Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента	*Швеця Владислава Ігоровича*
	(ПІБ)
академічної групи	*121М-22-1*
	(шифр)
спеціальності	*121 Інженерія програмного забезпечення*
	(код і назва спеціальності)
освітньої програми	*«121 Інженерія програмного забезпечення»*
	(назва освітньої програми)
на тему:	*Розробка та дослідження програмного забезпечення серверної частини веб-застосунку для онлайн курсів*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	*проф. Лактіонов І. С*			

Рецензент				

Нормоконтролер	*проф. Лактіонов І. С.*			

Дніпро
2023

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри

Програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 20 23 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

студенту *121М-22-1* *Швецю Владиславу Ігоровичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Розробка та дослідження програмного забезпечення серверної частини веб-застосунку для онлайн курсів*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 09.10.2023 р. № 1227-с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процеси та технології, пов'язані з розробкою серверної частини веб-застосунку, враховуючи особливості онлайн навчання.

Предмет досліджень – серверна частина веб-застосунку для онлайн курсів, яка включає в себе аспекти її розробки та функціональності за допомогою мови програмування Java.

Мета НДР – розробка та вдосконалення серверної частини веб-застосунку для онлайн курсів, з фокусом на покращенні функціональності, ефективності та забезпеченні зручного користувацького досвіду.

Вихідні дані для проведення роботи – теоретичні та експериментальні дослідження для вирішення задач проектування інформаційної системи застосунку та підходи до вибору технологій реалізації.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень полягає в розробці та оптимізації серверної частини веб-застосунку для онлайн курсів, а також вивченні та впровадженні перед.

Практична цінність результатів полягає у тому, що отримані в ході дослідження результати роботи має практичне значення для освітніх установ, корпоративних тренінгових центрів та індивідуальних користувачів.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень повинні бути подані у вигляді, який дозволяє побачити та оцінити безпосереднє використання технологій серверної сторіни розробки. В результаті роботи повинен бути розроблений програмний комплекс для веб-застосунку з онлайн курсами

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі	05.09.2023-24.09.2023
Збір, дослідження та систематизація інформації щодо проектування та реалізації безпечних функціональних застосунків на Java	25.09.2023-11.10.2023
Розробка і тестування автоматизованої системи для вирішення задачі забезпечення автоматизації управління об'єктами для тестування	12.10.2023-07.11.2023

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи очікується позитивним завдяки адаптивності даної системи до процесу тестування програмного продукту будь-якої установи, таким чином зменшити затрати на заробітну плату людям, які виконують мануальне тестування.

Соціальний ефект від реалізації результатів роботи очікується позитивним завдяки можливості вивчення та використання розробленої моделі ІС для вирішення задач проектування застосунків управління ресурсами тестування в корпоративних системах. Створена на нових технологіях архітектура інформаційної системи, що має стабільну та перевірену систему авторизації та безпеки, є ефективною та дозволить легко масштабувати та інтегрувати новий функціонал.

Завдання видав _____ *Лактіонов І.С.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Швець В.І.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *05.09.2023 р.*

Термін подання кваліфікаційної роботи до ЕК *15.12.2023*

РЕФЕРАТ

Пояснювальна записка: 95 с., 34 рис., 2 дод., 51 джерела.

Об'єкт розробки: процеси та технології, пов'язані з розробкою серверної частини веб-застосунку, враховуючи особливості онлайн навчання.

Предмет дослідження: серверна частина веб-застосунку для онлайн курсів, яка включає в себе аспекти її розробки та функціональності.

Мета кваліфікаційної роботи: розробка та вдосконалення серверної частини веб-застосунку для онлайн курсів, з фокусом на покращенні функціональності, ефективності та забезпеченні зручного користувацького досвіду.

Методи дослідження. Для виконання поставлених завдань використовуються різноманітні методи, зокрема аналіз літератури, проектування архітектури системи, програмування, тестування та апробація результатів.

Наукова новизна полягає в реалізації передових технологій, таких як асинхронне програмування, масштабованість та ефективна обробка даних, робить дане дослідження унікальним та передовим у контексті вирішення завдань, пов'язаних із створенням та управлінням онлайн курсами.

Практичне значення роботи. Розробка ефективної серверної частини веб-застосунку для онлайн курсів має практичне значення для освітніх установ, корпоративних тренінгових центрів та індивідуальних користувачів. Забезпечення стабільної роботи системи, оптимізованої для великої кількості користувачів, покращує якість освітнього процесу та забезпечує зручний та доступний інтерфейс для взаємодії.

Ключові слова: REST API, веб-застосунок, фреймворк, сервер, Spring Framework, http-запит, http-відповідь, інформаційна система, база даних, криптографія.

ABSTRACT

Explanatory note: 95 pages, 34 pictures, 2 appendices, 51 sources.

Object of research: processes and technologies related to the development of the server part of the web application, taking into account the peculiarities of online education.

Subject of research: the backend of a web application for online courses, which includes aspects of its development and functionality.

Purpose of Master's thesis: development and improvement of the server part of the web application for online courses, with a focus on improving functionality, efficiency and providing a convenient user experience.

Research methods. A variety of methods are used to perform the assigned tasks, including literature analysis, system architecture design, programming, testing and approbation of results.

Originality of research consists in the implementation of advanced technologies, such as asynchronous programming, scalability and efficient data processing, makes this research unique and advanced in the context of solving tasks related to the creation and management of online courses.

Practical value of the results. Developing an effective back-end web application for online courses is of practical importance for educational institutions, corporate training centers and individual users. Ensuring the stable operation of the system, optimized for a large number of users, improves the quality of the educational process and provides a convenient and accessible interface for interaction.

Keywords: REST API, web application, framework, server, Spring Framework, http-request, http-response, information system, database, cryptography.

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

СУБД – система управління базою даних;

UI – user interface;

GUI – graphical user interface;

СУБД – система управління базами-даних;

SQL – structured query language;

ПК – персональний комп’ютер;

ОС – операційна система;

HTTP – hyper text transfer protocol;

API – application programming interface;

ПЗ – програмне забезпечення.

ЗМІСТ

ВСТУП

Актуальність дослідження. Актуальність даного дослідження зумовлена невпинною динамікою розвитку сучасного освітнього простору та ростом популярності онлайн-освіти.

Розробка та вдосконалення серверної частини веб-застосунку для онлайн курсів є критично важливим завданням у забезпеченні якості та ефективності цих освітніх платформ. За допомогою високоякісної та ефективної серверної частини можна забезпечити стабільну роботу платформи, зручність взаємодії для користувачів та ефективне управління навчальним процесом.

З урахуванням зростаючого попиту на онлайн навчання та конкуренції в цьому сегменті, розробка оптимальної серверної частини веб-застосунку відкриває широкі можливості для поліпшення якості освіти та забезпечення конкурентоспроможності на ринку освітніх послуг. Дане дослідження пропонує вагомий внесок у вдосконалення та оптимізацію цього процесу, щоб найбільш ефективно відповісти на сучасні виклики в галузі освіти та технологій.

Мета дослідження: розробка та вдосконалення серверної частини веб-застосунку для онлайн курсів, з фокусом на покращенні функціональності, ефективності та забезпеченні зручного користувацького досвіду..

Завдання дослідження: для досягнення поставленої мети в роботі було передбачено вирішення наступних завдань:

1. Розробка серверної частини веб-застосунку для забезпечення стабільної роботи під час одночасної роботи багатьох користувачів;

2. Впровадження можливості взаємодії з онлайн курсами, включаючи функції реєстрації, авторизації та ведення статистики користувачів;

3. Проаналізувати доступні інструменти підключення до сховища даних та захисту комунікації, які доступні в IntelliJ IDEA;

4. Спроектувати та розробити відповідне програмне забезпечення, яке буде здатним до розгортання на різних конфігураціях ОС;

5. Перевірити ефективність системи та зробити висновки щодо доцільності її створення.

Об’єкт дослідження: процеси та технології, пов'язані з розробкою серверної частини веб-застосунку, враховуючи особливості онлайн навчання.

Предмет дослідження: серверна частина веб-застосунку для онлайн курсів, яка включає в себе аспекти її розробки та функціональності за допомогою мови програмування Java.

Методи дослідження. Для виконання поставлених завдань використовуються різноманітні методи, зокрема аналіз літератури, проектування архітектури системи, програмування, тестування та апробація результатів.

Наукова новизна полягає в розробці серверної частини веб-застосунку для онлайн курсів, пропонуючи новаторські підходи до оптимізації та покращення функціоналу системи.

Практичне значення. Проведене дослідження має практичне значення для освітніх установ, корпоративних тренінгових центрів та індивідуальних користувачів. Забезпечення стабільної роботи системи, оптимізованої для великої кількості користувачів, покращує якість освітнього процесу та забезпечує зручний та доступний інтерфейс для взаємодії..

Особистий внесок автора:

1. Наукові результати роботи отримані автором самостійно.

2. Розробка теоретичної частини роботи з дослідження і систематизування знань про існуючі підходи розробки;

4. Оцінка отриманих результатів.

Структура і обсяг роботи. Робота складається з вступу, трьох розділів і висновків. Містить 89 сторінку, в тому числі 60 сторінок тексту основної частини з 34 рисунками, списку використаних джерел з 51 найменуваннями на 5 сторінках, 3 додатка на 24 сторінках.

РОЗДІЛ 1.
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Життєвий цикл програмного забезпечення

Життєвий цикл програмного забезпечення є ключовою концепцією, яка охоплює період від моменту виникнення ідеї щодо створення програмного продукту до його завершення та подальшої підтримки. Цей цикл можна уявити як складну структуру, що включає в себе різноманітні процеси та завдання, які видаються та виконуються протягом розробки, використання та супроводу програмного продукту. Важливо зауважити, що ця концепція може бути відобразжена різними моделями, які, з свого боку, можна узагальнено розділити на три групи: інженерний підхід, з урахуванням специфіки задачі та сучасні технології швидкої розробки.

Однією з перших моделей розробки програм є каскадна модель життєвого циклу, часто називана "водоспадом"(рис 1.1) [1]. Ця модель, представлена ще у 1980-х роках, вражає послідовністю етапів, де перехід між ними є строго послідовним та неповоротним. За її основним принципом — завершення попереднього етапу перед переходом до наступного. Модель "водоспад" застосовується в проектах, де є потреба в чіткому плануванні термінів та вартості кожного етапу. Однак вона має недоліки у відносності реалізації, коли розробка системи рідко відбувається згідно з жорсткою схемою. Однак набагато відчутнішим є мінус нереалістичності даної моделі, коли на практиці розробка системи майже ніколи не проходить строго відповідно до негнучкої, заздалегідь продуманої схеми. Належить до першої групи моделей.

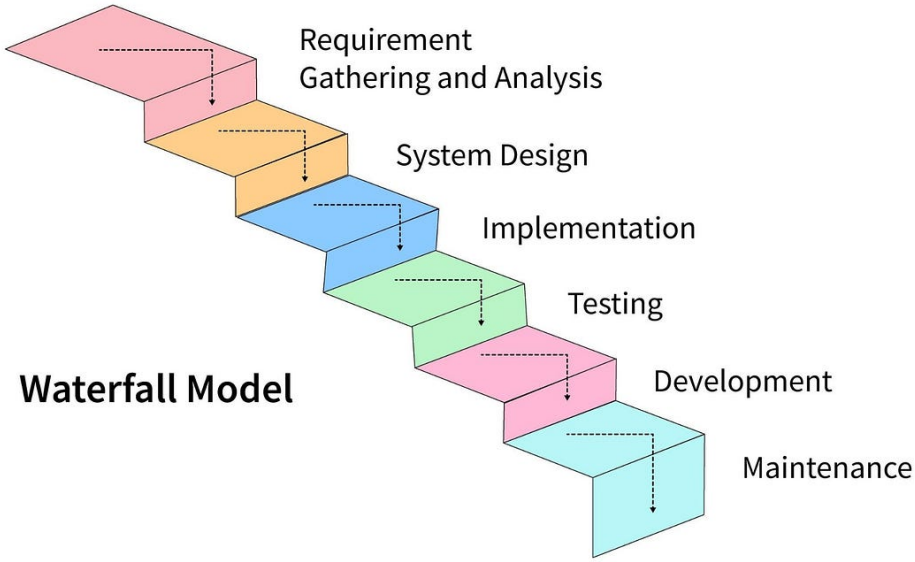

Рис. 1.1. Водоспадна модель життєвого циклу програмного забезпечення

Вдосконаленням каскадної моделі стала V-модель, представляючи розробку через тестування. (рис 1.2) [2].

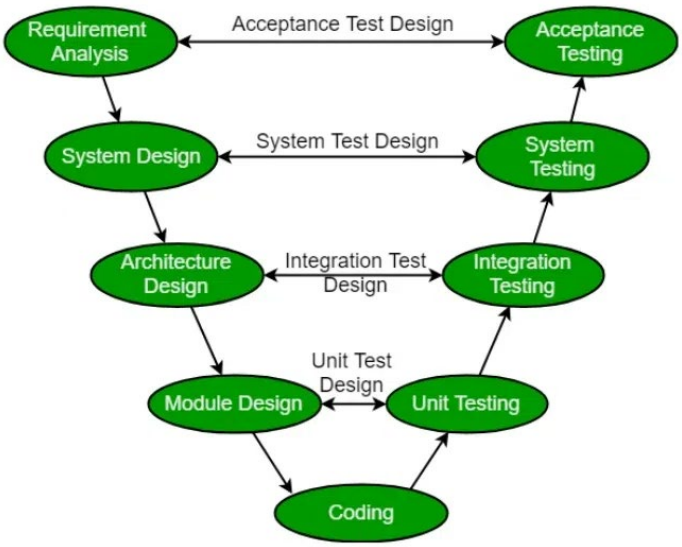

Рис. 1.2. V-подібна модель життєвого циклу програмного забезпечення

В представленій моделі, процес розробки зображено у формі знижувальної послідовності на лівій частині умовної літери "V", тоді як етапи тестування відображені на її правому краю.

Відповідність між етапами розробки та тестування визначена горизонтальними лініями. На кожному етапі проводиться контроль поточного процесу з метою переконатися в можливості переходу на наступний рівень. Початок тестування здійснюється з етапу написання вимог, і для кожного етапу встановлюється свій рівень тестового покриття.

Для кожного рівня тестування розробляється окремий тест-план з урахуванням критеріїв входу та виходу, і під час роботи над конкретним рівнем також розробляється стратегія тестування для наступного етапу. Таким чином, V-модель пропонує більш сучасний алгоритм, але залишається з деякими недоліками.

Її використання виправдано в проектах, які передбачають ширше тестування, і вона відноситься до основних практик екстремального програмування та третьої групи моделей.

Спіральна модель є комбінацією моделі "водоспаду" та каскадної, але враховує ризики (рис. 1.3)[3]. Кожна фаза починається з визначення мети та завершується переглядом прогресу. Ця модель, хоч і наближена до сучасних методів, все ж має свої обмеження.

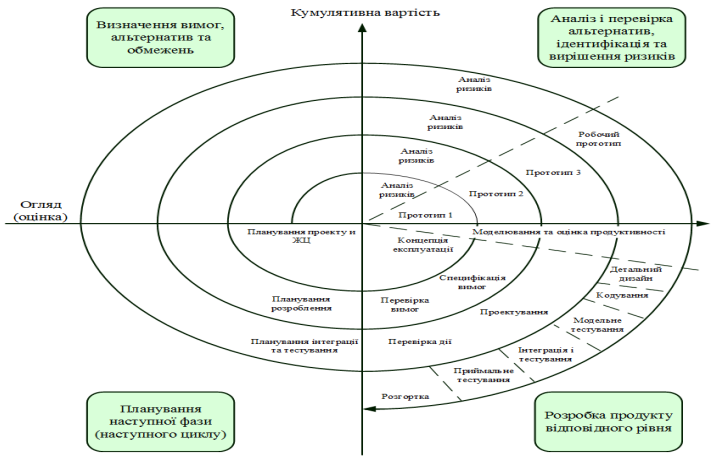

Рис. 1.3. Спіральна модель життєвого циклу програмного забезпечення

Дана модель належить до третьої групи та має переваги у швидкому отриманні результату, підвищенню конкурентоспроможності та змінюваності.

Однією з найбільш гнучких моделей розробки програмного забезпечення є Rational Unified Process (RUP). Rational Unified Process (RUP) відзначається гнучкістю, складаючись з чотирьох фаз. Команди можуть проводити ітерації доти, доки не виконають цілі.

Це робить RUP високоякісним вибором для проектів з потребою докладного контролю.Більшість видів тестування із залученням самих різних ресурсів відбувається під час етапу конструювання. Команда розробників розпочинає безпосереднє створення архітектури системи, а фахівці із забезпечення якості та тестувальники вивчають функціональність рішення та особливості інтерфейсу.

Розробка відлагодженого та перевіреного операційного програмного забезпечення – кінцева мета етапу конструювання.

Також не можна не сказати про методологію scrum (рис 1.4).

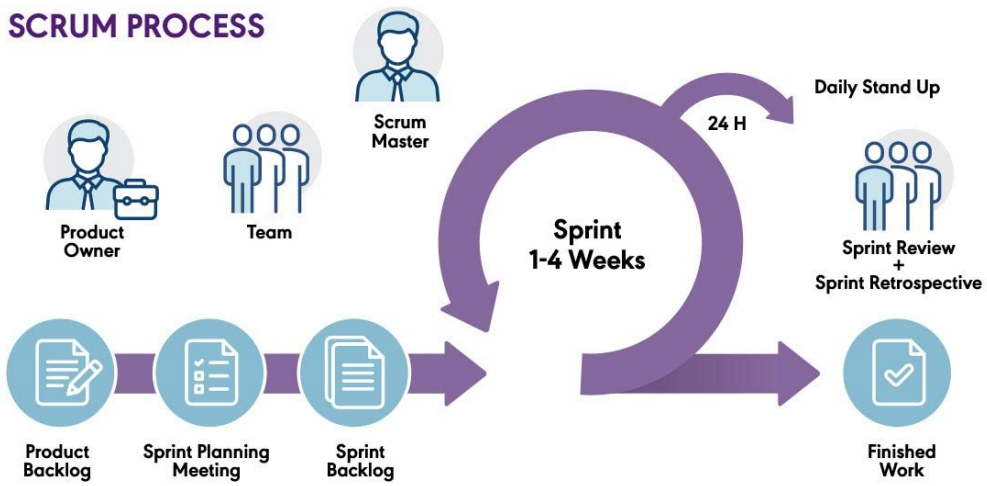

Рис. 1.4. Спіральна модель життєвого циклу програмного забезпечення

Методологія Scrum [4] є ітеративно-інкрементальною технологією зі спринтами.

Поняття «ітеративна» означає, що технологія розбивається на рівні за тривалістю проміжки часу – спринти, які тривають від одного до чотирьох тижнів. А слово «інкрементальна» – що в результаті ітерації виходить новий, потенційно робочий продукт, який вирішує бізнес-проблему.

Поняття «ітеративна» вказує на те, що технологія поділена на рівні за тривалістю проміжків часу, відомих як спринти, які зазвичай тривають від одного до чотирьох тижнів. У той час як термін "інкрементальна" вказує на те, що кожна ітерація призводить до нового, потенційно функціонального продукту, спрямованого на вирішення бізнес-проблеми.

Однак в процесі тестування ця модель має свої недоліки. В першу чергу, часто технічні завдання не формуються через швидке застаріння, що ускладнює створення та підтримку актуального комплекту автоматизованих тест-кейсів. Якщо програма взаємодіє з іншими додатками чи сервісами, тестування та ефективне управління ресурсами можуть стати проблемними.

Зазначене вище підкреслює, що тестування програмного забезпечення є обов'язковим етапом в будь-якій моделі розробки продукту. Воно, як справжнє розслідування, надає зацікавленим сторонам інформацію про якість тестованого програмного продукту або послуги. Також воно сприяє отриманню об'єктивного, незалежного уявлення про програмне забезпечення, що дає змогу бізнесу оцінити та зрозуміти ризики, пов'язані з його впровадженням. Методи випробувань включають в себе виконання програми за допомогою різноманітних ресурсів та зміни її стану відповідно до описаних сценаріїв для виявлення програмних помилок та перевірки придатності програмного продукту до використання.

1.2. Загальна характеристика досліджуваного об’єкту

Онлайн курси представляють собою новаторський підхід до навчання, що швидко займає центральне місце в освітньому ландшафті. Це особливо

актуально у зв'язку зі стрімким технологічним розвитком і високим рівнем доступності Інтернету. У рамках даного дослідження детально розглядається роль та вплив онлайн курсів на сучасну систему освіти.

Онлайн курси є не тільки засобом передачі знань, але й інструментом для демократизації освіти, забезпечуючи широкий доступ до навчання незалежно від фізичного місця або графіку зайнятості. Їхня значущість полягає в тому, що вони дозволяють студентам і професіоналам по всьому світу вдосконалювати свої навички та отримувати нові знання від провідних експертів у відповідних галузях.

Основні характеристики онлайн курсів, що розглядаються в даному дослідженні, включають їхню структуру, методику навчання, використання інтерактивних засобів, можливості адаптації до різних стилів навчання, інтеграцію технологій та підходи до оцінювання студентського успіху.

Розгляд технологічних аспектів включає аналіз використання різних платформ та засобів, таких як відео лекції, форуми, тести та інші інтерактивні елементи, які забезпечують ефективність передачі матеріалу та взаємодії між учасниками курсу.

Окрім того, в рамках дослідження визначається роль онлайн курсів у виборі професійного напрямку, розвитку навичок та вирішенні конкретних освітніх завдань. Детальний аналіз тенденцій у використанні онлайн курсів у сучасному світі дозволяє розуміти їхню потужність як інструменту для навчання та саморозвитку в різних соціокультурних та економічних контекстах.

Загальна характеристика онлайн курсів, яка буде представлена у цьому дослідженні, служитиме основою для подальшого розгляду розробки та вдосконалення програмного забезпечення для серверної частини веб-застосунку, спрямованого на підтримку цього важливого інструменту освіти.

1.3. Відомі методи, засоби і моделі розробки бекенду веб-застосунків для навчання

Вибір мови програмування та фреймворків є ключовим етапом у розробці бекенду веб-застосунку для навчання. Розглядаючи вибір мови програмування для розробки бекенду веб-застосунку для онлайн курсів, особлива увага приділяється порівнянню Java[4] з іншими мовами програмування. Вибір Java зумовлений рядом переваг, які вона пропонує в контексті даного проекту.

Однією з ключових переваг Java є її висока надійність та стабільність. Програми, написані на Java, відзначаються меншою кількістю помилок та продуктивною роботою протягом тривалого періоду. Це особливо важливо для великих та складних систем, таких як веб-застосунок для освіти.

Крім того, Java є крос-платформеною мовою, що означає, що розроблені застосунки можуть працювати на різних операційних системах без значних змін. Це надає гнучкість та універсальність, що є важливим аспектом у веб-розробці.

Технічні особливості Java також включають багатий екосистем фреймворків (рис 1.5), серед яких Spring Framework[5] та Spring Boot[6] виступають ключовими. Spring надає зручний та потужний інструментарій для розробки, забезпечуючи конвенції над конфігурацією та високий рівень абстракції.

У контексті розробки бекенду для онлайн курсів, Java виступає вдалим вибором через свою надійність, крос-платформеність та розширений інструментарій розробки, що дозволяє забезпечити високу якість та ефективність веб-застосунку.

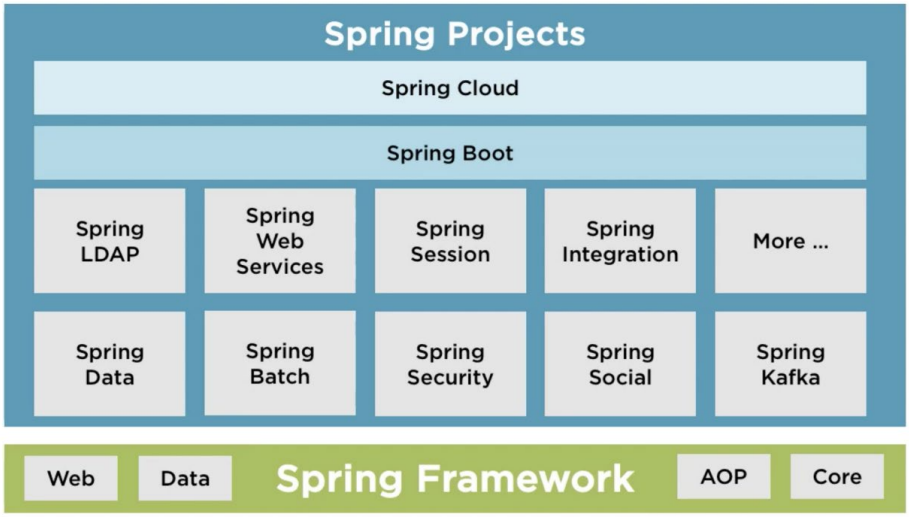

Рис. 1.5. Екосистема фреймворку Spring

Для спрощення розробки та підтримки вибрано Spring Framework, а зокрема його модуль Spring Boot. Його конвенції над конфігурацією дозволяють швидко створювати та розвивати застосунки, а модульна архітектура (рис 1.6) дозволяє використовувати тільки необхідні компоненти, що забезпечує ефективне використання ресурсів[7].

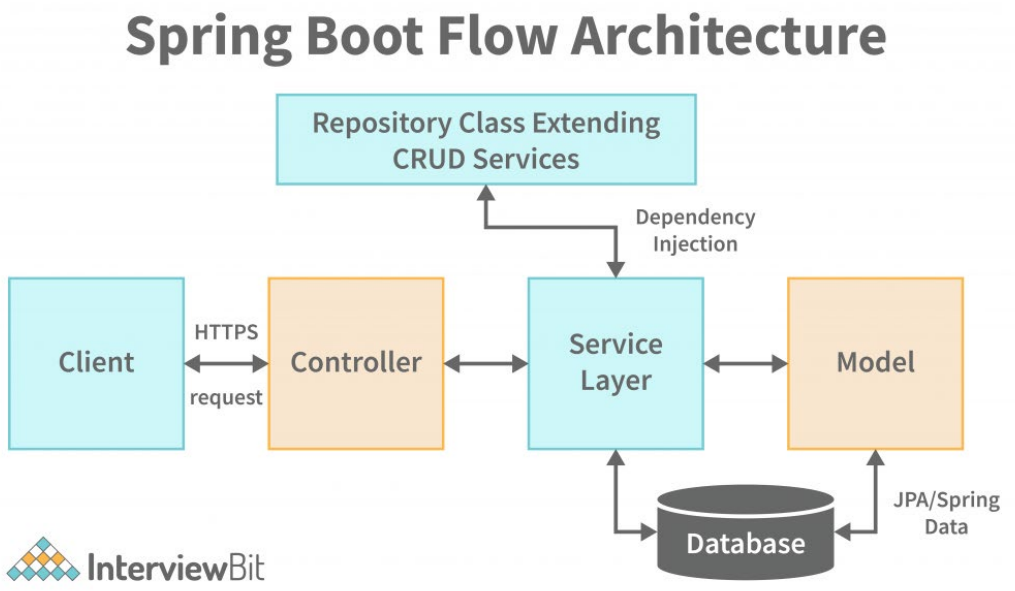

Рис. 1.6. Архітектура Spring Boot

Основна увага приділяється забезпеченню безпеки застосунку, і для цього використовується Spring Security [8]. Цей модуль дозволяє ефективно управляти аутентифікацією та авторизацією, забезпечуючи надійний контроль доступу до ресурсів системи.

Однією з ключових властивостей бекенду є база даних, і у цьому випадку обрано MongoDB [9]. Як NoSQL база даних, вона надає гнучкість у роботі з неструктурованими даними, що є важливим для ефективного зберігання та обробки інформації у веб-застосунках для навчання. MongoDB дозволяє легко масштабувати систему та швидко обробляти дані, що важливо для високопродуктивного та реактивного бекенду.

Обрана комбінація Java, Spring Framework і MongoDB створює стійку та ефективну основу для розробки бекенду веб-застосунку для онлайн курсів. Цей підхід дозволяє поєднати потужність мови програмування, гнучкість фреймворку та оптимізовану роботу з базою даних для створення високоякісного та високопродуктивного рішення.

Вибір бази даних має ключове значення для ефективності та масштабованості бекенду.

В контексті розробки бекенду веб-застосунку для онлайн курсів велику увагу приділяємо вибору та оптимізації бази даних, що визначається унікальними вимогами системи. При розробці веб-застосунків для навчання, реляційні бази даних, такі як MySQL чи PostgreSQL[10], залишаються популярними для забезпечення консистентності даних та швидкого доступу до них, однак однією з ключових розглядуваних баз є MongoDB, яка є NoSQL базою даних.

MongoDB вибрана з урахуванням гнучкості, яку надає NoSQL підхід (рис 1.7)[11]. Це дозволяє зберігати та опрацьовувати неструктуровані дані, що є важливим для ефективного зберігання інформації у веб-застосунках для навчання. MongoDB також легко масштабується та забезпечує швидкий доступ до даних, що є ключовим аспектом для інтерактивних елементів навчання.

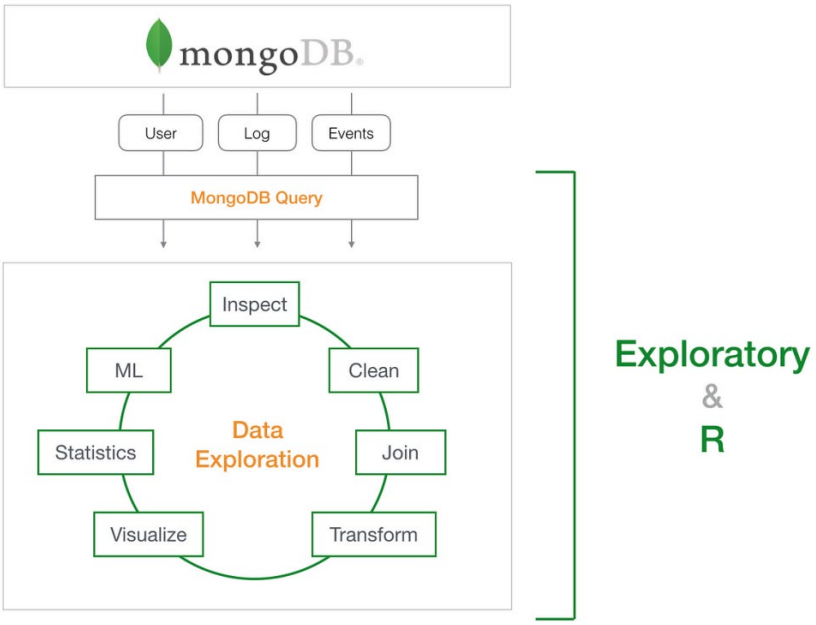

Рис. 1.7. Архітектура MongoDB

У порівнянні з реляційними базами даних, які використовують структурований підхід до даних, MongoDB відрізняється гнучкістю схеми. У реляційних базах даних потрібно строго визначити схему перед внесенням даних, що може бути обмежуючим у змінюваному середовищі онлайн навчання, де можливі зміни та доповнення вимог.

MongoDB також виграє у швидкості, особливо при обробці великої кількості неструктурованих даних. Реляційні бази даних, хоч і надійні, можуть бути менш продуктивними в сценаріях з інтенсивним читанням та записом.

MongoDB дозволяє легко масштабувати систему, додавати нові сервери та обробляти великий обсяг даних. Це важливо для систем, які мають потенціал зростання кількості користувачів та обсягу даних.

Однак MongoDB може бути менш ефективним у випадках, коли важливі відносини між даними, і вимагає додаткових зусиль для забезпечення цілості даних.

MongoDB є відмінним вибором для бекенду веб-застосунку для онлайн курсів через свою гнучкість, швидкість та можливість легкого масштабування. У порівнянні з реляційними базами даних, вона надає додаткові переваги для

систем, які вимагають гнучкості у зберіганні та обробці неструктурованих даних.

В реалізації бекенду веб-застосунку для онлайн курсів фокус приділяється інтерактивним можливостям, які допомагають оптимізувати процес навчання та створюють збалансоване та залучальне навчальне середовище.

Персоналізовані рекомендації виявляються особливо ефективним інструментом для користувачів. Аналізуючи їхню взаємодію з платформою, бекенд визначає їхні індивідуальні потреби та інтереси. Це веде до формування унікального набору рекомендацій, адаптованих до конкретного користувача, що робить процес навчання більш ефективним і зацікавлюючим.

Бекенд також впроваджує інтерактивний механізм обговорення та співпраці. Користувачі можуть активно обмінюватися думками та ідеями щодо курсів, що не лише сприяє формуванню спільноти, але й розширює можливості взаємодії та зрозуміння матеріалу.

У сфері інтерактивності важливою стає імплементація різноманітних завдань та вправ, які дають користувачам можливість застосовувати та вдосконалювати свої знання на практиці. Це може включати в себе відкриті завдання, тестування та інші інтерактивні елементи, що стимулюють активну участь та глибше засвоєння інформації.

Додатково, бекенд використовує інтерактивні можливості для збору та аналізу даних про взаємодію користувачів. Це дозволяє системі адаптувати рекомендації та контент, щоб краще відповідати потребам та очікуванням користувачів.

Такий підхід дозволяє створити не просто платформу для навчання, але і спільноту, де кожен учасник може знайти щось цікаве та корисне для себе. Технологічні можливості бекенду допомагають активно взаємодіяти з користувачами та надають їм інструменти для ефективного та зацікавленого навчання.

1.4. Порівняльний аналіз засобів досягнення ефективності,

масштабованості та швидкості роботи додатка

Тема архітектури програмного забезпечення стала надзвичайно важливою в області програмної інженерії у останні роки. У реальній практиці створення програм з чіткою архітектурою часто викликає значні труднощі.

Давайте розглянемо схему архітектури додатка (рис. 1.8)[12]. На зображенні представлені чотири круги, кожен з яких визначає окремий рівень програмного забезпечення. Проте важливо зауважити, що це лише схематичне зображення, і залежно від архітектурного рішення їх кількість може збільшитись чи зменшитись.

Ця схема містить наступні компоненти:

- Об'єкти: стійка частина програмного коду, яка залишається стабільною і не піддається зовнішнім змінам, таким як методи або структури даних.
- Сценарії використання: область для впровадження інкапсуляції та реалізації бізнес-логіки.
- Адаптери інтерфейсу: в цій частині відбувається трансформація та подання даних у сценаріях використання.
- Фреймворки та драйвери: ця область включає в себе фреймворки та інструменти для запуску програми.

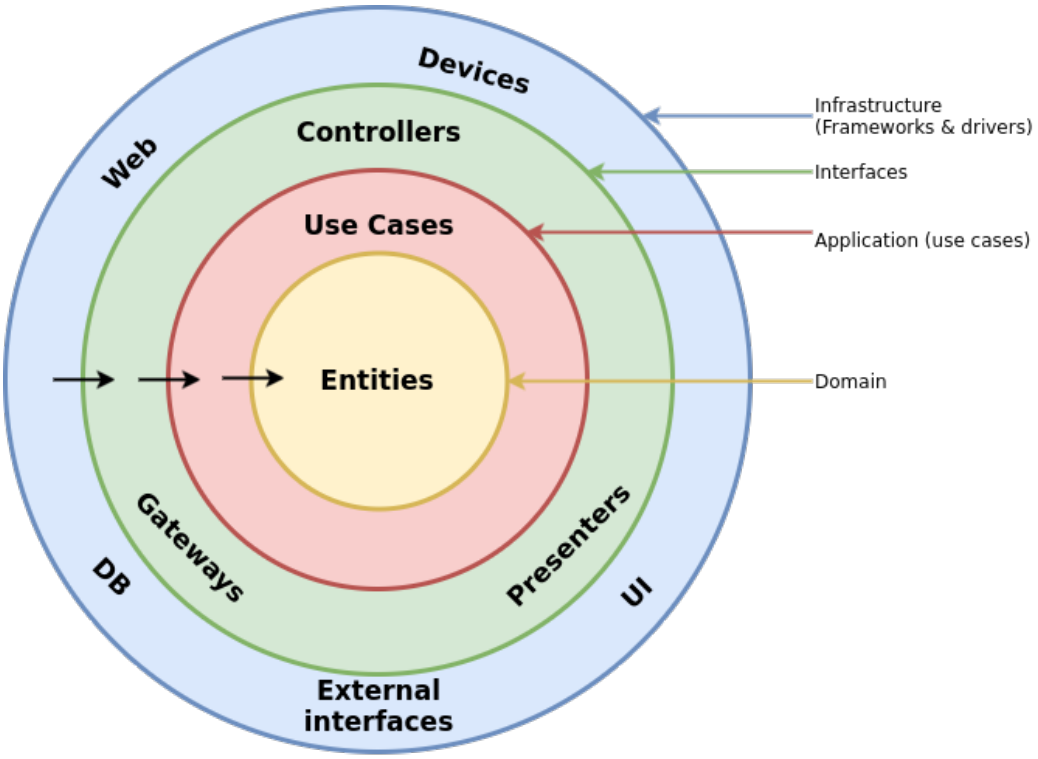

Рис. 1.8. Схема для "чистої" архітекури

Однією з основних принципів цієї архітектури є "Правило залежності"[13], яке відображено стрілками на представленому зображенні. Згідно з цим правилом, залежність є внутрішньою, а не зовнішньою. Це означає, що зовнішні круги знають та взаємодіють з внутрішніми, але внутрішні не мають інформації про зовнішні. Щоб відокремити клієнтський код від об'єктів, які можуть змінюватися, використовується ін'єкція залежностей.

У контексті тестування дотримання принципу інверсії залежностей також є великою допомогою, оскільки це дозволяє залежати від абстракцій, а не від конкретних реалізацій. Таким чином, макети можна легко передавати у тестах для використання подвійників.

Існують різні популярні технології для реалізації інверсії залежностей, такі як Java EE[14] та Spring [15]. Java EE – це платформа, яка надає API та середовище виконання для розробки та запуску великомасштабних, масштабованих, багаторівневих, надійних та безпечних програм.

Додатки Java EE володіють структурою, яка виокремлюється двома

основними характеристиками:

- Багаторівневість: У програмах Java EE використовується функціональний принцип розділення на ізольовані модулі;
- Вкладеність: У межах сервера Java EE розташовані контейнери компонентів.

Розділення програмного забезпечення на рівні є широко відомим:

- Клієнтський рівень;
- Середній рівень;
- Рівень доступу до даних.

На рівні клієнта програма запитує дані від сервера Java EE, який в свою чергу обробляє цей запит від клієнта та повертає йому відповідь. Наприклад, у ролі клієнтського додатка може виступати веб-браузер.

Середній рівень поділяється на дві складові: веб-рівень та бізнес-логіку. На веб-рівні за допомогою таких технологій, як Java Server Faces technology (JSF), Java Server Pages (JSP), Expression Language (EL), Servlets та Contexts and Dependency Injection for Java EE (CDI), забезпечується взаємодія між клієнтами та рівнем бізнес-логіки.

Рівень бізнес-логіки можна розглядати як ядро всієї системи та складається з таких компонентів, як Enterprise JavaBeans, JAX-RS RESTful web services, Java Persistence API entities та Java Message Service[16]. У цих компонентах реалізована вся бізнес-логіка програми, яка забезпечує функціональність для задоволення потреб конкретної сфери бізнесу, наприклад, електронна комерція або банківська справа.

Рівень доступу до даних іноді отримує назву рівня корпоративних інформаційних систем (Enterprise Information Systems – EIS). Цей рівень включає в себе різноманітні сервери баз даних, системи планування ресурсів (Enterprise Resource Planning – ERP) та інші джерела даних. Рівень бізнес-логіки звертається до цього рівня для отримання інформації з бази даних.

Технології на рівні EIS включають в себе Java Database Connectivity API

(JDBC)[17], Java Persistence API, Java EE Connector Architecture та Java Transaction API (JTA).

На рисунку 1.9 подано схему з простим Java EE додатком.

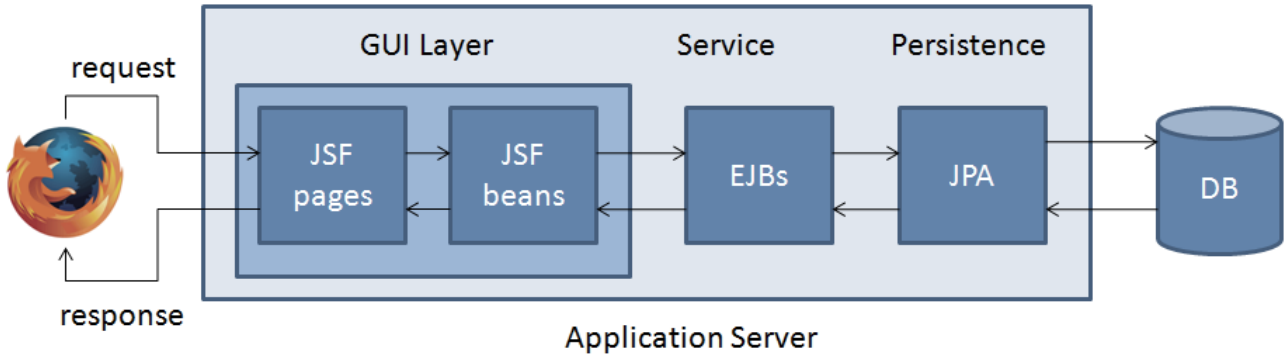

Рис. 1.9. Java EE додаток

Спочатку запит обробляється сторінкою JSF, а потім передається біну JSF. Подальше керування переходить до бізнес-логіки в EJB, яка взаємодіє з базою даних через JPA.

Основною перевагою є можливість масштабування, що означає, що при збільшенні навантаження на додаток на кілька порядків, його архітектура залишається незмінною. Для забезпечення працездатності, додаток може бути розгорнутий на кластері з декількома потужними вузлами, і все буде працювати в кластерному середовищі без необхідності внесення будь-яких змін. Однак, слабкими сторонами Java EE є складне середовище розробки додатків та висока остаточна вартість проекту, включаючи розробку, розгортання та тестування програми.

Spring Framework вважається конкурентним для Java EE.

Розвиток обох платформ виявився досить захоплюючим. Перші версії Java EE були створені за участю IBM, і хоча вони вражали своєю корисністю та функціоналом, виявилися досить великими та не зовсім зручними у використанні. Проблема полягала у важкості підтримки великої кількості конфігураційних файлів. Тим часом Spring IoC[18] вийшов у релізі майже одночасно. Це була компактна, легка у впровадженні та зручна у використанні

бібліотека. Як відзначалося, вона використовувала один конфігураційний файл, на відміну від Java EE. Таким чином, простота Spring призвела до того, що практично всі розробники почали впроваджувати цей фреймворк у своїх проектах.

Spring вирізняється своєю відомою системою впровадження залежностей та інверсії управління (IoC), що дозволяє легко створювати великомасштабні та слабко пов'язані додатки через використання IoC та AOP. Фреймворк Spring особливо ефективний у сфері фінансів та корпоративних програм завдяки своїм характеристикам безпеки, швидкості та простоти побудови транзакційних систем. На сьогоднішній день основними роботодавцями для розробників Spring є різноманітні банки, eBay, Visa та інші корпоративні гіганти.

Порівняно з Java EE, базовий додаток на Spring демонструє невелике відхилення в своєму принципі функціонування (рис. 1.10).

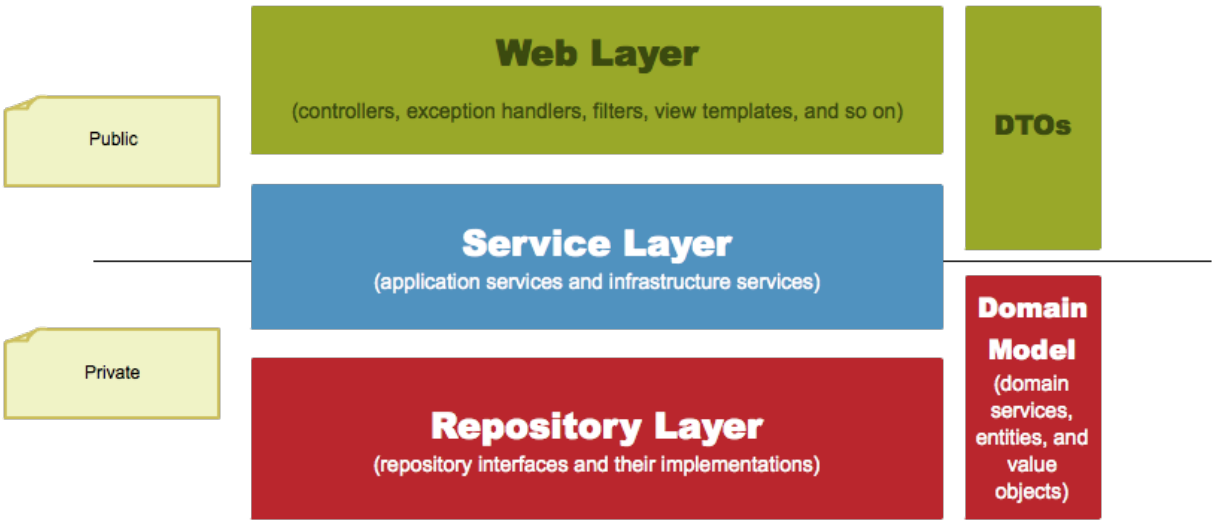

Рис. 1.10. Spring додаток

Spring спрощує процес розробки програм за допомогою консолідації системи залежностей та зменшення навантаження на окремі класи. Ця технологія, відома як інверсія управління, передає відповідальність за управління залежностями об'єктам самого фреймворку. У випадку інверсії управління (IoC), об'єкт вказує на необхідність певної залежності, і платформа

автоматично налаштовує та керує нею. Процес, яким Spring автоматично керує залежностями компонентів bean, відомий як впровадження залежностей[19]. Під час цього процесу Spring перевіряє, які саме компоненти bean є необхідними для правильної роботи обраного розробником bean і автоматично впроваджує їх як залежності.

Фреймворк Spring відзначається своєю підтримкою веб-архітектури MVC[20], яка розподіляє функціональні можливості між рівнями моделі, представлення та контролера. Приємним бонусом є обробка помилок, включаючи опрацювання винятків JDBC із системою їх ієрархії.

Широку популярність фреймворку Spring пояснюється значною кількістю переваг[21], серед яких:

- Легкість використання: Spring використовує звичайні об'єкти Java (POJO), а не сервери або корпоративні контейнери, що полегшує розробку;
- Ліцензія з відкритим кодом: Spring використовує ліцензію з відкритим кодом, що сприяє вільному використанню та модифікаціям;
- Модульність: Застосування принципів IoC та MVC дозволяє повторно використовувати компоненти у додатку без необхідності ручного керування їх залежностями;
- Сильна підтримка екосистеми Java: Spring інтегрується з існуючими технологіями, такими як ORM-фреймворки, JEE та таймери JDK;
- Масштабування транзакцій: Spring надає послідовний, масштабований інтерфейс управління транзакціями як для локальних, так і для глобальних транзакцій;
- Безпека: Простота впровадження готових модулів безпеки з функціями автентифікації покращує рівень безпеки додатків;
- Гнучкі конфігурації: Існує можливість використовувати інструкції на основі Java або конфігурацію XML для зручного налаштування;
- Простота використання: Spring Boot спрощує налаштування та ініціалізацію програми, а код Spring є простим для тестування.

До недоліків можна віднести обмежену документацію та паралельні механізми. Широкий спектр можливостей у Spring означає, що завдання можна вирішити декількома способами. Отже, для вибору ідеального рішення необхідно мати вичерпні знання доступних інструментів, щоб уникнути плутанини між командами. При цьому в документації Spring може бракувати чітких рекомендацій з різних тем, особливо щодо методів кібербезпеки.

На основі проведеного аналізу можна зазначити, що за основним функціоналом Spring і Java EE виявляють паритет та є найбільш ефективними інструментами для розробки корпоративних мережевих додатків на мові програмування Java.

Варто взяти до уваги, що при виборі технології важливо враховувати, що Java EE може бути використана лише в межах Enterprise Application Server, до яких не входить популярний сервер Tomcat[22]. У той час як додаток, побудований на технологіях фреймворку Spring, може запускатися на будь-якому сервері або навіть без сервера, оскільки має можливість запустити сервер самостійно.

Враховуючи описаний функціонал, можна визначити, що Spring є ідеальним інструментом для розробки невеликих програм з графічним інтерфейсом на Front-end або для мікросервісної архітектури[23]. З іншого боку, Java EE відмінно підходить для реалізації масштабованих монолітних кластерних додатків.

Висновок вірно підкреслює переваги використання Spring для побудови відносно невеликого додатку з автоматизацією зміни стану ресурсів тестування. Додатковою перевагою є легкість у підтримці і широка популярність на ринку розробки програмного забезпечення. Такий вибір забезпечить не лише зручність в роботі, але й певну стабільність та активну підтримку у майбутньому.

1.5. Аутентифікація та авторизація

Існування будь-якого веб-додатоку практично неможливио без

авторизації та аутентифікації, оскільки скрізь потрібно авторизуватися. У випадку програмного продукту з прямим доступом до цінної та захищеної бази даних цей крок стає в рази важливішим.

Для забезпечення безпеки створюваного додатку зі стеком технологій Spring буде актуальним використання Spring Security[24] – це платформа для додатків Java, яка надає спеціалізовані механізми для побудови систем авторизації та аутентифікації. Вона являє собою систему, що налаштовується для перевірки автентичності та контролю доступу до ресурсів додатків, а також захисту корпоративних додатків, створених за допомогою Spring (рис. 1.11).

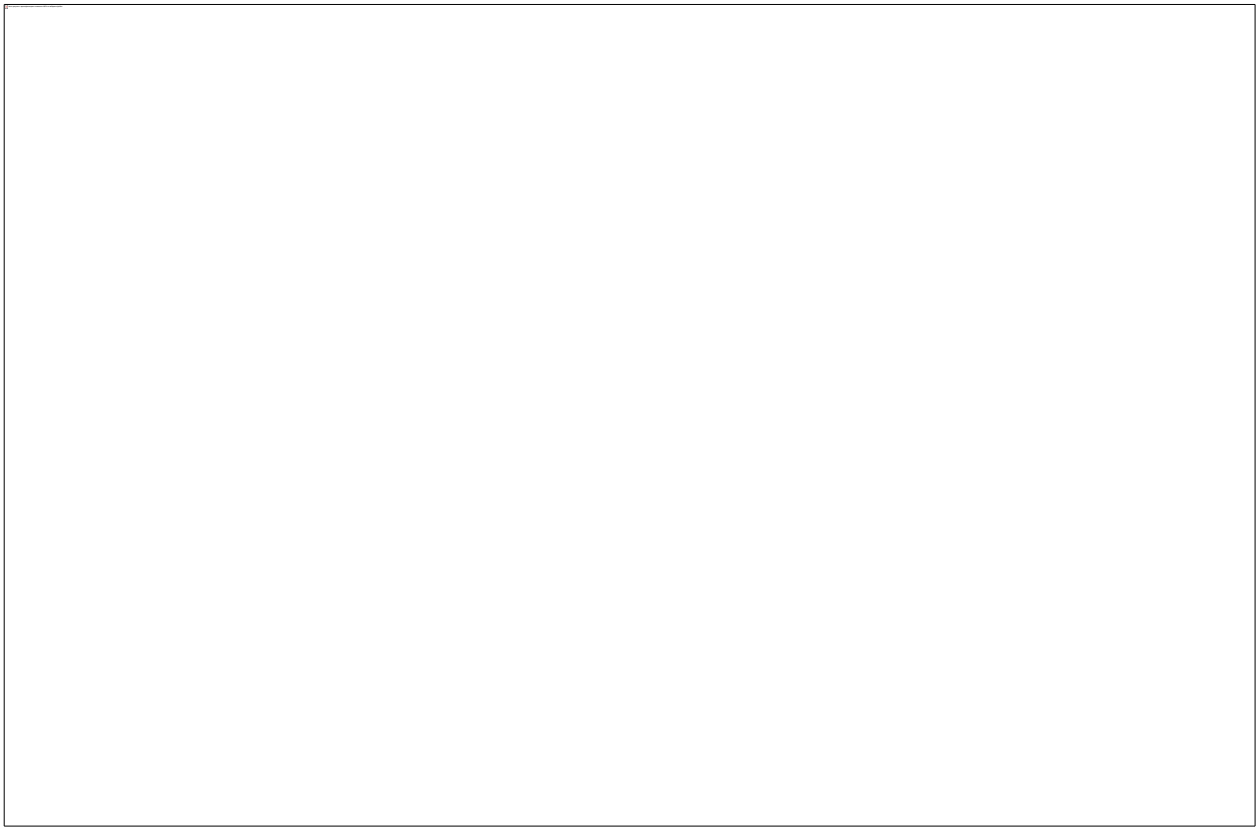

Рис. 1.11. Схематичне зображення реалізації Spring Security

Ключовим об'єктом є SecurityContextHolder, він зберігає інформацію про поточний контекст безпеки певної програми, включаючи приватну інформацію про самого користувача, який працює зараз із додатком. Усередині SecurityContext міститься об'єкт Authentication[25]. Spring Security використовує

цей об'єкт для надання інформації системи безпеки, яка пов'язана із запитами користувача. Сервіс UserDetails є представленням самого користувача, але в більш розширеному вигляді та з урахуванням специфіки програми Spring Security. У разі успішної аутентифікації, UserDetails використовується для створення Authentication об'єкта, який зберігається в SecurityContextHolder[26].

Широкі можливості авторизації в Spring Security є однією з найбільш вагомих причин її популярності. Розподіл програми на частини дозволяє відокремити доступ до певних зон окремим користувачам через функціонал GrantedAuthority[27]. Масив об'єктів GrantedAuthority складають певні повноваження, які надаються користувачу, наприклад роль адміністратора або тестувальника, які пізніше налаштовуються для веб-авторизації, а також авторизації методів. Даний розподіл зон доступу стає можливим завдяки широкій області видимості для дозволів, в яку входить вся програма.

Для створення безпечного додатка з автоматизації зміни стану ресурсів тестування знадобиться впровадження управління доступом до програми на основі ролей і привілеїв користувачів.

1.6. Аналіз техніко-функціональних характеристик систем-аналогів

У сучасному освітньому середовищі важливо не лише розробити ефективний веб-застосунок для онлайн курсів, але і вивчити досвід схожих систем, розглянути їх техніко-функціональні характеристики та здійснити порівняльний аналіз. Було зроблено детальний аналіз двох вітчизняних систем для онлайн-навчання - Prometheus та OpenEDX.

Основні технічні та функціональні характеристи онлайн ресурсу Prometheus:

- Prometheus використовує Python як основну мову програмування для реалізації своїх функцій та модулів.
- Система управління базами даних: PostgreSQL.

- Використання PostgreSQL гарантує надійність та ефективність зберігання та обробки даних в системі.
- Мікросервісна архітектура дозволяє розділити систему на невеликі, незалежні компоненти, що полегшує розширення та підтримку.

Prometheus є системою, що акцентує на важливості детального відстеження прогресу студентів та надає широкий спектр інструментів для ефективного керування курсами. Використання мікросервісної архітектури забезпечує гнучкість та масштабованість системи. Такий підхід може бути особливо корисним для освітніх інститутів, які прагнуть вдосконалити контроль та якість навчання.

Основні технічні та функціональні характеристи онлайн ресурсу OpenEDX:

- OpenEDX використовує Python та фреймворк Django для розробки, що забезпечує ефективну та стабільну основу для веб-застосунку.
- OpenEDX використовує різні системи управління базами даних (MySQL для традиційного зберігання та MongoDB для більш гнучких структур даних) дозволяє оптимізувати роботу з різними типами інформації.
- OpenEDX пропонує можливість вибору між монолітною та мікросервісною архітектурою, що дозволяє адаптувати систему до конкретних потреб користувачів.
- OpenEDX надає можливість викладачам створювати інтерактивні вправи та завдання для покращення активності студентів.
- Для технічних та інженерних курсів OpenEDX пропонує використання віртуальних лабораторій для практичного навчання та експериментів.

OpenEDX вирізняється своєю гнучкою архітектурою, що дозволяє користувачам обирати між монолітною та мікросервісною архітектурою в залежності від їхніх потреб. Інтерактивність та можливість використання віртуальних лабораторій роблять OpenEDX привабливим для різноманітних предметних областей, зокрема для технічних курсів. Здатність пристосовуватися

до різноманітних потреб робить OpenEDX сильним кандидатом для освітніх установ різного розміру та спрямованості.

Враховуючи різноманіття характеристик обох систем, вирішальними критеріями вибору будуть специфічні потреби користувачів та освітніх закладів. Якщо важливо враховувати інтерактивність та використання віртуальних лабораторій, OpenEDX може бути перевагою. З іншого боку, якщо важливий детальний аналіз успішності студентів та гнучкість архітектури, Prometheus може бути більш підходящим варіантом. Важливо враховувати конкретні вимоги та особливості вашого веб-застосунку для оптимального вибору системи-аналога.

1.7. Висновки

Автоматизоване тестування визволило час, який раніше був б витрачений на точкові та регулярні перевірки технічного стану продукту. Це підвищило ефективність роботи з масштабними процесами, зокрема імплементацією контролю якості на всіх рівнях управління проектом. Крім того, автоматизація тестування виявилася надзвичайно корисною в творчій та складній частині розробки програмного забезпечення, забезпечуючи ефективність у тестуванні нового функціоналу та виявленні дефектів при повторенні проблемних сценаріїв користувачів.

При мануальному тестуванні виникає виклик у підготовці ресурсів та систематичному зміні їх стану для забезпечення відповідності тестовим умовам реальній послідовності дій користувача та актуалізації його даних у базі даних додатка. У існуючих програмах для створення запитів та управління базами даних було виявлено кілька аспектів, які можна вдосконалити з метою підвищення ефективності процесу тестування. Наприклад, можливість розробки функціоналу для зміни даних в таблицях через інтуїтивний графічний інтерфейс, уникнення використання SQL-скриптів.

Також, для оптимізації процесу тестування можна реалізувати різні типи

таймерів. Наприклад, введення функціоналу для відкладеного запуску змін та їх застосування у визначений час або дату. Це дозволить більш гнучко налаштовувати тестові сценарії та оптимізувати використання ресурсів під час тестування.

Використання фреймворків при розробці надає можливість отримати швидший, якісніший та економічно більш вигідний результат, порівняно з розробкою проекту без використання будь-яких платформ. Розширений набір готових реалізацій найпоширеніших функціональних можливостей допомагає значно скоротити витрати часу та грошей на розробку програмного забезпечення. Крім того, це сприяє досягненню вищого рівня стабільності програми завдяки ефективному тестуванню та постійній підтримці рівня захисту від нових загроз.

Під час порівняльного аналізу доступних технологій для створення додатку був обраний Spring Framework, обумовлений наявністю великої кодової бази, активною спільнотою розробників, високою продуктивністю та успішними випадками використання Spring в компаніях, таких як Amazon, Google, Udemy, Trivago та інші.

Для забезпечення високого рівня безпеки системи та впровадження рольової схеми доступу обрано Spring Security. Ця потужна структура автентифікації та контролю доступу легко налаштовується під конкретний проект. Важливою особливістю Spring Security є його здатність захищати від різних атак, таких як фіксація сесії, викрадення кліків, підробка міжсайтових запитів і т.д. Фактично, цей фреймворк вважається стандартом для забезпечення безпеки програм, побудованих на основі Spring Framework, і фокусується на забезпеченні як автентифікації, так і авторизації для Java-програм.

РОЗДІЛ 2.
ЗБІР, СИСТЕМАТИЗАЦІЯ ТА ДОСЛІДЖЕННЯ ІНФОРМАЦІЇ ПРО СУЧАСНІ МОДЕЛІ ІНФОРМАЦІЙНИХ СИСТЕМ НА БАЗІ СУКУПНОСТІ ТЕХНОЛОГІЙ SPRING

2.1. Основи побудови IC у Spring Framework

Перед появою Enterprise Java Beans (EJB) розробники Java для створення веб-додатків зазвичай використовували JavaBeans[28]. Хоча JavaBeans були корисні для розробки компонентів користувацького інтерфейсу (UI), вони не забезпечували управління транзакціями та безпеку, що було вкрай необхідним для створення надійних та стабільних корпоративних систем. З появою EJB сподівалися вирішити цю проблему та розширити можливості веб- та корпоративних компонентів Java. Однак розробка додатків з використанням EJB залишалася складною, оскільки розробникам доводилося виконувати зайві завдання, такі як створення інтерфейсів Home і Remote, а також реалізацію методів зворотного виклику життєвого циклу.

Фреймворк Spring з'явився як відповідь на існуючі проблеми в розробці корпоративних додатків. Використовуючи інноваційні методи, такі як Аспектно-орієнтоване програмування (AOP), Plain Old Java Object (POJO) та ін'єкція залежностей (DI), Spring спрощує процес розробки, усуваючи складності, пов'язані з використанням Enterprise Java Beans (EJB). Фокус Spring полягає в наданні засобів для ефективного керування бізнес-об'єктами, що значно полегшує розробку веб-додатків у порівнянні з традиційними фреймворками Java та інтерфейсами прикладного програмування (API), такими як Java Server Pages, Java Database Connectivity та Java Servlet [29].

Фреймворк Spring можна зручно розглядати як набір підфреймворків, також відомих як шари, що включають Spring Web MVC, Spring AOP, Spring Object-Relational Mapping (Spring ORM) та Spring Web Flow, і багато інших. При розробці веб-додатку ви можете використовувати будь-який з цих модулів окремо, а також комбінувати їх, щоб отримати кращу функціональність вашого додатку. Особливості, такі як інверсія управління (IoC), аспектно-орієнтоване

програмування (AOP) та управління транзакціями, роблять Spring унікальним та потужним інструментом для розробки програмного забезпечення.

При створенні програмного забезпечення для зміни управління станом ресурсів можуть бути використані наступні функціональні можливості фреймворку Spring:

- контейнер IoC;

Spring надає два пакети, а саме org.springframework.beans та org.springframework.context, які допомагають у забезпеченні функціональності IoC. Він відноситься до основного контейнера, який використовує патерн DI або IoC для неявного надання посилання на об'єкт в класі під час виконання.

- фреймворк доступу до даних;

Дозволяє розробникам використовувати API персистентності, такі як JDBC та Hibernate[30], для зберігання персистентних даних в БД. Це допоможе у вирішенні різних проблем розробки: взаємодія з підключенням до бази даних, переконання у закритті з'єднання, робота з з винятками і реалізація управління транзакціями. Загалом дана частина Spring дозволить легко писати код для доступу до персистентних даних у всьому додатку.

- рівень абстракції JDBC;

Допомагає в обробці помилок у ефективній і простій формі.

- фреймворк Spring MVC;

Дозволяє створювати додатки на основі архітектури MVC, тобто всі запити від користувача спочатку проходять через контролер, а потім відправляються у візуальну обробку, такі як JSP-сторінки або сервлети.

- Spring Web Service;

Генерує кінцеві контексти та визначення вебсервісів на основі класів Java.

- Spring Security;

Цей фреймворк забезпечить різні функції безпеки, такі як: автентифікація та авторизація для створення захищеної програми Java Enterprise.

2.2. Архітектура Spring Framework

Spring фреймворк включає в себе сім модулів, які представлені у зображенні 2.1.

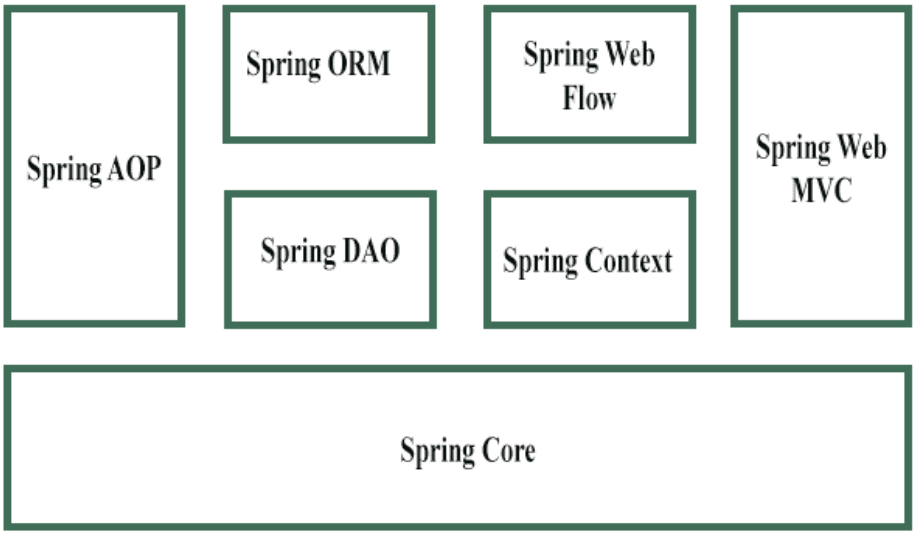

Рис. 2.1. Модулі Spring Framework

Це охоплює Spring Core, Spring AOP, Spring Web MVC, Spring DAO, Spring ORM, контекст Spring та Spring Web Flow [31]. Кожен з цих модулів пропонує різноманітні інструменти для створення корпоративних додатків; наприклад, модуль Spring Web MVC призначений для розробки додатків, використовуючи паттерн Model–View–Controller.

Ключовим компонентом у структурі фреймворку Spring є модуль Spring Core, який впроваджує контейнер IoC[32]. В Spring існують два типи реалізацій контейнера: bean factory та application context. Перший визначається через інтерфейс BeanFactory і служить як контейнер для бінів, дозволяючи відокремити конфігурацію та специфікацію залежностей від програмної логіки. Ця фабрика виступає в якості основного контейнера IoC та відповідає за створення екземплярів об'єктів, а також налаштовує та керує залежностями між цими об'єктами.

Аналогічно об'єктно-орієнтованому програмуванню, яке створює ієрархію об'єктів у додатках, AOP розкладає програми на аспекти або проблеми. Модуль Spring AOP дозволяє впроваджувати ці аспекти або проблеми в Spring-додаток. За допомогою анотації @Aspect [33], звичайні класи або біни можуть стати аспектами у Spring AOP, допомагаючи в моніторингу помилок у роботі програми, управлінні транзакціями і веденні журналів. Один з найпоширеніших прикладів використання управління транзакціями - це банківські операції, такі як переказ коштів з одного рахунку на інший.

Модуль Spring ORM[34] використовується в додатку для взаємодії з інформацією в базі даних. Він надає можливості для опрацювання баз даних з використанням Java Data Objects, Hibernate та iBatis[35]. Spring ORM підтримує DAO, що дозволяє зручно будувати ORM-рішення на основі DAO, такі як декларативне управління транзакціями, прозора обробка виключень, потокобезпечні легкі шаблонні класи та класи підтримки DAO.

Модуль веб-архітектури (Web-MVC) фреймворку реалізує концепцію Model-View-Controller для створення веб-додатків. Цей модуль розподіляє компоненти програми на моделі, контролери та представлення. Згідно з принципом роботи Spring MVC, при формуванні запиту від браузера, він спочатку надходить до класу DispatcherServlet[36], який за допомогою відображень-обробників направляє запит до контролера. Контролер витягує та обробляє інформацію із запиту та повертає результат у вигляді об'єкту моделі до класу DispatcherServlet. Потім використовуються класи ViewResolver для направлення результатів до представлення.

Модуль Spring Web Flow представляє собою розширення для Spring Web MVC. Там, де Spring Web MVC використовує контролери форм для втілення попередньо визначеного робочого процесу, такі як клас SimpleFormController, Spring Web Flow надає можливість створювати Java-клас або XML-файл, що дозволяє керувати робочим процесом між різними сторінками веб-додатку.

Модуль Spring Web DAO[37] включає в себе підтримку структурного шаблону, що дозволяє ізолювати прикладний або бізнес-рівень від рівня персистентності (зазвичай, це реляційна база даних) за допомогою абстрактного API. Цей модуль впроваджує рівень абстракції JDBC та надає програмні та декларативні класи для управління транзакціями.

Модуль контексту у програмі Spring базується на основному модулі Core. ApplicationContext, що є контекстом додатку, виступає як інтерфейс BeanFactory. Отже, цей модуль отримує свої можливості з пакету org.springframework.beans та підтримує різноманітні функціональні можливості, такі як інтернаціоналізація (I18N), валідація та завантаження ресурсів.

2.3. Структура діяльності автономного додатку

Spring Boot є спрощеною та автоматизованою версією Spring Framework[38], доступною для розширення іншим функціоналом фреймворку при необхідності. Тому кожен проект, включаючи додаток для автоматизації зміни стану ресурсів тестування, починає свій шлях з розробки базової структури на цій складовій фреймворку. Spring Boot використовує багаторівневу архітектуру, де кожен рівень взаємодіє з іншими незалежно від їх положення у ієрархії.

Spring Boot визначається головною метою — усунення непорозумінь та важких конфігурацій на основі XML та окремих анотацій у програмах. Паралельно із цим він пропонує переваги у вигляді гнучкості (можливості зміни конфігурації), автономності і здатності до запуску додатка на ранніх етапах розробки.

Spring Boot складається з чотирьох ключових компонентів[39], як показано на рисунку 2.2.

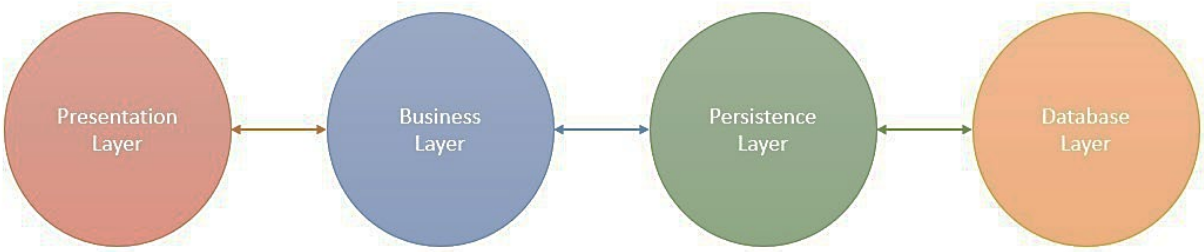

Рис. 2.2. Компоненти Spring Boot

1. Рівень представлення

Верхній рівень архітектури Spring Boot виступає першим презентаційним шаром. Цей рівень включає представлення (views), які відповідають за інтерфейсну частину програми. Рівень представлення обробляє HTTP-запити та виконує аутентифікацію, відповідаючи за перетворення параметрів поля JSON в об'єкти Java і навпаки [40]. Після успішної аутентифікації запит передається на наступний, бізнес-рівень.

2. Бізнес-рівень

Бізнес-рівень займається всією бізнес-логікою, і включає в себе класи сервісів, які відповідають за валідацію та авторизацію.

3. Рівень збереження

Рівень персистентності відповідає за збереження бази даних та виконує конвертацію між бізнес-об'єктами Java і рядками бази даних.

4. Рівень бази даних

Останній рівень включає всі бази даних додатку, такі як MySql, MongoDB і інші [22]. Зручність полягає в можливості оперувати декількома базами даних. Рівень бази даних відповідає за здійснення операцій CRUD.

Структура застосунка для бекєнду сайту з онлайн курсами буде ґрунтуватися на упорядкованій архітектурі Spring Boot, яка структурує ресурси у визначені процеси (рис. 2.3).

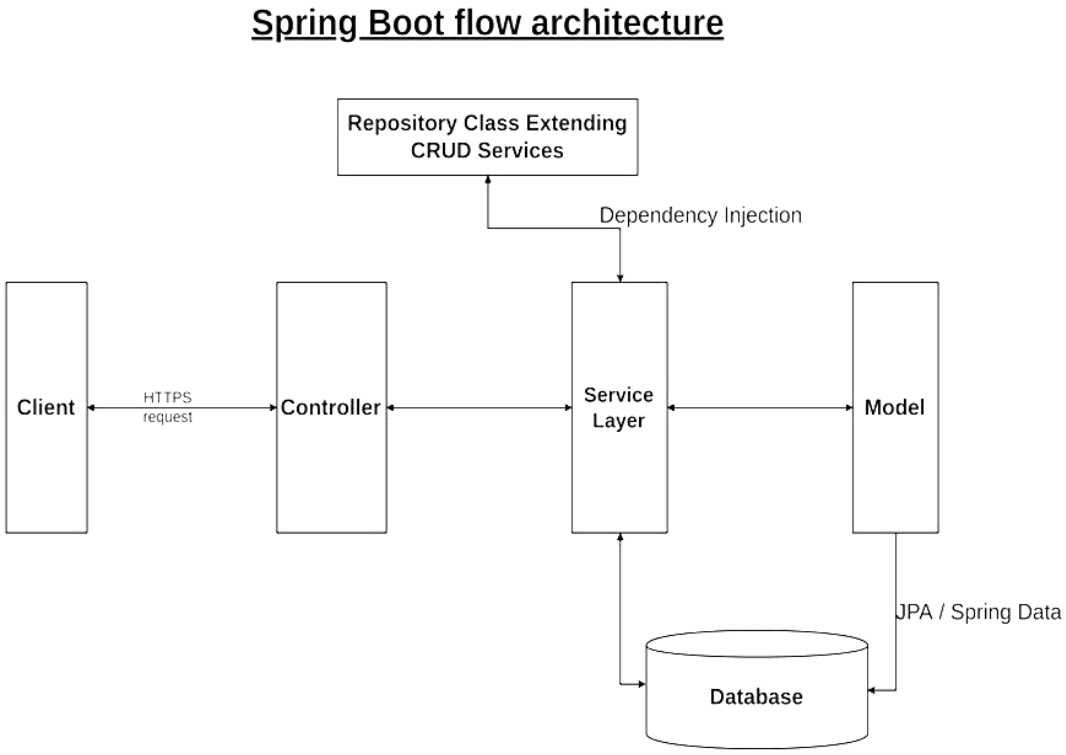

Рис. 2.3. Структура діяльності додатку

Згідно з поданою схемою, перший етап включає HTTP-запит від клієнта (може бути будь-якого типу: GET, PUT, POST і т. д.) [41]. Цей HTTP-запит передається до контролера, який його відображає, обробляє дескриптори та активує виконання логіки на сервері. На рівні служб відбувається виконання всієї бізнес-логіки. Дані з бази даних відображаються в класі моделі Spring Boot та обробляються за допомогою Java Persistence Library (JPA). У кінці, сторінка JavaServer Pages (JSP) повертається як відповідь від контролера.

Архітектура Spring Boot ґрунтується на фреймворку Spring, тому в значній мірі використовує всі функції та модулі Spring, такі як Spring MVC, Spring Core та інші.

2.4. Обґрунтування рівня залежності та зв'язаності

У парадигмі об'єктно-орієнтованого програмування термін "залежність" вказує на рівень прямого знання, яке один об'єкт має про інший, або, іншими словами, як часто зміни в класі A впливають на відповідні зміни в класі B.

На рисунку 2.4 представлено два види цієї залежності:

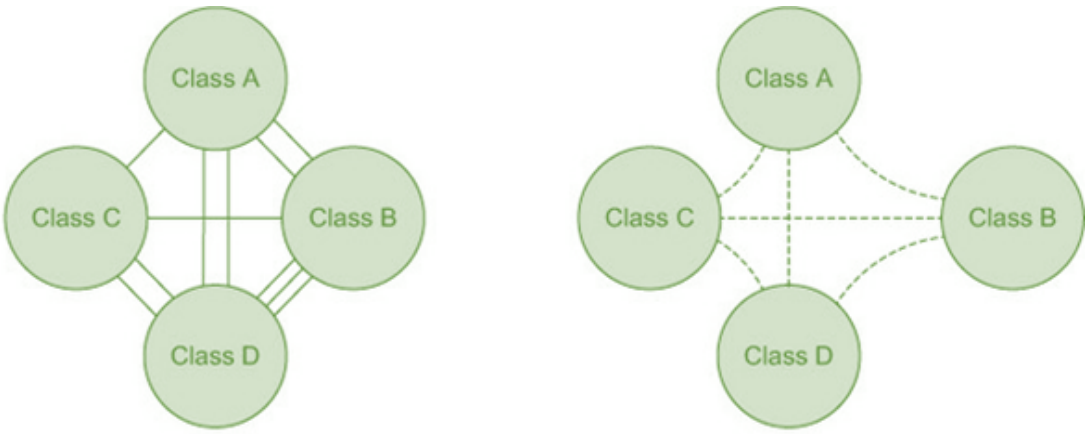

Рис. 2.4. Сильна та слабка залежність частин додатку

- Тісна залежність: в цьому контексті обидва класи часто взаємодіють та зазнають змін разом;

Клас А має більше інформації, ніж величезна необхідність, щодо того, як саме реалізовано клас В, що призводить до тісного зв'язку між ними.

- Слабка залежність: зв'язок вказує на те, що, в основному, класи функціонують незалежно один від одного.

Класи розглядаються як слабко зв'язані, коли клас А знає про клас В тільки те, що викривається через інтерфейс. Механізм ін'єкції залежностей у Spring Framework використовується для подолання проблем жорсткого зв'язку між об'єктами в додатку[42].

Для створення програми з управлінням станом тестових ресурсів важливо ретельно розглянути переваги та недоліки різних видів залежностей, таких як:

- Сильна залежність може сповільнити процес тестування, тоді як слабка залежність може покращити його ефективність;
- У випадку сильної залежності важко здійснити переміщення функціоналу між класами, в той час як у випадку слабкої залежності легше оперувати фрагментами коду, модулями, об'єктами чи компонентами;

- Жорстка залежність може бути швидшою та дешевшею для розробки невеликих додатків, але може виявитися менш ефективною в підтримці.

Проведений аналіз вказує на те, що для розробки додатку нежелажено використовувати жорсткий зв'язок, оскільки він ускладнює тестування, обмежує можливості повторного використання коду та гнучкість, а також робить систему менш адаптованою до змін. У порівнянні з цим, використання слабкої залежності виявляється більш вдалим, оскільки такий підхід сприяє полегшенню оновлень і масштабуванню програмного забезпечення, забезпечуючи більш легку адаптацію до змін вимог.

Відокремивши принцип слабкої залежності від абсолютного контексту, можна дійти до висновку, що всю функціональність можна розмістити в єдиному класі. Таким чином, залежностей від інших класів взагалі не буде, проте в той же час цей клас може включати абсолютно непов'язану між собою бізнес-логіку. Ця висновок відповідає принципам об'єктно-орієнтованого програмування, який підкреслює, що клас повинен мати єдину, чітко визначену мету. І чим більш фокусованою є мета класу, тим вище його ступінь зв'язності.

Основною перевагою великої групування є легкість утримання такого додатку завдяки меншим частим змінам і спрощеному процесу тестування. Ще однією перевагою великої взаємодії є вища можливість для повторного використання, що сприяє швидшому розвитку та зменшенню витрат на підтримку проекту.

Мінімальна залежність та висока логічна зв'язність є стандартом якості у розробці системи, який відповідає цілям додатка з автоматизації зміни стану ресурсів тестування. Загальний принцип полягає в створенні програмного забезпечення з слабко зв'язаними класами, які включають в себе чітко визначену бізнес-логіку. Дотримання цього принципу сприятиме можливості повторного використання створених класів та збереже зрозуміння їхніх обов'язків.

2.5. Технології Spring для побудови додатка

2.5.1. Платформа Spring Boot

Щоб спростити управління залежностями, Spring Boot автоматично включає необхідні сторонні бібліотеки для кожного типу програм на основі Spring і предоставляє їх розробникам за допомогою starter-пакетів (наприклад, spring-boot-starter-web, spring-boot-starter-data-jpa і інші)[43]. Для розробки додатка з автоматизації зміни стану тестових ресурсів будуть використані наступні пакети:

- "spring-boot-starter-web" використовується для створення веб-додатків на основі Spring, і він автоматично додає до проекту популярні бібліотеки, такі як spring-webmvc, jackson-json, validation-api та Tomcat;
- "spring-boot-starter-data-jpa" використовується для забезпечення доступу до бази даних без необхідності ручного пошуку та налаштування сумісних драйверів;
- "spring-boot-starter-security" створено для забезпечення середовища, в якому здійснюється аутентифікація та авторизація користувачів.

Отже, при використанні Spring Boot, спеціальний XML-файл pom.xml[44], який включає в себе інформацію про проект та різні деталі конфігурації, містить значно менше рядків, ніж у випадку зі звичайними Spring-додатками.

Ще однією важливою можливістю Spring Boot є автоматична конфігурація програми, яку можна повністю змінити у будь-який момент за допомогою користувацьких налаштувань.

Після вибору відповідного starter-пакету, Spring Boot автоматично намагається налаштувати Spring-додаток, враховуючи додані залежності jar-файлів. Наприклад, для обраного пакету spring-boot-starter-web, Spring Boot автоматично конфігурує біни, такі як DispatcherServlet та ResourceHandlers.

Спочатку в Spring Boot-додатку будуть розглядатися різні моделі програми та відповідні DTO (об'єкти передачі даних), які знаходяться у пакеті pojo. Питання використання DTO є предметом різних точок зору. У веб-додатку

для бекєнду сайту для курсів прийнято рішення про їх використання для полегшення зв'язку між шаром моделі програми та шаром інтерфейсу користувача.

DTO надають можливість передавати лише ті дані, які потрібно розкрити в користувальницькому інтерфейсі, а не весь об'єкт моделі, який під час розробки програмного забезпечення був створений за допомогою кількох вкладених об'єктів і збережений у базі даних.

Наступним елементом додатку будуть сервіси та об'єкти доступу до даних. Об'єкти доступу до даних (DAO) знаходяться у пакеті репозиторію services та є розширеннями інтерфейсу JpaRepository[45]. Методи цього інтерфейсу є базовими, як показано на рисунку 2.5, і допомагають сервісному рівню отримувати, зберігати та обробляти дані з бази даних.

Сервісний рівень розташований у пакеті сервісів services і завжди використовується з моделями, як вхідних або вихідних даних. Це визначено як одна з найкращих практик у розробці багаторівневої архітектури Spring-додатку для сайту онлайн курсів. Рівень контролера взаємодіє із сервісним рівнем, отримуючи запит від представлення (view), і при цьому контролер не має доступу до об'єктів моделі, завжди взаємодіє у термінах нейтральних DTO..

```
public interface JpaRepository<T, ID> extends PagingAndSortingRepository<T, ID>
  List<T> findAll();

  List<T> findAll(Sort sort);

  List<T> findAllById(Iterable<ID> ids);

  <S extends T> List<S> saveAll(Iterable<S> entities);

  void flush();

  <S extends T> S saveAndFlush(S entity);

  <S extends T> List<S> saveAllAndFlush(Iterable<S> entities);
```

Рис. 2.5. Методи взаємодії з базою даних

Рівень контролера координує весь процес, починаючи з перехоплення запиту і закінчуючи підготовкою та відправленням відповіді. Ця функціональність реалізована у пакеті controller.

Контролери додатку для бекєнду сайту з онлайн курсами працюватимуть на основі концепції Spring WebMVC. Вони будуть обробляти вхідні веб-запити та взаємодіяти з об'єктами Spring ModelAndView, які містять дані для відображення на відповідних представленнях/формах.

Головний метод для запуску програми у Spring Boot додатку буде визначений у класі, що включає анотацію @SpringBootApplication[46]. Ця анотація охоплює автоконфігурацію, сканування компонентів і конфігурацію завантаження Spring.

Вбудований сервер є важливою рисою у Spring Boot. Таким чином, все, що вам потрібно для запуску програми, - це створити виконуваний jar-файл за допомогою Maven і викликати метод, зображений на рисунку 2.6.

```
@SpringBootApplication
public class SpringSecurityApplication {

  public static void main(String[] args) {
    SpringApplication.run(SpringSecurityApplication.class, args);
  }
```

Рис. 2.6. Точка запуску Spring Boot додатку

2.5.2. Захист додатку з Spring Security

У 2020 році глобальний локдаун призвів до значного зростання цифрової активності на світовому ринку, зокрема, збільшився попит на створення веб-сайтів та додатків для надання товарів та послуг клієнтам. Проте, при такому швидкому впровадженні цифрової трансформації виникли проблеми безпеки,

оскільки багато організацій уперше зіткнулися з необхідністю забезпечення безпеки в цьому контексті. Зокрема, безпека корпоративних додатків стала особливо важливою, оскільки сучасні застосунки надаються в різних мережах і мають з'єднання з хмарними сервісами, що збільшує їхню вразливість перед загрозами та потенційними порушеннями безпеки.

Зростає необхідність забезпечення безпеки не лише на рівні мережі, але й на рівні самостійних додатків, оскільки атаки хакерів виявляються все частіше у напрямку цих додатків. Такий підхід надає значних переваг, і саме його використано при розробці корпоративного додатку для автоматизації зміни стану ресурсів тестування.

Модуль безпеки у Spring є інструментарієм для реалізації функцій авторизації, аутентифікації та контролю доступу, і він володіє розширеним спектром доступних конфігурацій. Загалом, це стандартний фреймворк, який використовується для захисту програмних додатків, побудованих на базі Spring.

В корпоративному додатку для бекєнду сайту з онлайн курсами, Spring Security виступатиме в ролі захисного шару, охоплюючи всі функціональні аспекти програми, як видно на діаграмі 2.7.

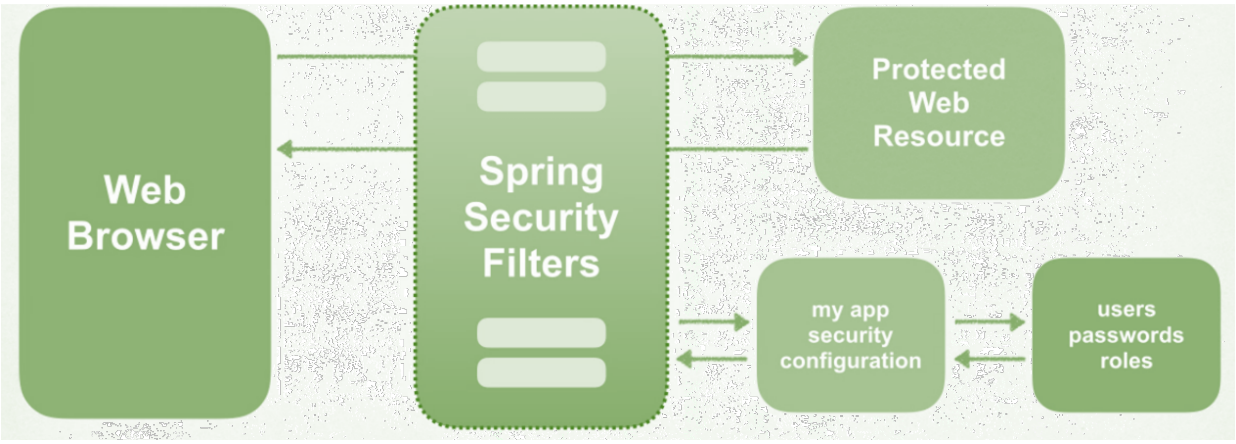

Рис. 2.7. Місце Spring Security у структурі програми

Spring Security Filters[47] в цьому випадку реалізує робочий процес, відображений на діаграмі у фігурі 2.8, і демонструє, як програма обробляє та

керує інформацією про користувача, такою як ім'я, пароль і повноваження, під час свого функціонування.

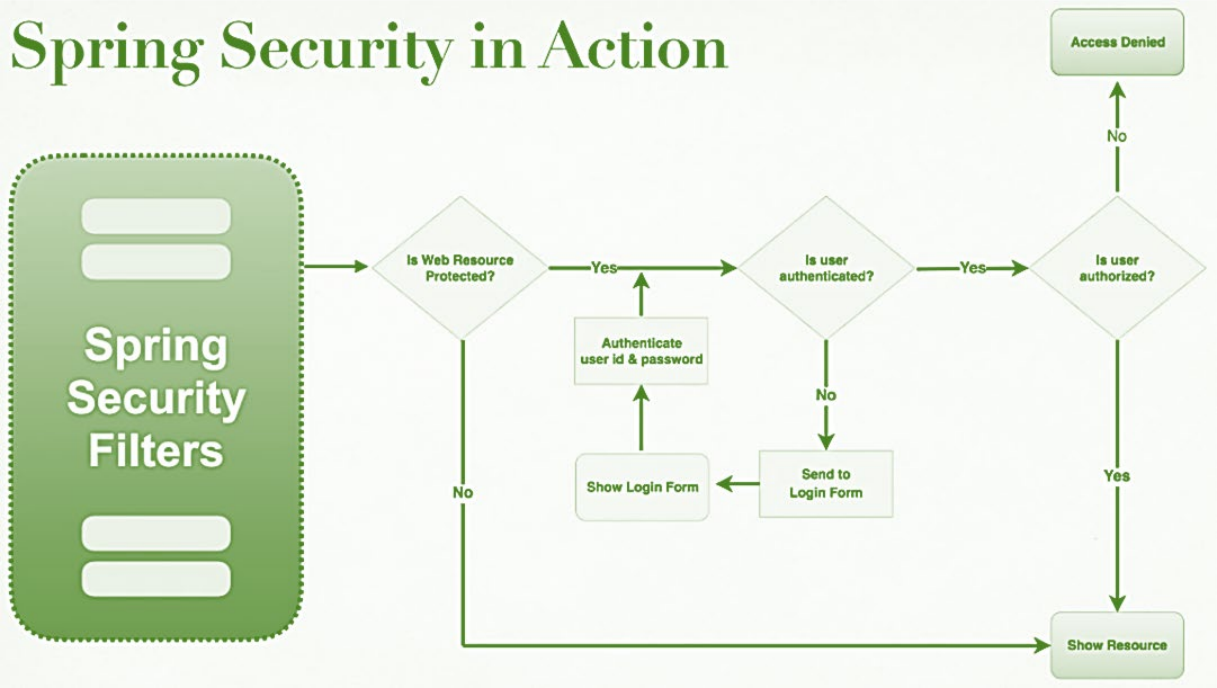

Рис. 2.8. Діаграма роботи фільтру Spring Security

Для впровадження Spring Security у програму з автоматизації зміни стану ресурсів тестування потрібно виконати деякі етапи [48].

На початку проекту додається стартер-пакет `spring-boot-starter-security`. Зазвичай додавання будь-якого стартера до POM-файлу не призводить до автоматичних змін у функціональності програми, і для їхньої реалізації зазвичай потрібно написати додатковий код. Проте, у випадку Spring Security ситуація відрізняється, оскільки за замовчуванням надається функціонал для користувача. Його ім'я встановлено як "user", а пароль генерується автоматично при запуску програми.

Отже, при включенні лише стартер-пакету `spring-boot-starter-security` до POM-файлу відбувається:

- Spring Security автоматично генерує користувача з ім'ям "user" та створює пароль, який можна переглянути в консолі працюючого додатку;

- Створюється сторінка, яка містить форму для введення імені та пароля, реалізуючи функціонал аутентифікації на основі форми;
- Інформація для входу в програму автоматично перевіряється, включаючи ім'я та пароль;
- Всі URL залишаються недоступними до завершення успішного процесу входу під реєстрованим користувачем.

Далі виконується налаштування InMemoryUserDetailsManager. Під час отримання параметрів користувача з запиту, що відбувається через Form-Based аутентифікацію, ім'я та пароль відправляються через форму та надходять на сервер як POST-параметри.

Для створення власного користувача в In-Memory автентифікації потрібно створити клас-конфігурацію SecurityConfig, який розширює клас WebSecurityConfigurerAdapter і позначити його анотацією @EnableWebSecurity для його біндування. AuthenticationManager відповідає за процес аутентифікації, і хоча він виконується автоматично, розробник може впливати на його конфігурацію, перевизначивши метод configure (AuthenticationManagerBuilder auth) класу WebSecurityConfigurerAdapter. Цей метод надає доступ до AuthenticationManagerBuilder, де можна визначити тип аутентифікації, тобто місце зберігання користувача. У випадку додатку з автоматизації зміни стану ресурсів тестування тип аутентифікації буде використовувати Jdbc.

Далі слід вказати конкретні налаштування для обраного AuthenticationManager, де визначається, як AuthenticationManager отримає збереженого користувача та як потім порівняти його з введеними даними. Фактично, AuthenticationManager формує не лише ім'я та пароль, але й дозволи користувача.

Для впровадження повноцінної системи безпеки в додатку буде створено механізм отримання доступу до окремих частин функціоналу через систему ролей. Ключовим етапом є створення PasswordEncoder, який визначає, як саме буде шифруватися пароль. У стандартній реалізації, якщо Encoder не

визначений, пароль залишається у своєму первісному вигляді. Проте в створюваному застосунку такий NoOpPasswordEncoder не призначений для використання, і замість цього пароль буде шифруватися з використанням BCryptPasswordEncoder з підвищеним рівнем захисту.

Один з методів авторизації в Spring Security використовує анотацію @PreAuthorize, де за допомогою виразів можна чітко визначити правила, за якими модуль авторизації вирішує, чи має бути дозволено чи заборонено виконання певної операції. У додатку для автоматизації зміни стану ресурсів тестування доступ до кожної частини функціоналу контролюється за допомогою вказаної анотації і вимагає відповідних дозволів. Наприклад, для простого перегляду наявних ресурсів тестування необхідний дозвіл користувача і визначається таким чином: @PreAuthorize("hasAuthority('read')").

Останнім етапом буде налаштування автентифікації та авторизації під час взаємодії з базою даних через DaoAuthenticationProvider[49]. Цей постачальник автентифікації, ймовірно, є одним із найбільш часто використовуваних у фреймворку. Подібно до більшості інших провайдерів автентифікації, DaoAuthenticationProvider використовує UserDetailsService для отримання імені користувача, пароля та повноважень, але, відмінно від більшості інших провайдерів, які також використовують UserDetailsService, цей провайдер автентифікації фактично очікує представлення пароля та перевіряє його валідність.

2.6. Аналіз безпечності та швидкості алгоритмів шифрування

Захист інформації шляхом використання криптографії є ключовим компонентом забезпечення безпеки даних у програмі. Криптографія виступає основою для забезпечення безпеки в сучасних інформаційних системах і використовується в різних аспектах, таких як зберігання облікових даних, електронна пошта, мобільні платежі і цифрові транзакції, що становлять лише частину найпоширеніших застосувань.

Криптографія вважається однією з найбільш складних тем в інформаційній безпеці. Застосовувати власні криптографічні бібліотеки може бути небезпечно, оскільки неможливо власноруч перевірити новий алгоритм на всі можливі загрози та постійно слідкувати за оновленням ризиків безпеки даних. Більшість сучасних мов програмування вже включають реалізовані криптографічні бібліотеки та модулі, тому перед прийняттям рішення щодо найкращого способу захисту інформації рекомендується провести детальний аналіз та порівняння існуючих можливостей.

Шифрування - це процес перетворення даних у такий формат, який може бути зрозумілий лише тим, хто володіє засобами для відновлення їх початкового вигляду. Технології шифрування вважаються ключовим елементом будь-якого безпечного обчислювального середовища.

Безпека шифрування полягає у здатності алгоритму генерувати зашифрований текст, який важко або навіть неможливо відновити до початкового відкритого стану. Для належного налаштування будь-якої основної схеми шифрування, вирішальною є правильна відповідь на наступні питання:

- Оцінка рівня безпеки алгоритму відповідно до сучасної криптографічної літератури;
- Розгляд характеристик продуктивності алгоритму, таких як підтримка паралельного шифрування та його ефективність;
- Розгляд політичної придатності рішення щодо використання конкретного алгоритму;
- Використання належно підібраних ключів і розмірів для забезпечення ефективності шифрування.

Ключовою є ретельна настройка всіх параметрів, оскільки навіть невеличкий промах в безпеці конфігурації може зробити криптосистему вразливою до атак.

У ході дослідження, використовуючи бібліотеку javax.crypto, було проведено шифрування за допомогою найпоширеніших алгоритмів JDK Sun:

AES128, AES256, SHA256, SHA512, DES, Triple DES і Blowfish [50]. Для кожної реалізації вимірювалася швидкість шифрування у мілісекундах (див. рис. 2.9), і були надані основні характеристики, важливі для безпеки даних (див. табл. 1).

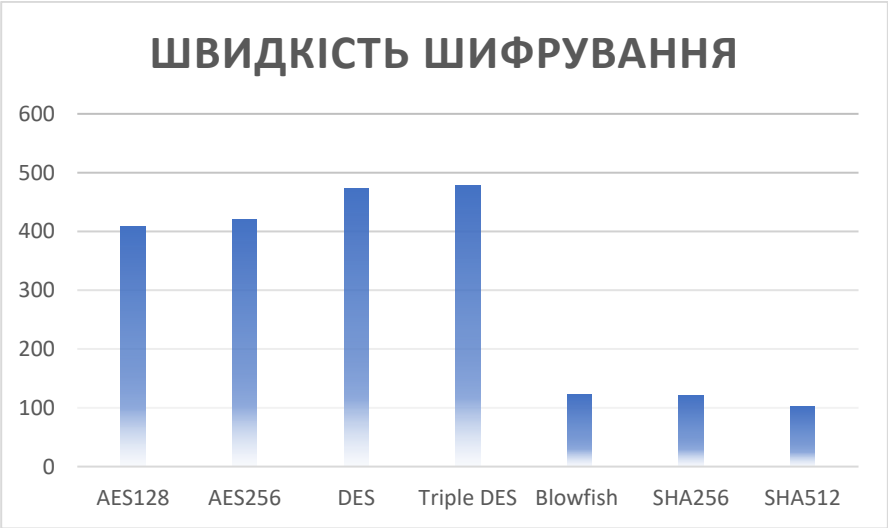

Рис. 2.9. Різниця швидкості роботи алгоритмів

Важливо відзначити, що AES визначається як Advanced Encryption Standard і представляє собою алгоритм, який більшість людей використовує в кінцевому рахунку, якщо вони не мають суттєвих причин використовувати щось інше. Це рішення є політично безпечним, оскільки стандарт шифрування Національного інституту стандартів і технологій США (NIST) рекомендує використання AES з 192 або 256-бітними ключами для шифрування конфіденційної інформації.

Таблиця 1

Порівняльна характеристика алгоритмів

Алгоритм	**Розміри ключа**	**Швидкість**	**Залежність швидкості від розміру ключа**	**Коментар**
AES	128-256	Середня	Так	Безпечний, його перевага полягає в тому, що він дозволяє використовувати 256-бітний

				розмір ключа, який має захистити від певних майбутніх атак (collision attacks і потенційних алгоритмів квантового обчислення)
DES	56	Повільна	Ні	Дуже небезпечно: навіть одна машина Copacobana вартістю 10000 USD може знайти ключ DES приблизно за тиждень.
Triple DES	112-168	Дуже повільна	Ні	Доволі безпечний, особливо для невеликих розмірів даних. 168-бітний варіант, за оцінками NIST, забезпечує безпеку даних до 20304 року, проте AES пропонує вищий рівень безпеки за меншої вартості CPU.
Blowfish	128-448	Швидка	Ні	Вважається безпечним, але пройшов менше крипто аналізу, ніж інші алгоритми. Blowfish було замінено Twofish, але останній не підтримується як стандарт у Java.
SHA	256-512	Дуже швидка	Так	Проблема полягає в тому, що лише SHA256 та SHA512 (тип SHA2, який генерує 256/512-бітове хеш-значення) залишається безпечним — SHA-1 (перша версія SHA, яка генерує 160-бітове хеш-значення) має відомі вразливості.

Покрім вибору алгоритму, важливо визначити розмір ключа, який гарантує "міцність" шифрування, припускаючи, що сам алгоритм в інших відносинах є безпечним. Розмір ключа зазвичай взаємозв'язаний з кількістю припущень, які зловмиснику потрібно буде зробити для знаходження ключа методом "brute force", припускаючи, що всі можливі ключі мають однакову ймовірність.

Використання ключів підвищує рівень безпеки методів захисту інформації. Алгоритми, що використовують ключі, можна розділити на дві основні категорії:

- Симетричні алгоритми шифрування використовують один і той же ключ для обох операцій – шифрування та дешифрування. Ці алгоритми можуть працювати в потоковому режимі (з бітами або байтами даних) або в блочному режимі (з фіксованими блоками даних). Зазвичай вони застосовуються для шифрування даних, файлів і, особливо, інформації у мережах зв’язку, таких як електронні листи, TLS, миттєві повідомлення і т. д.;
- Асиметричні алгоритми шифрування використовують два відмінних ключі – один для шифрування та інший для дешифрування. Ці алгоритми застосовуються для обчислення цифрових підписів і протоколів встановлення ключів.

Як визначити розмір ключа? Зазвичай мінімально необхідний розмір ключа визначається як комбінація максимально можливого розміру ключа для конкретного алгоритму, екстрапольованого на кількість років, протягом яких слід зберігати зашифровані конфіденційні дані. Наприклад, Національний інститут стандартів і технологій США (NIST) рекомендує мінімум "128 біт міцності" для забезпечення конфіденційності даних "після 2030 року". Згідно з найкращими практиками з шифрування даних, представленими компанією Visa для передачі номерів кредитних карток, "ключі повинні мати потужність принаймні 112 еквівалентних бітів"[51]. Вони фактично вважають 128 біт (як у мінімальному розмірі ключа AES) "міцнішими, ніж потрібно", можливо через обмежений термін служби кредитних карток.

2.7. Висновки

Під час проведення досліджень та аналізу було виявлено, що Spring Framework представляє собою обширну та зручну платформу для розробки веб-проектів на Java. Його склад включає велику кількість незалежних модулів, які

призначені для різних завдань, починаючи від простих веб-додатків і закінчуючи розробкою проектів Big Data.

Основна перевага фреймворку Spring полягає в здатності розробляти додаток як набір слабко зв'язаних компонентів за допомогою Dependency Injection. Чим менше компоненти програми залежать один від одного, тим простіше стає розробка нового функціоналу та підтримка існуючого. Проте в цьому підході до створення програми важливо зберігати баланс і групувати код у класах за їхньою бізнес-логікою, щоб дотримуватися принципу "Low Coupling, High Cohesion" у створенні додатку.

Для створення програми автоматизації зміни стану ресурсів тестування будуть використані наступні модулі фреймворку:

- Spring Boot;
- Spring Data JPA;
- Spring Security.

Кожен з них внесе свій внесок у розробку та функціонування програмного забезпечення. Spring Boot автоматично налаштує проект на основі стартових пакетів, ефективно управлятиме залежностями та спростить розгортання додатку на Spring за допомогою вбудованого сервера для його запуску. Специфікація Java для управління реляційними даними Spring Data JPA дозволить отримувати доступ і зберігати дані між об'єктом/класом Java і реляційною базою даних. Фреймворк Spring Security забезпечить аутентифікацію, авторизацію та захист від розповсюджених атак.

З придатними зусиллями практично будь-яку криптографічну систему можна успішно атакувати. Ключовим аспектом є те, скільки часу та ресурсів необхідно для її злому. Як вбачено з проведеного аналізу, існує багато деталей, на які варто звернути увагу, щоб правильно впровадити та реалізувати схему шифрування у додатку.

На даний момент найбільш перевіреним та збалансованим алгоритмом шифрування є AES. Для забезпечення безпеки даних протягом принаймні 20-30

років рекомендується використовувати 256-бітний ключ. 128-бітний ключ може використовуватися, якщо враховано вдумливий захист від методу "brute force" та при умові, що протягом терміну зберігання даних невелика ймовірність квантових обчислень. Наприклад, для даних кредитних карток можна бути впевненим, що термін їхнього дії закінчиться до появи квантових комп'ютерів у нашому повсякденному житті.

Отже, для шифрування особливо цінних данних у додатку було обрано алгоритм шифрування AES з 256-бітний ключем.

РОЗДІЛ 3.
РОЗРОБКА ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ЗМІНИ СТАНУ РЕСУРСІВ ТЕСТУВАННЯ

3.1. Структура інформаційної системи автоматизації зміни стану ресурсів тестування

Внутрішня структура створюваної системи являє собою веб-сайт, який складається з серверної частини (backend). Користувач системи взаємодіє безпосередньо через HTTP-протоколи, через функціонал якої має змогу:

- зареєструватись;
- увійти в систему;
- переглянути список курсів и т.д.;
- додати/видалити/змінити курси/заннятя, при умові наявності необдних дозволів у його ролі;
- оперувати дозволами та ролями користувачів, доступно для ролі admin.

Для автентифікації, авторизації, отримання дозволів та оперування даними з БД користувач взаємодіє з backend-частиною системи за допомогою API. Система комунцкації серверної частини додатку зображена на рис. 3.1.

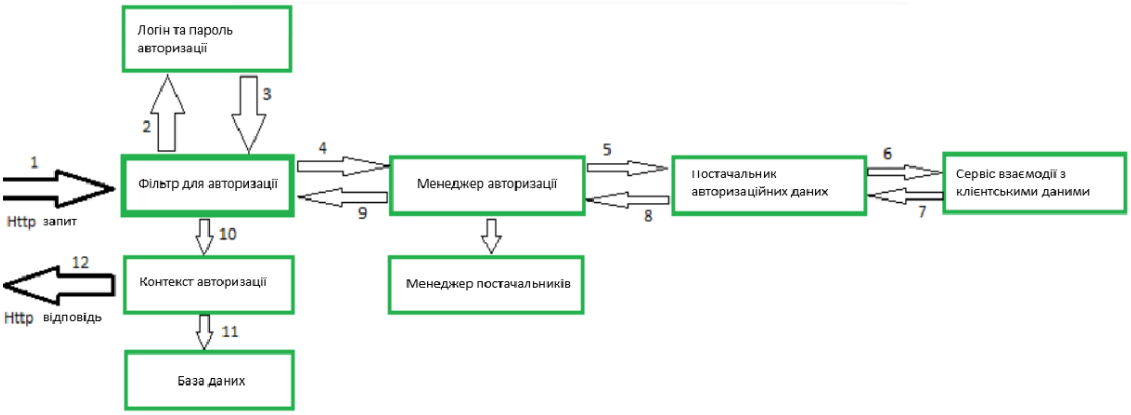

Рис. 3.1. Архітектура комунікації частин програми

Алгоритм взаємодії має наступну структуру:

1. Користувач, використовуючи веб-клієнт (браузер), ініціює запит до конкретної веб-сторінки застосунку через API;
2. Незалежно від початкової сторінки, система управління курсами та заннятями (тепер під назвою "SkillHub") перенаправляє користувача на сторінку авторизації;
3. Перед обробкою бізнес-логіки та відправкою результату, запит фільтрується ланцюжком фільтрів безпеки для аутентифікації та авторизації.;
4. Коли запит перехоплюється фільтром аутентифікації (AuthenticationFilter), він отримує дані із запиту, такі як ім'я користувача та пароль, і створює об'єкт аутентифікації;
5. Фільтр викликає метод аутентифікації менеджера аутентифікації (Authentication Manager), який, в свою чергу, викликає метод автентифікації менеджера провайдерів (Provider Manager);
6. Менеджер провайдерів викликає метод автентифікації відповідного AuthenticateProvider, який повертає Основний об'єкт автентифікації (Principal Authentication Object), якщо автентифікація успішна;
7. Для інтерфейсу з єдиним методом автентифікації в SkillHub обрано провайдер автентифікації DaoAuthenticationProvider;
8. Провайдер автентифікації отримує об'єкт користувача з бази даних, порівнює його з вхідними обліковими даними об'єкта автентифікації. У разі успішної автентифікації повертається Основний об'єкт автентифікації;
9. Після перевірок, якщо клієнт ввів URL, відмінний від авторизаційного, він перенаправляється на обрану функціональність або на початкову сторінку з усіма ресурсами тестування з урахуванням наявності необхідних дозволів;
10. У разі потреби зв'язку з базою даних, взаємодія з сервером Mongo відбувається через підключення, налаштоване при запуску додатку;

11. Контролер завершує обробку запиту, формує результат і надає відповідь для відображення презентаційній частині програми.

3.2. Структура та опис бази даних

Хоча у більшості систем управління базами даних існують стандартні функції для створення та отримання доступу до інформації у базі даних, методи виконання завдань можуть варіюватися.

Додаткові можливості, функції та підтримка є унікальними для кожної (СУБД). Для проекту SkillHub було вибрано систему MongoDB, враховуючи перераховані переваги:

- MongoDB сумісна з усіма сучасними платформами, такими як Windows, Unix, Linux;
- простота завантаження та управління;
- MongoDB забезпечує повну безпеку даних, оскільки лише авторизовані користувачі мають доступ до бази даних;
- масштабованість та можливість безкоштовного використання.

У застосунку SkillHub доступ до бази даних MongoDB налаштовано за допомогою Spring Data, яке є зручним механізмом для взаємодії з об'єктами бази даних, організації їх у репозиторії, а також для внесення змін та вилучення даних. В деяких випадках для виконання описаних операцій в додатку може бути достатньо лише оголосити інтерфейс та метод у ньому, без необхідності надання конкретної реалізації..

Однією з ключових точок взаємодії між базою даних та додатком SkillHub є репозиторій, який служить засобом, за допомогою якого сервіс отримує потрібні дані для виконання бізнес-логіки, як показано на рис. 3.2.

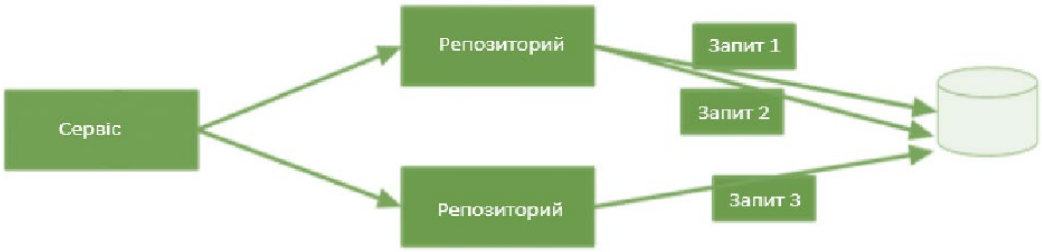

Рис. 3.2. Архітектура взаємодії з базою даних

Використований MongoRepository надає базові функції для виконання операцій пошуку, збереження та видалення даних (CRUD операції). Додаткові запити до даних можна конструювати безпосередньо за ім'ям методу, як це було реалізовано для пошуку користувача за адресою електронної пошти: Optional<User> findByEmail(String email);;

Організація бази даних може змінюватися відповідно до потреб користувачів у створенні необхідних тестових ресурсів. У вихідному наборі вже наявні стандартні ресурси, і структура бази даних має наступний формат:

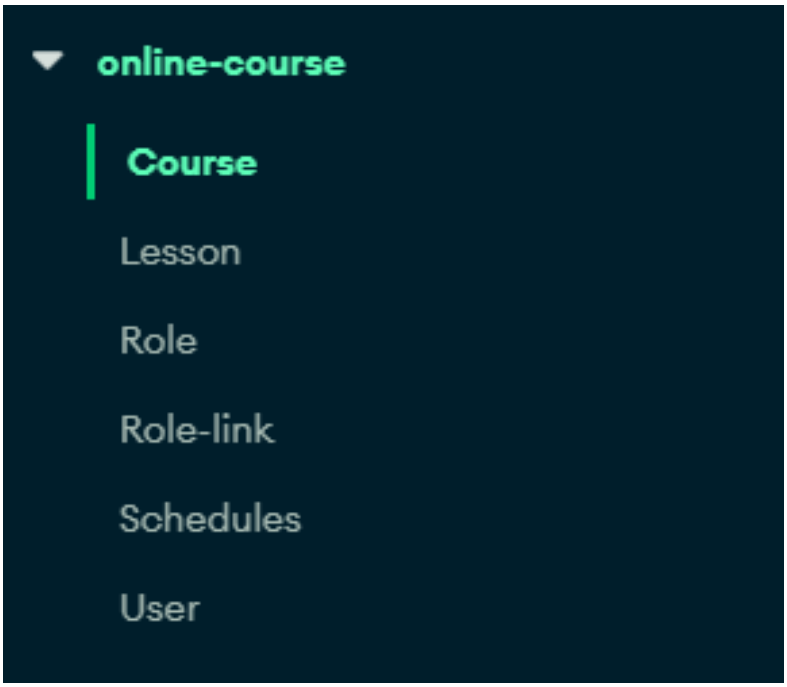

Рис. 3.3. Структура бази даних

3.3. Опис ІС SkillHub та алгоритмів її функціонування

Обробка основних даних в системі відбувається на серверній стороні додатку. Вхідні дані для програмного комплексу надходять у вигляді текстових даних з форм, які користувач заповнює для керування курсами або заннятями. Результатом роботи веб-додатку є набір збережених у системі об’єктів, готових до використання та відображених на сторінках перегляду, згрупованих за їх типом.

Основна мета системи полягає в забезпеченні клієнту інтуїтивно зрозумілого та зручного інструменту для ефективного керування онлайн курсами, а також можливості створення сценаріїв для їхньої зміни у майбутньому.

Для розробки застосунку було застосовано наступні технології:

- Java 21 – створення серверної логіки;
- Spring Security – для аутентифікації та авторизації;
- Spring Data – API для роботи з базою даних;
- Spring Boot – створення бізнес логіки додатку;
- Mongo Compass– керування БД;
- Project Lombok – скорочення шаблонного коду;

Розроблений застосунок для керування онлайн курсами має відповідати наступним вимогам:

- інтуїтивно-зрозумілий принцип для роботи з системою;
- наявна система автентифікації та авторизації;
- реєстрація користувачів;
- зручна навігація за допомогою API;
- можливість роботи в браузерах на ПК, ноутбуках та мобільних пристроях з різними характеристиками та конфігураціями;
- шифування особливо цінних даних користувачів;
- наявність модулів створення, оновлення та видалення курсів;

- гнучкість для модифікацій та масштабування.

Після переходу за інтернет-адресою листування програмного комплексу SkillHub, користувач побачить сторінку авторизації, що зображена на рис. 3.4.

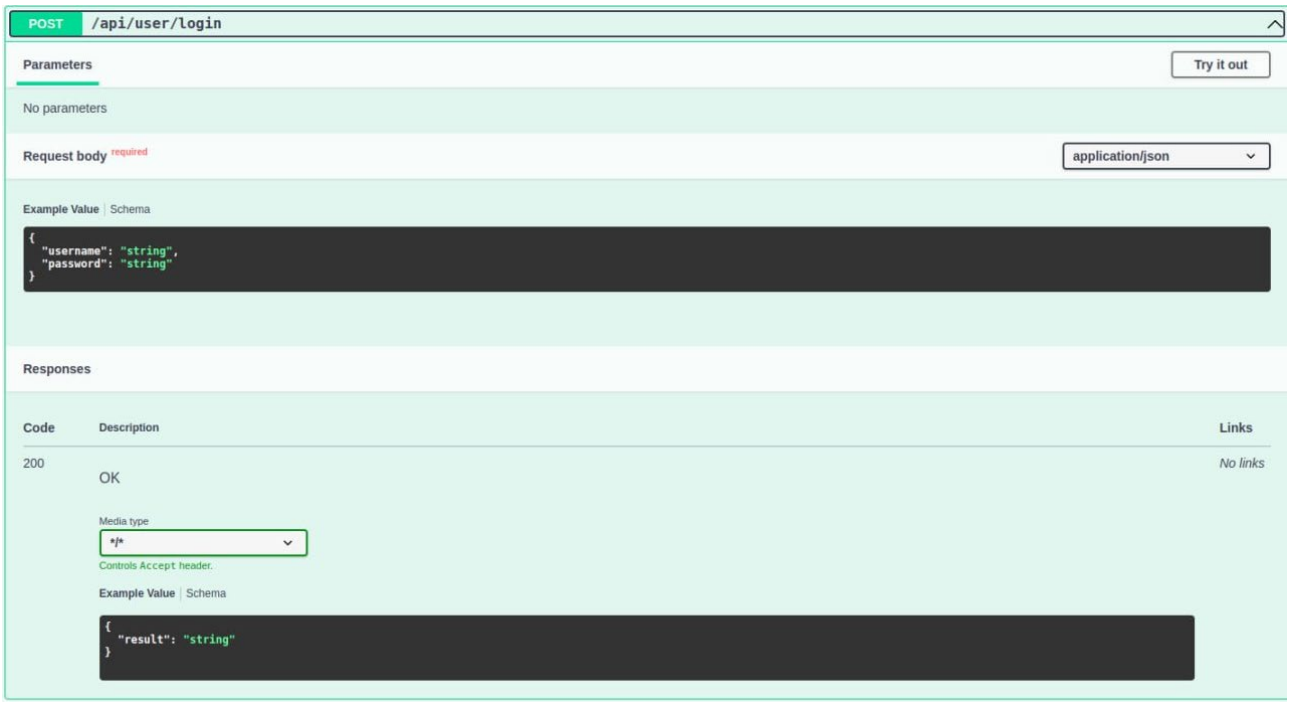

Рис. 3.4. API авторизації

У разі, якщо користувач має відповідні авторизаційні дані, він може увійти до системи, перейшовши за допомогою API "login" та введенням свого логіну та пароля у відповідні поля форми. Для реєстрації в системі слід скоростатись API "register" та вказати облікові дані у формі, яка зображена на рис. 3.5.

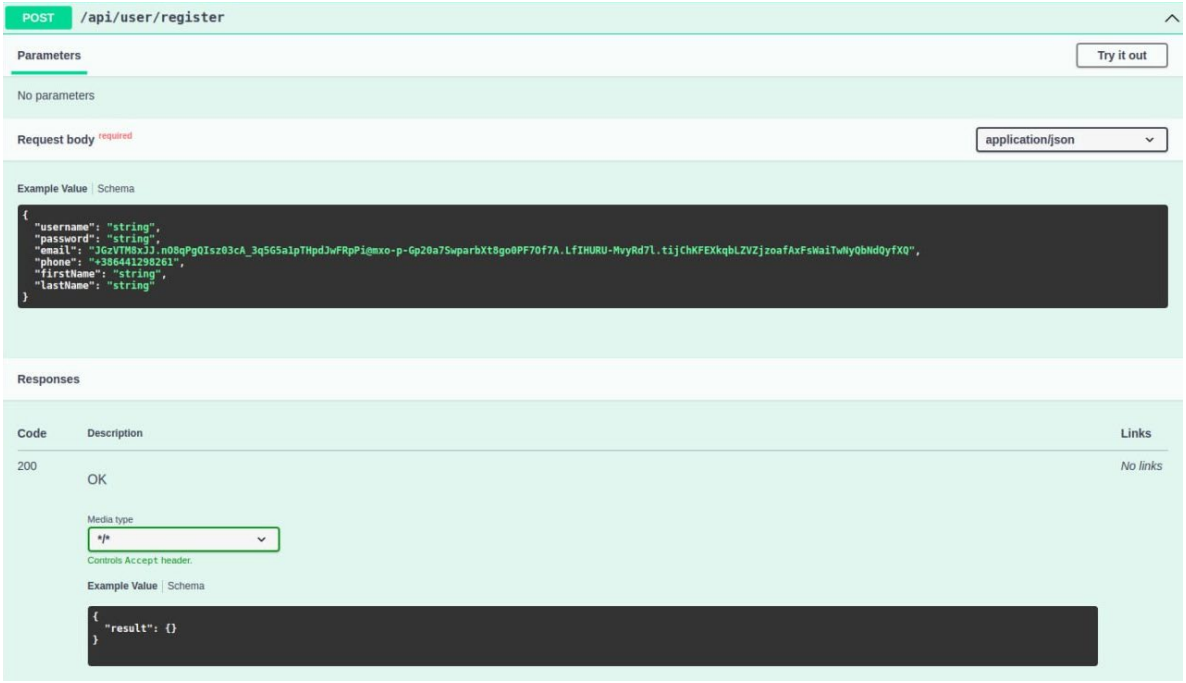

Рис. 3.5. API реєстрації в системі

Після успішної аворизації користувач отримує детальну інформацію про user, зображену на рис. 3.6.

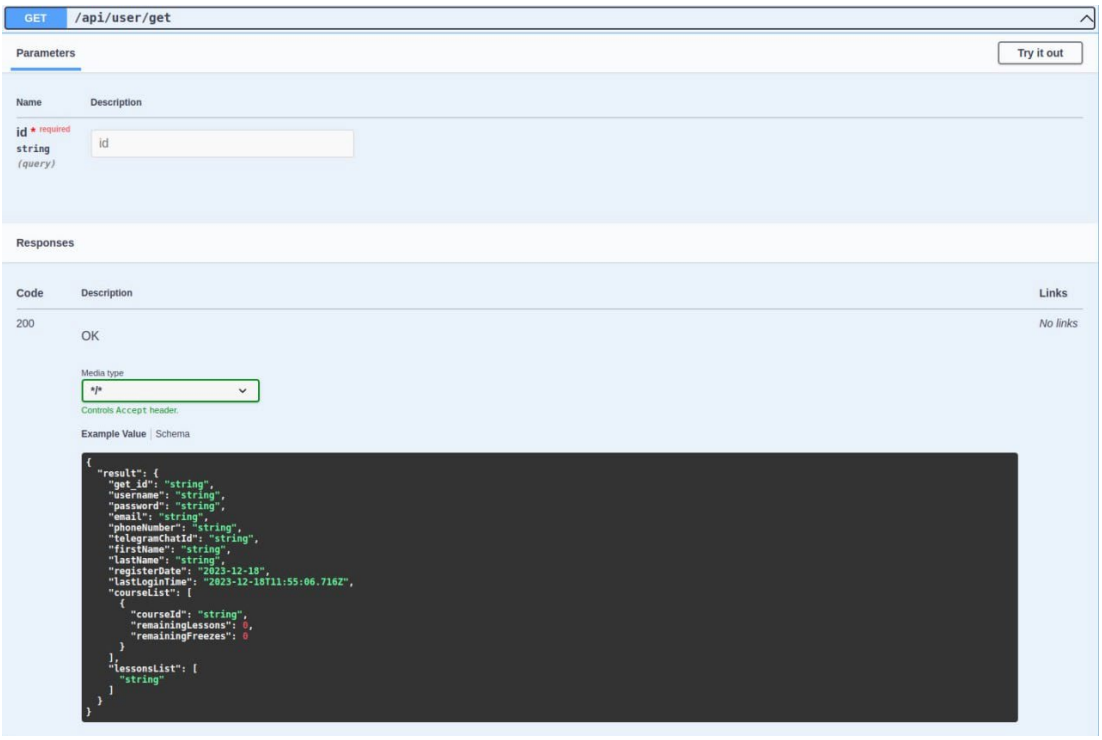

Рис. 3.6. API детальної інформації про користувача

Якщо користувач не може авторизуватись, він може відновити свій логін та пароль, за допомогою відправленного посилання через API flow (рис. 3.7 – 3.8).

Рис. 3.7. API отримання посилання для відновлення паролю

Рис. 3.8. API встановлення нового паролю

Для взаємодії з курсами або заннятями користувач може використовувати ряд схожих API (рис. 3.9 – 3.14).

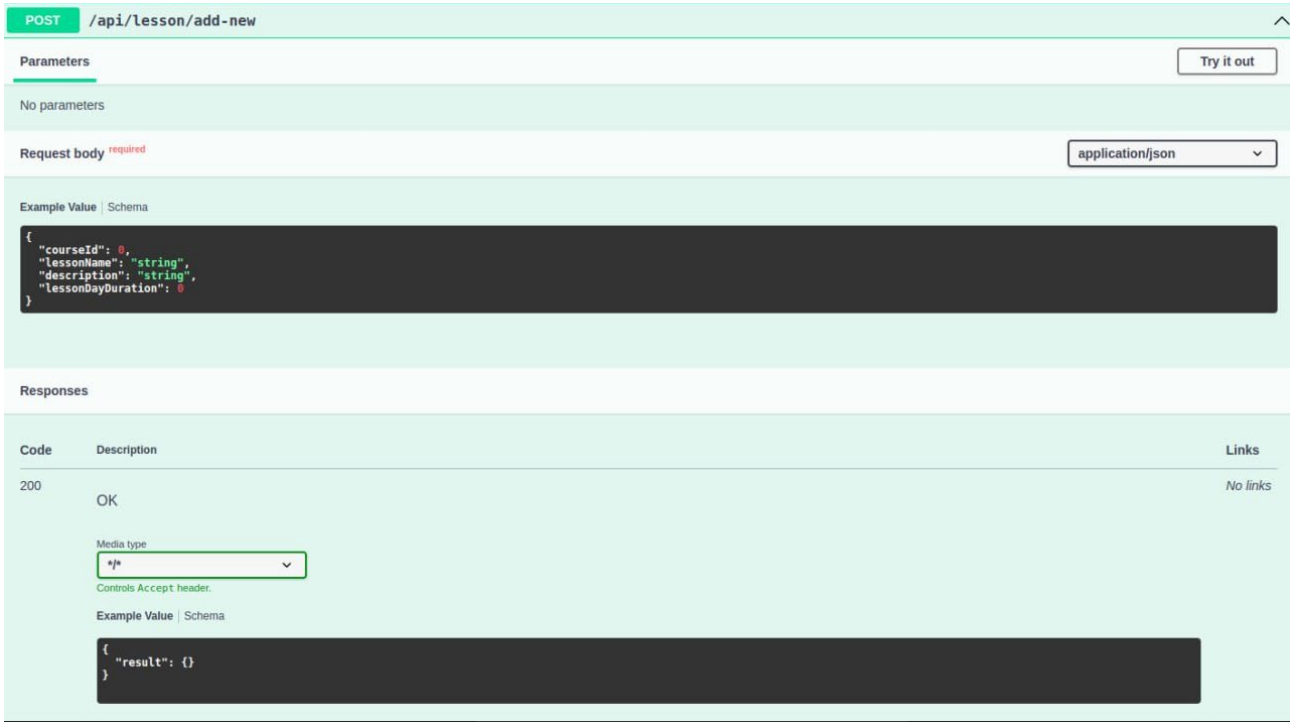

Рис. 3.9. API створення нового заннятя/курсу

При необхідності внесення змін у ресурс, є можливість створення, редагування та видалення об’єктів з відкладенням у часі або без нього. Для створення нового ресурсу з можливим планування, сторінка буде мати такий вигляд, як на рисунках 3.11 – 3.12.

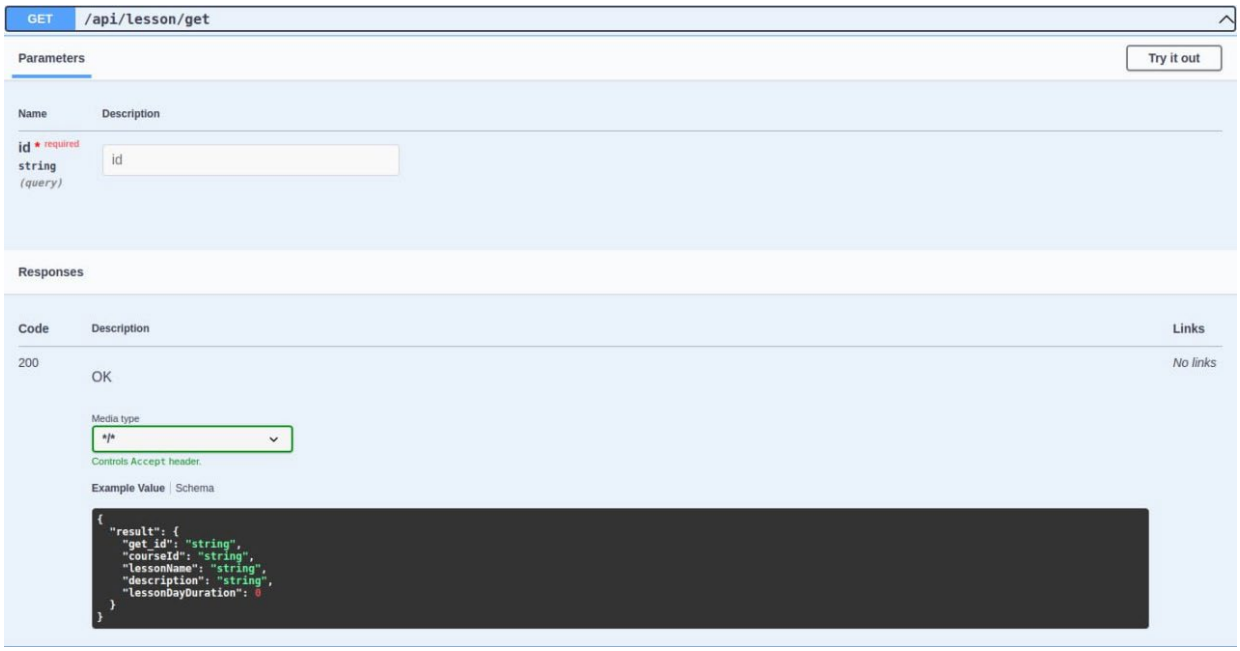

Рис. 3.10. API отримання курсу/заннятя по його id

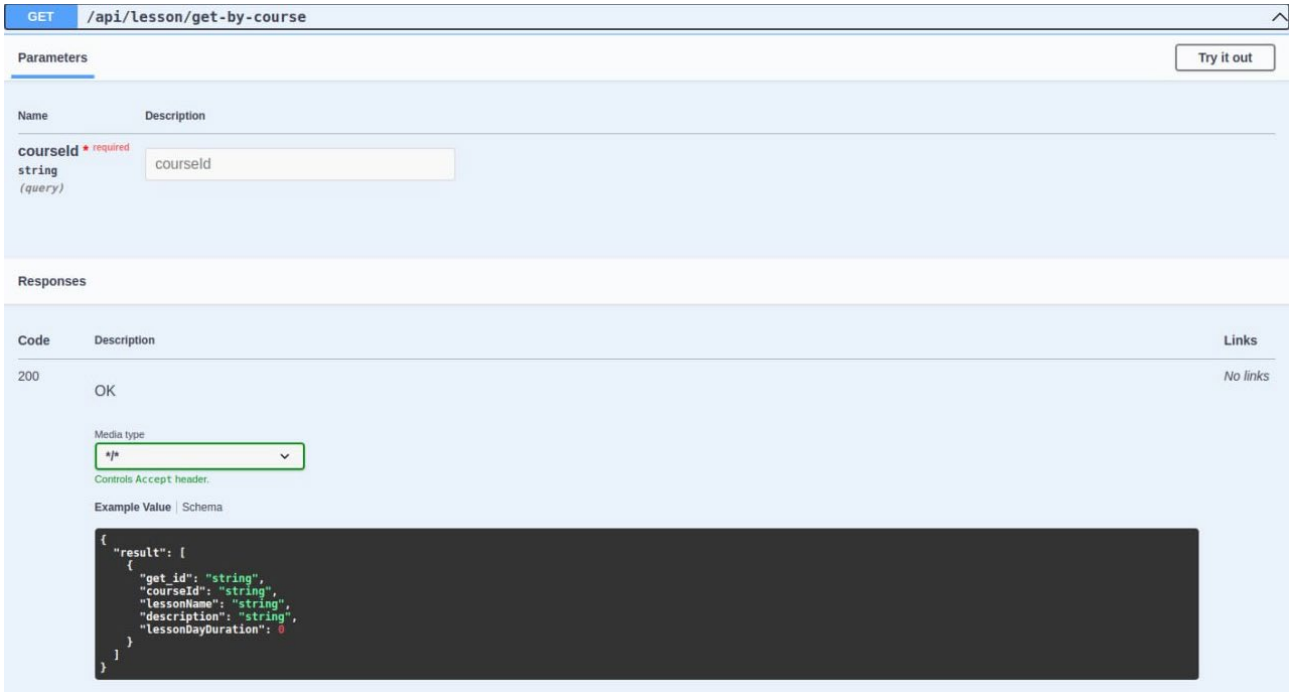

Рис. 3.11. API отримання списку заннять по курсу

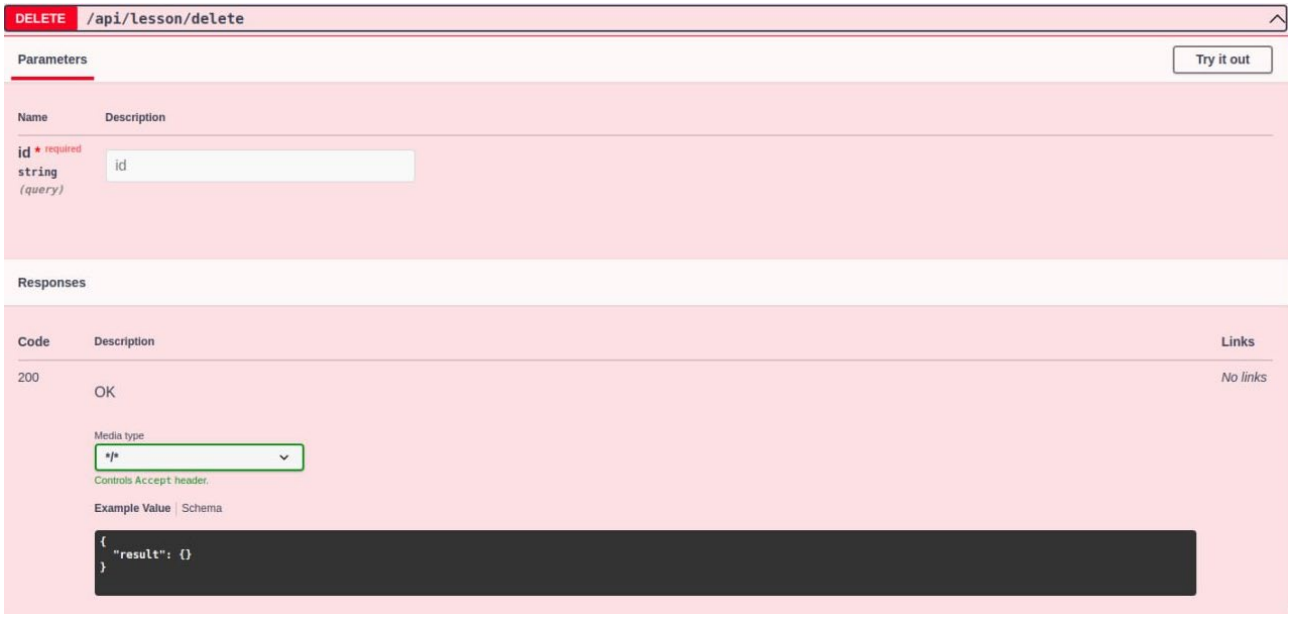

Рис. 3.12. API видалення курсу/заннятя

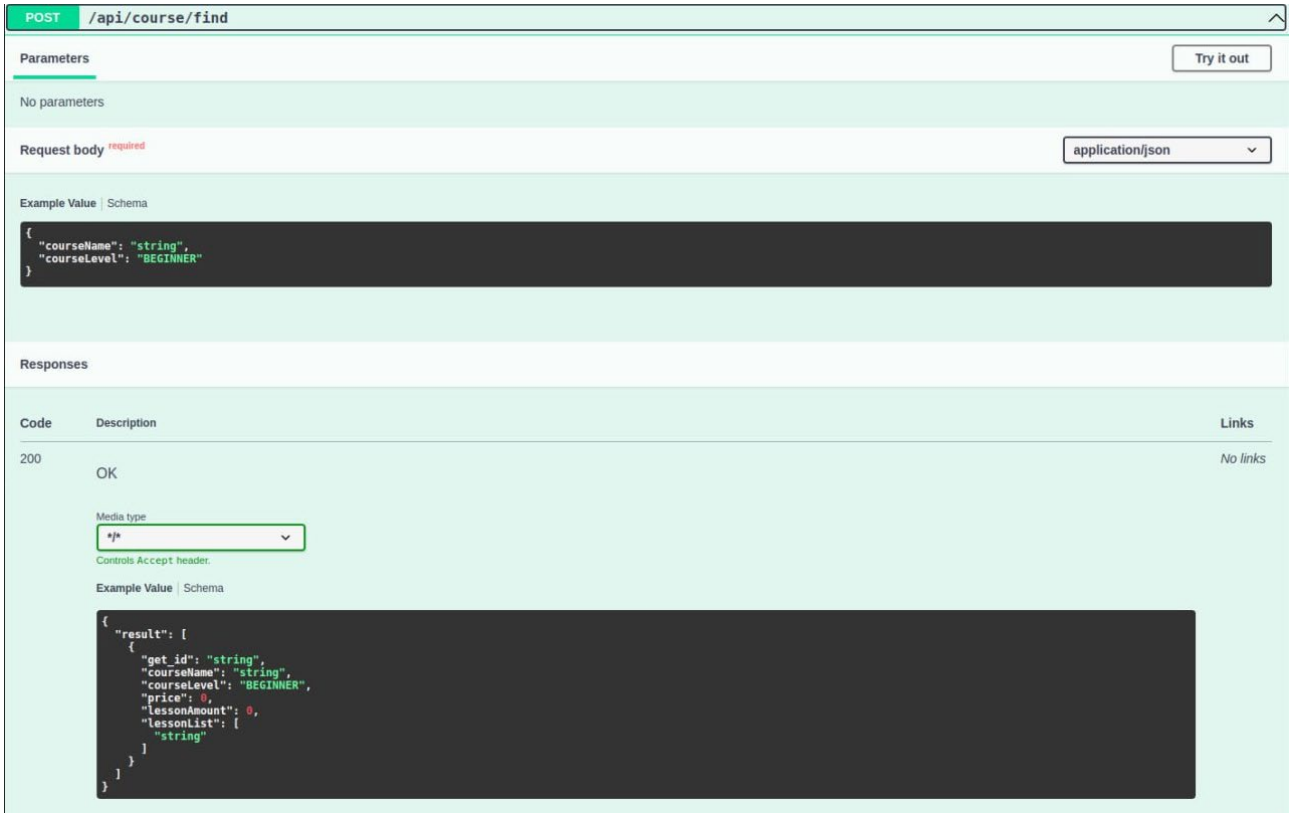

Рис. 3.13. API пошуку курсу за вказанними параметрами

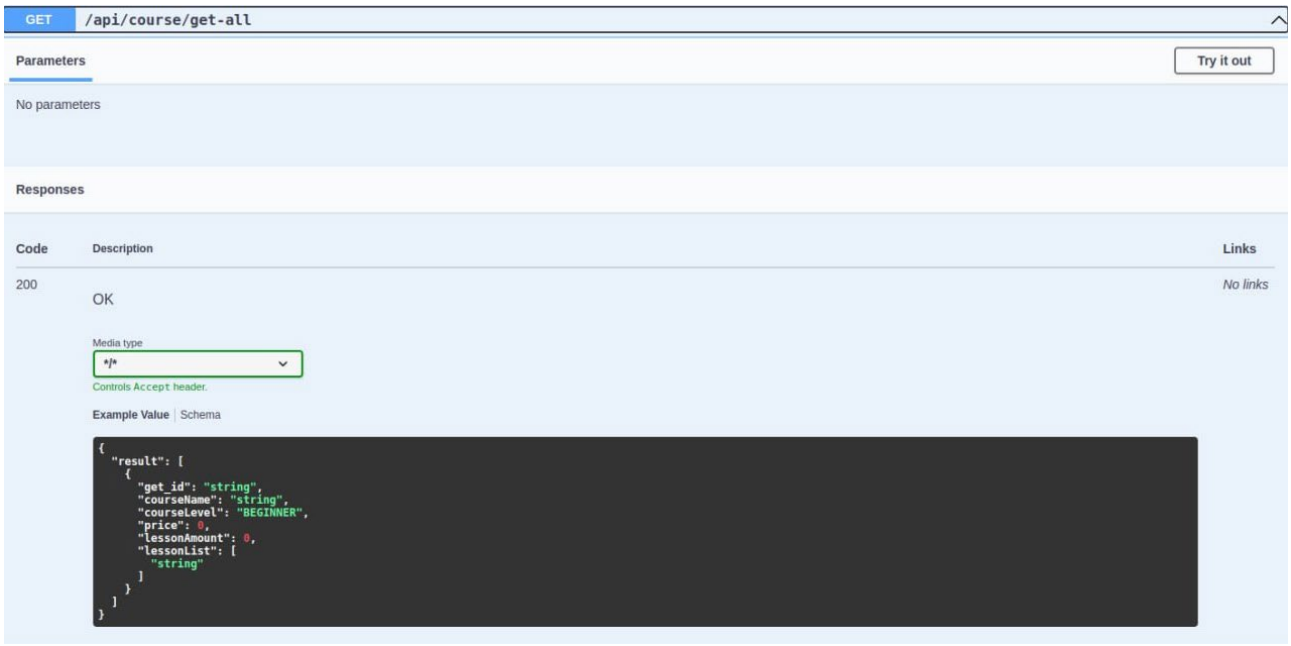

Рис. 3.14. API отримання списку усіх активних курсів

Якщо користувач спробую зробити дію, на яку не має належних прав або ролі, метою безпечності вихідних даних відобразиться узагальнена помилка.

При редагуванні обраного курсу або заннятя, id є незмінним, а всі інші параметри для зручності проставляються автоматично з вже наявної інформації.

Також слід зазначити, що при створенні та редагуванні курсу або заняття з паролем користувач вводить його без шифрування заздалегідь, всі процеси проходять автоматично на серверній частині додатку, зберігаючи час клієнту.

3.4. Оцінка параметрів і характеристик веб-застосунку

Для оцінки ергономічності веб-застосунку використовується система параметрів, яка не враховує користувацький інтерфейс, навігацію та взаємодію з системою.

Методика включає:

- Зручність використання: Середня кількість кроків для виконання основних завдань: 3; Середній час на виконання основних завдань: 15 секунд.
- Виявлення Вразливостей: Автоматизовані сканери виявили 3 потенційні вразливості, які були виправлені.

- Стійкість до Атак: "Penetration testing" показав високий рівень стійкості до типових атак; Моніторинг системи: Не зафіксовано ніякої аномальної активності.

Порівняння із конкурентами показало, що розроблений веб-застосунок має конкурентоздатну ергономіку та вищий рівень захищеності. Недоліків, які могли б суттєво вплинути на безпеку чи зручність використання, виявлено не було.

Отже, розроблений веб-застосунок вирізняється високою ергономічністю та ефективністю, а також демонструє високий рівень захищеності, роблячи його важливим та конкурентоздатним продуктом на ринку онлайн курсів.

3.4. Висновки

У цьому розділі було проведено проектування та розробку системи керування для сайту з онлайн курсами SkillHub, використовуючи різноманітні технології, що є актуальними на сучасному ринку java-застосунків.

У процесі розробки було використано Spring Data, а також методи інтерфейсу MongoRepository для взаємодії з базою даних. Основний фокус був спрямований на створення гнучкої системи ролей та контролю доступу до різних функціональних частин. Для зручного керування курсами та заннятями користувачеві був наданий інтуїтивно-зрозумілий API функціонал, який дозволяє створювати, змінювати та видаляти курси або заннятя без необхідності володіння навичками взаємодії з БД.

Також було проведено ретельне тестування додатку, під час якого не було виявлено очевидних можливостей порушення логіки системи. Вся структура зберігає можливість масштабування та додавання нової бізнес-логіки в найкоротші терміни та з мінімальними витратами.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи була створена інформаційна система для керування онлайн курсами та заннятями "SkillHub", що включає клієнтські та адміністративні компоненти.

У даній роботі розроблено надійну архітектуру та втілено її у сучасну реалізацію, базуючись на відомому та ефективному фреймворку Spring. Клієнтська частина додатку дозволяє виконувати всі операції з CRUD без необхідності взаємодії з кодом, забезпечуючи доступ до управління курсами та заннятями відповідно до наданих дозволів ролі користувача. Адміністративна частина є інтуїтивно зрозумілою і не вимагає спеціальної підготовки для управління ролями та статусом користувачів у системі.

Для операційного використання додатку було створено та початково заповнено базу даних, що забезпечує високу ефективність завдяки своїй документо-орієнтованій структурі.

Створений додаток сприятиме підвищенню продуктивності, швидкості та складності сценаріїв для повноцінного керування онлайн курсами та заннятями. Основа додатку є багатофункціональною, що дозволяє застосовувати отримані результати як у практичній розробці програмного забезпечення зі зміною бізнес-логіки, так і для поглиблення рівня самостійного освоєння галузі. Для створення застосунку використовувалися сучасні технології програмування: Java 21, Spring Boot, Spring Data, MongoDB, Spring Security, Lombok.

Створений проект SkillHub спрямований на підвищення зручності управління курсами або заннятями та зменшення дій для внесення змін в об'єкти. Отже, він буде корисним для підтримки сайтів для більшості надавачів послуг. Таким чином, в процесі виконання кваліфікаційної роботи було досягнуто всіх цілей та виконано усі вимоги, які висувалися при проектуванні інформаційної системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Neal Ford, Mark Richards, Pramod Sadalage, Zhamak Dehghani. Software Architecture: The Hard Parts: Modern Trade-Off Analyses for Distributed Architectures – O'Reilly Media, 2021. – 459 p.

2. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Grady Booch. Design Patterns: Elements of Reusable Object-Oriented Software – Addison-Wesley Professional, 1994. – 416 p.

3. Jim Blandy, Jason Orendorff, Leonora Tindall. Programming Rust: Fast, Safe Systems Development – O'Reilly Media, 2021. – 735 p.

4. Martin Fowler. Refactoring: Improving the Design of Existing Code (2nd Edition) – Addison-Wesley Professional, 2018. – 448 p.

5. Greg L. Learning Spring Boot 2.0 - Second Edition: Simplify the development of lightning fast applications based on microservices and reactive programming – Apress, 2017. – 370 p.

6. Sam Newman. Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith – O'Reilly Media, 2019. – 272 p.

7. Програмуваня по-українськи URL: programming.in.ua (дата звернення: 03.05.2021).

8. Heckler M. Spring Boot: Up and Running: Building Cloud Native Java and Kotlin Applications – O'Reilly Media, 2021. – 328 p.

9. Building RESTful Web Services / URL: https://www.tutorialspoint.com/spring_boot/spring_boot_building_restful_web_servic.htm (дата звернення: 08.10.2023).

10. Deinum M. Spring 5 Recipes – A problem-solution approach – Apress, 2017. – 870 p.

11. What is Spring Framework? An Unorthodox Guide / URL: https://www.marcobehler.com/guides/spring-framework (дата звернення: 11.09.2023).

12. Java EE at a Glance/ URL: https://www.oracle.com/java/technologies/java-ee-glance.html (дата звернення: 29.09.2023).

13. Spring vs. the World: Comparing Spring Boot Alternatives / URL: https://www.jrebel.com/blog/spring-boot-alternatives (дата звернення: 21.09.2023).

14. Benjamin Evans, Martijn Verburg, Jason Clark. The Well-Grounded Java Developer, Second Edition – Manning, 2023. – 74 p.

15. Security with Spring / URL: https://www.baeldung.com/security-spring (дата звернення: 07.11.2023).

16. Excellent Ways to Secure Your Spring Boot Application / URL: https://developer.okta.com/blog/2018/07/30/10-ways-to-secure-spring-boot (дата звернення: 22.11.2023).

17. Spring Framework Annotations / URL: https://springframework.guru/spring-framework-annotations/ (дата звернення: 26.02.2021).

18. Greg L. Learning Spring Boot 2.0 - Second Edition: Simplify the development of lightning fast applications based on microservices and reactive programming – Apress, 2017. – 370 p.

19. Antonov A. Spring Boot 2.0 Cookbook - Second Edition: Configure, test, extend, deploy, and monitor Spring Boot application – Packt Publishing, 2018. – 286 p.

20. Spring Boot / URL: https://spring.io/projects/spring-boot (дата звернення: 17.10.2023).

21. Eric Freeman. Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software 2nd Edition – O'Reilly Media, 2021. – 669 p.

22. Spring Boot : Auto Configuration / URL: https://javagyansite.com//2020/02/17/spring-boot-auto-configuration/ (дата звернення: 11.10.2023).

23. Spring Security @PreAuthorize Annotation / URL: https://www.appsdeveloperblog.com/spring-security-preauthorize-annotation-example/ (дата звернення: 12.10.2023).

24. Moises Macero. Learn Microservices with Spring Boot: A Practical Approach to RESTful Services using RabbitMQ, Eureka, Ribbon, Zuul and Cucumber – Apress, 2017. – 344 p.

25. How to secure REST with Spring Security / URL: https://www.infoworld.com/article/3630107/how-to-secure-rest-with-spring-security.html (дата звернення: 15.11.2023).

26. Walls C. Spring in Action (5th Edition) –Manning, 2018. –520 p.

27. Inversion of Control(IoC) in Spring / URL: https://javagyansite.com//2018/07/22/inversion-of-controlioc/ (дата звернення: 18.10.2023)

28. Spring MVC Execution Flow / URL: https://javagyansite.com//2020/03/01/spring-mvc-execution-flow/ (дата звернення: 20.10.2023).

29. Spilca L. Spring Security in Action –Manning, 2020. –560 p.

30. Робимо серіалізацію в JSON та назад з Jackson. [Електронний ресурс]. URL:https://blog.ithillel.ua/ru/articles/delaem-serializatsiyu-v-json-i-obratno-c-jackson(дата звернення 15.05.2023).

31. Паттерн проєктування MVC.[Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-%D0%B2%D0%B8%D0%B4-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80 (дата звернення 14.05.2023).

32. MySQL Documentation / URL: https://dev.mysql.com/doc/ (дата звернення: 22.10.2023).

33. MongoDB Documentation.[Електронний ресурс]. URL: https://www.mongodb.com/docs/(дата звернення 15.05.2023)

34. Додання бібліотек у проект Intellij IDEA. [Електронний ресурс].URL: https://javadevblog.com/kak-dobavitbiblioteku-jar-fajl-v-proekt-intellij-idea.html(дата звернення 11.05.2023).

35. Налаштування CORS за допомогою SpringBoot.[Електронний ресурс].URL: https://www.baeldung.com/spring-cors (дата звернення 08.06.2023)

36. Інтерфейси в Java. [Електронний ресурс].URL: https://www.fandroid.info/interfejsy/(дата звернення 23.05.2023).

37. Common Application Properies. [Електронний ресурс].URL: https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html(дата звернення 15.05.2023).

38. Walls C. Spring Boot in Action –Manning, 2016. –264 p.

39. Scheduled Tasks Documentation / URL: https://www.aquaclusters.com/app/home/project/public/aquadatastudio/wikibook/Documentation21.0/page/Scheduled-Tasks/Scheduled-Tasks / (дата звернення: 02.11.2023).

40. Transaction Status / URL: https://epayments-support.ingenico.com/en/get-started/transaction-status-full/ (дата звернення: 15.11.2023).

41. Cosmina I., Harrop R., Schaefer C., Ho C. Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools (5th edition) –Apress, 2017. –878p.

42. Spring scheduler – виконання коду за розкладом / URL: https://java-master.com/spring-scheduler (дата звернення: 29.11.2023).

43. K. Siva Prasad Reddy. Beginning Spring Boot 2: Applications and Microservices with the Spring Framework – Apress, 2017. – 324 p.

44. Java Spring vulnerabilities / URL: https://cybersecurity.att.com/blogs/labs-research/java-spring-vulnerabilities (дата звернення: 13.09.2023).

45. Sam Newman. Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith – O'Reilly Media, 2019. – 272 p.

46. Michael Feathers. Working Effectively with Legacy Code – Pearson, 2004. – 464 p.

47. Titus Winters, Tom Manshreck, Hyrum Wright. Software Engineering at Google: Lessons Learned from Programming Over Time – O'Reilly Media, 2020. – 599 p.

48. Merih Taze. Engineers Survival Guide: Advice, tactics, and tricks After a decade of working at Facebook, Snapchat, and Microsoft –Taze, 2021. – 245 p.

49. Sam Newman. Building Microservices: Designing Fine-Grained Systems – O'Reilly Media, 2021. – 612 p.

50. Java AES Encryption and Decryption / URL: https://www.baeldung.com/java-aes-encryption-decryption (дата звернення: 26.09.2023).

51. Encryption and Decryption in Java Cryptography / URL: https://www.veracode.com/blog/research/encryption-and-decryption-java-cryptography (дата звернення: 14.09.2023).

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

MongoConfig.java

```
package com.example.onlinecourse.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.mongodb.MongoDatabaseFactory;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.data.mongodb.core.convert.DefaultDbRefResolver;
import org.springframework.data.mongodb.core.convert.DefaultMongoTypeMapper;
import org.springframework.data.mongodb.core.convert.MappingMongoConverter;
import org.springframework.data.mongodb.core.mapping.MongoMappingContext;

@Configuration
public class MongoConfig{

   @Bean
   public MongoTemplate mongoTemplate(MongoDatabaseFactory mongoDbFactory, MongoMappingContext
context) {

      MappingMongoConverter converter = new MappingMongoConverter(new
DefaultDbRefResolver(mongoDbFactory), context);
      converter.setTypeMapper(new DefaultMongoTypeMapper(null));

      MongoTemplate mongoTemplate = new MongoTemplate(mongoDbFactory, converter);

      return mongoTemplate;
   }
}
```

ValidationUtils.java

```
package com.example.onlinecourse.utils;

import lombok.Data;
import lombok.Getter;

@Data
public class ValidationUtils {
   public static final String PARAM_NOT_SPECIFIED_VALIDATION_MESSAGE = "Required param is not
specified!";
```

```
    public static final String INCORRECT_PARAM_VALIDATION_MESSAGE = "Specified param is
incorrect!";

    public static final String EMAIL_VALIDATION_REGEX = "^[\\w\\.-]+@[a-zA-Z\\d\\.-]+\\.[a-zA-Z]{2,}$";
    public static final String PHONE_VALIDATION_REGEX = "^\\+38\\d{10}$";

    public static void checkArgument(boolean expression, String message) {
      if (!expression) {
        throw new IllegalArgumentException(message);
      }
    }

}
```

UserController.java

```
package com.example.onlinecourse.controllers;

import com.example.onlinecourse.controllers.dto.CommonApiResponseWrapper;
import com.example.onlinecourse.controllers.dto.user.ForgetPasswordWebReqDto;
import com.example.onlinecourse.controllers.dto.user.LoginUserWebReqDto;
import com.example.onlinecourse.controllers.dto.user.RegisterNewUserWebReqDto;
import com.example.onlinecourse.controllers.dto.user.SetNewPasswordWebReqDto;
import com.example.onlinecourse.services.user.UserComponent;
import com.example.onlinecourse.services.user.pojo.ForgotPasswordParamsDto;
import com.example.onlinecourse.services.user.pojo.UserDto;
import jakarta.validation.Valid;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@Slf4j
@RestController
@RequestMapping("/api/user")
public class UserController {

    private final UserComponent userComponent;

    // <editor-fold defaultstate="collapsed" desc="***Init and setters***">

    @Autowired
    public UserController(UserComponent userComponent) {
      this.userComponent = userComponent;
```

```
    }

    // </editor-fold>

    @PostMapping("/register")
    public CommonApiResponseWrapper registerUser(@RequestBody @Valid
                                    RegisterNewUserWebReqDto paramsDto) {
      try {
        userComponent.registerNewUser(paramsDto);

        return new CommonApiResponseWrapper();
      } catch (Exception exception) {
        log.warn(exception.getMessage());
        throw exception;
      }
    }

    @PostMapping("/login")
    public CommonApiResponseWrapper<String> loginUser(@RequestBody @Valid LoginUserWebReqDto
paramsDto) {
      try {
        return new CommonApiResponseWrapper<>(userComponent.loginUser(paramsDto.getUsername(),
paramsDto.getPassword()));
      } catch (Exception exception) {
        log.warn(exception.getMessage());
        throw exception;
      }
    }

    @PostMapping("/forgot-pass")
    public CommonApiResponseWrapper<String> forgetPassword(@RequestBody @Valid
ForgetPasswordWebReqDto paramsDto) {
      try {
        String userId = userComponent.forgotPassword(ForgotPasswordParamsDto.builder()
            .username(paramsDto.getUsername())
            .email(paramsDto.getEmail())
            .phone(paramsDto.getPhone())
          .build());
        return new CommonApiResponseWrapper<>(userId);
      } catch (Exception exception) {
        log.warn(exception.getMessage());
```

```
            throw exception;
        }
    }

    @PostMapping("/set-new-password")
    public CommonApiResponseWrapper setNewPassword(@RequestBody @Valid
SetNewPasswordWebReqDto paramsDto) {
        try {
            userComponent.setNewUserPassword(paramsDto.getUserId(), paramsDto.getPassword());
            return new CommonApiResponseWrapper<>();
        } catch (Exception exception) {
            log.warn(exception.getMessage());
            throw exception;
        }
    }

    @GetMapping("/get")
    public CommonApiResponseWrapper<UserDto> getUser(@RequestParam(name = "id") String userId) {
        try {
            return new CommonApiResponseWrapper<>(userComponent.getUserById(userId));
        } catch (Exception exception) {
            log.warn(exception.getMessage());
            throw exception;
        }
    }
}
```

UserComponent.java

```
package com.example.onlinecourse.services.user;

import com.example.onlinecourse.controllers.dto.user.RegisterNewUserWebReqDto;
import com.example.onlinecourse.services.crypt.CryptService;
import com.example.onlinecourse.services.roles.RoleComponent;
import com.example.onlinecourse.services.roles.enums.RoleEnum;
import com.example.onlinecourse.services.telegram.TelegramService;
import com.example.onlinecourse.services.user.pojo.ForgotPasswordParamsDto;
import com.example.onlinecourse.services.user.pojo.UserDto;
import jakarta.validation.constraints.NotNull;
import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
```

```
import java.time.LocalDate;
import java.util.Collections;

import static com.example.onlinecourse.utils.ValidationUtils.checkArgument;
import static io.micrometer.common.util.StringUtils.isNotBlank;

@Slf4j
@Component
public class UserComponent {
   private final UserService userService;
   private final CryptService cryptService;
   private final TelegramService telegramService;
   private final RoleComponent roleComponent;

   // <editor-fold defaultstate="collapsed" desc="***Init and setters***">

   @Autowired
   public UserComponent(UserService userService,
                        CryptService cryptService,
                        TelegramService telegramService, RoleComponent roleComponent) {
      this.userService = userService;
      this.cryptService = cryptService;
      this.telegramService = telegramService;
      this.roleComponent = roleComponent;
   }

   // </editor-fold>

   public void registerNewUser(@NotNull RegisterNewUserWebReqDto paramsDto) {
      checkArgument(paramsDto != null, "Required params is are not specified!");
      checkRegistrationParams(paramsDto);

      userService.insertNewUser(UserDto.builder()
                        .username(paramsDto.getUsername())
                        .password(cryptService.cryptPassword(paramsDto.getPassword()))
                        .email(paramsDto.getEmail())
                        .phoneNumber(paramsDto.getPhone())
                        .firstName(paramsDto.getFirstName())
                        .lastName(paramsDto.getLastName())
                        .registerDate(LocalDate.now())
```

```
                .courseList(Collections.emptyList())
                .lessonsList(Collections.emptyList())
            .build());

    UserDto user = userService.getUserByUsername(paramsDto.getUsername());

    roleComponent.createRoleLink(user.get_id(), RoleEnum.CLIENT_ROLE);

}

public String loginUser(String username, String password) {
    Boolean usernameCheck = userService.usernameExist(username);

    UserDto userData = userService.getUserByUsername(username);

    Boolean passwordCheck = cryptService.checkPassword(password, userData.getPassword());

    if (usernameCheck && passwordCheck) {
        return userData.get_id();
    } else {
        return null;
    }
}

public String forgotPassword(ForgotPasswordParamsDto paramsDto) {
    checkArgument(paramsDto != null, "Required params are not specified!");
    paramsDto.checkValid();

    UserDto user = userService.getUserToForgotPassword(paramsDto);
    checkArgument(user != null, "User with specified params not found!");

    telegramService.sendMessageToChat(user.getTelegramChatId(), buildForgotPasswordMessage());

    return user.get_id();
}

public void setNewUserPassword(String userId, String newPassword) {
    checkArgument(userId != null, "User ID is not specified!");
    checkArgument(isNotBlank(newPassword), "New password is not specified!");

    UserDto user = userService.getUserById(userId);
```

```
    checkArgument(user != null, "User with this ID not found!");

    user.setPassword(cryptService.cryptPassword(newPassword));

    userService.updateUserData(user);
}

public UserDto getUserById(String id) {
    return userService.getUserById(id);
}

//====== Private Elements =======

private String buildForgotPasswordMessage() {
    StringBuilder message = new StringBuilder();
    return message.toString();
}

private void checkRegistrationParams(RegisterNewUserWebReqDto paramsDto) {
    checkUsername(paramsDto.getUsername());
    checkPhone(paramsDto.getPhone());
    checkEmail(paramsDto.getEmail());
}

private void checkUsername(String username) {
    try {
        if (userService.usernameExist(username)) {
            throw new IllegalArgumentException("Username is already used!");
        }
    } catch (Exception exception) {
        log.info("Check username exception: " + exception.getMessage());
    }
}

private void checkEmail(String email) {
    try {
        if (userService.emailExist(email)) {
            throw new IllegalArgumentException("Username is already used!");
        }
    } catch (Exception exception) {
        log.info("Check username exception: " + exception.getMessage());
```

```
        }
    }

    private void checkPhone(String phone) {
        try {
            if (userService.phoneExist(phone)) {
                throw new IllegalArgumentException("Username is already used!");
            }
        } catch (Exception exception) {
            log.info("Check username exception: " + exception.getMessage());
        }
    }
}
```

UserService.java

```
package com.example.onlinecourse.services.user;

import com.example.onlinecourse.services.telegram.TelegramService;
import com.example.onlinecourse.services.user.pojo.ForgotPasswordParamsDto;
import com.example.onlinecourse.services.user.pojo.UserDto;
import com.mongodb.MongoException;
import lombok.extern.slf4j.Slf4j;

import org.springframework.stereotype.Service;

import java.util.Optional;

import static com.example.onlinecourse.utils.ValidationUtils.checkArgument;
import static io.micrometer.common.util.StringUtils.isNotBlank;

@Slf4j
@Service
public class UserService {

    private final UserMngRepository userMngRepository;
    // <editor-fold defaultstate="collapsed" desc="***Init and setters***">

    public UserService(UserMngRepository userMngRepository) {
        this.userMngRepository = userMngRepository;}

    // </editor-folder>
```

```
public UserDto getUserById(String id) {
    Optional<UserDto> user;
    try {
        user = userMngRepository.findById(id);
    } catch (MongoException exception) {
        log.warn(exception.getMessage());
        throw exception;
    } catch (Exception exception) {
        log.warn(exception.getMessage());
        user = Optional.empty();
    }
    return user.get();
}

public UserDto getUserByUsername(String username) {
    UserDto user;
    try {
        user = userMngRepository.getByUsername(username);
    } catch (MongoException exception) {
        log.warn(exception.getMessage());
        throw exception;
    } catch (Exception exception) {
        log.warn(exception.getMessage());
        user = null;
    }
    return user;
}

public void insertNewUser(UserDto userDto) {
    try {
        userMngRepository.insert(userDto);
    } catch (Exception exception) {
        log.warn(exception.getMessage());
        throw exception;
    }
}

public void updateUserData(UserDto userDto) {
    try {
        userMngRepository.save(userDto);
    } catch (Exception exception) {
```

```
            log.warn(exception.getMessage());
            throw exception;
        }
    }

    public Boolean usernameExist(String username) {
        try {
            return userMngRepository.existsByUsername(username);
        } catch (Exception exception) {
            log.warn(exception.getMessage());
            throw exception;
        }
    }

    public Boolean phoneExist(String phone) {
        try {
            return userMngRepository.existsByPhoneNumber(phone);
        } catch (Exception exception) {
            log.warn(exception.getMessage());
            throw exception;
        }
    }

    public Boolean emailExist(String email) {
        try {
            return userMngRepository.existsByEmail(email);
        } catch (Exception exception) {
            log.warn(exception.getMessage());
            throw exception;
        }
    }

    public UserDto getUserToForgotPassword(ForgotPasswordParamsDto paramsDto) {
        checkArgument(paramsDto != null, "Required params aren't specified!");
        paramsDto.checkValid();

        UserDto user;

        if (isNotBlank(paramsDto.getUsername())) {
            user = userMngRepository.getByUsername(paramsDto.getUsername());
        } else if (isNotBlank(paramsDto.getEmail())) {
```

```
            user = userMngRepository.getByEmail(paramsDto.getEmail());
        } else if (isNotBlank(paramsDto.getPhone())) {
            user = userMngRepository.getByPhoneNumber(paramsDto.getPhone());
        } else user = null;

        checkArgument(user != null, "User with specified params didn't found");

        return user;
    }

    //======== Private Elements ===========
}
```

UserDto.java

```
package com.example.onlinecourse.services.user.pojo;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

import java.io.Serial;
import java.io.Serializable;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.List;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Document(collection = "User")
public class UserDto implements Serializable {

    @Serial
    private static final long serialVersionUID = 1L;

    @Id
    private String _id;
```

```
        private String username;
        private String password;
        private String email;
        private String phoneNumber;
        private String telegramChatId;
        private String firstName;
        private String lastName;
        private LocalDate registerDate;
        private LocalDateTime lastLoginTime;
        private List<UserCourseInfo> courseList;
        private List<String> lessonsList;

        @Data
        @Builder
        @AllArgsConstructor
        @NoArgsConstructor
        public static class UserCourseInfo implements Serializable {

            @Serial
            private static final long serialVersionUID = 1L;

            private String courseId;
            private Integer remainingLessons;
            private Integer remainingFreezes;
        }
    }
```

SecurityConfig.java

```
package ua.testerossa.config;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
```

```
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;

@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {
 private final UserDetailsService userDetailsService;
 public static final String LOGIN_ENDPOINT = "/auth/login";
 public static final String LOGOUT_ENDPOINT = "/auth/logout";
 public static final String SUCCESS_ENDPOINT = "/auth/success";
 public static final String FAILURE_ENDPOINT = "/auth/error";
 @Autowired
 public SecurityConfig(@Qualifier("userDetailsServiceImpl") UserDetailsService userDetailsService) {
  this.userDetailsService = userDetailsService;
 }
 @Override
 protected void configure(HttpSecurity http) throws Exception {
  http
      .csrf().disable()
      .authorizeRequests()
      .antMatchers("/auth/*").permitAll()
      .anyRequest()
      .authenticated()
      .and()
      .formLogin()
      .loginPage(LOGIN_ENDPOINT).permitAll()
      .failureUrl(FAILURE_ENDPOINT).permitAll()
      .defaultSuccessUrl(SUCCESS_ENDPOINT)
      .and()
      .logout()
      .logoutRequestMatcher(new AntPathRequestMatcher(LOGOUT_ENDPOINT, "POST"))
      .invalidateHttpSession(true)
      .clearAuthentication(true)
      .deleteCookies("JSESSIONID")
      .logoutSuccessUrl(LOGIN_ENDPOINT);
 }
 @Override
 protected void configure(AuthenticationManagerBuilder auth) throws Exception {
  auth.authenticationProvider(daoAuthenticationProvider());
 }
```

```
  @Bean
  protected PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder(12);
  }
  @Bean
  protected DaoAuthenticationProvider daoAuthenticationProvider() {
    DaoAuthenticationProvider daoAuthenticationProvider = new DaoAuthenticationProvider();
    daoAuthenticationProvider.setPasswordEncoder(passwordEncoder());
    daoAuthenticationProvider.setUserDetailsService(userDetailsService);
    return daoAuthenticationProvider;
  }
  @Override
  public void configure(WebSecurity web) throws Exception {
    web.ignoring().antMatchers("/resources/**", "/static/**");
  }
}
```

UserDetails.java

```
package ua.testerossa.security;

import lombok.Data;
import lombok.extern.slf4j.Slf4j;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import ua.testerossa.model.Status;
import ua.testerossa.model.dto.User;
import java.util.Collection;
import java.util.List;

@Data
@Slf4j
public class SecurityUser implements UserDetails {
    private final String username;
    private final String password;
    private final List<SimpleGrantedAuthority> authorities;
    private final boolean isActive;
```

```
    public SecurityUser(String username, String password, List<SimpleGrantedAuthority> authorities,
boolean isActive) {
        this.username = username;
        this.password = password;
        this.authorities = authorities;
        this.isActive = isActive;
    }
    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return authorities;
    }
    @Override
    public String getPassword() {
        return password;
    }
    @Override
    public String getUsername() {
        return username;
    }
    @Override
    public boolean isAccountNonExpired() {
        return isActive;
    }
    @Override
    public boolean isAccountNonLocked() {
        return isActive;
    }
    @Override
    public boolean isCredentialsNonExpired() {
        return isActive;
    }
    @Override
    public boolean isEnabled() {
        return isActive;
    }
    public static UserDetails fromUser(User user) {
        log.info("fromUser user:{}, authorities:{}", user.toString(), user.getRole().getAuthorities());
```

```
        return new org.springframework.security.core.userdetails.User(
                user.getEmail(), user.getPassword(),
                user.getStatus().equals(Status.ACTIVE),
                user.getStatus().equals(Status.ACTIVE),
                user.getStatus().equals(Status.ACTIVE),
                user.getStatus().equals(Status.ACTIVE),
                user.getRole().getAuthorities()
        );
    }
}
```

SecurityUtils.java

```
package ua.testerossa.utils;
import com.sun.mail.util.BASE64EncoderStream;
import lombok.extern.slf4j.Slf4j;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.spec.KeySpec;
import java.util.Arrays;
import java.util.Base64;

@Slf4j
public class SecurityUtils {
  public static final String SECRET_KEY_64 = "KaPdSgVk";
  public static final String SECRET_KEY_128 = "cRfUjXnZr4u7x!A%";
  public static final String SECRET_KEY_256 = "q4t7w!z%C*F-JaNcRfUjXn2r5u8x/A?D";
  public static final String SECRET_KEY_512 = "WmZq4t7w!z%C*F-
JaNcRfUjXn2r5u8x/A?D(G+KbPeSgVkYp3s6v9y$B&E)H@McQ";
  private static final String SALT = "ssshhhhhhhhhhh!!!!";
```

```
public static String des(String strToEncrypt) {
  try {
    SecretKey key = KeyGenerator.getInstance("DES").generateKey();
    Cipher ecipher = Cipher.getInstance("DES");
    ecipher.init(Cipher.ENCRYPT_MODE, key);
    byte[] utf8 = strToEncrypt.getBytes(StandardCharsets.UTF_8);
    byte[] enc = ecipher.doFinal(utf8);
    enc = BASE64EncoderStream.encode(enc);
    return new String(enc);
  } catch (Exception e) {
    System.out.println("Error occured during des encryption: " + e.toString());
  }
  return null;
}
public static String tripleDes(String strToEncrypt) {
  try {
    final MessageDigest md = MessageDigest.getInstance("md5");
    final byte[] digestOfPassword = md.digest(SECRET_KEY_128.getBytes(StandardCharsets.UTF_8));
    final byte[] keyBytes = Arrays.copyOf(digestOfPassword, 24);
    for (int j = 0, k = 16; j < 8; ) {
      keyBytes[k++] = keyBytes[j++];
    }
    final SecretKey key = new SecretKeySpec(keyBytes, "DESede");
    final Cipher cipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
    cipher.init(Cipher.ENCRYPT_MODE, key);
    byte[] plainTextBytes = strToEncrypt.getBytes(StandardCharsets.UTF_8);
    return Base64.getEncoder()
        .encodeToString(cipher.doFinal(plainTextBytes));
  } catch (Exception e) {
    System.out.println("Error occured during des encryption: " + e.toString());
  }
  return null;
}
public static String blowfish(String strToEncrypt) {
  try {
    byte[] KeyData = SECRET_KEY_128.getBytes(StandardCharsets.UTF_8);
    SecretKeySpec KS = new SecretKeySpec(KeyData, "Blowfish");
```

```
    Cipher cipher = Cipher.getInstance("Blowfish");
    cipher.init(Cipher.ENCRYPT_MODE, KS);
    return Base64.getEncoder().
        encodeToString(cipher.doFinal(strToEncrypt.getBytes(StandardCharsets.UTF_8)));
  } catch (Exception e) {
    System.out.println("Error occured during blowfish encryption: " + e.toString());
  }
  return null;
}
public static String md5(String strToEncrypt) {
  try {
    StringBuilder code = new StringBuilder();
    MessageDigest messageDigest = MessageDigest.getInstance("MD5");
    byte[] bytes = strToEncrypt.getBytes(StandardCharsets.UTF_8);
    byte[] digest = messageDigest.digest(bytes);
    for (byte aDigest : digest) {
      code.append(Integer.toHexString(0x0100 + (aDigest & 0x00FF)).substring(1));
    }
    return code.toString();
  } catch (Exception e) {
    System.out.println("Error occured during md5 encryption: " + e.toString());
  }
  return null;
}
public static String sha256(String strToEncrypt) {
  try {
    MessageDigest digest = MessageDigest.getInstance("SHA-256");
    byte[] hash = digest.digest(strToEncrypt.getBytes(StandardCharsets.UTF_8));
    return Base64.getEncoder().
        encodeToString(hash);
  } catch (Exception e) {
    System.out.println("Error occured during sha256 encryption: " + e.toString());
  }
  return null;
}
public static String sha512(String passwordToHash) {
  try {
```

```
    MessageDigest md = MessageDigest.getInstance("SHA-512");
    byte[] bytes = md.digest(passwordToHash.getBytes(StandardCharsets.UTF_8));
    StringBuilder sb = new StringBuilder();
    for (byte aByte : bytes) {
      sb.append(Integer.toString((aByte & 0xff) + 0x100, 16).substring(1));
    }
    return sb.toString();
  } catch (Exception e) {
    System.out.println("Error occured during sha512 encryption: " + e.toString());
  }
  return null;
}
public static String aes(String strToEncrypt, String key) {
  log.info("aes: strToEncrypt={}", strToEncrypt);
  try {
    byte[] iv = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
    IvParameterSpec ivspec = new IvParameterSpec(iv);
    SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
    KeySpec spec = new PBEKeySpec(key.toCharArray(), SALT.getBytes(), 65536, 256);
    SecretKey tmp = factory.generateSecret(spec);
    SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivspec);
    return Base64.getEncoder()
        .encodeToString(cipher.doFinal(strToEncrypt.getBytes(StandardCharsets.UTF_8)));
  } catch (Exception e) {
    System.out.println("Error while encrypting: " + e.toString());
  }
  return null;
}
public static String decrypt(String strToDecrypt, String key) {
  log.info("decrypt: strToEncrypt={}", strToDecrypt);
  try {
    byte[] iv = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
    IvParameterSpec ivspec = new IvParameterSpec(iv);
    SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
    KeySpec spec = new PBEKeySpec(key.toCharArray(), SALT.getBytes(), 65536, 256);
```

```
    SecretKey tmp = factory.generateSecret(spec);
    SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
    cipher.init(Cipher.DECRYPT_MODE, secretKey, ivspec);
    return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
   } catch (Exception e) {
    System.out.println("Error while decrypting: " + e.toString());
   }
  return null;
}
        }
```

ДОДАТОК Б

ПЕРЕЛІК ФАЙЛІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_Швець_В.І.doc	Пояснювальна записка до проекту. Документ Word.
Диплом_Швець_В.І.pdf	Пояснювальна записка до проекту в форматі PDF.
Програма	
Program.rar	Архів, що містить коди програми для запуску проекту.
Презентація	
Презентація_Швець.ppt	Презентація для проекту.