УДК 004.75
DOI https://doi.org/10.32782/2521-6643-2023.1-65.5

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

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

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

Орлова Д. І., магістр за спеціальністю «Комп'ютерні науки»
Полтавського університету економіки і торгівлі
ORCID: 0009-0004-4764-9859

ПРОЄКТУВАННЯ ТА РОЗРОБКА WEB-СЕРВІСУ ДЛЯ ГЕНЕРУВАННЯ ТА РОЗСИЛКИ PDF-ДОКУМЕНТІВ

В роботі розглядається проєктування та розробка web-сервісу для автоматичної генерації, розсилки на email та перевірки PDF-сертифікатів або інших PDF-документів, який може працювати незалежно або в інтеграції з іншими програмними продуктами. Задача генерації PDF-документу реалізується з допомогою бібліотеки iText, що дозволяє створювати файли різними способами. В цьому проєкті PDF-файли генеруються на основі HTML-шаблонів, що дозволяє просто створювати документи різної розмітки та різного оформлення. Для розсилки електронних листів використовується SMPT-сервер від GMail. Створення листів здійснюється з допомогою бібліотеки JavaMailSender. Для перевірки PDF-сертифікату на справжність є дві можливості. Можна перейти по посиланню, яке розміщене на сертифікаті. У випадку, якщо інформація про такий документ є в базі даних, буде виводитися назва заходу та ім'я учасника. Якщо інформації в базі даних немає, то буде виведена інформація про це. Інший спосіб – введення коду з документу в спеціальне поле на web-сторінці. Програмний продукт розроблений у вигляді web-сервісу, що реалізує підхід REST. Проєкт реалізований на мові програмування Java з використанням сімейства фреймворків Spring: Spring Boot, Spring Data JPA, Spring Web, Spring Security. Проєкт реалізує патерн проєктування Controller-Service-Repository і складається із відповідних рівнів. На рівні entity описані сутності з бази даних у вигляді об'єктів. Тобто для кожної із таблиць в базі даних створений клас, поля якого відповідають атрибутам таблиці. На рівні репозиторію описуються запити до бази даних з використанням інтерфейсу JpaRepository. Логіка роботи web-сервісу реалізована на рівні сервісів. Для виконання кожної із задач написаний окремий клас-сервіс або кілька класів, якщо задача велика і її доцільно розбити на підзадачі. REST-контролери відповідають за зовнішній інтерфейс. Для кожної із дій, які може здійснити користувач або інший застосунок, буде створена окрема адреса доступу (endpoint). Основні дії: створення заходу, отримання списку заходів, видалення заходів, додавання інформації про учасника заходу, отримання списку учасників, генерація сертифікатів, розсилка сертифікатів, автентифікація користувачів, перевірка наявності сертифіката по унікальному коду. Web-сервіс має простий інтерфейс користувача у вигляді web-сторінки, на якій можна перевірити сертифікат на справжність. Для роботи адміністраторів створений кабінет за допомогою бібліотеки Swagger.

Ключові слова: *генерація pdf, web-сервіс, java, smpt, spring.*

Oleksiichuk Yu. F., Olkhovska O. V., Olkhovsky D. M., Orlova D. I. Design and development of a web service for generating and sending PDF documents

The design and development of a web service for automatic generation, email distribution and verification of PDF certificates or other PDF documents are considered in the work. This web service can work independently or in integration with other software products. The task of generating a PDF document is implemented using the iText library, which allows you to create files in various ways. In this project, PDF files are generated based on HTML templates, which allows you to easily create documents with different markup and different design. GMail's SMPT server is used to send e-mails. Letters are created using

the JavaMailSender library. There are two ways to verify the authenticity of a PDF certificate. You can follow the link on the certificate. If information about such a document is available in the database, the name of the event and the name of the participant will be displayed. If there is no information in the database, information about this will be displayed. Another way is to enter the code from the document into a special field on the web page. The software product is developed in the form of a web service that implements the REST approach. The project is implemented in the Java programming language using the Spring family of frameworks: Spring Boot, Spring Data JPA, Spring Web, Spring Security. The project implements the Controller-Service-Repository design pattern and consists of the corresponding levels. At the entity level, entities from the database are described in the form of objects. That is, a class is created for each of the tables in the database, the fields of which correspond to the attributes of the table. At the repository level, requests to the database using the JpaRepository interface are described. The logic of the web service is implemented at the service level. A separate service class or several classes are written to perform each of the tasks, if the task is large and it is advisable to divide it into subtasks. REST controllers are responsible for the external interface. A separate access address (endpoint) will be created for each of the actions that the user or other application can perform. Main actions: creating an event, receiving a list of events, deleting events, adding information about an event participant, receiving a list of participants, generating certificates, sending certificates, authenticating users, checking the presence of a certificate by a unique code. The web service has a simple user interface in the form of a web page where you can check the certificate for authenticity. For the work of administrators, a cabinet was created using the Swagger library.

Key words: *pdf generation, web service, java, smpt, spring.*

Постановка проблеми. Автоматизація різноманітних процесів, зокрема, в освіті, дозволяє суттєво підвищити продуктивність праці, позбавити необхідність виконувати рутинну роботу. В Полтавському університеті економіки і торгівлі створене та інтегроване в систему різноманітне програмне забезпечення, що вирішує ряд задач, що стосуються дистанційного навчання, складання розкладу, розподілу та обліку навантаження викладачів, планування навчального процесу тощо. Водночас залишаються задачі, які також можна автоматизувати. Наприклад, зараз постійно проводиться багато різних заходів: конференцій, тренінгів, семінарів, курсів, за результатами яких учасникам можуть видаватися сертифікати (або інші документи) в PDF-форматі. Якщо учасників заходу досить багато, то ручне генерування сертифікатів для кожного може зайняти багато часу. Наступною задачею є розсилка цих сертифікатів по email, автоматизація якої буде економити час організаторам. Також важливою є проблема захисту PDF-документів від підробок. Враховуючи сучасні можливості графічних редакторів та редакторів PDF-документів, зробити копію та відредагувати файл не займає багато часу. Тому необхідно забезпечити можливість перевірки створених PDF-документів.

Проаналізувавши програмні продукти, що існують, не було знайдено таких, щоб повністю задовольняли всі вимоги. Тому актуальною є розробка програмного забезпечення, для розв'язання описаних задач.

Для більшої універсальності для реалізації програмного забезпечення вибраний підхід створення web-сервісу. Це дозволяє використовувати продукт як незалежно, так і в інтеграції з іншим програмним забезпеченням, використовуючи прикладний програмний інтерфейс (API).

Аналіз останніх досліджень і публікацій. Тенденція останнього десятиліття в розробці програмних продуктів з API полягає у поступовому переході від технологій SOAP (Simple Object Access Protocol) та RPC (Remote Procedure Call) до технології REST (Representational State Transfer) [1–3].

REST-підхід до архітектури мережевих протоколів, які надають доступ до інформаційних ресурсів, що був описаний Роєм Філдінгом в дисертації в 2000 році [4]. Реалізація Web-сервісів REST відповідає чотирьом базовим принципам проектування:

- явне використання HTTP-методів;
- незбереження стану;
- надання URI, аналогічних структурі каталогів;
- передача даних в XML, JavaScript Object Notation (JSON) або в обох форматах [4–5].

Розробники можуть створювати API з використанням кількох архітектур. API-інтерфейси, які відповідають архітектурному стилю REST, називаються REST API. Веб-служби, що реалізують архітектуру REST, називаються веб-службами RESTful. Як правило, термін RESTful API відноситься до мережевих RESTful API. Однак REST API та RESTful API є взаємозамінними термінами.

REST web-сервіси є популярною технологією розробки різноманітних програмних продуктів [5–6]. Основна перевага такого підходу – можливість легкої інтеграції в інші застосунки.

Web-сервіси активно використовуються в розподілених системах, що базуються на мікросервісній архітектурі, що є дуже популярними зараз [7].

Задача створення PDF-документів актуальна і для інших університетів. Так в [8] розглядається розробка мікросервісу для створення PDF-документів. Для роботи мікросервісу використано наступний алгоритм: спочатку отримується запит та вибирається відповідні дані з бази даних. Потім обробляються отримані дані та використовується шаблон для створення HTML-документу. Далі відкривається отриманий документ у вікні браузера та з використанням функціоналу браузера здійснюється його конвертація в PDF. Цей мікросервіс є невід'ємною частиною системи управління бізнес-процесами університету. Він доповнює наявні мікросервіси, такі як відділ кадрів, розклад навчальних занять та електронна залікова книжка, відповідно до вимог системи.

Мета. Мета полягає у проєктуванні та розробці web-сервісу на мові програмування Java за допомогою фреймворку Spring. Основні функції web-сервісу перелічені нижче.

1. Генерація PDF-документів. Сервіс має працювати з використанням технології REST API. Тобто до нього можна буде звертатися з іншого програмного забезпечення або використовувати як окремий застосунок.

2. Автоматична розсилка електронних листів із вкладеними згенерованими сертифікатами. Для цього використовуватиметься сторонній SMTP-сервіс. Робота з відкритим програмним інтерфейсом (API) буде здійснюватися з допомогою бібліотеки JavaMailSender.

3. Перевірка сертифікату: на сертифікаті буде унікальний код, який потрібно ввести на сайті, та гіперпосилання. Якщо сертифікат справжній, то буде виводитися інформація про учасника заходу. Якщо код неправильний, то інформація, що такого коду немає. Це дозволить уникати підробок сертифікатів.

Виклад основного матеріалу. Для розробки Web-сервісу для роботи з PDF-документами вибрана мова програмування Java. Для більш ефективної розробки використовується сімейство фреймворків Spring, а саме Spring Boot, Spring Data JPA, Spring Web, Spring Security. Використання цих технологій дозволяє суттєво скоротити час розробки та підвищити надійність створеного програмного продукту.

Для генерації pdf-файлів існує ряд бібліотек.

а) бібліотека Adobe PDF. Ця бібліотека надає API такими мовами програмування, як C++, .NET та Java. Можна редагувати, переглядати, роздруковувати та витягувати текст із PDF-файлів;

б) процесор форматування об'єктів – Форматер друку з відкритим вихідним кодом, керований об'єктами форматування XSL та незалежним форматом виводу;

в) PDF Box – Apache PDFBox. Це бібліотека Java з відкритим вихідним кодом, підтримує розробку та перетворення документів PDF. Використовуючи цю бібліотеку можна розробляти Java-програми, які створюють, конвертують та обробляють документи PDF;

г) звіти Jasper. Це інструмент звітів Java, який генерує звіти у форматі PDF, включаючи Microsoft Excel, RTF, ODT, значення через кому та файли XML;

д) iText – це бібліотека Java, за допомогою якої можна розробляти програми, які створюють, конвертують і обробляють документи PDF [9–10].

Після аналізу всіх переваг та недоліків бібліотек, вибрана iText. Головними перевагами цієї бібліотеки є детальна документація, велика кількість прикладів, різні варіанти генерації PDF-документів, можливість генерації на основі HTML-шаблонів.

Web-сервіс використовує базу даних MySQL для надійного збереження необхідних даних. Самі PDF-документи не будуть зберігатися в базі даних, але вся необхідна для їх генерації інформація буде збережена. Це дозволить суттєво скоротити кількість пам'яті, що використовується. При потребі новий документ може буди повторно згенерований і відправлений на електронну пошту. Основні таблиці бази даних (рис. 1) описані далі.

1. Event – тут буде зберігатися інформація про заходи, учасникам яких будуть генеруватися та розсилатися PDF-сертифікати. В атрибутах таблиці зберігається інформація про назву та дату заходу, а також дані, що будуть використані при генерації сертифікату: заголовок, кількість годин, та довільний текст, який забажає організатор заходу. Для тексту передбачено 2 поля, тому інформація може міститися в двох різних місцях документу, але не обов'язково використовувати всі поля, частину із них можна залишити порожніми. Також в цій таблиці міститься інформація про шаблон, на основі якого буде генеруватися PDF-документ.

2. Certificate – в цій таблиці буде зберігатися інформація про учасників заходів. Основні атрибути:

- full_name – повне ім'я;
- email – електронна адреса;
- personal_info – інша інформація про учасника, наприклад, місце роботи, посада тощо;
- date – дата останньої успішної генерації та надсилання PDF-документу. Використовується для уникнення помилкових повторних розсилок;
- has_link – інформація про те, чи потрібно, на документі показувати посилання та код для перевірки на справжність. Якщо значення true, то унікальний код буде згенерований і посилання буде відображатися на документі, якщо false, то ні;
- link – посилання для перевірки справжності PDF-документа, використовується для захисту від підробок у тому випадку, коли значення атрибуту has_link дорівнює true.

3. Users – в цій таблиці буде зберігатися інформація про користувачів сервісу, яка необхідна для автентифікації. Основні атрибути:

- email – адреса електронної пошти користувача, яка також буде використовуватися як логін при автентифікації;
- password – пароль користувача, який зберігається є вигляді геш-коду отриманого з допомогою функції BCrypt [11];
- status – статус користувача, що може бути активним або неактивним. Неактивний користувач не може здійснювати жодних дій в системі;

• role – роль користувача в системі. На даний час можливі 3 ролі: супер-адміністратор, адміністратор та звичайний користувач, в майбутньому за потреби кількість ролей може бути розширена. Роль визначає можливості та обмеження користувача в системі.

4. Code – тут будуть зберігатися унікальні ідентифікатори для сертифікатів. Основним атрибутом тут є uid – унікальний текстовий ідентифікатор, що використовується для перевірки PDF-документу на справжність. Ця таблиця логічно не зв'язана з іншими і використовується лише для зберігання цих ідентифікаторів.

5. Таблиця user_to_event є допоміжною таблицею для реалізації зв'язку «багато до багатьох» між таблицями users та event.

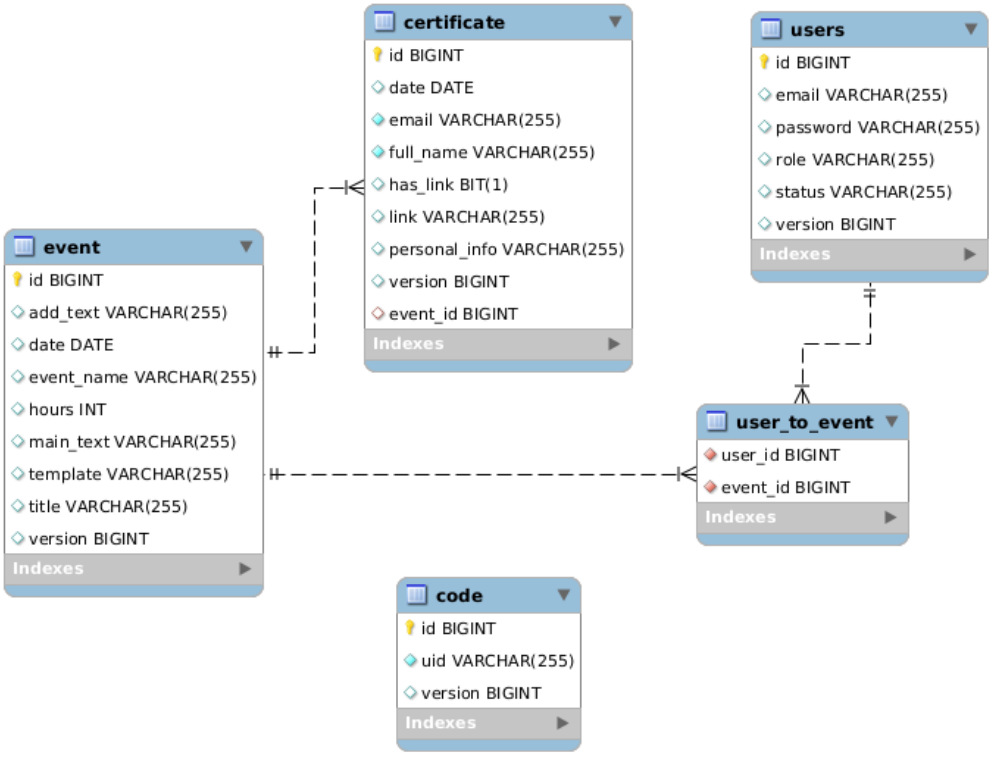

Рис. 1. ER-діаграма бази даних

Структурно код web-сервісу має багатошарову організацію, кожен рівень (layer) якої відповідає за певний функціонал. Застосунок реалізує патерн проєктування Controller-Service-Repository [12-13].

На рівні entity (model) описуються сутності з бази даних у вигляді об'єктів. Тобто для кожної із таблиць бази даних, окрім user_to_event, буде створений клас, поля якого відповідають атрибутам таблиці. Це потрібно для об'єктно-реляційного відображення даних. Наприклад, код класу User, що відповідає таблиці users:

```
@Entity
@Table(name = "users")
@Getter
@NoArgsConstructor
@EqualsAndHashCode(of = "id")
@FieldDefaults(level = PRIVATE)
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO, generator = "native")
    @GenericGenerator(name = "native", strategy = "native")
    @Column(updatable = false)
    Long id;
    @Version
    @Column(updatable = true)
    Long version;
```

```
    String email;
    String password;
    @Enumerated(EnumType.STRING)
    UserStatus status;
    @Enumerated(EnumType.STRING)
    UserRole role;
    @ManyToMany
    @JoinTable(
        name = "user_to_event",
        joinColumns = @JoinColumn(name = "user_id"),
        inverseJoinColumns = @JoinColumn(name = "event_id"))
    List<Event> events;
    public User(String email, String password) {
      this.email = email;
      this.password = password;
    }
}
```

Для зменшення кількості коду використовуються анотації бібліотеки lombok (@Getter, @NoArgsConstructor, @EqualsAndHashCode, @FieldDefaults). Ці анотації створюють гетери для всіх полів, конструктор без параметрів, перевизначають методи equals() та hashCode() та роблять всі поля приватними відповідно. Призначення інших анотацій:

- анотації @Entity та @Table зв'язують клас із відповідною таблицею в базі даних;
- анотації @Id, @GeneratedValue та @GenericGenerator призначені для створення первинного ключа в таблиці, автоматичної генерації його значення та прив'язуванні до відповідного поля класу;
- анотація @Column дозволяє встановити властивості для атрибуту в базі даних;
- анотації @ManyToMany та @JoinTable створюють зв'язок типу «багато до багатьох» з іншою таблицею бази даних.

На рівні репозиторію реалізуються запити до бази даних. Оскільки в проєкті використовується фреймворк Spring Data JPA, то базовий функціонал для роботи з базами даних (збереження, отримання всіх записів із таблиці, отримання запису по id тощо) реалізований автоматично. Для цього для кожної сутності створюється інтерфейс, що успадковує узагальнений інтерфейс JpaRepository. Для інших запитів потрібно створювати абстрактні методи, реальні запити будуть генеруватися автоматично фреймворком Spring Data JPA, використовуючи назву методу.

Приклад репозиторію:

```
public interface CertificateRepository extends JpaRepository<Certificate, Long> {
  List <Certificate> findAllByEvent(Event event);
}
```

В даному випадку метод буде повертати всі записи в базі даних, пов'язані із конкретним заходом.

Логіка роботи web-сервісу реалізована на рівні сервісів (services). Для виконання кожної із задач написаний окремий сервіс або кілька сервісів, якщо доцільно розбити задачу на підзадачі. Основні задачі: генерація PDF-документу, надсилання email, робота з таблицями бази даних, автентифікація користувачів тощо.

Контролери відповідають за зовнішній інтерфейс. Для кожної із дій, які може здійснити користувач або інший застосунок, створена окрема адреса доступу (endpoint). Основні дії: створення події, отримання списку подій, видалення події, додавання інформації про учасника заходу, отримання списку учасників, генерація сертифікатів, розсилка сертифікатів, автентифікація користувачів, перевірка наявності сертифіката по унікальному коду.

Web-сервіс має простий інтерфейс користувача у вигляді web-сторінки. Для роботи адміністраторів створений кабінет за допомогою бібліотеки Swagger [13]. Ця бібліотека вибрана тому, що для створення простого кабінету достатньо лише підключити її до проєкту та написати кілька рядків коду. Незважаючи на простоту інтерфейсу, Swagger дозволяє виконувати адміністратору всі необхідні дії.

Для доступу до Swagger користувач має пройти автентифікацію. Електронна адреса використовується як логін.

Система не передбачає самостійну реєстрацію користувача. Лише користувач з правами адміністратора може зареєструвати нового користувача. Для цього він має ввести email нового користувача. Новому користувачу на цей email прийде інформація про успішну реєстрацію та пароль. Після цього користувач може пройти автентифікацію та змінити пароль на новий.

Звичайний користувач може створювати заходи, додавати учасників цих заходів, переглядати інформацію про учасників та заходи, розсилати сертифікати. Користувач має доступ лише до заходів, що він створив, та до тих, права роботи з якими надав адміністратор.

Адміністратор, крім прав звичайного користувача, може реєструвати користувачів, блокувати їх та надавати користувачам доступ до заходів. Адміністратор має доступ до всіх заходів.

Супер-адміністратор, на додачу до прав адміністратора, може змінювати роль учасників: звичайного користувача зробити адміністратором і навпаки.

Для генерації та розсилки PDF-документів користувач повинен зробити наступні дії.

1. Створити новий захід.
2. Додати інформацію про учасників цього заходу. Передбачена можливість імпорту даних із файлів.
3. Відправити лист з PDF-документом. Можна відправляти як одному конкретному учаснику так і всім учасникам деякого заходу.

Також додатково є можливість завантажити PDF-документ через Swagger.

Для розгортання сервісу необхідний будь-який Linux-сервер зі встановленими програмами JDK17 та Maven. Spring Boot включає в себе вбудований контейнер сервлетів Tomcat. Окремо необхідно мати доступ до MySql бази даних.

Можливо також створити Docker-образ та запускати web-сервіс в як контейнер в Docker [15].

Також проєкт може бути дещо модифікований і використовуватися як один із мікросервісів у системі.

Висновки з дослідження і перспективи подальших розвідок у цьому напрямі. В роботі розглянуто проєктування та програмна реалізацію web-сервісу для генерації PDF-сертифікатів. Web-сервіс дозволяє автоматично генерувати, розсилати та перевіряти сертифікати. Діапазон застосування може буди досить широким, наприклад, обслуговування конференцій, семінарів, дистанційних курсів, інших заходів. На даний час web-сервіс перебуває на завершальному етапі тестування та розробки. В подальшому можливості web-сервісу можуть бути покращені за рахунок створення кабінету користувача з використанням можливостей сучасних JavaScript-бібліотек та фреймворків, який буде більш зручним ніж Swagger.

Основні переваги Web-сервісу:

1. Можливість роботи як окремого web-застосунку.
2. Можливість інтеграції з іншими програмними продуктами, використовуючи REST API.
3. Простота у розгортанні та використанні.

Список використаних джерел:

1. Kopecký, Jacek, Paul Fremantle, and Rich Boakes. A history and future of Web APIs. *IT-Information Technology*. No. 3, 2014. P. 90–97.
2. Bülthoff, Frederik, and Maria Maleshkova. 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, P. 64–74.
3. Зінченко А. Ю. Проектування розподілених інформаційних систем на основі використання технології слабозв'язаних компонентів. *Системи та технології, 1(63)*, 2022. С. 5–14. https://doi.org/10.32782/2521-6643-2022.1-63.1
4. Fielding, Roy Thomas. Architectural styles and the design of network-based software architectures. University of California, Irvine, 2000.
5. Neumann, Andy, Nuno Laranjeiro, and Jorge Bernardino. An analysis of public REST web service APIs. *IEEE Transactions on Services Computing* 14, No. 4, 2018, P. 957–970.
6. Chen, Yixiong, Yang Yang, Zhanyao Lei, Mingyuan Xia, and Zhengwei Qi. Bootstrapping automated testing for RESTful web services. *Fundamental Approaches to Software Engineering: 24th International Conference, FASE 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 – April 1, 2021*, Proceedings 24, pp. 46–66. Springer International Publishing, 2021.
7. Massaga, Aristide, and Georges Edouard Kouamou. Towards a Framework for Evaluating Technologies for Implementing Microservices Architectures. *Journal of Software Engineering and Applications 14*, No. 8, 2021, P. 442–453.
8. Коломієць О. Е. Проєктування та розроблення сервісної архітектури управління бізнес-процесами університету. Генерація PDF документів засобами системи : кваліфікаційна робота на здобуття ступеня вищої освіти «магістр». Херсонський держ. ун-т. Херсон : ХДУ, 2021. 54 с.
9. Lowagie, Bruno. IText in Action. Simon and Schuster, 2010.
10. Bluck, Alan S. PDF Document Creation Using Java. In IBM Software Systems Integration: *With IBM MQ Series for JMS, IBM FileNet Case Manager, and IBM Business Automation Workflow*, p. 991-1269. Berkeley, CA: Apress, 2023. https://doi.org/10.1007/978-1-4842-8861-0_6
11. Provos, Niels, and David Mazieres. Bcrypt algorithm. *USENIX*. 1999.
12. Fowler, Martin. Patterns of Enterprise Application Architecture: Pattern Enterpr Applica Arch. Addison-Wesley, 2012.
13. Walls, Craig. Spring in action. Simon and Schuster, 2022.
14. Dos Santos, Jéssica Soares, Leonardo Guerreiro Azevedo, Elton FS Soares, Raphael Melo Thiago, and Viviane Torres da Silva. Analysis of Tools for REST Contract Specification in Swagger/OpenAPI. *ICEIS (2),* 2020. P. 201–208.
15. Miell, Ian, and Aidan Sayers. Docker in practice. Simon and Schuster, 2019.

References:

1. Kopecký, Jacek, Paul Fremantle, and Rich Boakes (2014). A history and future of Web APIs. *IT-Information Technology*. No. 3. P. 90–97.

2. Bülthoff, Frederik, and Maria Maleshkova (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,* Revised Selected Papers 11, P. 64–74.

3. Zinchenko A. Yu. (2022) Proektuvannia rozpodilenykh informatsiinykh system na osnovi vykorystannia tekhnolohii slabozviazanykh komponentiv [Design of distributed information systems based on the use of the technology of loosely coupled components]. *Systemy ta tekhnolohii – Systems and Technologies*, 1(63). P. 5–14. https://doi.org/10.32782/2521-6643-2022.1-63.1 [in Ukrainian].

4. Fielding, Roy Thomas (2000). Architectural styles and the design of network-based software architectures. University of California, Irvine.

5. Neumann, Andy, Nuno Laranjeiro, and Jorge Bernardino (2018). An analysis of public REST web service APIs. *IEEE Transactions on Services Computing* 14, No. 4, P. 957–970.

6. Chen, Yixiong, Yang Yang, Zhanyao Lei, Mingyuan Xia, and Zhengwei Qi (2021). Bootstrapping automated testing for RESTful web services. *Fundamental Approaches to Software Engineering: 24th International Conference, FASE 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 – April 1,* Proceedings 24, pp. 46–66. Springer International Publishing.

7. Massaga, Aristide, and Georges Edouard Kouamou (2021). Towards a Framework for Evaluating Technologies for Implementing Microservices Architectures. *Journal of Software Engineering and Applications 14*, No. 8, P. 442–453.

8. Kolomiiets O. E. (2021). Proiektuvannia ta rozroblennia servisnoi arkhitektury upravlinnia biznes-protsesamy universytetu. Heneratsiia PDF dokumentiv zasobamy systemy: kvalifikatsiina robota na zdobuttia stupenia vyshchoi osvity «mahistr» [Design and development of the university's business process management service architecture. Generation of PDF documents by means of the system: qualifying work for obtaining the degree of higher education "master"]. Khersonskyi derzh. un-t. Kherson : KhDU, 54 p.

9. Lowagie, Bruno (2010). IText in Action. Simon and Schuster [in English].

10. Bluck, Alan S (2023). PDF Document Creation Using Java. In IBM Software Systems Integration: *With IBM MQ Series for JMS, IBM FileNet Case Manager, and IBM Business Automation Workflow*, p. 991–1269. Berkeley, CA: Apress. https://doi.org/10.1007/978-1-4842-8861-0_6

11. Provos, Niels, and David Mazieres (1999). Bcrypt algorithm. *USENIX*. [in English].

12. Fowler, Martin (2012). Patterns of Enterprise Application Architecture: Pattern Enterpr Applica Arch. Addison–Wesley.

13. Walls, Craig (2022). Spring in action. Simon and Schuster.

14. Dos Santos, Jéssica Soares, Leonardo Guerreiro Azevedo, Elton FS Soares, Raphael Melo Thiago, and Viviane Torres da Silva (2020). Analysis of Tools for REST Contract Specification in Swagger/OpenAPI. *ICEIS (2)*. P. 201–208.

15. Miell, Ian, and Aidan Sayers (2019). Docker in practice. Simon and Schuster.