вибрана мова програмування Java та сімейство фреймворків Spring [15]: Spring Boot, Spring Data JPA, Spring Web, Spring Security. Як сховище даних використовується СКБД PostgreSQL [16].

База даних є необхідним елементом web-сервісу і забезпечує збереження даних. Структура бази даних представлена на рис. 1.

Розглянемо основні таблиці бази даних.

1. admin – зберігає дані користувачів з адміністративними правами, ролями: SUPERADMIN, ADMIN, MENTOR, PROFESSOR, має наступні атрибути:
- username – унікальне ім’я користувача;
- email – електронна адреса;
- full_name – повне ім’я;
- title – регалії, посада тощо;
- password – пароль користувача, який зберігається у вигляді геш-коду отриманого з допомогою функції BCrypt [17];

2. admin_role – зберігає записи з ролями адмін-користувачів, має поля:
- admin_id – зовнішній ключ на id адмін-користувача;
- roles – роль адмін-користувача.

3. student – зберігає дані користувачів з неадміністративними правами, роллю STUDENT, має поля:
- username – унікальне ім’я користувача;
- full_name – повне ім’я;
- password – пароль користувача, який зберігається у вигляді геш-коду отриманого з допомогою функції BCrypt [17];
- group_id – зовнішній ключ на id групи студента.

4. student_group – зберігає записи студентських груп, має поля:
- name – унікальна назва групи;
- full_name – повна назва групи (як у розкладі);
- archived – статус групи (активна чи архівна).

5. group_limit – таблиця для визначення максимальної кількості тем, які викладач може запропонувати студентській групі, має поля:
- group_id – зовнішній ключ на id студентської групи;
- professor_id – зовнішній ключ на id викладача;
- upper_limit – числове значення максимальної кількості тем.

6. thesis – зберігає теми, має поля:
- course_name – формулювання теми курсової роботи;
- diploma_name – формулювання теми дипломної роботи;
- comment – коментар викладача;
- owner_id – зовнішній ключ на id викладача, який є власником теми.

7. offered_thesis – зберігає дані тем, які були запропоновані студентський групі і їх статуси, має поля:
- thesis_id – зовнішній ключ на id запропонованої теми;
- assignee_id – зовнішній ключ на id студента, який працює з темою;
- status – актуальний статус теми, можливі значення: PENDING, OPEN, RESERVED, SELECTED, APPROVED, COMPLETED, UNCOMPLETED.

8. offered_thesis_student_group – таблиця поєднує запропоновані теми зі студентськими групами, має поля:
- offered_thesis_id – зовнішній ключ на id запропонованої теми;
- group_id – зовнішній ключ на id студентської групи.

Web-сервіс передбачає наступні ролі:

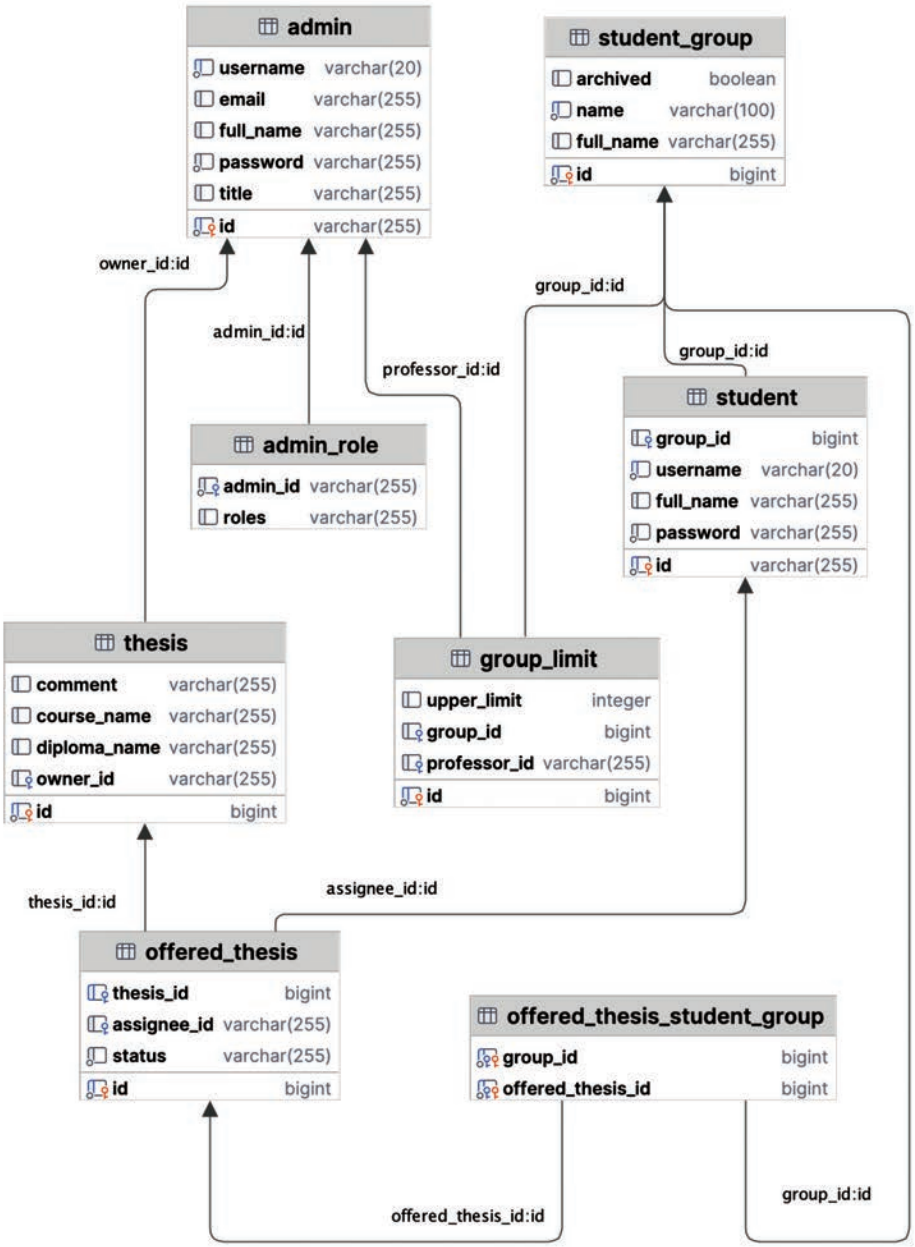

Рис. 1. ER-діаграма бази даних web-сервісу

1. Супер-адміністратор. Має всі права, один в системі.
2. Адміністратор. Основна функція додавати користувачів.
3. Куратор. Має бачити загальну інформацію про те, хто яку тему вибрав, на якому етапі вибір тем.
4. Викладач. Може додавати теми, робити їх видимими для груп.
5. Студент. Може вибирати теми, відноситься до окремої групи.

Кожен студент відноситься до однієї групи. Група може бути:

1) активною, якщо студенти ще здійснюють вибір тем;
2) архівною, якщо студенти вже завершили вибір тем.

Статус змінюється адміном або куратором. Студенти архівної групи можуть тільки подивитися свою тему, інші функції для них недоступні. Назва групи – унікальна.

Якщо студент пише роботу кілька разів (наприклад, бакалаврську і магістерську роботи), то кожного разу буде створюватися новий обліковий запис.

Студенти групи мають ліміт на вибір тем конкретного викладача. При створенні групи потрібно задати її назву. Потім задати ліміти для кожного викладача.

Викладач може створити, змінити та видалити тему. Викладач може додати коментар до теми, який студент буде бачити при виборі теми.

Викладач може вибрати теми з свого списку та запропонувати їх групі. Якщо тема запропонована активній групі, то її не можна змінити чи видалити. Можна запропонувати кільком групам одну й ту ж тему.

Тема може мати наступні статуси:

1. Редагується – тема не запропонована активній групі.

2. Невибрана – тема запропонована активній групі, але її ще ніхто не вибрав.

3. Забронювана – студент може забронювати тему. В цьому статусі інші студенти будуть бачити її, але не зможуть вибрати (буде інший колір або помітка). Після додаткового обмірковування або консультації з викладачем студент приймає рішення: вибирає тему (статус Вибрана) або відмовляється від неї (статус Невибрана).

4. Вибрана – студент вибирає тему, після цього самостійно відмовитися від теми не може, тільки через адміна або викладача у виняткових ситуаціях.

5. Затверджена – куратор, адміністратор чи викладач можуть затвердити тему. Після цього нічого змінювати не можна.

6. Виконана – куратор, адміністратор чи викладач можуть змінити тему на цей статус, якщо студент вже захистив роботу.

7. Невиконана – куратор, адміністратор чи викладач можуть встановити цей статус для теми, якщо студент відрахований або не виконав роботу з іншої причини.

В майбутньому для web-сервісу будуть створені кабінети користувачів або будуть під'єднані інші сервіси. В контролері передбачені методи для таких кабінетів.

1. Кабінет викладача. Викладач бачить всі свої теми, може фільтрувати їх по статусах, додавати нові, редагувати, якщо тема в статусі Редагована, бачить, хто вибрав чи забронював його тему.

2. Кабінет студента. Студент бачить список тем, їх статус, імена викладачів, що запропонували ці теми. Може забронювати тему, вибрати тему. Відмовитися можна від забронюваної, але не від вибраної. Якщо для якогось викладача досягнуто ліміт тем для цієї групи, то його теми вже не можна вибрати чи забронювати.

3. Кабінет куратора. Куратор бачить список студентів групи, теми, які вони вибрали або забронювали, дату останнього входу в кабінет кожного студента. Також куратор може бачити список тем, що запропонував кожний викладач.

Загальний алгоритм роботи з системою може виглядати наступним чином.

1. Автоматично створюється супер-адміністратор після першого запуску системи.

2. Супер-адміністратор додає адміністратора чи адміністраторів.

3. Адміністратори додають викладачів та кураторів.

4. Викладачі працюють зі своїми темами.

5. Адміністратор або куратор створює групу, задає ліміт для кожного викладача.

6. Адміністратор або куратор додає студентів у групу.

7. Викладачі пропонують теми кожній групі. Система відстежує, щоб викладачі запропонували достатню кількість тем. Деякі теми можуть пропонуватися різним групам, деякі – лише одній.

8. Студенти вибирають теми відповідно до порядку описаного вище.

9. Куратор вибирає теми замість студентів, що не зробили вчасно вибір.

10. Куратор або адміністратор затверджує теми. Група стає архівною.

11. Куратор або адміністратор формує наказ відповідно до вибраних тем.

12. Після захисту робіт куратор, адміністратор або викладач може перевести теми в статус виконаних чи невиконаних.

13. Новий семестр, перехід на крок 4.

Структура проєкту побудована на основі патерну проєктування Controller-Service-Repository [18] і складається із відповідних шарів.

Шар контролера (Controller) визначає зовнішній інтерфейс (API) сервісу і відповідає за надання за надання даних клієнтам. Шар сховища (Repository) відповідає за зберігання та отримання даних. Сервісний шар (Service) – це місце де розташовується вся бізнес-логіка. Якщо бізнес-логіка вимагає отримання/збереження даних, вона підключається до Repository. Щоб отримати доступ до бізнес-логіки, клієнти викликають класи з шару Controller.

Шар Controller включає наступні класи: AdminController, GroupController, GroupLimitController, OfferedThesisController, StudentController, ThesisController, UserController. В кожному із класів визначаються методи для роботи із відповідними сутностями.

Шар Service містить такі класи: AdminService, GroupLimitService, GroupService, OfferedThesisService, StudentService, ThesisService, UserService.

Шар Repository складається із інтерфейсів: AdminRepository, GroupLimitRepository, GroupRepository, OfferedThesisRepository, StudentRepository, ThesisRepository, UserRepository.

Сутності (класи) доменної моделі відображаються у відповідні таблиці бази даних, проєкт має наступні класи: Admin, Group, GroupLimit, OfferedThesis, OfferedThesisStatus, Student, Thesis, User, UserRole.

Наприклад, програмний код сутності Group виглядає так:

```
@Getter
@Setter
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(name = «student_group»)
public class Group {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   @Column(name = «name», nullable = false, unique = true, length = 100)
   private String name;
   @Column(name = «full_name»)
   private String fullName;
   @Column(name = «archived»)
   private boolean archived;
}
```

Тестування – невід'ємний етап розробки програмного забезпечення. Для тестування Spring Boot web-сервісів існує кілька типів тестів.

1. Unit-тести. Ці тести перевіряють окремі компоненти програми (наприклад, класи, методи) без залучення залежностей. Для тестування компонентів Spring Boot використовуються фреймворки, такі як JUnit або TestNG.

2. Інтеграційні тести. Ці тести перевіряють взаємодію між різними компонентами застосунку. У випадку Spring Boot це може бути, наприклад, тестування взаємодії сервісів, репозиторіїв та інших компонентів.

3. Тести API. Це тести, які перевіряють працездатність API web-сервісу. Вони можуть включати автоматизовані тести, які взаємодіють з API за допомогою HTTP-запитів і перевіряють повернені відповіді.

Для ручного тестування в проєкт інтегрований сервіс Swagger [19], який дозволяє візуалізувати наявні кінцеві точки (endpoints) і виконати різні запити до web-сервісу. Для виконання HTTP-запитів також використовується застосунок Postman [20]. Postman – це програмний інструмент, який використовується для тестування, розробки та взаємодії з веб-сервісами і API. Він дозволяє користувачам створювати, виконувати та автоматизувати тести API, відправляти HTTP-запити та отримувати відповіді для валідації функціональності та виконання різних сценаріїв. Postman також забезпечує можливості документування API та спільної роботи в команді.

Testcontainers [21] є бібліотекою тестування, що дозволяє створювати тести з використанням реальних залежностей за допомогою одноразових контейнерів Docker. Ця бібліотека надає програмне API для генерації необхідних залежних сервісів у вигляді контейнерів Docker, що дозволяє писати тести, використовуючи реальні сервіси замість макетів. Таким чином, будь-які типи тестів, незалежно від того, чи це модульні тести, тести API або інтеграційні тести, можуть бути реалізовані з використанням реальних залежностей за допомогою однієї і тієї ж моделі програмування.

Висновки з дослідження і перспективи подальших розвідок у цьому напрямі. В роботі розглянуто проєктування, програмну реалізацію та тестування web-сервісу для вибору тем дипломних робіт студентами. Web-сервіс після впровадження дозволить спростити роботу викладачів при підготовці тем дипломних робіт. Для студентів вибір тем стане більш зручним, оскільки його можна буде робити в режимі онлайн. В подальшому планується створення web-версій кабінетів студента, викладача та куратора або інтеграція з іншими системами, що використовуються в університеті.

Список використаних джерел:

1. Ольховська, О. В., Кошова, О. П., Ольховський, Д. М., Семикоз, Д. С. Розробка web-застосунку для формування розкладу в закладі вищої освіти. *Вісник Херсонського національного технічного університету*, 2023, 1 (84), 155-162. https://doi.org/10.35546/kntu2078-4481.2023.1.21

2. Педченко, Н. С. Моніторинг розвитку кадрового потенціалу Вищого навчального закладу Укоопспілки» Полтавський університет економіки і торгівлі». *Актуальні проблеми та перспективи розвитку соціально-трудових відносин в умовах цифрової економіки : матеріали XII Міжнародної науково-практичної конференції.* Полтава : ПУЕТ, 2021, 84-88.

3. Kopecký, J., Fremantle, P., Boakes, R. A history and future of Web APIs. it-Information Technology, 2014, 56(3), 90-97.

4. Bülthoff, F., Maleshkova, M. RESTful or RESTless–current state of today's top web APIs. In The Semantic Web: ESWC 2014 Satellite Events: ESWC 2014 Satellite Events, Anissaras, Crete, Greece, May 25-29, 2014, Revised Selected Papers 11 (pp. 64-74). Springer International Publishing.

5. Зінченко, А. Ю. Проектування розподілених інформаційних системна основі використання технології слабозв'язаних компонентів. *Системи та технології*, 2023, 63 (1), 5-14. https://doi.org/10.32782/2521-6643-2022.1-63.1

6. Олексійчук, Ю. Ф., Ольховська, О. В., Ольховський, Д. М., Орлова, Д. І. Проєктування та розробка web-сервісу для генерування та розсилки pdf-документів. *Системи та технології*, 2023, 65(1), 39-45. https://doi.org/10.32782/2521-6643-2023.1-65.5

7. Кошова, О. П., Ольховська, О. В., Тацій, Д. С., Олексійчук, Ю. Ф., & Черненко, О. О. Розробка веб-додатків та сервісів на платформі node.js. *Таврійський науковий вісник. Серія: Технічні науки*, 2023, (2), 78-89.

8. Fielding, R. T. Architectural styles and the design of network-based software architectures. University of California, Irvine, 2000.

9. Neumann, A., Laranjeiro, N., & Bernardino, J. An analysis of public REST web service APIs. IEEE Transactions on Services Computing, 2018, 14(4), 957-970.

10. Chen, Y., Yang, Y., Lei, Z., Xia, M., Qi, Z. Bootstrapping automated testing for RESTful web services. *In International Conference on Fundamental Approaches to Software Engineering* (pp. 46-66). Cham: Springer International Publishing, 2021, March.

11. Massaga, A., Kouamou, G. E. Towards a Framework for Evaluating Technologies for Implementing Microservices Architectures. *Journal of Software Engineering and Applications*, 2021, 14(8), 442-453.

12. Соболєв, П. О. Автоматизована система керування дипломними роботами на кафедрі (серверна частина) (Bachelor's thesis, КПІ ім. Ігоря Сікорського), 2021.

13. Касьяненко, І. І. iOS додаток управління педагогічними та науковими аспектами роботи кафедри (Master's thesis, Київ), 2018.

14. Штокал, С. С. Модуль «Додатковий персонал» системи управління дипломними проектами (Bachelor's thesis, КПІ ім. Ігоря Сікорського), 2019.

15. Walls, C. Spring in action. Simon and Schuster, 2022.

16. Drake, J. D., Worsley, J. C. Practical PostgreSQL. O'Reilly Media, Inc, 2002.

17. Provos, N., Mazieres, D. Bcrypt algorithm. In USENIX, 1999.

18. Fowler, M. Patterns of enterprise application architecture. Addison-Wesley, 2012.

19. Dos Santos, J. S., Azevedo, L. G., Soares, E. F., Thiago, R. M., da Silva, V. T. Analysis of Tools for REST Contract Specification in Swagger/OpenAPI. In ICEIS (2) (pp. 201-208), 2020.

20. Westerveld, D. API Testing and Development with Postman: A practical guide to creating, testing, and managing APIs for automated software testing. Packt Publishing Ltd, 2021.

21. Sharp, T. R. Testing the Persistence Tier with Testcontainers. In Introducing Micronaut: Build, Test, and Deploy Java Microservices on Oracle Cloud (pp. 81-88). Berkeley, CA: Apress, 2022.

References:

1. Olkhovska, O. V., Koshova, O. P., Olkhovskyi, D. M., Semikoz, D. S. (2023). Development of a web application for creating a schedule in a higher education institution. *Bulletin of the Kherson National Technical University*, 1 (84), 155-162. https://doi.org/10.35546/kntu2078-4481.2023.1.21

2. Pedchenko, N. S. (2021). Monitoring the development of personnel potential of the higher educational institution of the Ucoopspilka «Poltava University of Economics and Trade». *Actual problems and prospects for the development of social and labor relations in the conditions of the digital economy: materials of the 12th International Scientific and Practical Conference. Poltava:* PUET, 84-88.

3. Kopecký, J., Fremantle, P., Boakes, R. (2014). A history and future of Web APIs. it-Information Technology, 56(3), 90-97.

4. Bülthoff, F., Maleshkova, M. (2014). RESTful or RESTless–current state of today's top web APIs. In The Semantic Web: ESWC 2014 Satellite Events: ESWC 2014 Satellite Events, Anissaras, Crete, Greece, May 25-29, 2014, Revised Selected Papers 11 (pp. 64-74). Springer International Publishing.

5. Zinchenko, A. Yu. (2023). The design of distributed information systems is based on the use of the technology of loosely coupled components. *Systems and Technologies*, 63 (1), 5-14. https://doi.org/10.32782/2521-6643-2022.1-63.1

6. Oleksiichuk, Y. F., Olkhovska, O. V., Olkhovskyi, D. M., Orlova, D. I. (2023). Design and development of a web service for generating and sending pdf documents. *Systems and Technologies*, 65(1), 39-45. https://doi.org/10.32782/2521-6643-2023.1-65.5

7. Koshova, O. P., Olkhovska, O. V., Tatsii, D. S., Oleksiichuk, Y. F., & Chernenko, O. O. (2023). Development of web applications and services on the node.js platform. *Taurian Scientific Bulletin. Series: Technical Sciences*, (2), 78-89.

8. Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures. University of California, Irvine.

9. Neumann, A., Laranjeiro, N., & Bernardino, J. (2018). An analysis of public REST web service APIs. IEEE Transactions on Services Computing, 14(4), 957-970.

10. Chen, Y., Yang, Y., Lei, Z., Xia, M., Qi, Z. (2021, March). Bootstrapping automated testing for RESTful web services. In International Conference on Fundamental Approaches to Software Engineering (pp. 46-66). Cham: Springer International Publishing.

11. Massaga, A., & Kouamou, G. E. (2021). Towards a Framework for Evaluating Technologies for Implementing Microservices Architectures. *Journal of Software Engineering and Applications,* 14(8), 442-453.

12. Sobolev, P. O. (2021). Automated management system for theses at the department (server part) (Bachelor's thesis, KPI named after Igor Sikorskyi).

13. Kasyanenko, I. I. (2018). iOS application for managing pedagogical and scientific aspects of the department's work (Master's thesis, Kyiv).

14. Shtokal, S. S. (2019). Module «Additional personnel» of the diploma project management system (Bachelor's thesis, KPI named after Igor Sikorskyi).

15. Walls, C. (2022). Spring in action. Simon and Schuster.

16. Drake, J. D., & Worsley, J. C. (2002). Practical PostgreSQL. O'Reilly Media, Inc.

17. Provos, N., & Mazieres, D. (1999). Bcrypt algorithm. In USENIX.

18. Fowler, M. (2012). Patterns of enterprise application architecture. Addison-Wesley.

19. Dos Santos, J. S., Azevedo, L. G., Soares, E. F., Thiago, R. M., da Silva, V. T. (2020). Analysis of Tools for REST Contract Specification in Swagger/OpenAPI. In ICEIS (2) (pp. 201-208).

20. Westerveld, D. (2021). API Testing and Development with Postman: A practical guide to creating, testing, and managing APIs for automated software testing. Packt Publishing Ltd.

21. Sharp, T. R. (2022). Testing the Persistence Tier with Testcontainers. In Introducing Micronaut: Build, Test, and Deploy Java Microservices on Oracle Cloud (pp. 81-88). Berkeley, CA: Apress.