БАКАЛАВРСЬКА РОБОТА

БР. ІП - 07.00.00.000 ПЗ

Група ІП-21-1

Дубей Едуард

2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Дубей Едуард Володимирович

(прізвище, ім'я, по батькові)

УДК 004

(індекс)

БАКАЛАВРСЬКА РОБОТА

Розробка методології автоматизації процесу генерації веб-аплікацій та сторінок

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач освітнього рівня Дубей Е.В.

(підпис, ініціали та прізвище здобувача)

Науковий керівник Михайлюк Ірина Романівна, к.п.н., доцент

(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ ____ ” _____2025 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Дубею Едуарду Володимировичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) “ Розробка методології автоматизації процесу генерації веб-аплікацій та сторінок”

керівник проекту (роботи) Михайлюк І.Р., доцент

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 р. № 264/7

2. Строк подання студентом проекту (роботи) 10 червня 2025 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної області процесу генерації веб- сторінок та аплікацій
2. Алгоритмічна реалізація системи генерації веб-аплікацій
3. Проектування структури бази даних
4. Розробка ER діаграм проекту
5. Програмна реалізація системи автоматизації процесу генерації веб-аплікацій та сторінок

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Платформа Eclipse (рис. 1.1)
2. Офіційна сторінка Apache Tomcat (рис. 1.2)
3. MySQL workbench (рис. 1.3)
4. Спіральна модель розробки програмного забезпечення (рис. 1.4)
5. Архітектурний шаблон Model-View-Controller (MVC) (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **28 квітня 2025 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Аналіз предметної області процесу генерації веб-сторінок та аплікацій	03.05.2025	виконано
2	Алгоритмічна реалізація системи генерації веб-аплікацій	15.05.2025	виконано
3	Проектування структури бази даних	26.05.2025	виконано
4	Розробка ER діаграм проекту	01.06.2025	виконано
5	Програмна реалізація системи автоматизації процесу генерації веб-аплікацій та сторінок	06.06.2025	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	10.06.2025	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Бакалаврська робота містить 75 сторінок, 85 рисунків, список використаних джерел із 39 найменуваннями, 1 додаток.

Мета роботи - розробка системи автоматизованої генерації веб-аплікацій і веб-сторінок із використанням сучасних фреймворків, бібліотек та інструментів управління життєвим циклом програмного забезпечення.

Об'єкт дослідження - процес створення веб-аплікацій і веб-сторінок у рамках сучасних архітектурних підходів.

Предмет дослідження - методи та засоби автоматизації генерації веб-аплікацій, зокрема програмні фреймворки, архітектурні моделі, технології клієнт-серверної взаємодії.

В першому розділі проаналізовано потребу в автоматизованій генерації веб-додатків, обґрунтовано актуальність, вибір інструментів і архітектурного підходу для подальшої реалізації.

В другому розділі запропоновано оптимальні фреймворки та протоколи, спроєктовано структуру бази даних, що забезпечує узгоджену взаємодію між компонентами системи.

В третьому розділі реалізовано конфігурацію, запуск і розгортання системи автоматизації генерації веб-додатків із використанням BPMN і web-сервісів.

Висновок: реалізовано конфігурацію системи, модель клієнта, налаштування веб-сервера Tomcat, створення та розгортання проекту з використанням потоків процесів BPMN. Результатом роботи є діюча система, що забезпечує автоматизацію процесів створення веб-аплікацій та оптимізацію життєвого циклу їх розробки

КЛЮЧОВІ СЛОВА: АВТОМАТИЗАЦІЯ РОЗРОБКИ, ГЕНЕРАЦІЯ ВЕБ-АПЛІКАЦІЙ, SPRING FRAMEWORK, HIBERNATE, MVC, ВЕБ-СЕРВІС, TOMCAT, БАЗА ДАНИХ, ER-ДІАГРАМА.

ANNOTATION

The bachelor's thesis contains 75 pages, 85 figures, a list of used sources with 39 names, 1 appendix.

The purpose of the work is to develop a system for automated generation of web applications and web pages using modern frameworks, libraries and software life cycle management tools.

The object of the study is the process of creating web applications and web pages within the framework of modern architectural approaches.

The subject of the study is methods and tools for automating the generation of web applications, in particular software frameworks, architectural models, client-server interaction technologies.

The first section analyzes the need for automated generation of web applications, justifies the relevance, choice of tools and architectural approach for further implementation.

The second section proposes optimal frameworks and protocols, and designs a database structure that ensures coordinated interaction between system components.

In the third section, the configuration, launch and deployment of the system for automating the generation of web applications using BPMN and web services are implemented.

Conclusion: the system configuration, client model, Tomcat web server settings, project creation and deployment using BPMN process flows are implemented. The result of the work is a working system that provides automation of web application creation processes and optimization of their development life cycle

KEYWORDS: DEVELOPMENT AUTOMATION, WEB APPLICATION GENERATION, SPRING FRAMEWORK, HIBERNATE, MVC, WEB SERVICE, TOMCAT, DATABASE, ER-DIAGRAM

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

BPMN - Business Process Model Notation

ASF - Apache Software Foundation

JSP - Java Server Pages

XML - Extensible Markup Language

MDE – Model-Driven Engineering

MDWE – Model-Driven Web Engineering

MVC – Model-View-Controller

SPA – Single Page Application

REST – Representational State Transfer

PWA – Progressive Web Application

DSL – Domain-Specific Language

SSR – Server-Side Rendering

ВСТУП

Однією з основних складнощів у дизайні веб-додатків є трудомісткість створення нових веб-сторінок з нуля. Для традиційних проектів веб-додатків клієнти зазвичай потребують знайти компанії, які можуть спроектувати та реалізувати веб-додатки клієнтів, слідуючи методологіям розробки програмного забезпечення для проектування веб-додатків клієнтів за останні кілька років. Нещодавно з'явилося багато онлайн-генераторів веб-додатків. Weebly та WordPress (інструмент для створення веб-додатків онлайн) стали найбільш використовуваними онлайн-генераторами веб-проектів, але їхні клієнти все ще повинні змінювати свої проекти, такі як перекомпонування сторінок та додавання додаткових елементів до своїх веб-додатків. Метою проекту є зробити його більш зручним для клієнтів, які не знайомі з загальною технологією веб-додатків. Проект використовуватиме методологію, орієнтовану на потік, для керування проектом за допомогою нотації бізнес-процесів (BPMN). Проект також включатиме теорії інженерії програмного забезпечення, оскільки деякі способи генерації веб-додатків призначені для того, щоб клієнти надсилали свої спеціальні вимоги на налаштовані проекти замовлень. Проект використовуватиме відповідні методології розробки програмного забезпечення для відповідності вимогам веб-додатків клієнтів. Після циклу життя розробки програмного забезпечення клієнти отримають свій продукт, а потім вони повинні додати його до своїх проектів.

Стрімкий розвиток інформаційних технологій зумовлює підвищення вимог до швидкості, якості та гнучкості створення веб-аплікацій. У відповідь на ці виклики особливої актуальності набувають системи автоматизації процесів генерації веб-сторінок і застосунків. Автоматизація дозволяє скоротити час розробки, зменшити кількість помилок, забезпечити простоту

масштабування та підтримки застосунків протягом усього їхнього життєвого циклу.

Актуальність роботи

Актуальність роботи обумовлена необхідністю прискорення розробки веб-аплікацій у сучасних умовах цифрової трансформації. Традиційні методи створення застосунків є трудомісткими та ресурсозатратними, що створює потребу у використанні автоматизованих систем для оптимізації процесів розробки. Запропоноване рішення базується на використанні перевірених архітектурних підходів (MVC), популярних фреймворків (Spring, Hibernate), протоколів взаємодії (SOAP, REST) і сучасних засобів управління процесами (BPMN), що забезпечує його відповідність вимогам ринку і можливість широкого застосування у практиці розробки веб-систем.

У межах даної роботи було проведено аналіз предметної області, здійснено обґрунтований вибір інструментів і технологій, розроблено архітектуру та реалізовано працездатну систему автоматизованої генерації веб-аплікацій. Робота базується на сучасних фреймворках і технологіях, що дозволяє інтегрувати рішення у реальні проєкти та адаптувати його до різноманітних вимог користувачів.

Мета роботи - розробка системи автоматизованої генерації веб-аплікацій і веб-сторінок із використанням сучасних фреймворків, бібліотек та інструментів управління життєвим циклом програмного забезпечення.

Завдання дослідження:

- Провести аналіз предметної області генерації веб-аплікацій.
- Обґрунтувати вибір інструментів, моделей та архітектурних підходів.
- Реалізувати алгоритм генерації веб-аплікацій.
- Спроєктувати базу даних для підтримки роботи системи.
- Розробити програмне рішення та протестувати його функціональність.

Об’єкт дослідження - процес створення веб-аплікацій і веб-сторінок у рамках сучасних архітектурних підходів.

Предмет дослідження - методи та засоби автоматизації генерації веб-аплікацій, зокрема програмні фреймворки, архітектурні моделі, технології клієнт-серверної взаємодії.

Методи дослідження:

- Аналіз та порівняння існуючих рішень.

- Моделювання архітектури програмних систем.

- Проєктування баз даних.

- Програмна реалізація з використанням об’єктно-орієнтованих підходів.

- Тестування та емпірична перевірка працездатності системи.

Наукова новизна

Удосконалено підхід до автоматизації створення веб-аплікацій шляхом інтеграції BPMN-моделювання з технологіями Spring, Hibernate, REST/SOAP та MVC, що забезпечує скорочення часу на розробку і підвищення масштабованості програмних рішень.

Практичне застосування

Результати можуть бути використані для створення внутрішніх корпоративних систем, CRM-систем, адміністративних панелей або сервісів, що швидко масштабуються та легко розгортаються в хмарному середовищі.

Бакалаврська робота містить 75 сторінок, 85 рисунків, 3 розділи список використаних джерел із 39 найменуваннями, 1 додаток.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПРОЦЕСУ ГЕНЕРАЦІЇ ВЕБ- СТОРІНОК ТА АПЛІКАЦІЙ

1.1. Особливості та актуальність розробки система автоматизованої генерації та управління життєвим циклом веб-аплікацій

Одним із суттєвих викликів у розробці програмного забезпечення для веб-платформ є висока трудомісткість та ресурсна інтенсивність процесу створення нових веб-додатків, особливо при необхідності реалізації унікальних функціональних вимог та користувацьких інтерфейсів. Традиційні методології розробки часто передбачають послідовне, деталізоване проєктування та імплементацію компонентів "з нуля" для кожного окремого проєкту. З метою оптимізації та прискорення цих процесів запропоновано та розроблено програмну систему автоматичної генерації веб-додатків.

Основне призначення системи полягає у забезпеченні можливості генерації універсальних та кастомізованих веб-додатків на основі принципів та теорій інженерії програмного забезпечення. Система використовує підходи, що дозволяють автоматизувати значну частину рутинних завдань, спираючись на стандартизовані патерни проєктування, архітектурні стилі та техніки кодогенерації.

Управління проєктами та оркестрація процесу генерації в системі здійснюється за допомогою методології, орієнтованої на потоки робіт (workflow-oriented methodology). Центральним елементом цієї методології є використання нотації моделювання бізнес-процесів (BPMN). BPMN використовується не лише для візуального моделювання процесів розробки та генерації, але й слугує основою для рушія виконання потоків (BPMN flow engine), який автоматично керує послідовністю завдань, транзакціями даних та станами у процесі генерації веб-додатків. Залежність функціонування

системи від BPMN-рушія визначає його як ключовий компонент архітектури. BPMN-рушій (Business Process Model and Notation engine) – це програмний компонент, який слугує для інтерпретації та автоматизованого виконання бізнес-процесів, змодельованих за допомогою стандарту BPMN. Основна функція BPMN-рушія полягає у взятті змодельованого бізнес-процесу (зазвичай представленого у вигляді XML-файлу відповідно до специфікації BPMN 2.0), аналізі його логіки та подальшому управлінні потоком виконання.

Система автоматичної генерації має модульну структуру, що забезпечує гнучкість та масштабованість. Основними функціональними модулями є:

- Модуль бази даних, що відповідає за персистентне зберігання даних, включаючи репозиторій шаблонів веб-додатків, конфігурацій проєктів, метаданих згенерованих артефактів, а також визначень бізнес-процесів у форматі BPMN.

- Модуль веб-сервера - надає середовище для розгортання згенерованих веб-додатків, обслуговування HTTP-запитів та функціонування програмних інтерфейсів (API) для взаємодії з іншими модулями системи.

- Модуль генерації HTML-сторінок - відповідає за створення представницького шару веб-додатків на основі шаблонів та конфігурацій, генеруючи відповідний HTML, CSS та клієнтський JavaScript-код.

- Модуль реалізації функціональності призначений для генерації або інтеграції компонентів бізнес-логіки та серверного коду, що забезпечує функціональні вимоги веб-додатків.

- Модуль фінансово-аналітичної моделі - спеціалізований компонент, що може генерувати або інтегрувати функціонал, пов'язаний з фінансовим аналізом або моделюванням, як частину цільового веб-додатку. Це може бути реалізовано як специфічний тип функціонального модуля або набору конфігурацій.

Модуль клієнтського інтерфейсу системи - надає користувачеві (замовнику або розробнику) інтерфейс для взаємодії з Системою Генерації, включаючи вибір шаблонів, налаштування параметрів генерації та моніторинг процесу.

- BPMN-рушій. Ядро системи, що інтерпретує та виконує змодельовані у BPMN процеси, керуючи послідовністю дій всіх інших модулів під час генерації веб-додатку.

Система підтримує два основні режими взаємодії з користувачем/замовником, що визначають процес генерації:

1. Стандартний режим (на основі шаблонів). Користувач ініціює процес через клієнтський інтерфейс Системи. Взаємодія передбачає вибір одного з доступних шаблонів веб-додатків у репозиторії та активацію функції генерації. BPMN-рушій виконує стандартизований потік робіт, визначений для обраного шаблону, автоматично генеруючи веб-додаток з мінімальною необхідністю додаткової конфігурації.

2. Режим спеціалізованого замовлення. Цей режим активується для замовлень з унікальними або нестандартними вимогами від клієнтів. Процес може включати більш глибокий етап збору та формалізації вимог, можливо, створення нового або модифікацію існуючого BPMN-процесу генерації та/або розробку нових компонентів, що інтегруються в процес генерації під управлінням BPMN-рушія.

Після завершення циклу виконання процесу генерації (що відображає життєвий цикл розробки програмного забезпечення в рамках автоматизованого процесу) клієнти отримують фінальний програмний продукт. Цей продукт, як правило, включає розгорнутий веб-додаток, що функціонує у середовищі веб-сервера, та може додатково містити згенерований вихідний код, конфігураційні файли та, потенційно, документацію, що описує структуру та функціонування згенерованого

рішення. Використання BPMN забезпечує прозорість процесу генерації та відповідність результату попередньо змодельованим потокам робіт.

1.2. Вибір інструментів та середовища розробки

Є чотири основні інструменти розробки, які було використано для проектування та розробки мого проекту та веб-сервісу. По-перше, Eclipse використовується для розробки проекту та веб-сервісу. По-друге, Apache Tomcat використовується як веб-хост середовище для проекту та веб-сервісу. По-третє, MySQL використовується як сховище даних, де дані будуть збережені. І останній — це редактор UML, який використовується для моделювання зв'язку між базою даних та компонентами проекту.

Eclipse, який є інтегрованим середовищем розробки (IDE) і найбільш широко використовуваним Java IDE, пропонує базове робоче середовище та розширюваний плагін компонент для налаштування середовища розробки [19].

Eclipse – це потужне, розширюване та безкоштовне інтегроване середовище розробки (IDE), яке здобуло широку популярність серед розробників програмного забезпечення. Хоча спочатку Eclipse був орієнтований переважно на розробку на мові Java, його модульна архітектура та система плагінів дозволяють використовувати його для роботи з багатьма іншими мовами програмування та для різноманітних завдань розробки.

Завдяки своїй розширюваності та відкритості, Eclipse став популярним вибором для розробників, що працюють з Java, а також з C++, PHP, Python, Android та багатьма іншими технологіями, для яких існують відповідні плагіни.

Через Windows-базований комп'ютер сторонні плагіни можуть бути легко вбудовані в середовище розробки Eclipse та допомагати розробникам Java веб-додатків розробляти проекти, такі як SVN. Eclipse також пропонує

потужну підтримку для розробки та тестування проекту та веб-сервісів, включаючи можливості налагоджування, перегляду змінних та поетапного виконання коду.

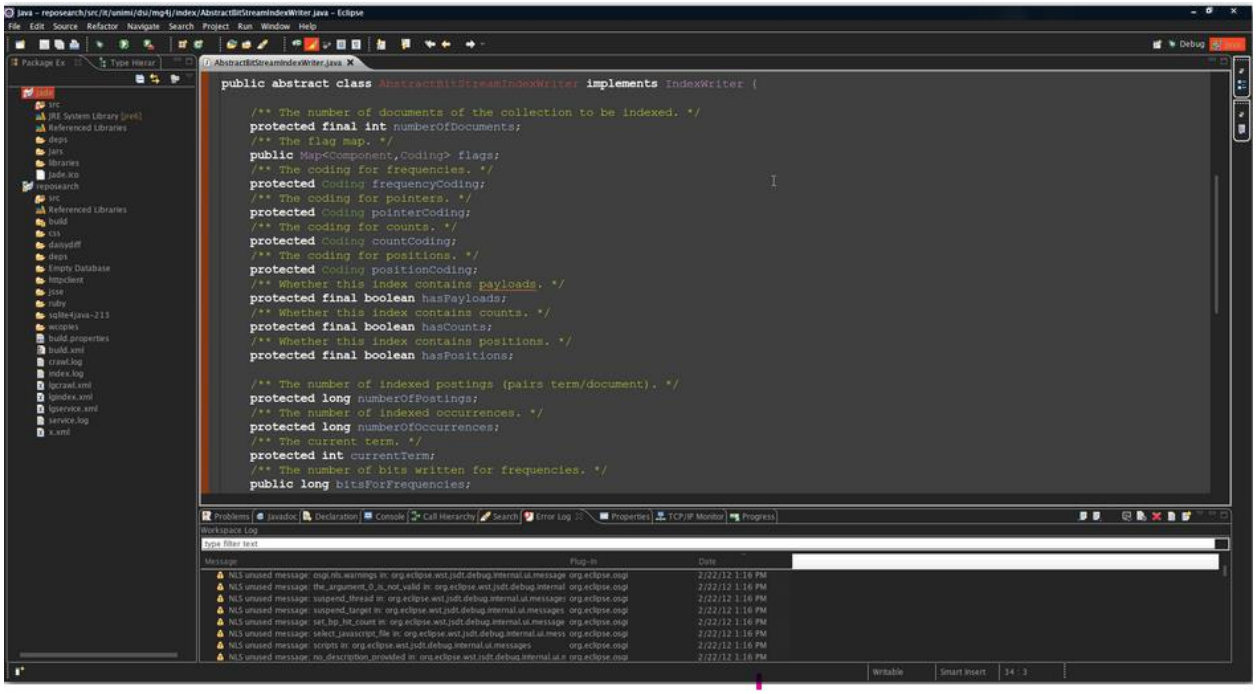

Рисунок 1.1 – Платформа Eclipse

Apache Tomcat — це відкритий вихідний код і широко використовуваний Java Servlet Container, розроблений Apache Software Foundation (ASF). Tomcat реалізує кілька специфікацій Java EE, включаючи Java Servlet та Java Server Pages (JSP). На платформі Eclipse веб-сервіс tomcat можна керувати за допомогою IDE Eclipse [20].

Apache Tomcat – це популярний сервер застосунків та контейнер сервлетів з відкритим вихідним кодом, розроблений Apache Software Foundation. Він є однією з ключових технологій для виконання вебзастосунків, написаних на мові програмування Java.

Tomcat в першу чергу функціонує як контейнер сервлетів. Це означає, що він надає середовище, в якому можуть виконуватися Java-сервлети та JSP-сторінки. Сервлети – це невеликі Java-програми, які розширюють

можливості вебсерверів, обробляючи запити клієнтів та генеруючи динамічний контент. JSP (JavaServer Pages) – це технологія, що дозволяє вбудовувати Java-код у HTML-сторінки для створення динамічного вебвмісту.

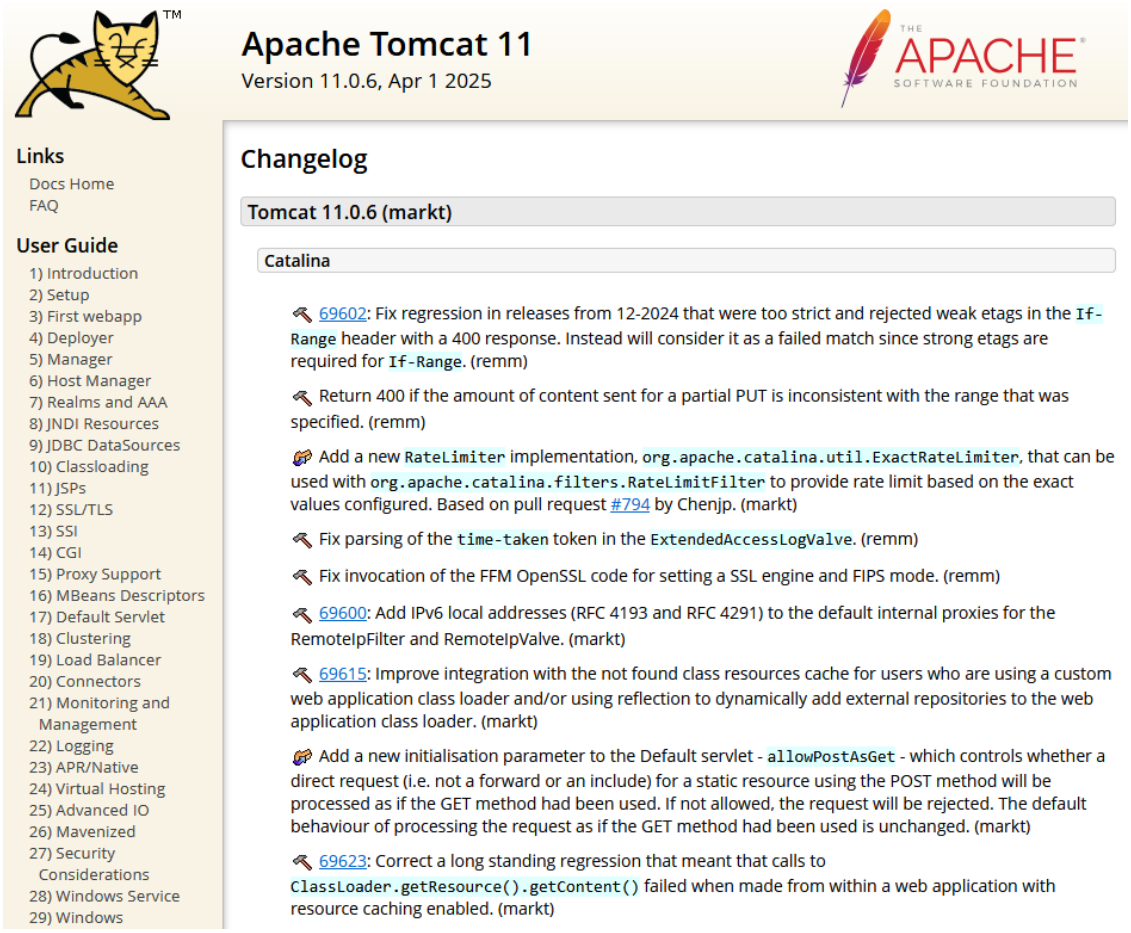

Рисунок 1.2 – Офіційна сторінка Apache Tomcat

Крім того, Tomcat може виступати як вебсервер, здатний обслуговувати статичні вебсторінки. Однак його основна сила полягає саме в обробці динамічного контенту, створеного за допомогою Java-технологій.

MySQL — це відкрита реляційна система управління базами даних (RDBMS). Eclipse пропонує багато інтерфейсів баз даних, включаючи MySQL, і його легко налаштувати на Windows-базованому середовищі для створення бази даних проекту за допомогою інструменту Eclipse database tool [21].

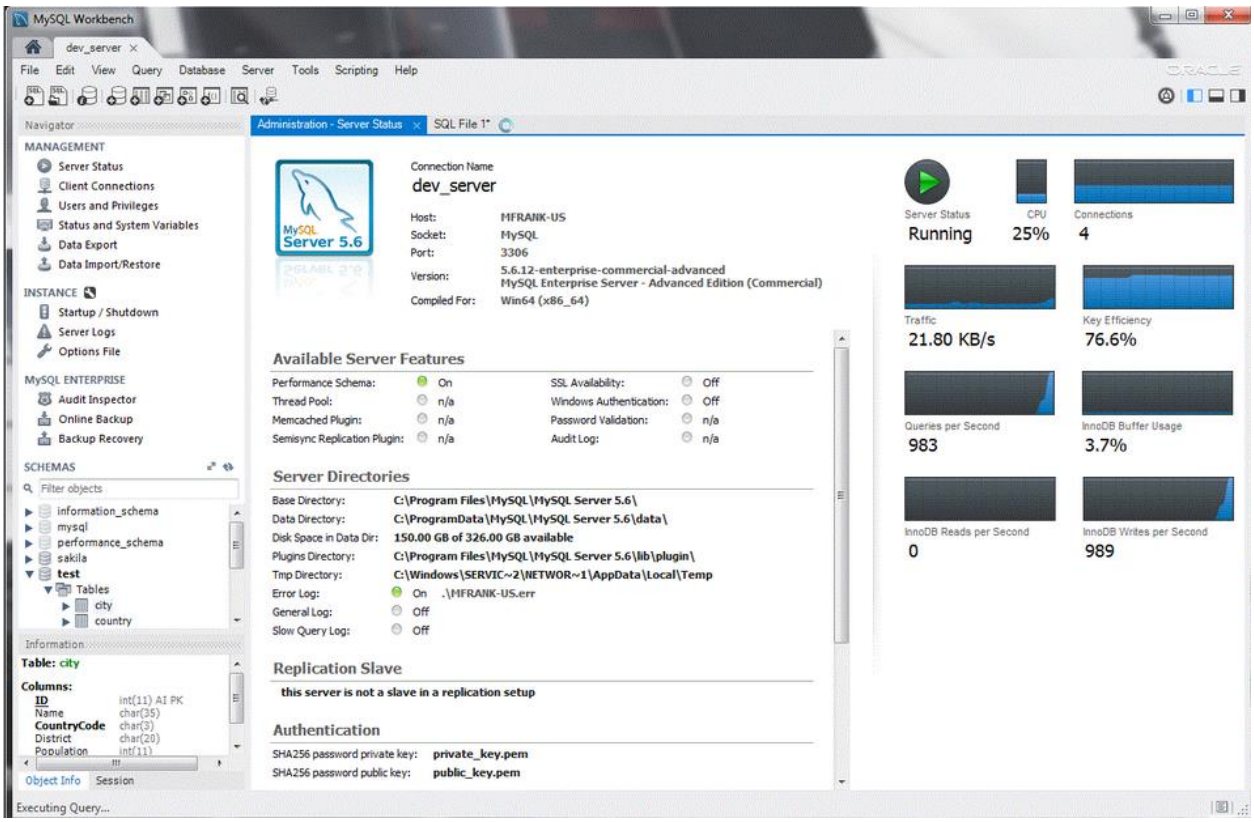

Рисунок 1.3 - MySQL workbench

MySQL Workbench – це інтегроване візуальне середовище, розроблене компанією Oracle, яке надає архітекторам баз даних, розробникам та адміністраторам БД повний набір інструментів для проектування, розробки та адміністрування баз даних MySQL. Він об'єднує функціональність трьох основних напрямків: дизайн, розробка SQL та адміністрування.

MySQL Workbench дозволяє створювати складні ER-діаграми (діаграми "сутність-зв'язок") для візуалізації структури бази даних. Підтримується пряме і зворотне проектування, що дає можливість генерувати SQL-скрипти для створення бази даних на основі моделі, або ж створювати модель на основі існуючої бази даних чи SQL-скриптів. Інструменти редагування таблиць, колонок, зв'язків, індексів та інших об'єктів роблять процес моделювання інтуїтивно зрозумілим.

Вбудований SQL-редактор надає потужні можливості для написання, виконання та оптимізації SQL-запитів. Серед ключових функцій:

підсвічування синтаксису, автодоповнення коду, шаблони запитів, історія виконання, а також візуальний інструмент EXPLAIN для аналізу продуктивності запитів. Редактор дозволяє працювати з кількома вкладками запитів та переглядати результати у зручному форматі.

У галузі інженерії програмного забезпечення UML (Unified Modeling Language) — це загальноприйнята мова моделювання, призначена для надання стандартного способу візуалізації проектування проекту [22].

Unified Modeling Language (UML), або уніфікована мова моделювання, – це стандартизована візуальна мова, яка використовується для специфікації, візуалізації, конструювання та документування артефактів програмних та інших систем. Вона відіграє ключову роль в об'єктно-орієнтованому аналізі та проектуванні, надаючи розробникам та іншим зацікавленим сторонам спільний засіб для розуміння та обговорення складних систем.

UML не є методологією розробки програмного забезпечення, але він може використовуватися в рамках різних методологій (наприклад,тративних та інкрементальних). Хоча UML найбільш широко застосовується у програмній інженерії, його також можна використовувати для моделювання бізнес-процесів та інших систем.

1.3. Вибір моделі реалізації

Процес реалізації системи автоматичної генерації вебзастосунків та системи управління ними здійснюється згідно зі спіральною моделлю розробки програмного забезпечення [7]. Ця модель обрана як структурована методологія, що передбачає ітеративний підхід з акцентом на аналізі ризиків та послідовному нарощуванні функціональності системи на кожному витку спіралі.

На поточному етапі, за результатами завершення першого циклу розробки, реалізовано базовий функціональний обсяг, що становить близько

40% від загальної запланованої функціональності системи. Ключовим елементом, реалізованим у рамках цієї ітерації, є інтеграція та початкове використання технології BPMN (Business Process Model and Notation) [3].

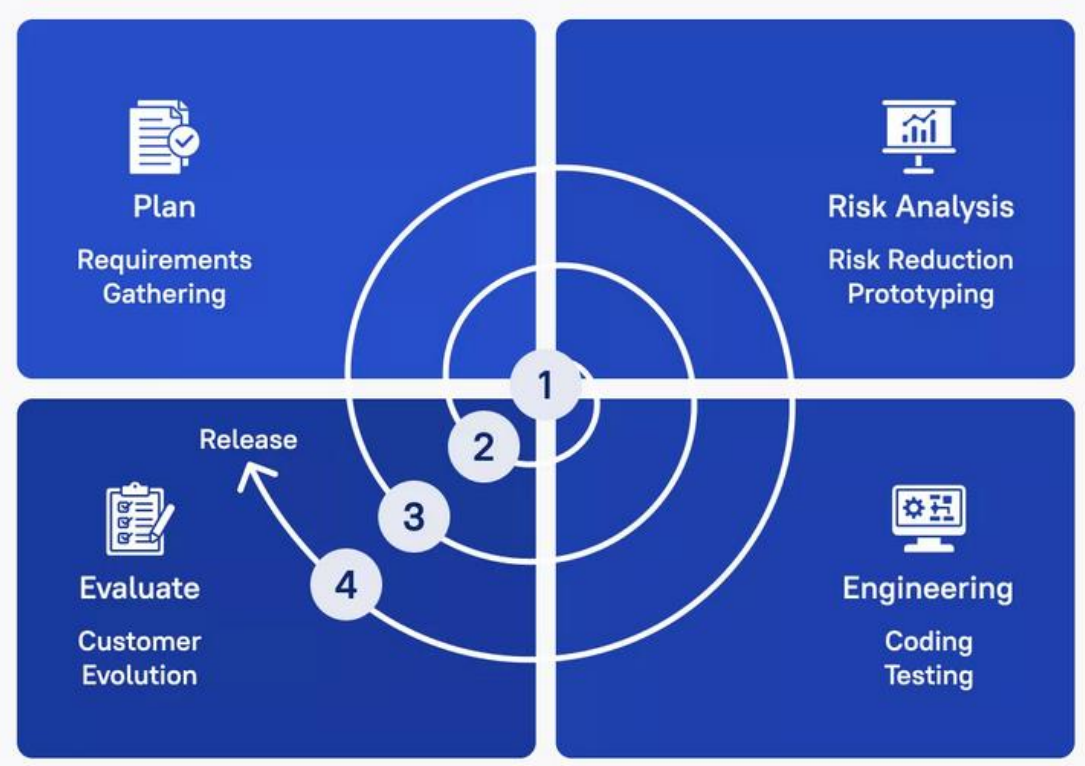

Рисунок 1.4 – Спіральна модель розробки програмного забезпечення

BPMN застосовується як систематичний підхід до моделювання, виконання, документування, вимірювання, моніторингу та контролю бізнес-процесів, що охоплює як автоматизовані, так і неавтоматизовані складові, з метою ефективного досягнення цілей та реалізації стратегій компанії [3]. У контексті даної системи, BPMN виконує функцію керування потоками вебсторінок (web page flows). Моделі потоків, створені за допомогою відповідних інструментів моделювання BPMN, інтерпретуються рушієм BPMN. На основі цих моделей рушій визначає послідовність відображення вебсторінок та необхідних даних у відповідь на запити користувачів, тим самим керуючи навігацією та взаємодією користувача з вебзастосунком. Реалізовано підтримку для типових логічних потоків, таких як процеси

онлайн-купівлі, бронювання готелів та резервування столиків у ресторанах, управління якими здійснюється за допомогою спеціалізованого контролера потоку. Система передбачає можливість подальшої модифікації визначених потоків клієнтами через спеціалізовану платформу управління клієнтами, що забезпечує гнучкість у налаштуванні логіки вебзастосунків без безпосереднього втручання у програмний код.

Для забезпечення гнучкості та зручності користувацького інтерфейсу (UI), архітектура системи передбачає декомпозицію вебсторінок та їхньої функціональності на дискретні фрагменти коду. Ці фрагменти зберігаються у централізованій базі даних та динамічно збираються під час виконання для формування повноцінних вебсторінок. На рівні представлення активно використовуються вебтехнології, зокрема HTML [6] для структурування контенту та jQuery [9] для реалізації клієнтської логіки, інтерактивності та підвищення зручності користування.

Окрім базової функціональності генерації та управління потоками, система включає ряд спеціалізованих компонентів та контролерів:

- Контролер даних вебзастосунків. Надає користувацькі інтерфейси для перегляду та маніпулювання даними, асоційованими зі згенерованими вебзастосунками. Цей компонент розмежовує доступ: клієнти можуть працювати безпосередньо з даними застосунку, тоді як системні адміністратори мають додаткові повноваження щодо управління структурою бази даних застосунків.

- Контролер електронної пошти. Вбудовується у згенеровані вебзастосунки для забезпечення функціональності сповіщень та комунікації через електронну пошту.

- Контролер вебсервера. Надає системним адміністраторам інтерфейс для віддаленого адміністрування середовища виконання вебзастосунків, зокрема вебсервера Apache Tomcat [5]. Можливо, також залучається Apache

HTTP Server [2] як фронтальний сервер для розподілу навантаження або обробки статичного контенту.

- Платформа адміністрування. Централізований компонент для системних адміністраторів, що дозволяє керувати загальною конфігурацією системи генерації, шаблонами доступних вебзастосунків, переліком функціональних модулів та базами даних, які використовуються згенерованими застосунками.

Реалізація проекту виконується переважно на мові програмування Java. Для побудови надійної та масштабованої архітектури застосовується інтегрований стек фреймворків Java SSH, який включає Spring [1] (для управління залежностями та бізнес-логіки), Hibernate [8] (як ORM-рішення для роботи з базою даних) та Struts (для реалізації архітектури Model-View-Controller) [8]. Управління збиранням проекту, залежностями та життєвим циклом розробки здійснюється за допомогою інструменту Maven [10]. Сервером застосунків для виконання згенерованих вебдодатків слугує Apache Tomcat [5].

Генератор вебзастосунків надає можливість інтеграції у вебсторінки широкого спектра типових елементів користувацького інтерфейсу, включаючи: текстові поля (text fields), текстові області (text areas), кнопки (кнопки відправки – submit buttons, кнопки скидання – reset buttons, настроювані кнопки – custom buttons), області пошуку (search areas), одиночні зображення (single images) та слайд-шоу зображень (image sliders), текстові мітки (text labels), спливаючі вікна (windows), розділи (sections) та довільні області (areas) для організації контенту.

1.4. Висновки до розділу

У першому розділі було проведено комплексний аналіз предметної області процесу генерації веб-сторінок та аплікацій. Встановлено, що

розробка систем автоматизованої генерації та управління життєвим циклом веб-аплікацій є актуальною задачею в умовах стрімкого розвитку цифрових технологій та підвищення вимог до швидкості і якості створення програмних продуктів. Автоматизація дозволяє значно скоротити час розробки, мінімізувати кількість помилок та забезпечити гнучкість подальшої підтримки та розвитку застосунків.

На основі аналізу сучасних тенденцій та вимог було обґрунтовано вибір інструментів і середовища розробки, які забезпечують необхідну функціональність, масштабованість та інтеграцію із сучасними технологіями. Особливу увагу приділено вибору фреймворків, мов програмування та засобів для управління життєвим циклом розробки.

Також розглянуто можливі моделі реалізації системи автоматизованої генерації веб-аплікацій. Вибір моделі був здійснений з урахуванням потреб забезпечення модульності, гнучкості та можливості розширення системи у майбутньому. Таким чином, результати аналізу заклали основу для формування вимог до майбутньої системи та визначили напрямки її подальшої розробки.

РОЗДІЛ 2. АЛГОРИТМІЧНА РЕАЛІЗАЦІЯ СИСТЕМИ ГЕНЕРАЦІЇ ВЕБ-АПЛІКАЦІЙ ТА ПРОЕКТУВАННЯ СТРУКТУРИ БАЗИ ДАНИХ

2.1. Вибір бібліотек і фреймворків для розробки

2.1.1. Maven як засіб управління проектом

Для реалізації даного проекту було обрано інструмент управління проектом та автоматизації збірки Apache Maven. Використання Maven зумовлене його фундаментальними перевагами у спрощенні та стандартизації процесів розробки програмного забезпечення, зокрема на платформі Java.

Історично Maven виник як ініціатива з уніфікації процесу збірки в рамках проекту Jakarta Turbine, де існувала множина проектів з різними сценаріями збірки на базі Ant та несистематизованим управлінням залежностями через CVS [17]. Ключова мета Maven полягає у підвищенні прозорості та керованості процесу розробки, дозволяючи розробникам швидко осягнути загальний стан проекту [17].

Для досягнення цієї мети Maven вирішує низку критичних завдань [17]:

- Спрощення процесу збірки. Абстрагування розробників від низькорівневих деталей процесу компіляції, тестування та пакування.

- Забезпечення уніфікованої системи збірки. Використання стандартизованої моделі проекту (Project Object Model - POM) та набору плагінів гарантує, що процес збірки є послідовним та відтворюваним для будь-якого Maven-проекту.

- Надання якісної інформації про проект. Автоматична генерація звітів щодо залежностей, тестового покриття, документації тощо.

- Заохочення дотримання кращих практик розробки. Структура проекту за замовчуванням та конвенції Maven спрямовують розробників до

використання загальноприйнятих підходів (наприклад, розташування вихідного коду та тестів, життєвий цикл збірки).

- Забезпечення прозорої міграції до нових функціональних можливостей. Спрощення інтеграції нових бібліотек та оновлень фреймворків за рахунок централізованого управління залежностями.

У контексті даного проекту Maven використовується для автоматизації всього життєвого циклу збірки – від компіляції вихідного коду Java до пакування готових артефактів (наприклад, WAR-файлів для розгортання на Tomcat) та управління залежностями проекту від зовнішніх бібліотек. Однією з вагомих переваг Maven є автоматичне resolved (визначення та завантаження) необхідних бібліотек (JAR-файлів) з центральних або корпоративних репозиторіїв на основі декларативного опису залежностей у POM-файлі (pom.xml). Це виключає необхідність ручного завантаження та додавання JAR-файлів до проекту.

Крім того, Maven забезпечує стійкість до зміни робочого простору проекту: при перенесенні проекту або роботі на іншій машині Maven автоматично відновлює всі необхідні залежності, усуваючи потребу у повторному ручному налаштуванні шляхів до бібліотек, що є суттєвим спрощенням порівняно з підходами, де управління залежностями не централізовано. Також Maven надає вбудовані плагіни, які можуть бути використані для виконання специфічних завдань, таких як розгортання застосунку на вебсервері, зокрема Tomcat.

2.1.2. Архітектурний шаблон Model-View-Controller (MVC)

Архітектура розробленої системи базується на використанні шаблону проектування Model-View-Controller (MVC) (рисунок 2.1). MVC є усталеним архітектурним патерном, що застосовується для структуризації програмного забезпечення, зокрема при розробці користувацьких інтерфейсів, шляхом розділення його на три взаємопов'язані логічні компоненти. Основна ідея

полягає у відокремленні представлення даних від бізнес-логіки та логіки обробки користувацького вводу, що сприяє підвищенню модульності, гнучкості та зручності підтримки коду.

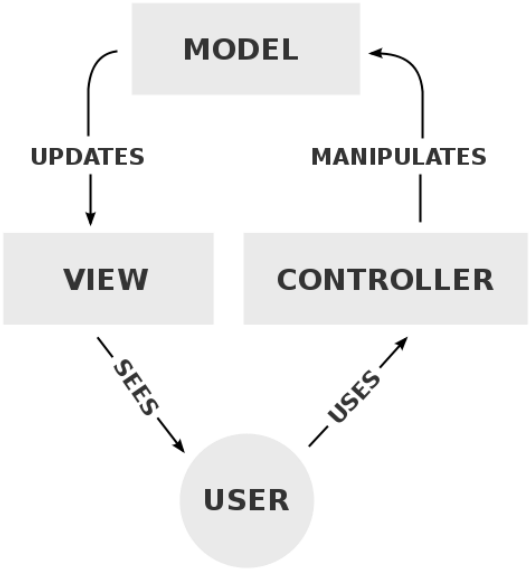

Рисунок 2.1 - Архітектурний шаблон Model-View-Controller (MVC)

Три ключові компоненти шаблону MVC та їхні функції:

- Модель (Model). Цей компонент відповідає за представлення даних застосунку, бізнес-логіку та правила, що регулюють роботу з даними. Модель інкапсулює стан даних та надає методи для доступу та маніпулювання ними. Вона є незалежною від способів відображення цих даних користувачеві. У даному проекті для зв'язку об'єктів моделі з базою даних та спрощення роботи з ними застосовуються Java Entity (сутності), що позначаються відповідними Java анотаціями, такими як @Entity (в рамках фреймворку персистентності, наприклад, Hibernate, який інтегровано через Spring).

- Представлення (View). Цей компонент відповідає за відображення даних з Моделі користувачеві та взаємодію з ним (наприклад, відображення форм введення даних). Представлення отримує дані від Моделі, але не містить бізнес-логіки. Одна і та ж Модель може мати декілька різних

Представлень, що дозволяє адаптувати спосіб відображення інформації для різних типів користувачів або цілей (наприклад, табличні дані для бухгалтерів та графічне представлення для менеджерів).

- Контролер (Controller). Цей компонент виступає посередником між Моделлю та Представленням. Він отримує вхідні дані від користувача (через Представлення), інтерпретує їх та перетворює на команди для Моделі або Представлення. Контролер викликає методи Моделі для оновлення даних або отримання необхідної інформації, а також обирає відповідне Представлення для відображення результатів користувачеві. Контролер не містить бізнес-логіки, його основне завдання – обробка вводу та координація взаємодії між іншими компонентами.

Для реалізації зв'язків та координації між компонентами Моделі, Представлення та Контролера в даному проекті активно використовується Spring Framework. Spring надає потужний інструментарій, зокрема механізми інверсії керування (IoC) та впровадження залежностей (DI), а також підтримку Java анотацій, що значно спрощує конфігурацію та зв'язування компонентів MVC. Наприклад, анотації використовуються для позначення класів як контролерів (@Controller), мапінгу URL-запитів на відповідні методи контролерів (@RequestMapping), автоматичного зв'язування компонентів тощо, забезпечуючи ефективну та гнучку реалізацію архітектурного шаблону MVC.

2.1.3. Spring Framework

В архітектурі системи використовується Spring Framework, який позиціонується як комплексний фреймворк для розробки додатків та інвертований контейнер для платформи Java. Spring надає фундаментальну підтримку, яка може бути застосована у будь-якому Java-додатку, проте він також містить розширення, спеціалізовані для створення корпоративних вебзастосунків на базі платформи Java EE (нині Jakarta EE). Відмінною

рисою Spring є його неінвазивний підхід та відсутність прив'язки до конкретної моделі програмування, що сприяло його широкому поширенню та популярності у спільноті Java-розробників.

Spring Framework пропонує всеохоплюючу програмну та конфігураційну модель, що істотно спрощує розробку складних Java-базованих вебзастосунків. Ключовою перевагою фреймворку є реалізація принципу інверсії керування (Inversion of Control - IoC) та механізму впровадження залежностей (Dependency Injection - DI). Ці механізми дозволяють управляти життєвим циклом об'єктів (компонентів) та їхніми залежностями, делегуючи ці функції контейнеру Spring. Такий підхід значно спрощує інтеграцію різних компонентів та технологій, таких як фреймворки для роботи з базами даних (наприклад, Hibernate), рушії бізнес-процесів (BPMN engine), а також архітектурні шаблони, як Model-View-Controller (MVC).

Інтеграція зовнішніх компонентів у середовище Spring здійснюється шляхом їх оголошення як Beans (керованих об'єктів) в конфігурації Spring. Традиційно це виконувалося за допомогою XML-файлів конфігурації, однак сучасні підходи активно використовують Java анотації для декларативного визначення Beans та їхніх залежностей, що зменшує обсяг конфігураційних файлів та підвищує читабельність коду. Spring контейнер відповідає за створення екземплярів цих компонентів та "ін'єкцію" їхніх залежностей автоматично, звільняючи розробника від необхідності явного створення об'єктів та управління їхніми зв'язками.

2.1.4. Hibernate Framework

Для ефективної роботи з реляційною базою даних у проекті застосовується Hibernate ORM (Object-Relational Mapping). Hibernate є потужним інструментом для об'єктно-реляційного відображення, призначеним для мови програмування Java. Він надає фреймворк для

відображення об'єктно-орієнтованої доменної моделі даних (представленої Java-класами) на структуру реляційної бази даних (представленої таблицями).

Основне завдання Hibernate полягає у вирішенні проблеми невідповідності парадигм (impedance mismatch) між об'єктно-орієнтованим підходом, де дані моделюються у вигляді об'єктів зі станом та поведінкою, та реляційним підходом, де дані зберігаються у вигляді таблиць зі зв'язками. Hibernate замінює необхідність прямого, низькорівневого доступу до бази даних (наприклад, через JDBC) вищими рівнями абстракції та автоматизованої обробки даних.

Ключові функції Hibernate включають:

- Визначення відповідності між класами сутностей (entity classes) у доменній моделі та таблицями в реляційній схемі.

- Відображення типів даних Java на типи даних бази даних.

- Автоматичне збереження стану об'єктів у базі даних та завантаження даних з БД в об'єкти.

- Підтримка власної об'єктно-орієнтованої мови запитів HQL (Hibernate Query Language), Criteria API та можливість виконання "рідних" SQL-запитів, абстрагуючи розробника від ручного написання SQL для більшості типових операцій (CRUD – Create, Read, Update, Delete).

- Hibernate самостійно генерує оптимізовані SQL-запити на основі об'єктних операцій та конфігурації відображення, звільняючи розробника від необхідності ручного управління SQL та конвертації наборів результатів.

Використання Hibernate суттєво спрощує роботу з базою даних порівняно з прямим використанням низькорівневих API, таких як JDBC. Відображення між Java-сутностями та таблицями бази даних конфігурується декларативно за допомогою Java анотацій (наприклад, @Entity, @Table, @Column) або XML-файлів відображення. Hibernate бере на себе відповідальність за реалізацію підключення до бази даних, управління пулом

з'єднань, управління транзакціями та автоматичне відображення даних між об'єктною та реляційною формами. Це контрастує з підходом прямого JDBC, де розробнику довелося б вручну реалізовувати управління з'єднаннями, пули з'єднань, синхронізацію потоків та процедури ручного відображення даних з ResultSet в об'єкти Java.

2.1.5. BPMN (Business Process Model and Notation)

Для моделювання та виконання бізнес-процесів у системі використовується стандарт Business Process Model and Notation (BPMN) [16]. BPMN є графічною нотацією, призначеною для візуального представлення бізнес-процесів у вигляді стандартизованої моделі.

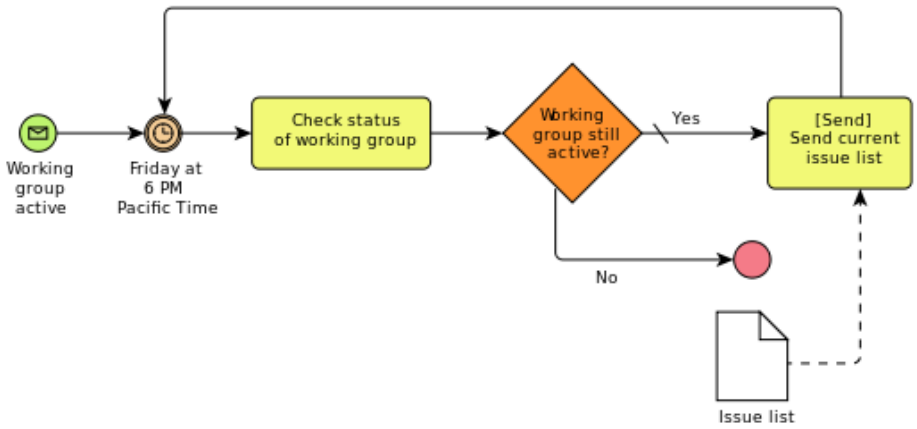

Рисунок 2.2 – Представлення бізнес-процесу засобами BPMN моделі

Стандарт BPMN був початково розроблений ініціативою Business Process Management Initiative. Після злиття цієї організації з Object Management Group (OMG) у 2005 році, розвиток та управління стандартом перейшли під егіду OMG. Важливою віхою став випуск версії BPMN 2.0 у січні 2011 року. У цій версії назва була розширена до "Business Process Model and Notation", щоб явно підкреслити включення не тільки графічної нотації та елементів діаграм, але й введення формальної семантики виконання, що зробило моделі BPMN виконуваними.

Використання BPMN у проекті дозволяє формалізувати логіку виконання процесів (наприклад, послідовність кроків користувача на вебсайті) у візуальному форматі, зрозумілому як бізнес-аналітикам, так і технічним спеціалістам. Оскільки BPMN може бути інтегрований з фреймворками, такими як Spring (через відповідні рушії BPMN), це істотно спрощує реалізацію функціональності управління робочими процесами (workflow). Замість імплементації складної логіки виконання процесу безпосередньо у програмному коді, розробник фокусується на проектуванні діаграми потоку BPMN, яка декларативно описує послідовність дій, шлюзи, події та завдання. Рушій BPMN, інтегрований зі Spring, відповідає за інтерпретацію цієї діаграми та керування виконанням процесу. Реалізація специфічної логіки для окремих завдань процесу може потребувати написання додаткового коду (наприклад, обробників завдань, сервісів), часто з використанням мов сценаріїв, таких як JavaScript, або більш комплексних компонентів, які викликаються рушієм BPMN.

2.2. Протоколи взаємодії та клієнтські технології

2.2.1. Протокол SOAP (Simple Object Access Protocol)

SOAP (Simple Object Access Protocol) являє собою специфікацію протоколу, призначену для структурованого обміну інформацією в розподілених середовищах, зокрема при реалізації вебсервісів у комп'ютерних мережах. Фундаментальними принципами, закладеними в основу SOAP, є забезпечення розширюваності, нейтральності та незалежності.

SOAP використовує XML Information Set як формат для своїх повідомлень, що забезпечує структуроване представлення даних. Передача цих повідомлень покладається на протоколи прикладного рівня, найбільш поширеними серед яких є Hypertext Transfer Protocol (HTTP) та Simple Mail

Transfer Protocol (SMTP) [12]. Така архітектура дозволяє процесам, що виконуються на різнорідних програмних та апаратних платформах (наприклад, операційні системи Windows та Linux), здійснювати взаємодію шляхом обміну повідомленнями у стандартному форматі XML. Завдяки опорі на загальноприйняті вебпротоколи, такі як HTTP, які підтримуються переважною більшістю операційних систем, SOAP забезпечує можливість клієнтам викликати віддалені вебсервіси та отримувати відповіді незалежно від мови програмування чи платформи реалізації самих сервісів та клієнтів.

У контексті архітектури даного проекту, SOAP використовується для реалізації взаємодії між основною системою та спеціалізованою підсистемою управління файлами. Ця підсистема відповідає за виконання всіх операцій, пов'язаних з файлами та каталогами на сервері, включаючи створення, копіювання/дублювання, видалення та оновлення файлів. Основна система ініціює команди до цієї підсистеми, передаючи необхідні параметри в структурованих SOAP-повідомленнях. Зокрема, через цей інтерфейс реалізуються такі функції, як копіювання та оновлення файлів конфігурації вебсервера (Tomcat), а також завантаження файлів з клієнтської системи на сервер. Реалізація вебсервісної частини та клієнтської частини взаємодії здійснювалася з використанням відповідних інструментів інтегрованого середовища розробки Eclipse.

2.2.2. Архітектурний стиль REST (Representational State Transfer)

Representational State Transfer (REST), або RESTful Web services, є архітектурним стилем, який пропонує певний набір обмежень для проектування розподілених систем, що взаємодіють через мережу Інтернет. Основний принцип REST полягає у роботі з ресурсами, які ідентифікуються за допомогою уніфікованих ідентифікаторів ресурсів (URI), та маніпулюванні їхніми текстовими представленнями (representation) за

допомогою стандартизованого набору операцій без стану (stateless operations).

Взаємодія в RESTful системах переважно базується на використанні протоколу HTTP, при цьому для виконання операцій над ресурсами застосовуються його стандартні методи, такі як GET (для отримання представлення ресурсу), POST (для створення нового ресурсу або відправки даних), PUT (для повного оновлення ресурсу) та DELETE (для видалення ресурсу). Відсутність стану на сервері між запитами клієнта спрощує масштабування системи.

У даному проекті архітектурний стиль REST застосовується для організації зв'язку між основною системою та рушієм BPMN. Зокрема, через RESTful інтерфейс рушій BPMN може надавати дані (наприклад, у форматі JSON-рядка) основній системі, яка потім використовує ці дані для відображення відповідного контенту на інтерфейсі користувача. Порівняно з протоколами, що вимагають підтримки стану сесії на сервері або використання специфічних клієнтських бібліотек для обробки повідомлень (як це часто буває з SOAP), RESTful підхід, що базується на стандартних механізмах HTTP, може вимагати менших зусиль на розробку клієнтської частини взаємодії, оскільки більшість сучасних клієнтських платформ та браузерів мають вбудовану підтримку для виконання HTTP-запитів. Це сприяє зниженню трудовитрат при реалізації з'єднань між компонентами системи.

2.2.3. Бібліотека jQuery Ajax

Для реалізації асинхронної взаємодії на клієнтській стороні вебзастосунків у проекті використовується бібліотека jQuery у поєднанні з технологією Ajax. Вибір бібліотеки jQuery обґрунтовано її широкою популярністю та функціональністю, спрямованою на спрощення клієнтського скриптингу HTML. jQuery є найбільш поширеною JavaScript бібліотекою, що

використовується значною часткою вебсайтів у мережі, що свідчить про її ефективність та надійність.

Ajax (Asynchronous JavaScript and XML) являє собою сукупність вебтехнологій, що застосовуються на стороні клієнта (у веббраузері) для створення асинхронних вебзастосунків. Використання Ajax дозволяє вебдодаткам здійснювати обмін даними з сервером у фоновому режимі, тобто асинхронно, без необхідності перезавантаження всієї вебсторінки. Це суттєво покращує інтерактивність та зручність користувацького інтерфейсу, оскільки дозволяє динамічно оновлювати частини сторінки, відправляти дані на сервер або отримувати їх, не перериваючи взаємодію користувача з іншим вмістом сторінки.

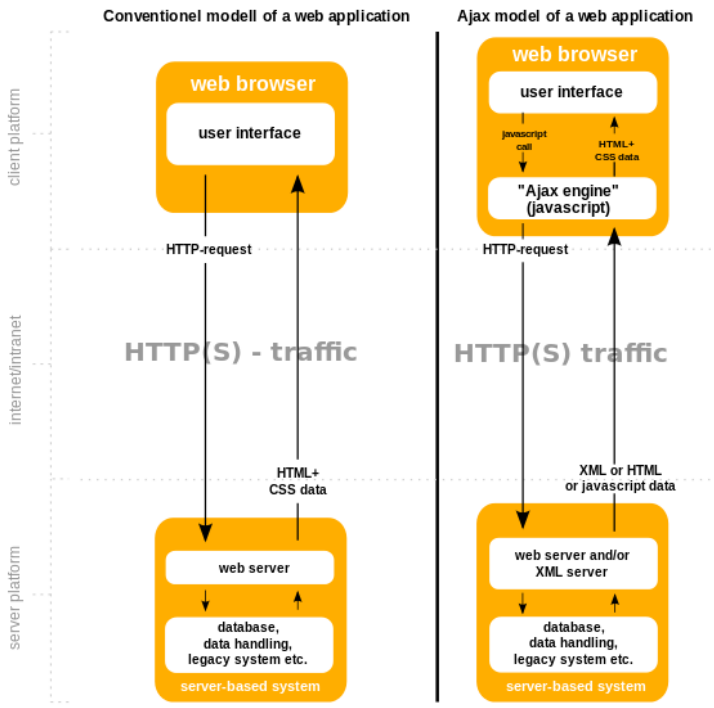

Рисунок 2.3 - Типовий робочий процес взаємодії з використанням jQuery Ajax

Бібліотека jQuery надає абстрагований та зручний API для виконання Ajax-запитів, що значно спрощує процес розробки асинхронної клієнтської логіки порівняно з безпосереднім використанням нативного об'єкта

XMLHttpRequest, який є основою технології Ajax у браузерах. jQuery обробляє відмінності у реалізації Ajax між різними браузерами та надає простий інтерфейс для надсилання запитів та обробки відповідей (наприклад, у форматах JSON, XML, HTML).

Workflow взаємодії з використанням jQuery Ajax у типовому випадку може бути представлений схемою (рис. 2.3). Клієнтська подія (наприклад, натискання кнопки) ініціює JavaScript-функцію, яка за допомогою методів jQuery Ajax відправляє асинхронний HTTP-запит на сервер. Сервер обробляє запит, виконує необхідні дії (наприклад, звертається до бази даних або викликає бізнес-логіку) та повертає відповідь (часто у форматі JSON). JavaScript-функція на клієнті отримує цю відповідь та динамічно оновлює вміст вебсторінки без її повного перезавантаження, забезпечуючи плавну та швидку взаємодію з користувачем.

2.3. Проектування системи з використанням use case діаграм

На рисунку 2.4 показано загальний вигляд системи.

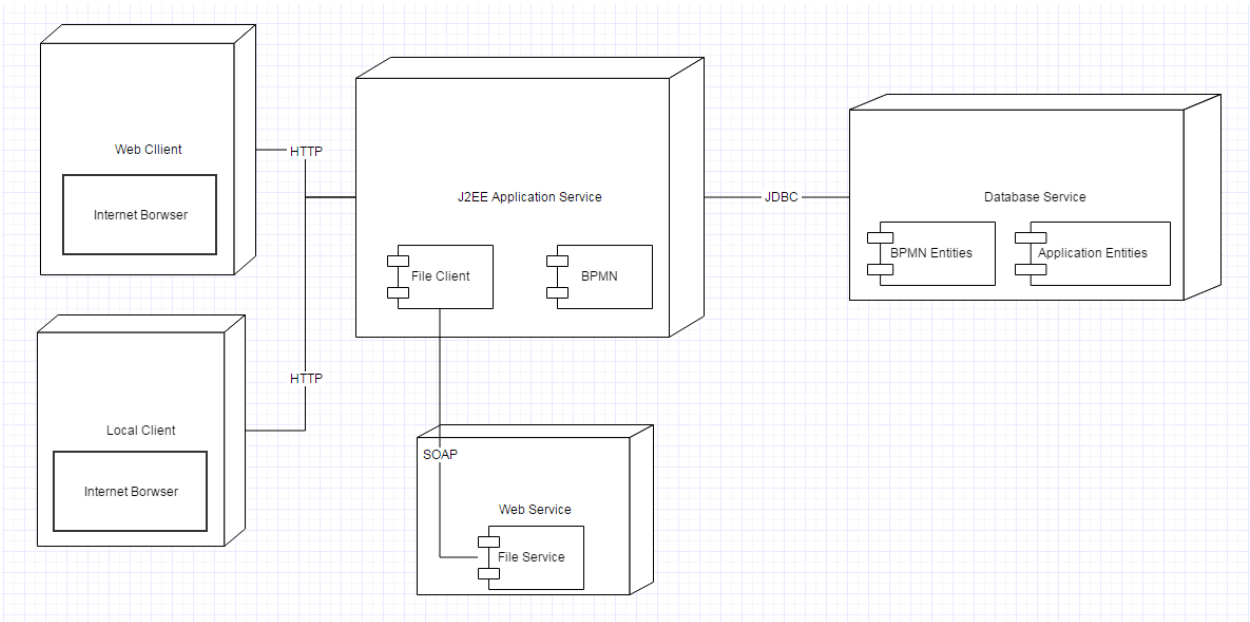

Рисунок 2.4 - Огляд конструкції системи

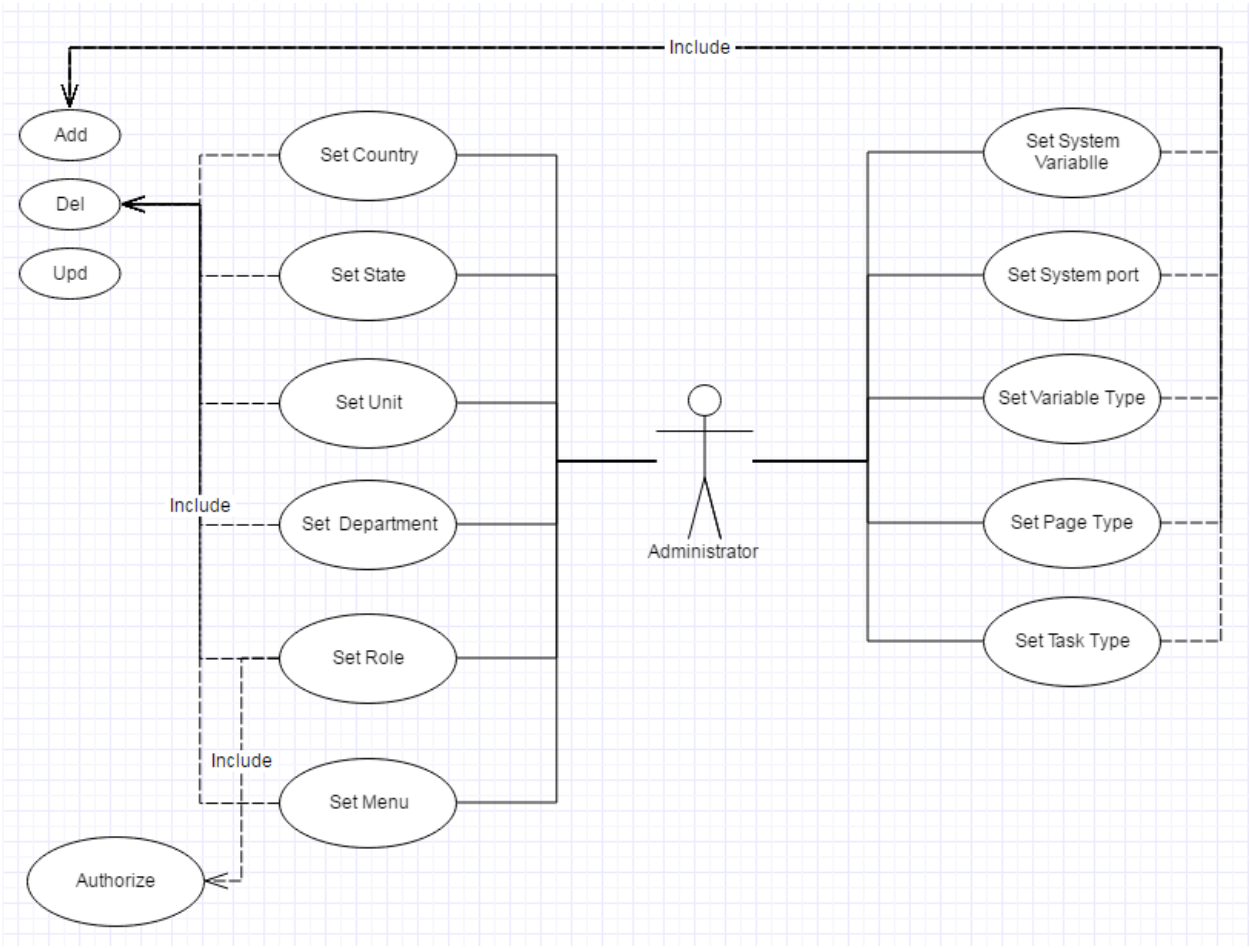

Рисунок 2.5 - Варіант використання компонента конфігурації системи

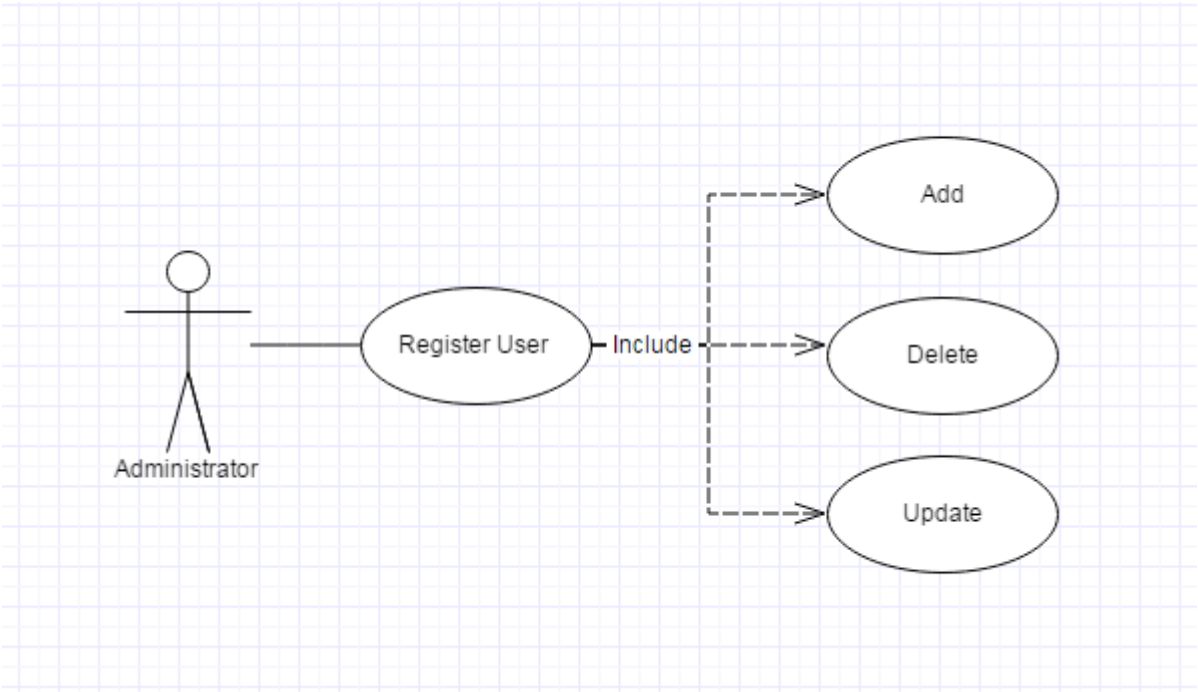

Рисунок 2.6 - Варіант використання клієнтського компонента

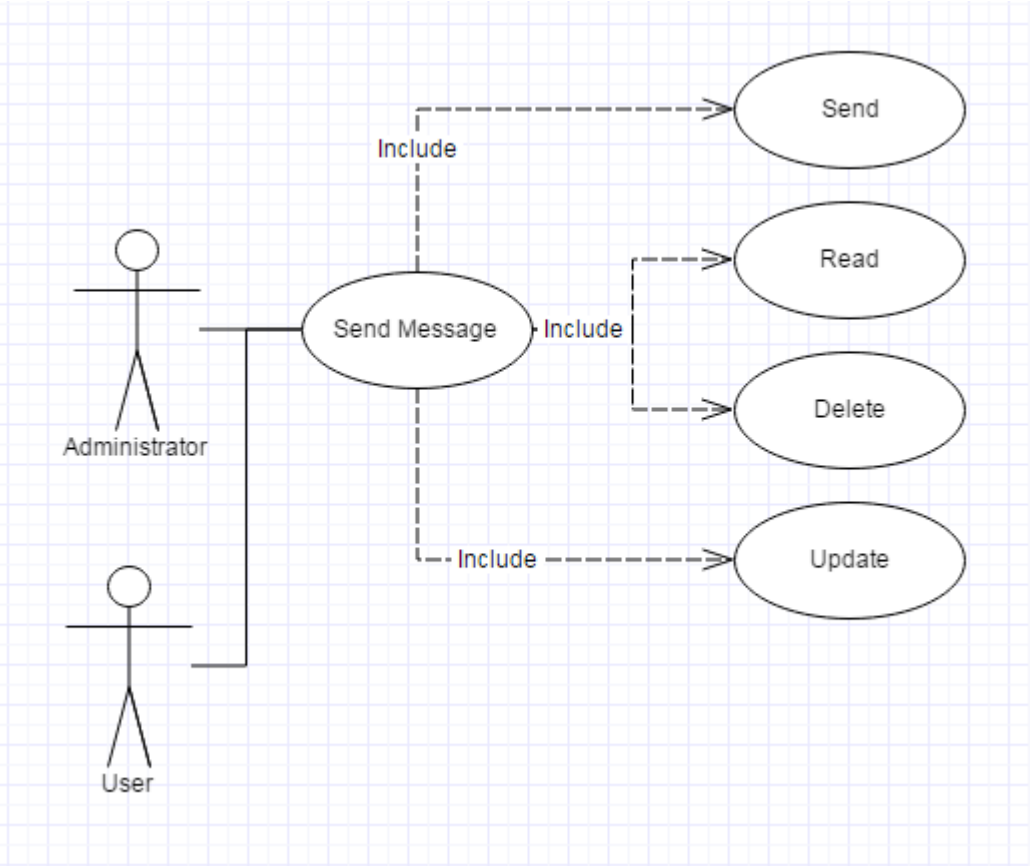

Рисунок 2.7 - Варіант використання компонента повідомлень.

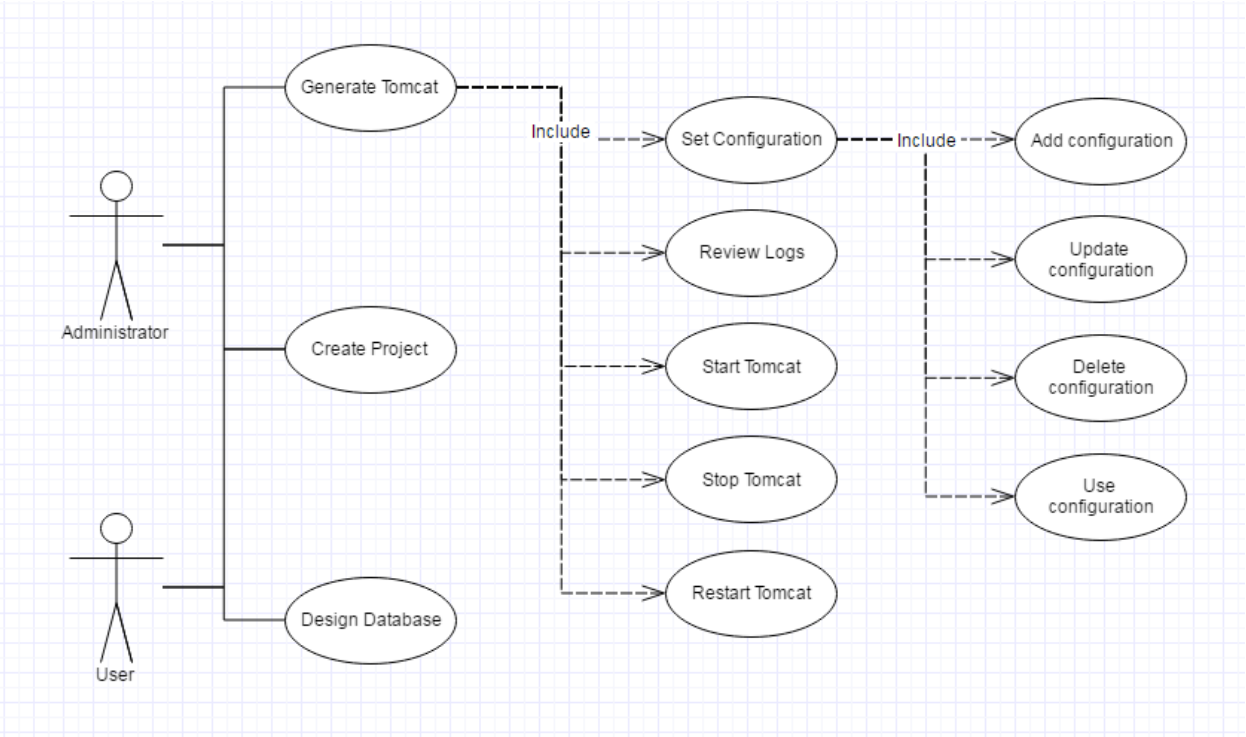

Рисунок 2.9 - Варіант використання компонента Tomcat

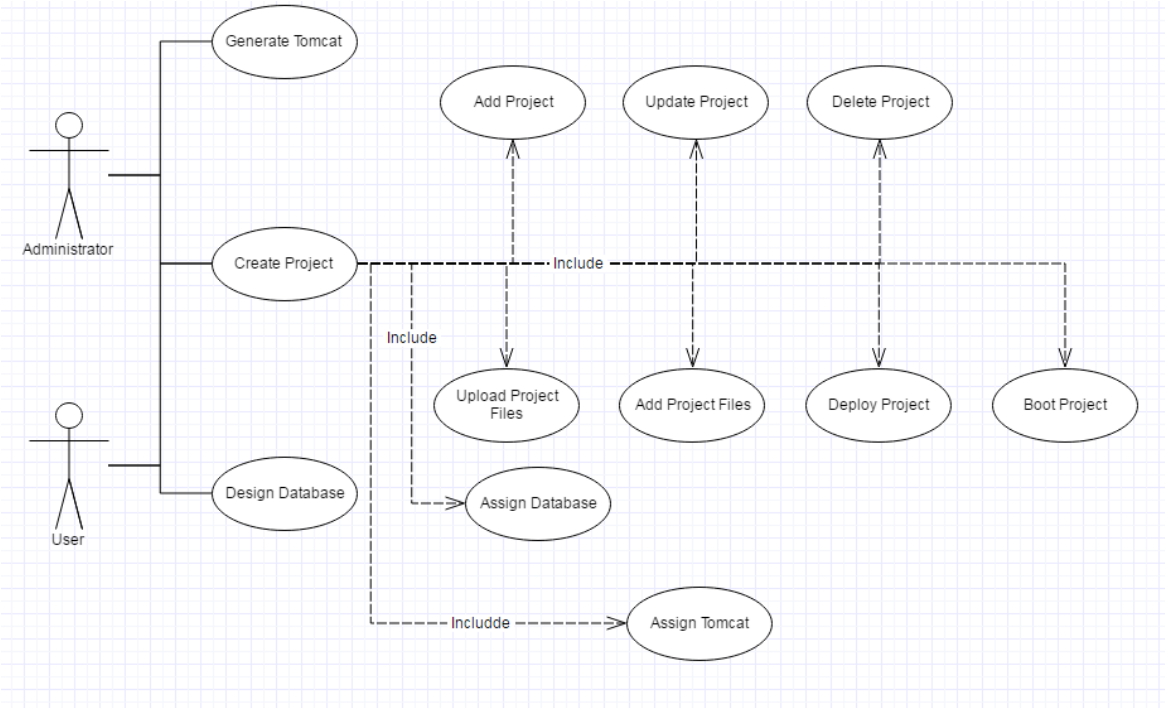

Рисунок 2.10 - Варіант використання компонента проекту

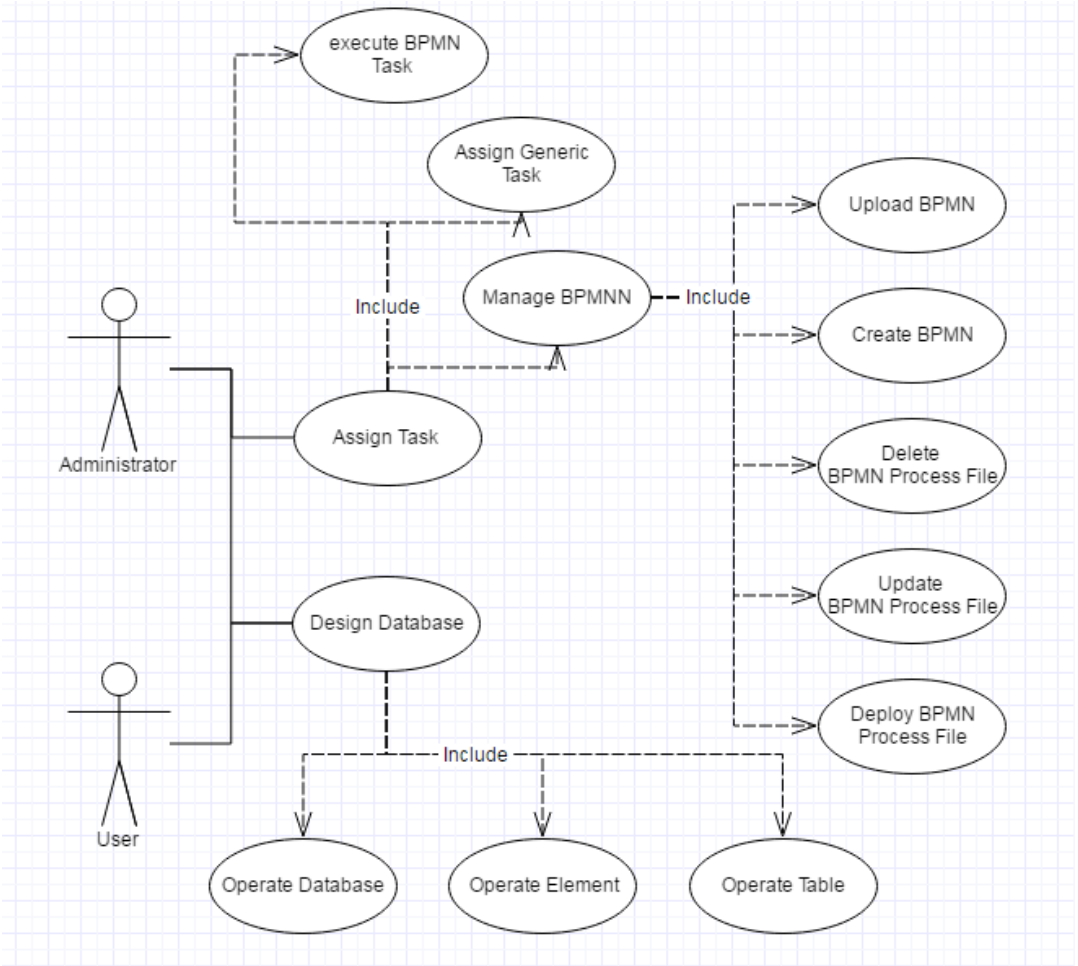

Рисунок 2.11 - Варіант використання компонента завдання та бази даних

2.4. Розробка ER діаграм проекту

ER-діаграма — це візуальний інструмент, що використовується для моделювання структури бази даних. Вона відображає сутності (Entity), їхні атрибути (Attribute) та зв'язки (Relationship) між цими сутностями. Це концептуальна модель, яка допомагає зрозуміти організацію даних та зв'язків між ними перед створенням фактичної бази даних.

Основні компоненти ER-діаграми:

- Сутності (Entities): Об'єкти або поняття, про які потрібно зберігати інформацію. На діаграмі вони зазвичай представлені прямокутниками. (На вашій діаграмі це Menu, Authority, Role, User, Country, State, Department, Unit).

- Атрибути (Attributes): Характеристики або властивості сутності. Вони зазвичай перераховані всередині прямокутника сутності. Ключові атрибути (первинні ключі), що унікально ідентифікують сутність, часто позначаються (наприклад, id). (На вашій діаграмі це menuName, path для Menu, username, upassword, email для User тощо).

- Зв'язки (Relationships): Взаємодія або асоціація між двома або більше сутностями. Вони зазвичай представлені лініями, що з'єднують сутності. Тип зв'язку (один-до-одного, один-до-багатьох, багато-до-багатьох) позначається символами на кінцях ліній (наприклад, "вороняча лапка" для "багатьох", пряма лінія для "одного"). Зв'язки також можуть мати назву, що описує характер взаємодії (наприклад, Belongs, Owns).

На рисунку 2.12 показана ER діаграма на якій змодельовано структуру даних для системи, яка, пов'язана з управлінням користувачами, їхніми ролями, дозволами в системі меню, а також географічною та організаційною приналежністю.

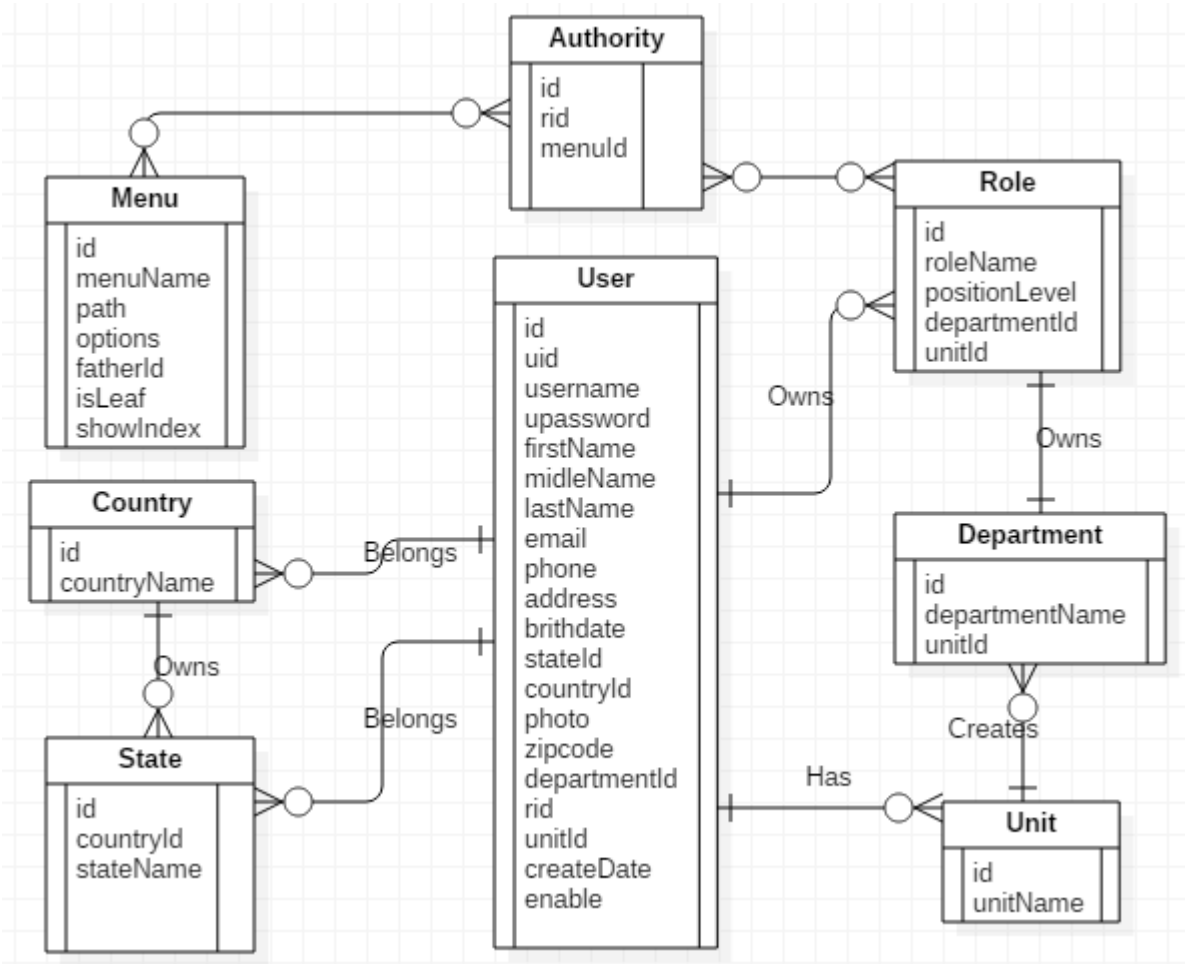

Рисунок 2.12 - ER-компонент клієнта

Сутності: Menu, Authority, Role, User, Country, State, Department, Unit.

Атрибути: Кожна сутність має набір атрибутів, які описують її властивості (наприклад, у User є username, email, phone, address, birthdate, createDate тощо). Атрибути id у більшості сутностей, імовірно, є первинними ключами.

Зв'язки: Діаграма показує різні зв'язки між сутностями:

- User Belongs to Country (Багато користувачів належать до однієї країни - багато-до-одного).

- User Belongs to State (Багато користувачів належать до одного штату/області - багато-до-одного).

- Country Owns State (Одна країна має багато штатів/областей - один-до-багатьох).

- User Owns Role (Користувач має роль - ймовірно, один-до-одного або один-до-багатьох, залежно від символу на кінці зв'язку біля Role - тут виглядає як один-до-одного, якщо User може мати тільки одну роль, або багато-до-одного, якщо багато користувачів можуть мати одну роль).

- Role Owns Department (Роль належить до відділу - багато-до-одного).

- Department Owns Unit (Відділ належить до підрозділу - багато-до-одного).

- Unit Creates User (Підрозділ створює користувача - один-до-багатьох).

- User Has Unit (Користувач має підрозділ - багато-до-одного).

Authority пов'язана з Menu та Role (моделює дозволи, де певна Role має Authority на певні Menu пункти). Зв'язок між Authority та Role виглядає як багато-до-одного (багато повноважень належать до однієї ролі). Зв'язок між Authority та Menu виглядає як багато-до-одного (багато повноважень стосуються одного пункту меню). Таким чином, сутність Authority є зв'язуючою таблицею для зв'язку багато-до-багатьох між Role та Menu, показуючи, які ролі мають доступ до яких пунктів меню.

Ця діаграма є прикладом того, як ER-моделювання допомагає візуалізувати складні взаємозв'язки між різними частинами системи даних, що є критично важливим етапом при проектуванні реляційних баз даних.

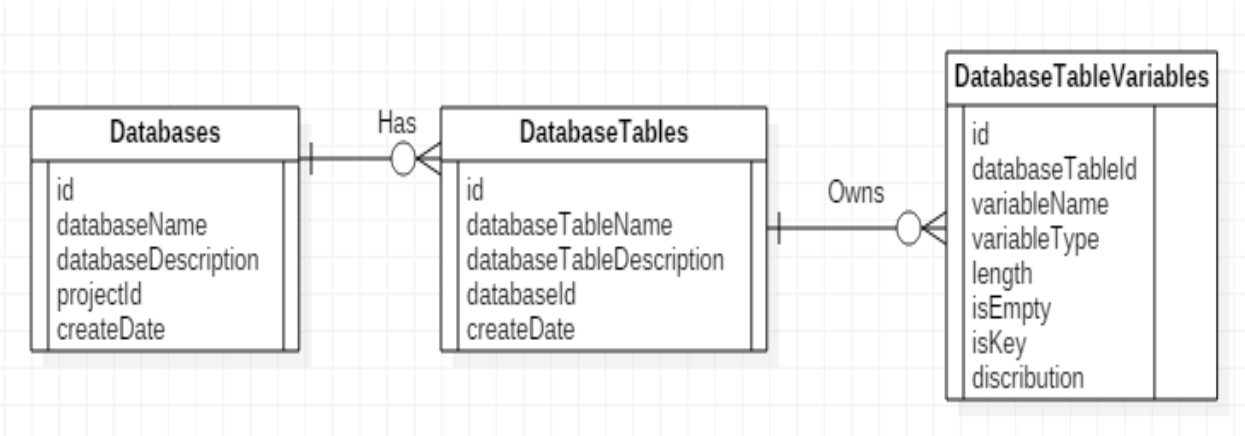

Рисунок 2.13 - Компонент бази даних

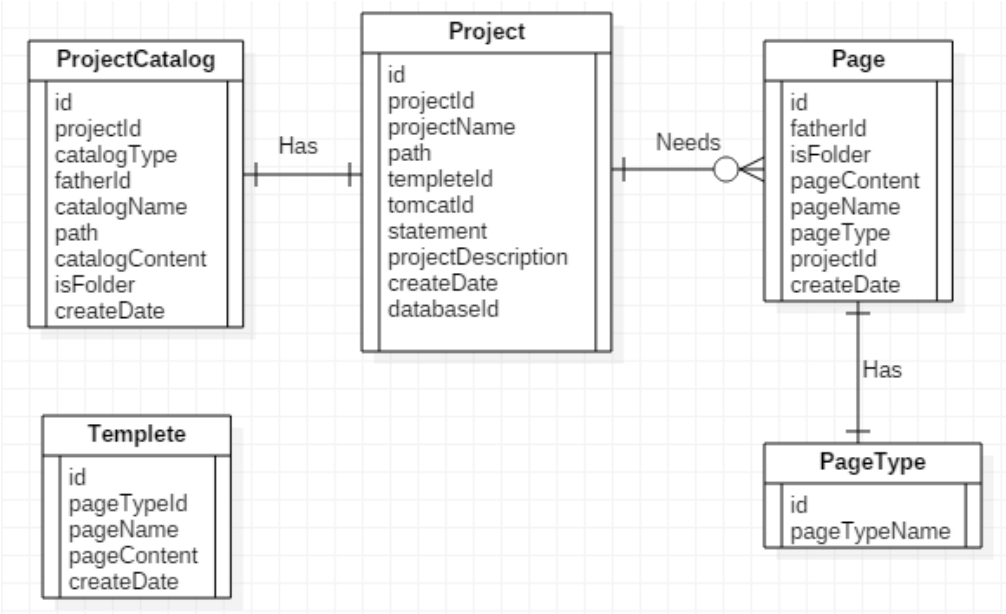

Рисунок 2.14 - ER-компонент проекту

На цій діаграмі (рис. 2.14) представлені наступні сутності:

- ProjectCatalog (Каталог проектів): використовується для організації проектів у каталоги або групи.

- Project (Проект): Основна сутність, що представляє вебпроект.

- Page (Сторінка): Сутність, що представляє окрему вебсторінку в рамках проекту.

- Template (Шаблон): Сутність, що представляє шаблони, які можуть використовуватися при створенні сторінок або проектів.

- PageType (Тип сторінки): Сутність для класифікації сторінок за їх типом.

Атрибути кожної сутності описують її характеристики (наприклад, id, name, description, path, createDate тощо). Атрибути з суфіксом Id (наприклад, projectId, templateId, databaseId, pageTypeId) в одній сутності, які відповідають id іншої сутності, вказують на зв'язки між ними (зовнішні ключі).

Зв'язки між сутностями показані лініями та символами:

- ProjectCatalog Has Project: Зв'язок типу "один-до-багатьох" (один каталог проектів може містити багато проектів).

- Project Needs Page: Зв'язок типу "один-до-багатьох" (один проект складається з багатьох сторінок).

- Page Has PageType: Зв'язок типу "багато-до-одного" (багато сторінок мають один і той самий тип сторінки).

Атрибут templateId у сутності Project вказує на зв'язок типу "багато-до-одного" з сутністю Template (багато проектів можуть використовувати один шаблон), навіть якщо пряма лінія зв'язку не зображена на цьому фрагменті діаграми.

Атрибут databaseId у сутності Project, ймовірно, вказує на зв'язок з сутністю "База даних", яка, можливо, не включена у цей фрагмент діаграми, але фігурувала у попередньому обговоренні компонентів системи.

Загалом, ця ER-діаграма моделює структуру даних, яка підтримує систему управління проектами вебзастосунків, їхню організацію в каталоги, структурування проектів за сторінками, класифікацію сторінок за типами та використання шаблонів при створенні проектів.

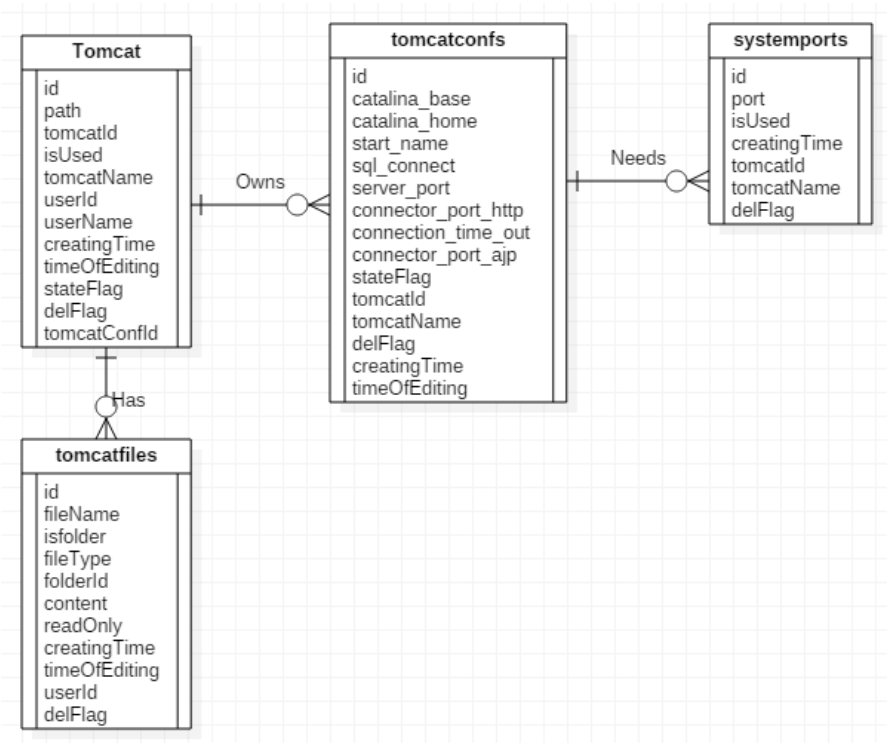

Рисунок 2.15 - ER-компонент Tomcat

2.5. Розробка таблиць бази даних

Нижче на рисунках представлені таблиці розробленої бази даних.

id	int	11	0	☐	🔑
countryName	varchar	50	0	☑	

Рисунок 2.16 - Таблиця країн

id	int	11	0	☐	🔑
countryId	int	11	0	☑	
stateName	varchar	50	0	☑	

Рисунок 2.17 – Таблиця штату

id	int	11	0	☐	🔑
departmentName	varchar	50	0	☐	
unitId	int	11	0	☐	

Рисунок 2.18 – Таблиця департаменту

id	int	11	0	☐	🔑
unitName	varchar	50	0	☑	

Рисунок 2.19 – Таблиця одиниці

id	int	11	0	☐	🔑
rid	int	11	0	☐	
menuId	int	11	0	☐	

Рисунок 2.20 – Таблиця авторизації

id	int	11	0	☐	🔑
variableTypeName	varchar	50	0	☐	
variableName	varchar	255	0	☑	

Рисунок 2.21 – Таблиця змінних

id	int	11	0	☐	🔑
roleName	varchar	50	0	☐	
positionLevel	int	11	0	☐	
departmentId	int	11	0	☐	
unitId	int	11	0	☐	

Рисунок 2.22 – Таблиця ролей

id	int	11	0	☐	🔑
menuName	varchar	50	0	☐	
path	varchar	50	0	☑	
options	varchar	255	0	☑	
fatherId	int	11	0	☐	
isLeaf	int	11	0	☐	
showIndex	int	11	0	☐	

Рисунок 2.23 – Таблиця меню

id	int	11	0	☐	🔑
taskTypeName	varchar	50	0	☐	

Рисунок 2.24 – Таблиця типів завдань

id	int	11	0	☐	🔑
pageTypeName	varchar	50	0	☐	

Рисунок 2.25 – Таблиця типів сторінок

id	int	11	0	☐	🔑
port	varchar	255	0	☐	
isUsed	int	11	0	☐	
creatingTime	varchar	255	0	☐	
tomcatId	varchar	255	0	☑	
tomcatName	varchar	255	0	☑	
delFlag	int	11	0	☐	

Рисунок 2.26 – Таблиця портів системи

id	int	11	0	☐	🔑
originalTomcatPath	varchar	100	0	☑	
targetTomcatPath	varchar	100	0	☑	
projectPath	varchar	100	0	☑	
photoPath	varchar	100	0	☑	
bpmnProcessFilePath	varchar	100	0	☑	
pageSize	int	11	0	☑	

Рисунок 2.27 – Таблиця змінних конфігурації системи

id	int	11	0	☐	🔑
uid	varchar	50	0	☐	
username	varchar	50	0	☐	
upassword	varchar	50	0	☐	
firstName	varchar	50	0	☑	
midleName	varchar	50	0	☑	
lastName	varchar	50	0	☑	
email	varchar	50	0	☐	
phone	varchar	50	0	☑	
address	varchar	100	0	☑	
brithdate	varchar	50	0	☑	
stateId	int	11	0	☑	
countryId	int	11	0	☑	
photo	varchar	100	0	☑	
zipcode	varchar	50	0	☑	
departmentId	int	50	0	☑	
rid	int	11	0	☐	
unitId	int	11	0	☐	
createDate	varchar	50	0	☐	
enable	int	11	0	☐	

Рисунок 2.28 – Таблиця клієнтів

id	int	11	0	☐	🔑
pageTypeId	int	11	0	☐	
pageName	varchar	50	0	☐	
pageContent	text	0	0	☑	
createDate	varchar	50	0	☐	

Рисунок 2.29 – Таблиця типів сторінок

id	int	11	0	☐	🔑
taskName	varchar	50	0	☐	
receiverId	varchar	50	0	☐	
departmentId	int	11	0	☐	
taskFromId	varchar	50	0	☐	
taskTypeId	int	11	0	☐	
taskContent	text	0	0	☐	
isAccept	int	11	0	☑	
startDate	varchar	50	0	☑	
dueDate	varchar	50	0	☐	
process	int	11	0	☑	
createDate	varchar	255	0	☐	

Рисунок 2.30 – Таблиця завдань

id	int	11	0	☐	🔑
projectId	varchar	50	0	☑	
projectName	varchar	50	0	☐	
path	varchar	255	0	☐	
templeteId	int	11	0	☑	
tomcatId	varchar	50	0	☑	
statement	int	11	0	☑	
projectDescription	text	0	0	☑	
createDate	varchar	50	0	☐	
databaseId	int	11	0	☑	

Рисунок 2.31 – Таблиця проектів

id	int	11	0	☐	🔑
projectId	int	11	0	☐	
catalogType	varchar	50	0	☐	
fatherId	int	11	0	☐	
catalogName	varchar	50	0	☐	
path	varchar	255	0	☑	
catalogContent	longtext	0	0	☑	
isFolder	int	11	0	☐	
createDate	varchar	50	0	☐	

Рисунок 2.32 – Таблиця файлів проектів

id	int	11	0	☐	🔑
accountName	varchar	50	0	☑	
address	varchar	100	0	☑	
cpassword	varchar	50	0	☑	
countryId	int	11	0	☐	
createDate	varchar	50	0	☑	
customerId	varchar	50	0	☑	
DOB	varchar	50	0	☑	
email	varchar	50	0	☑	
enable	int	11	0	☐	
firstName	varchar	50	0	☑	
midleName	varchar	50	0	☑	
lastName	varchar	50	0	☑	
phone	varchar	50	0	☑	
stateId	int	11	0	☐	
zipcode	varchar	50	0	☑	
apassword	varchar	255	0	☑	
dateofbrith	varchar	255	0	☑	

Рисунок 2.32 – Таблиця клієнтів

id	int	11	0	☐	🔑
bpmnProcessName	varchar	50	0	☐	
realName	varchar	50	0	☐	
bpmnProcessPath	varchar	100	0	☐	
bpmnDescription	text	0	0	☑	
uploadDate	varchar	50	0	☐	

Рисунок 2.33 – Таблиця завантажених файлів BPMN

id	int	11	0	☐	🔑
databaseName	varchar	50	0	☐	
databaseDescription	text	0	0	☐	
projectId	int	11	0	☑	
createDate	varchar	50	0	☐	

Рисунок 2.34 – Таблиця бази даних

id	int	11	0	☐	🔑
databaseTableName	varchar	255	0	☐	
databaseTableDescription	text	0	0	☑	
databaseId	int	11	0	☐	
createDate	varchar	255	0	☐	

Рисунок 2.35 – Таблиця таблиць бази даних

id	int	11	0	☐	🔑
databaseTableId	int	11	0	☐	
variableName	varchar	50	0	☐	
variableType	varchar	50	0	☐	
length	int	11	0	☐	
isEmpty	int	11	0	☐	
isKey	int	11	0	☐	
discribution	text	0	0	☑	

Рисунок 2.36 – Таблиця змінних таблиць бази даних

id	int	11	0	☐	🔑
createDate	varchar	255	0	☑	
fatherId	int	11	0	☐	
isFolder	int	11	0	☐	
pageContent	varchar	255	0	☑	
pageName	varchar	255	0	☑	
pageType	varchar	255	0	☑	
projectId	int	11	0	☐	

Рисунок 2.37 – Таблиця сторінок

id	int	11	0	☐	🔑
title	varchar	50	0	☐	
senderName	varchar	50	0	☑	
senderId	varchar	50	0	☑	
receiverName	varchar	50	0	☑	
receiverId	varchar	50	0	☑	
content	text	0	0	☑	
isRead	int	11	0	☐	
isReply	int	50	0	☐	
replyId	int	50	0	☐	
createDate	varchar	50	0	☐	

Рисунок 2.38 – Таблиця повідомлень

id	int	11	0	☐	🔑
catalina_base	varchar	255	0	☐	
catalina_home	varchar	255	0	☐	
start_name	varchar	255	0	☐	
sql_connect	text	0	0	☑	
server_port	varchar	255	0	☐	
connector_port_http	varchar	255	0	☐	
connection_time_out	varchar	255	0	☐	
connector_port_ajp	varchar	255	0	☐	
stateFlag	int	11	0	☐	
tomcatId	varchar	255	0	☐	
tomcatName	varchar	255	0	☐	
delFlag	int	11	0	☐	
creatingTime	varchar	255	0	☐	
timeOfEditing	varchar	255	0	☐	

Рисунок 2.39 – Таблиця конфігурації Tomcat

id	varchar	255	0	☐	🔑
fileName	varchar	50	0	☐	
isfolder	int	11	0	☐	
fileType	int	11	0	☑	
folderId	varchar	255	0	☐	
content	text	0	0	☑	
readOnly	int	11	0	☐	
creatingTime	varchar	50	0	☐	
timeOfEditing	varchar	50	0	☐	
userId	int	11	0	☐	
delFlag	int	11	0	☐	

Рисунок 2.40 – Таблиця файлів

id	int	11	0	☐	🔑
path	varchar	255	0	☐	
tomcatId	varchar	255	0	☐	
isUsed	int	11	0	☐	
tomcatName	varchar	255	0	☐	
userId	int	11	0	☐	
userName	varchar	50	0	☐	
creatingTime	varchar	50	0	☐	
timeOfEditing	varchar	50	0	☐	
stateFlag	int	11	0	☐	
delFlag	int	11	0	☐	
tomcatConfId	int	11	0	☑	

Рисунок 2.41 – Таблиця Tomcat

2.6. Висновки до розділу

У другому розділі було розглянуто алгоритмічну реалізацію системи генерації веб-аплікацій та проєктування структури бази даних. На основі аналізу було здійснено обґрунтований вибір основних бібліотек і фреймворків для реалізації системи. Maven обрано як ефективний засіб управління залежностями та проєктною структурою. Архітектурний шаблон Model-View-Controller (MVC) забезпечив логічне розділення компонентів системи, що сприяє її масштабованості та зручності супроводу.

Spring Framework було використано як основний каркас для створення серверної частини, завдяки його гнучкості та широкій підтримці модульності. Для роботи з базами даних обрано Hibernate Framework, що дозволяє реалізувати ефективне об'єктно-реляційне відображення (ORM) та спростити взаємодію із СУБД. Також у розробці використовувалася нотація BPMN для моделювання бізнес-процесів, що дозволило структурувати логіку бізнес-операцій на високому рівні абстракції.

Розглянуто і обґрунтовано вибір протоколів взаємодії між клієнтською та серверною частинами системи. Протокол SOAP та архітектурний стиль REST забезпечили можливість гнучкої побудови сервісів із врахуванням різних вимог до інтеграції. Бібліотека jQuery Ajax була обрана для реалізації

асинхронної взаємодії на стороні клієнта, що підвищує інтерактивність веб-аплікацій.

Окрему увагу приділено проєктуванню структури бази даних. Було розроблено ER-діаграми проєкту, що дозволили визначити сутності, їх атрибути та взаємозв'язки. На основі діаграм були спроєктовані таблиці бази даних, що забезпечують цілісність даних та ефективну організацію зберігання інформації.

Таким чином, виконаний аналіз і вибір технологій створив міцне підґрунтя для подальшої реалізації функціональної системи автоматизованої генерації веб-аплікацій.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ АВТОМАТИЗАЦІЇ ПРОЦЕСУ ГЕНЕРАЦІЇ ВЕБ-АПЛІКАЦІЙ ТА СТОРІНОК

3.1. Реалізація конфігурації системи

Під час входу в систему адміністратор повинен налаштувати кілька елементів, як показано на рисунку 3.1. Перший елемент — це коренева папка оригінального tomcat та цільова папка tomcat, де повинні зберігатися клієнтські tomcat. Оригінальна папка tomcat розміщена разом з проектом у папці розгорнутого проекту веб-сервісу tomcat. Другий елемент — це шлях до проекту, де повинні зберігатися проекти клієнтів. Третій елемент — це шлях до папки для зображень, де повинні зберігатися всі завантажені зображення. Четвертий елемент — це шлях до файлу BPMN, де повинні зберігатися всі завантажені файли процесу BPMN. Останній елемент — це розмір сторінки, який визначає, скільки елементів повинно відображатися на сторінці списку.

Рисунок 3.1 - Конфігурація змінних системи

Конфігурація портів системи (рисунок 3.2) призначена для веб-сервісу tomcat. Кожен веб-сервіс tomcat потребує трьох системних портів, щоб

система могла працювати. Адміністратор систем повинен перевірити та переконатися, що достатньо системних портів для клієнта.

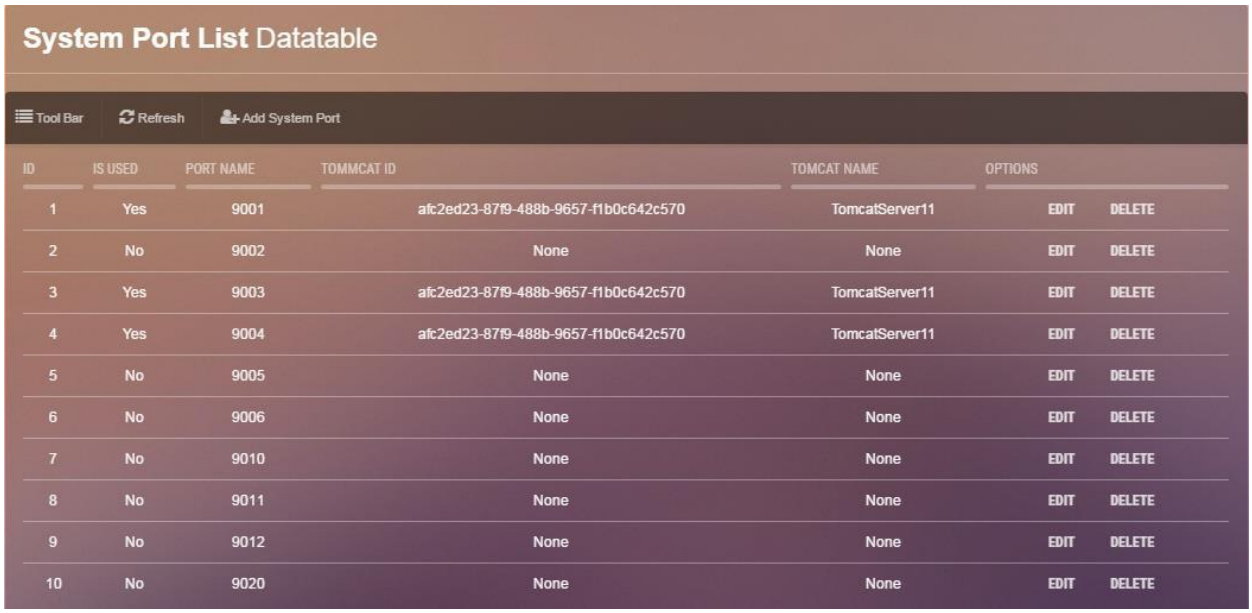

Рисунок 3.2 - Конфігурація системних портів

Конфігурація типів змінних (рисунок 3.3) призначена для створення баз даних. Клієнти можуть створювати бази даних через цей проект, і їм потрібні ці типи змінних для оголошення кожного елемента.

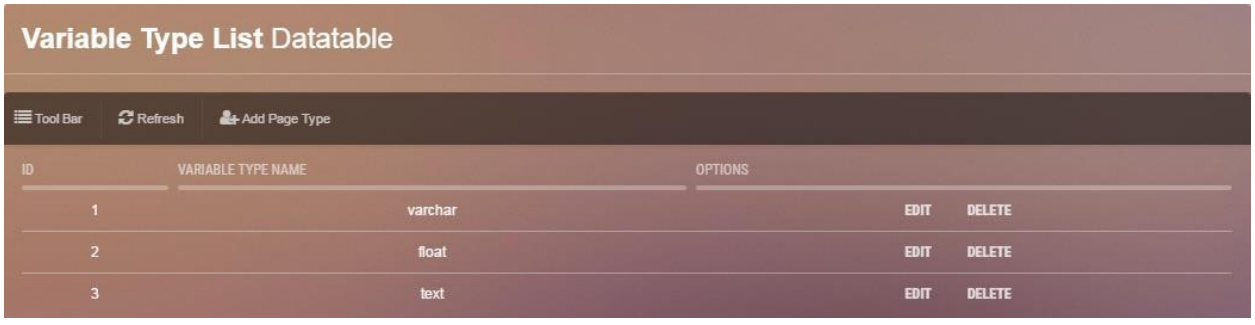

Рисунок 3.3 - Конфігурація типів змінних

Конфігурація типів сторінок (рисунок 3.4) призначена для створення веб-сторінок. Клієнти можуть створювати дерево проектних файлів з різними типами веб-файлів та папок, такими як JSP, js або CSS. Якщо файл є JSP, тоді клієнти можуть писати ці файли мовою JSP. Після того, як клієнти

розгортають свої проекти, усі файли будуть розгорнуті на їхні веб-сервіси tomcat.

Рисунок 3.4 - Конфігурація типів сторінок

Конфігурація типів завдань (рисунок 3.5) призначена для завдань BPMN. Клієнти можуть призначати різні типи завдань BPMN іншим клієнтам.

Рисунок 3.5 - Конфігурація типів завдань

3.2. Налаштування моделі клієнта

Щоб створити обліковий запис компанії, адміністратору даної системи потрібно використовувати конфігурацію штату та країни, що показано на рисунках 3.6 та 3.7.

Рисунок 3.6 - Конфігурація штату

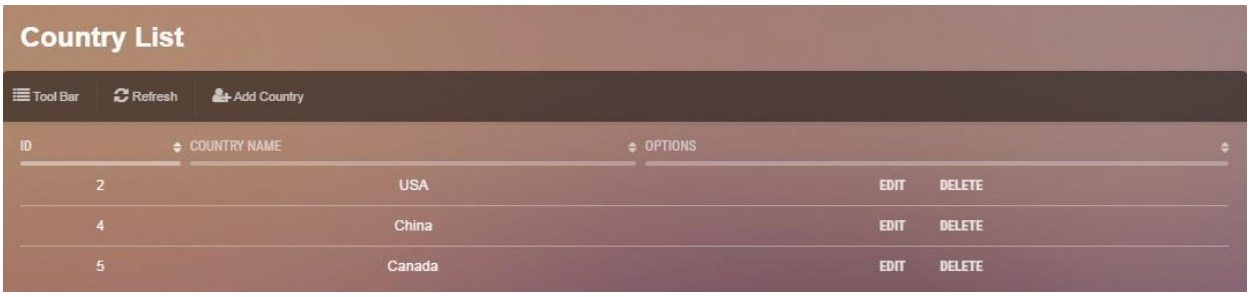

Рисунок 3.7 - Конфігурація країни

Конфігурація компанії (рисунок 3.8) використовується для розділення різних компаній. Коли корпоративні клієнти входять в систему, вони не бачать один одного.

Рисунок 3.8 - Конфігурація компанії

Як ми знаємо, кожна компанія має багато відділів. Кожен відділ має свої ролі. У цій системі кількість меню, до яких можуть отримати доступ клієнти, контролюється конфігурацією ролей (рисунок 3.9). Кожна компанія має системного адміністратора, який може отримувати доступ до ролей

(рисунок 3.9), відділів (рисунок 3.10) та меню (рисунок 3.11), але адміністратор може надавати тільки ті права, якими він володіє.

Рисунок 3.9 - Конфігурація ролей

Рисунок 3.10 - Конфігурація відділів

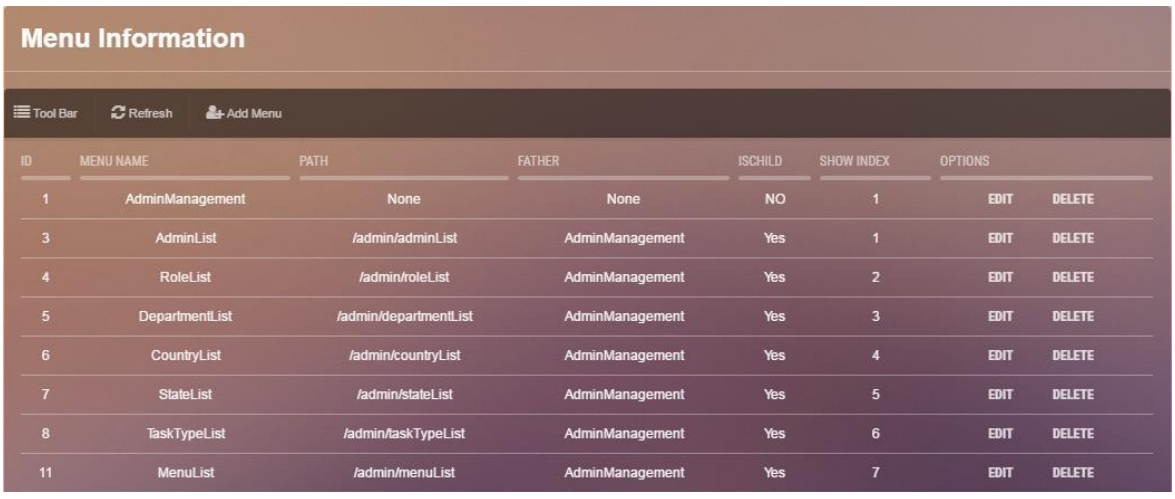

Рисунок 3.11 - Інформація про меню

3.3. Налаштування веб-сервісу Tomcat

Щоб додати веб-сервіси tomcat, клієнти повинні перейти на вкладку управління проектами. Під цією вкладкою клієнти можуть знайти список

веб-сервісів tomcat (рисунок 3.12). Щоб додати tomcat, клієнти повинні натиснути кнопку додавання tomcat на панелі інструментів відображення списку, потім клієнти побачать сторінку додавання tomcat (рисунок 3.13). Зараз їм потрібно ввести унікальну назву веб-сервісу tomcat, оскільки система перевірить, чи є назва унікальною.

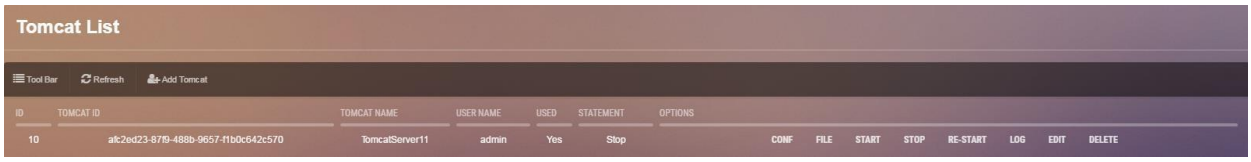

Рисунок 3.12 - Список веб-сервісів tomcat

Рисунок 3.13 - Сторінка додавання веб-сервісу tomcat

Коли клієнти успішно додають свій веб-сервіс tomcat, вони зможуть потрапити на сторінку конфігурації файлів tomcat (рисунок 3.14). Клієнти побачать список конфігураційних файлів і зможуть вибрати один з них для застосування до tomcat. Після цього клієнти побачать базову інформацію про конфігурацію tomcat (рисунок 3.15), включаючи порт сервера, порт HTTP, порт ALP та поточний стан конфігурації tomcat.

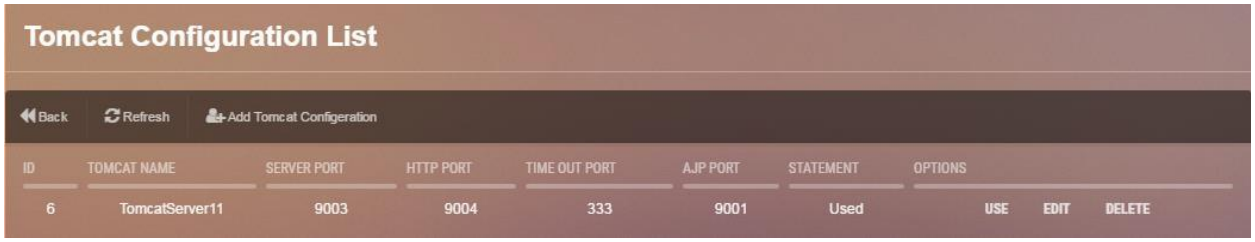

Рисунок 3.14 - Список конфігурацій tomcat

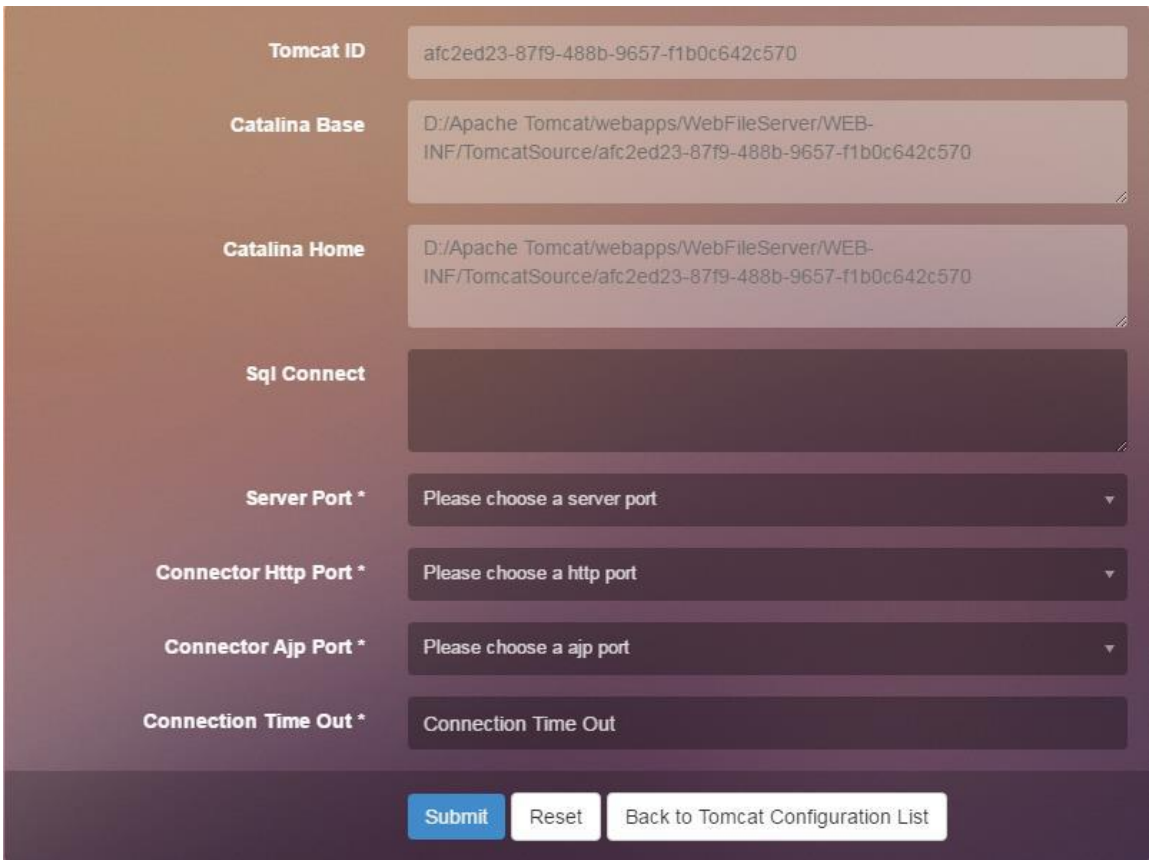

Рисунок 3.15 - Сторінка додавання веб-сервісу tomcat

Після створення клієнтських серверів tomcat клієнти можуть побачити дерево файлів tomcat на сторінці списку файлів tomcat (рисунок 3.16). Там вони також можуть побачити фізичний файл, як показано на рисунку 3.17.

Рисунок 3.16 - Список конфігурацій tomcat

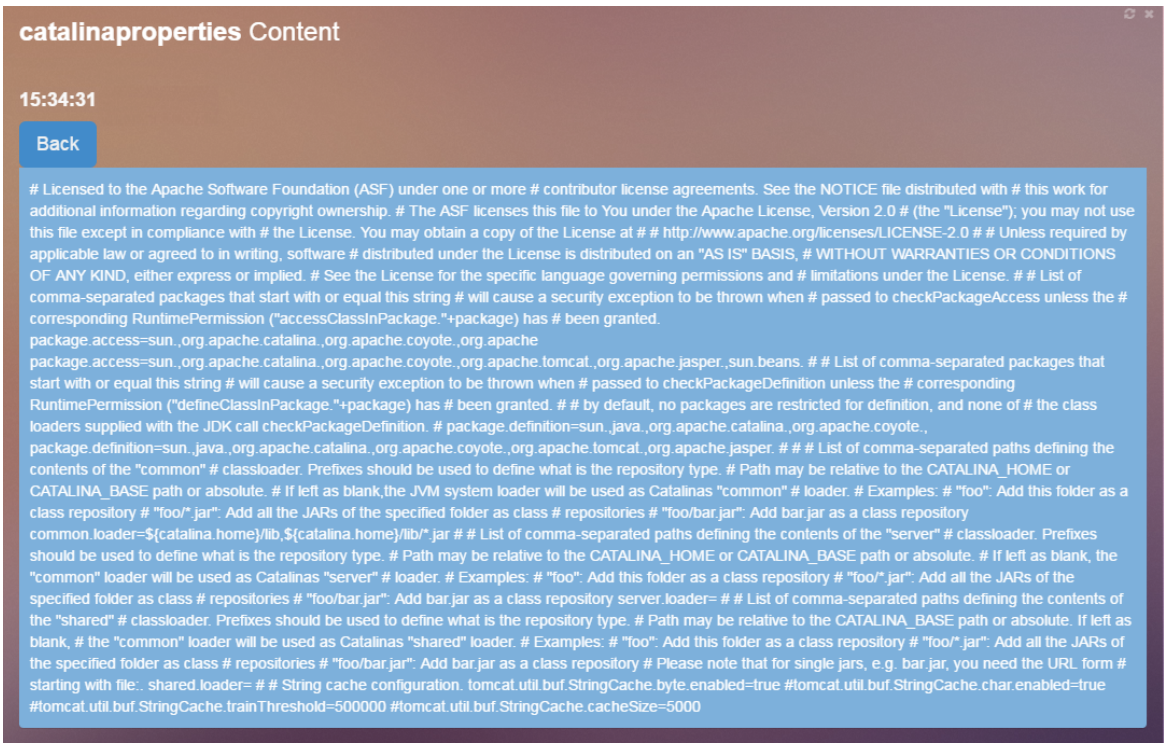

Рисунок 3.17 - Вміст файлу tomcat

Клієнти можуть налаштовувати файли та папки tomcat, як показано на рисунках 3.18 та 3.19. Клієнти також можуть додати додатковий jar-файл для запуску tomcat у папку lib для своїх проектів.

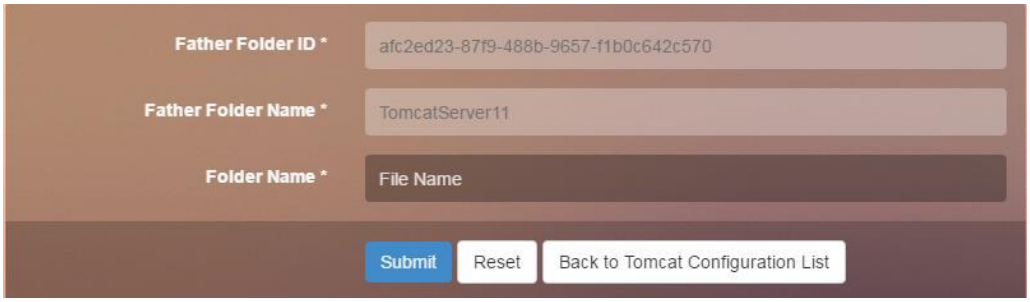

Рисунок 3.18 - Додавання папки tomcat

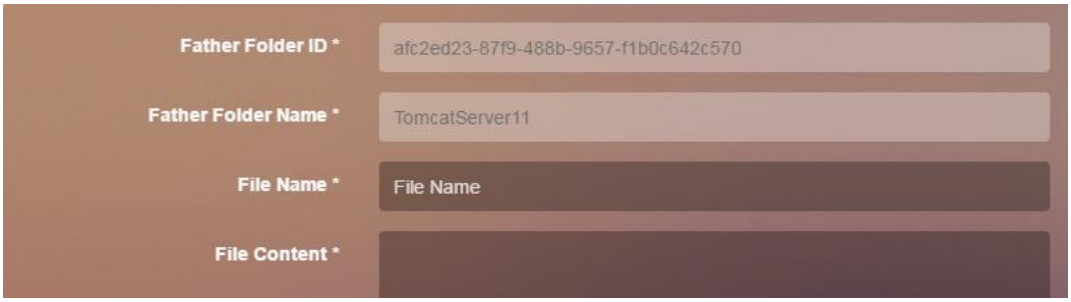

Рисунок 3.19 - Додавання файлу tomcat

Коли все налаштовано, клієнт може виконувати такі операції з tomcat, як запуск, зупинка та перезапуск на сторінці списку веб-сервісів tomcat.

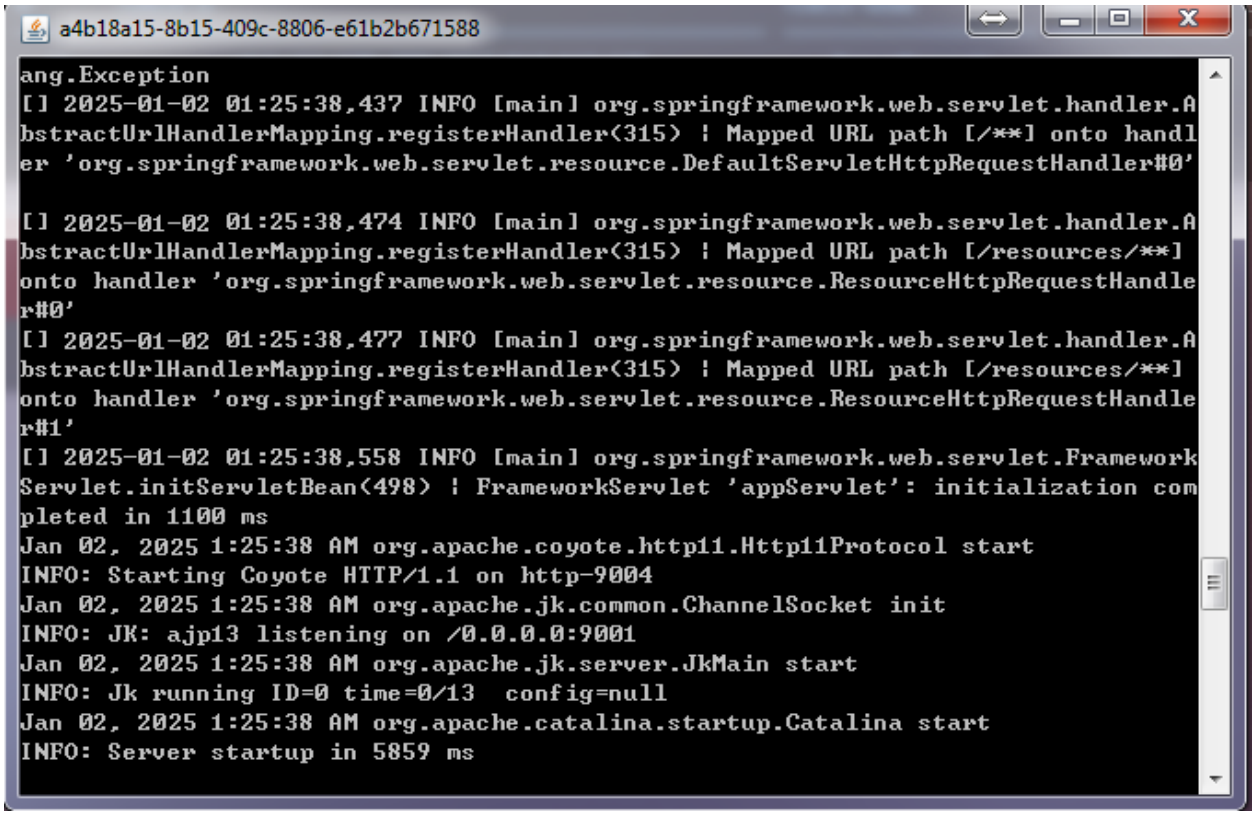

Рисунок 3.19 - Консоль веб-сервісу tomcat

3.4. Процес створення проекту

Під вкладкою управління проектами клієнти можуть додавати проекти до списку проектів (рисунок 3.20) після того, як вони створять веб-сервіси tomcat.

Рисунок 3.20 - Список проектів

Щоб додати проект, клієнти повинні ввести принаймні назву проекту та опис проекту, щоб клієнти могли написати деякі нотатки, щоб пояснити, що робить проект і що це таке.

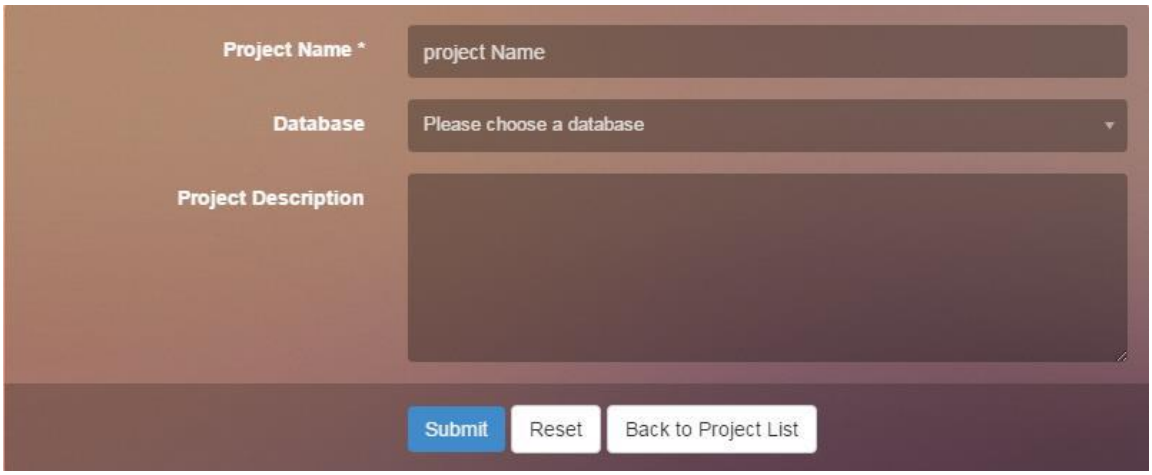

Рисунок 3.21 - Сторінка додавання проекту

Коли проект створено, клієнт може визначити, який тип проектного файлу він хоче створити. Якщо клієнт вибере створення проектного файлу, натиснувши кнопку додавання каталогу (рисунок 3.22), клієнт створить онлайн-кодування файлу. Якщо клієнт вибере кнопку завантаження каталогу (рисунок 3.22), клієнт завантажить файл проекту tomcat war у веб-сервіс tomcat і запустить цей проект.

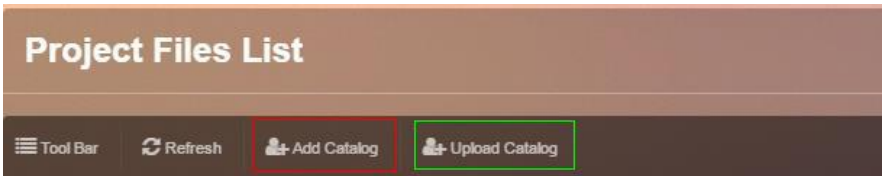

Рисунок 6.22. Два способи додавання файлів проекту

3.5. Розгортання проекту та запуск веб-сервісу

Коли клієнт завершив свій проектний файл, йому потрібно спочатку запустити проект. Запуск проекту (рисунок 3.24) означає, що проектні файли,

створені клієнтом онлайн, будуть записані на диск з бази даних, а потім клієнт повинен натиснути кнопку завантаження (рисунок 3.24), щоб завантажити ці файли на веб-сервіс tomcat. Потім клієнт може успішно запустити свій веб-сервіс tomcat.

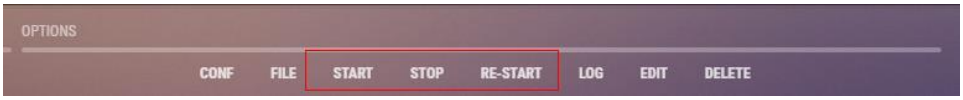

Рисунок 3.23 - Опції веб-сервісу tomcat

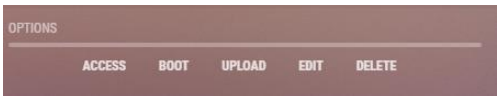

Рисунок 3.24 - Опції проекту

3.5.1. Розгортання потоку процесу BPMN

Щоб розгорнути файл процесу BPMN, клієнт повинен визначити, який спосіб він хоче використовувати. Я пропоную два способи розгортання файлу процесу BPMN. Під вкладкою управління проектами є два списки відображення BPMN. Перший список (рисунок 3.25) покаже клієнту внутрішній список файлів BPMN, які створені системою з вбудованим редактором файлів BPMN (рисунок 3.30). Інший список (рисунок 3.26) призначений для того, щоб клієнт міг завантажити та розгорнути файл процесу BPMN у двигун BPMN.

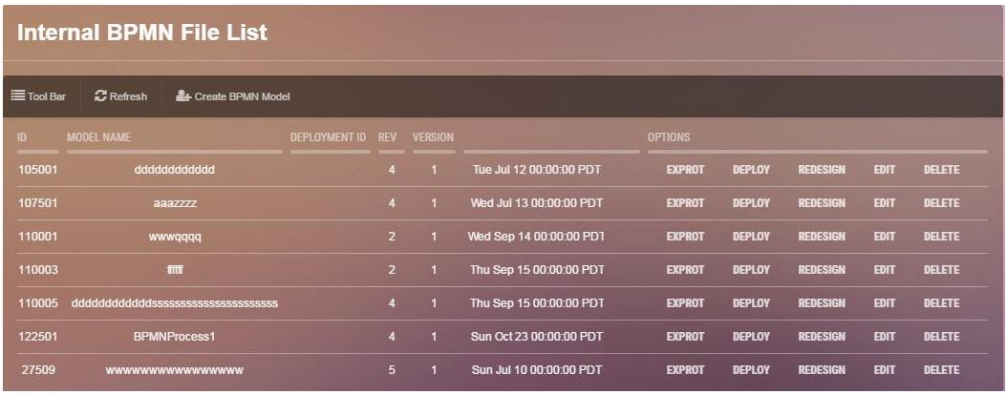

Рисунок 3.25 - Список процесів BPMN, створених на боці системи

Зовнішній BPMN буде відображатися у списку відображення зовнішнього BPMN (рисунок 3.27).

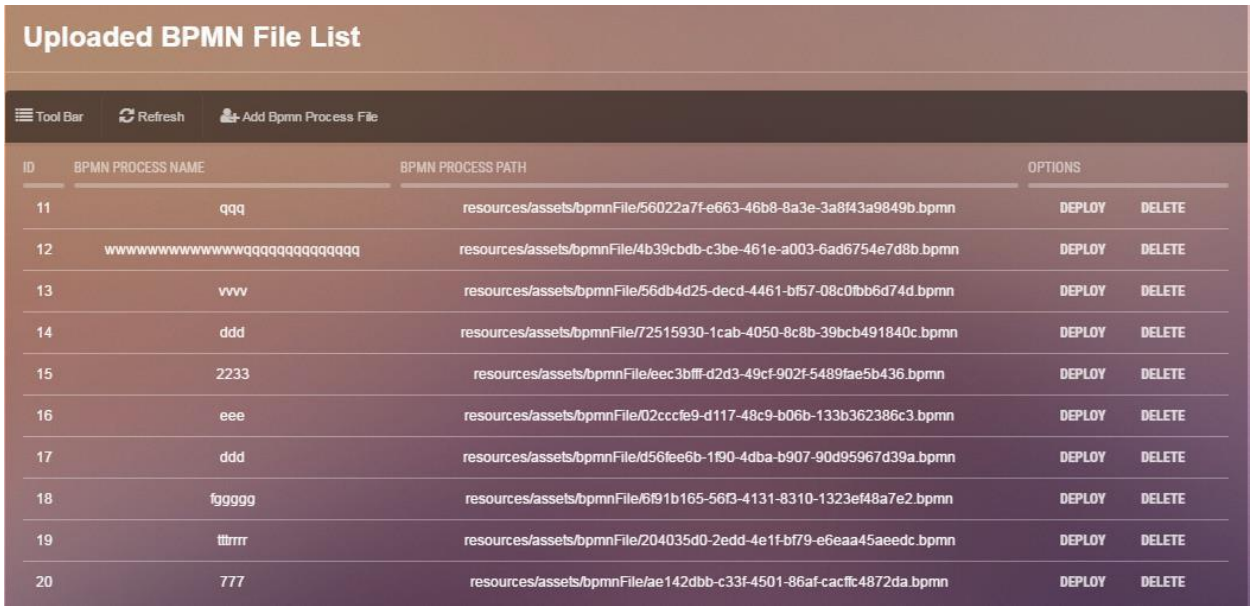

Рисунок 3.26 - Список завантажених файлів процесу BPMN

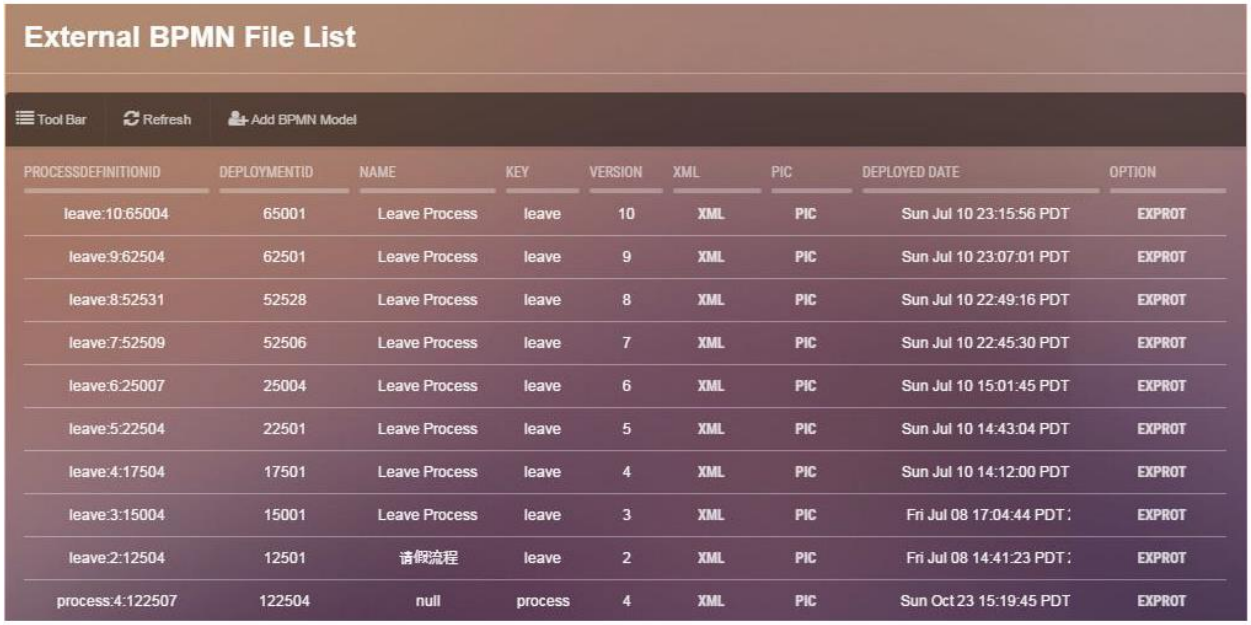

Рисунок 3.27 - Список розгорнутих завантажених файлів процесу BPMN

У списку відображення зовнішнього BPMN клієнт може переглянути файл процесу BPMN двома способами. Перший спосіб — побачити файл

процесу в форматі XML (рисунок 3.28). Інший спосіб — побачити файл процесу як зображення (рисунок 3.29).

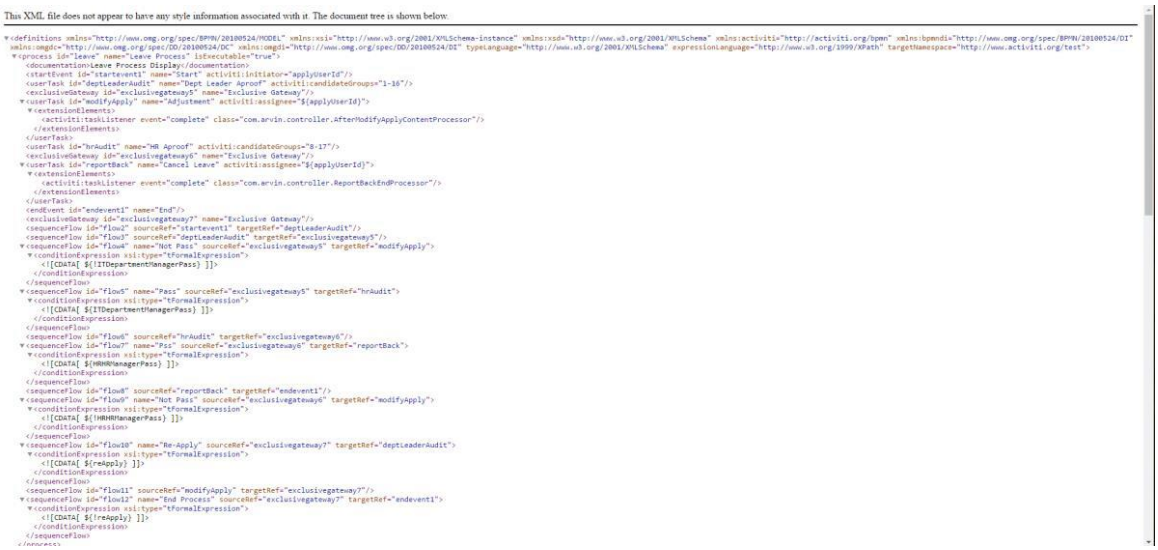

Рисунок 3.28 - Файл процесу BPMN у форматі XML

Цей процес BPMN для відпустки показує, якщо клієнти запитують відпустку, двигун BPMN запустить процеси та передасть завдання керівнику відділу. Якщо керівник відділу прийме ці запити, двигун передасть це завдання до відділу кадрів. Якщо керівник відділу кадрів також прийме ці запити, тоді запити будуть відправлені назад клієнтам. Якщо керівник відділу кадрів або керівник відділу не прийме ці запити, запити також будуть відправлені назад клієнтам, але клієнти можуть або відхилити ці запити, або зробити деякі коригування, щоб перезапустити ці процеси запитів.

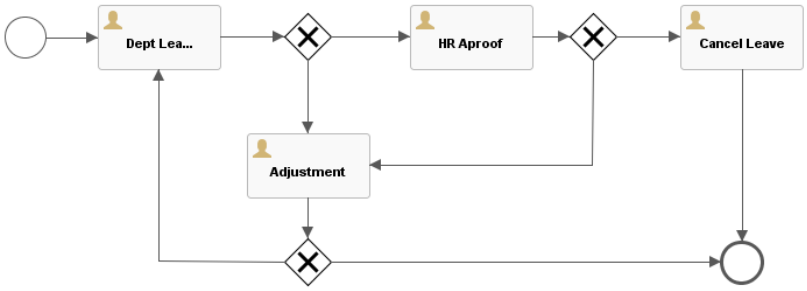

Рисунок 3.29 - Відображення процесу BPMN за допомогою зображення

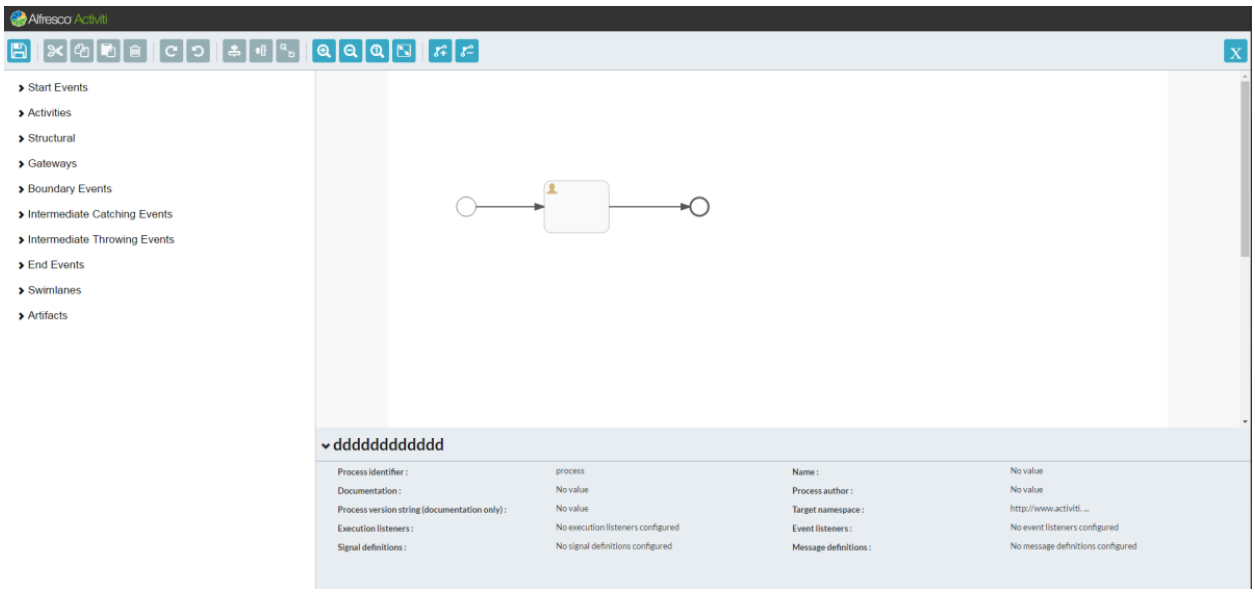

Рисунок 3.30 - Вбудований редактор процесу BPMN

3.5.2. Призначення завдань

Ця система пропонує компонент завдань, за допомогою якого клієнт може призначити завдання іншим клієнтам або запитати завдання, таке як запит на відпустку. Я спроектував компонент завдань, який містить два типи завдань. Перший тип — це завдання BPMN (рисунок 3.31), а другий — це загальне завдання (рисунки 3.35 та 3.36). Компонент завдань BPMN містить три списки даних: поточні завдання (рисунок 3.31), завдання в процесі (рисунок 3.32) та завершені завдання (рисунок 3.33).

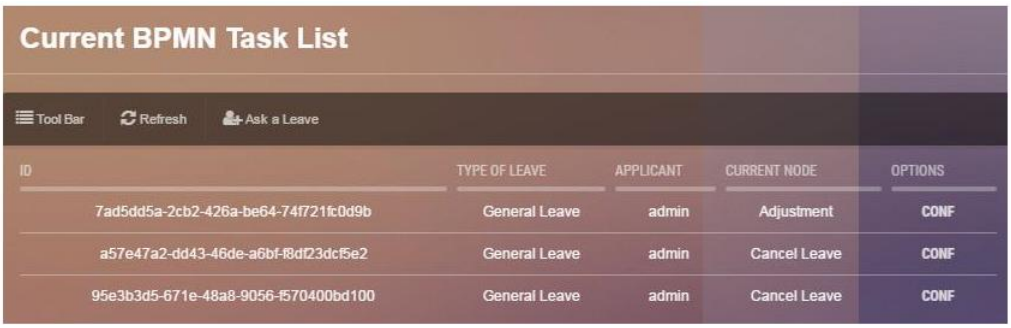

Рисунок 3.31 - Список поточних завдань процесу BPMN

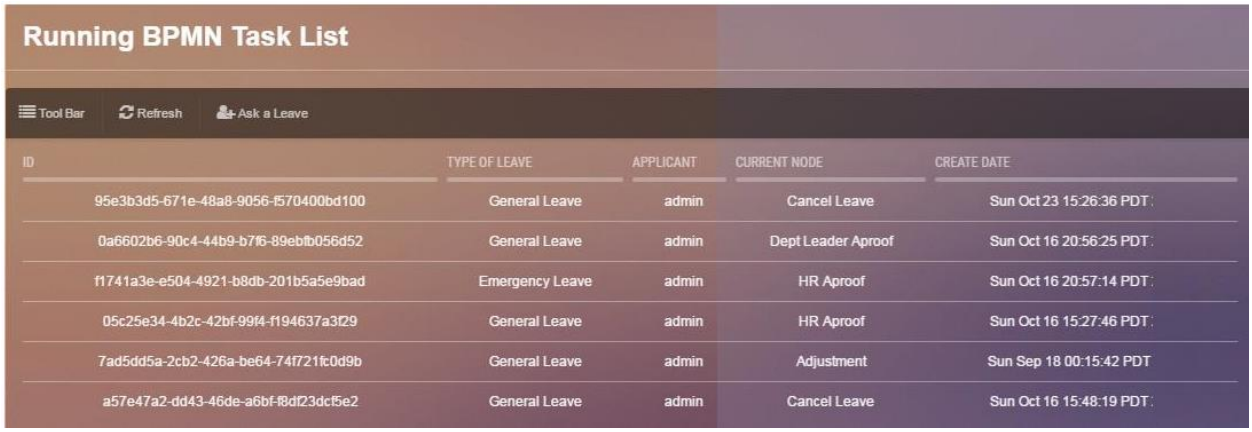

Рисунок 3.32 - Список завдань процесу BPMN в процесі

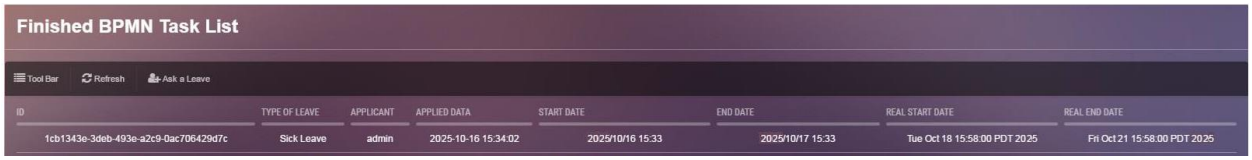

Рисунок 3.33 - Список завершених завдань процесу BPMN

Щоб додати завдання BPMN, клієнту потрібно заповнити форму завдання BPMN (рисунок 3.34). Форма включає деяку основну інформацію, що допоможе керівнику відділу та керівнику відділу кадрів визначити, чи потрібно відхилити запит, чи прийняти його.

Рисунок 3.34 - Сторінка додавання завдання BPMN

Щоб призначити типові завдання (рисунок 3.35), клієнти повинні мати високорівневі ролі, такі як керівник відділу. Після того, як клієнти

призначають типові завдання іншим клієнтам, отримувачі можуть знайти завдання або у панелі системних інструментів (рисунок 3.36), сторінці списку отриманих завдань (рисунок 3.37), або у списку справ (рисунок 3.38).

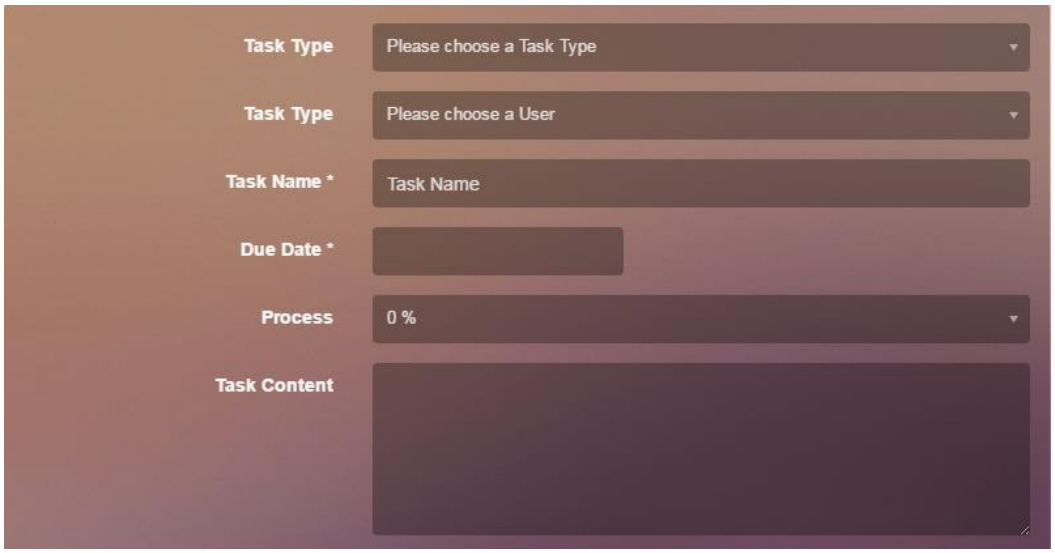

Рисунок 3.35 - Сторінка додавання загального завдання

Рисунок 3.36 - Список отриманих завдань

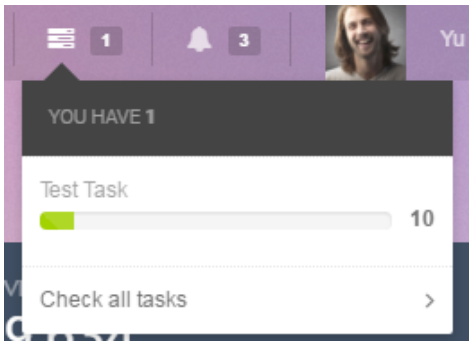

Рисунок 3.37 - Список завдань на панелі інструментів системи

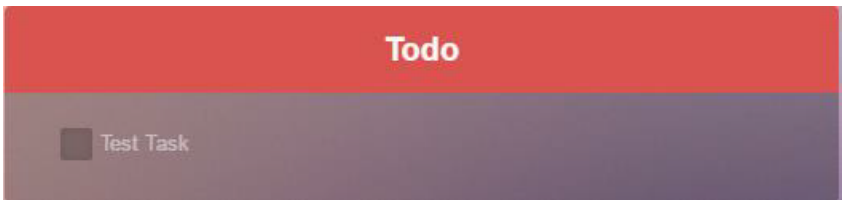

Рисунок 3.38 - Список завдань для виконання

Після того, як клієнт завершує типове завдання, він/вона може натиснути на це завдання, щоб позначити його як виконане, як показано на рисунку 3.39.

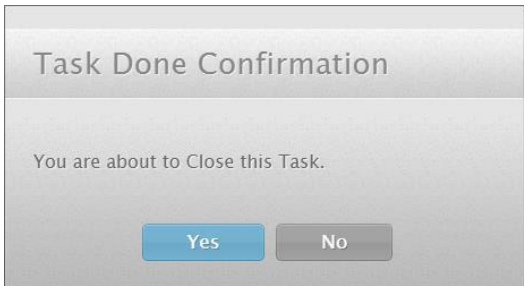

Рисунок 3.39 - Завершення завдання

3.5.3. Компонент надсилання повідомлень

Компонент повідомлень дозволяє клієнтам легше спілкуватися один з одним шляхом надсилання внутрішніх повідомлень. Потім отримувачі можуть знайти повідомлення на сторінці списку повідомлень (рисунок 3.40) або у панелі системних інструментів (рисунок 3.41).

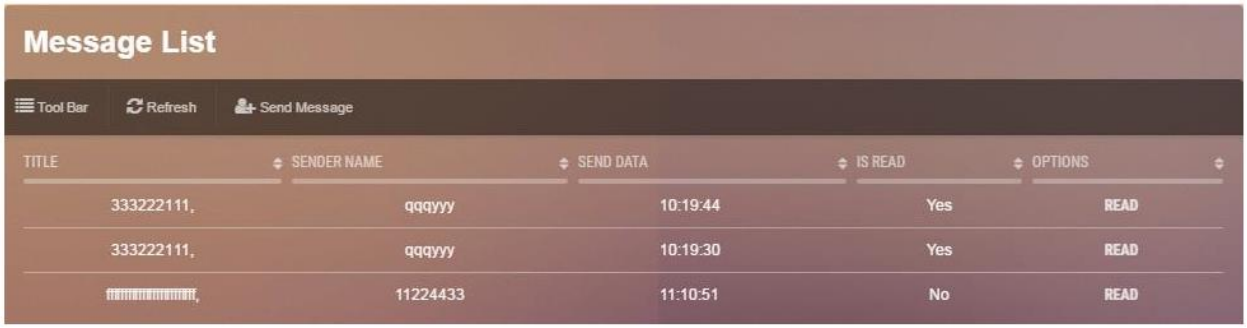

Рисунок 3.40 - Сторінка списку повідомлень

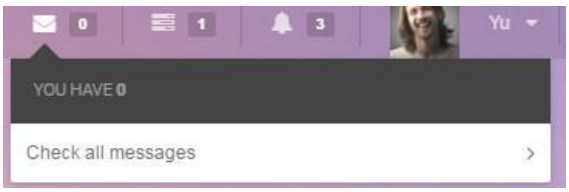

Рисунок 3.41 - Список повідомлень на панелі інструментів системи

3.6. Висновки до розділу

У третьому розділі було здійснено практичну реалізацію системи автоматизації процесу генерації веб-аплікацій та сторінок. Описано процес налаштування конфігурації системи, що забезпечує її гнучкість, адаптивність та можливість масштабування відповідно до вимог користувачів і середовища розгортання.

Було розроблено та налаштовано модель клієнта, яка забезпечує взаємодію між клієнтською та серверною частинами, а також підвищує ефективність обміну даними. Налаштування веб-сервера Tomcat дозволило створити стабільне середовище для роботи веб-сервісів та управління процесами розгортання застосунку.

Детально описано процес створення проєкту — від ініціалізації базової структури до підключення необхідних залежностей та модулів. Також розглянуто етап розгортання проєкту та запуску веб-сервісу, включно з розгортанням потоків процесів BPMN, що забезпечують автоматизацію бізнес-логіки всередині системи.

Окрема увага приділена механізмам призначення завдань користувачам у процесах BPMN та розробці компонента надсилання повідомлень, який підвищує оперативність взаємодії між учасниками процесу та системою.

У результаті програмної реалізації створено працездатну систему, що відповідає поставленим вимогам і забезпечує автоматизацію основних етапів створення веб-аплікацій, від генерації коду до організації життєвого циклу застосунків.

ВИСНОВКИ

В дипломній роботі розроблена система автоматичної генерації веб-аплікацій та система управління, що спрямована на вирішення актуальної задачі спрощення та автоматизації процесу створення веб-додатків. Досягнення цієї мети реалізовано шляхом застосування методології, орієнтованої на управління потоками, та використання стандартизованої технології BPMN (Business Process Model and Notation) для формалізації, автоматизації та моніторингу як процесів генерації вебзастосунків, так і логіки їхнього подальшого функціонування.

Архітектура розробленої системи є багатокомпонентною та охоплює ключові функціональні блоки, зокрема: підсистеми управління базою даних, взаємодії з веб-сервером, обробки HTML-контенту та реалізації функціональності, компонент фінансово-аналітичного моделювання, модуль клієнтського інтерфейсу та інтегроване ядро BPMN-рушія.

Реалізована функціональність системи охоплює такі основні аспекти:

1. Автоматизована генерація веб-аплікацій. Система надає користувачам можливість ініціювати створення веб-додатків шляхом вибору та налаштування попередньо визначених шаблонів. Інтегровані інструменти забезпечують автоматизацію значної частини етапів генерації, що суттєво скорочує час та ресурси, необхідні для запуску нового вебзастосунку.

2. Управління проектами та завданнями. Система надає користувачам інструментарій для ефективного управління проектами, пов'язаними зі створенням та супроводом вебзастосунків, а також для контролю виконання окремих завдань в рамках цих проектів через спеціалізовані інтерфейси відстеження статусу та прогресу.

3. Інтеграція з BPMN. Використання BPMN для моделювання та виконання робочих потоків (workflows) та логічних процесів забезпечує високий рівень гнучкості та адаптивності системи. BPMN-орієнтований

підхід дозволяє легко модифікувати бізнес-логіку та послідовність виконання завдань без безпосереднього втручання у програмний код.

4. Системна конфігурація та адміністрування. Система надає системним адміністраторам засоби для централізованого конфігурування ключових компонентів середовища виконання, включаючи параметри вебсерверів та баз даних, що забезпечує необхідну керованість, стабільність та надійність функціонування всієї платформи.

Успішна реалізація даного проекту демонструє досягнення поставлених цілей щодо оптимізації процесу створення веб-аплікацій. Розроблена система є функціональним рішенням, яке може бути застосоване для генерації різноманітних типів веб-аплікацій, значно мінімізуючи технічні зусилля з боку кінцевих користувачів. Наявність інтегрованих інструментів для управління проектами, завданнями та можливостей моніторингу процесів робить систему цінним та ефективним інструментом для суб'єктів, які шукають сучасні та ефективні рішення для автоматизації розробки та супроводу веб-аплікацій.

Подальший розвиток системи може бути зосереджений на розширенні бібліотеки доступних шаблонів та функціональних модулів, інтеграції з ширшим спектром зовнішніх сервісів та платформ, а також на вдосконаленні аналітичних можливостей та інструментів моніторингу для надання користувачам більш детальної інформації про процеси генерації та ефективність функціонування згенерованих застосунків.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Pressman, R. S., & Maxim, B. R. (2014). Software Engineering: A Practitioner's Approach (8th ed.). McGraw-Hill Education.
2. Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley.
3. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
4. Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice (3rd ed.). Addison-Wesley.
5. Larman, C. (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd ed.). Prentice Hall.
6. Apache Tomcat 11 (11.0.6) – Changelog - https://tomcat.apache.org/tomcat-11.0-doc/changelog.html#Tomcat_11.0.6_(markt)
7. Hughes, J., & Shoffner, M. (2002). Java 2 Web Developer Certification Study Guide. Sybex.
8. Reas, C., & Fry, B. (2007). Processing: A Programming Handbook for Visual Designers and Artists. MIT Press.
9. A. Leonard, JBoss Tools 3 Developers Guide, Packet Publishing Ltd, April 2009.
10. B. Laurie and P. Laurie, Apache the definitive Guide 3rd, O'REILLY, Dec. 2002.
11. B. Rücker, Real-Life BPMN: Using BPMN 2.0 to Analyze, Improve, and Automate Processes in Your Company, 2 edition, CreateSpace Independent Publishing Platform, Dec. 5, 2014.
12. B. Silver, Bpmn Method and Style, 2nd Edition, with Bpmn Implementer's Guide: A Structured Approach for Business Process Modeling and Implementation Using Bpmn 2, Cody-Cassidy Press, Oct. 17, 2011.

13. Kumar, A., & Singh, R. (2018). "Automated Web Application Generation Using Model-Driven Architecture." International Journal of Computer Applications, 179(7), 1–5.
14. Lee, S., & Kim, J. (2019). "A Framework for Automatic Generation of Web Applications Using UML." Journal of Systems and Software, 150, 1–15.
15. Chen, X., & Zhang, Y. (2020). "Automating Web Application Development with Model-Driven Engineering." Software: Practice and Experience, 50(4), 1–20.
16. Garcia, J., & Perez, L. (2017). "Template-Based Code Generation for Web Applications." Proceedings of the 2017 ACM Symposium on Applied Computing, 1234–1240.
17. Wang, H., & Li, M. (2016). "Automated Generation of Web Applications from Business Process Models." Information and Software Technology, 72, 1–12.
18. Zhou, Q., & Sun, Y. (2015). "A Model-Driven Approach to Web Application Development." Journal of Web Engineering, 14(3), 1–18.
19. García-Muñoz, J., & Gómez, J. (2019). "A Low-Code Approach for Model-Driven Web Development." Computer Standards & Interfaces, 64, 101–113.
20. Brambilla, M., Cabot, J., & Wimmer, M. (2017). Model-Driven Software Engineering in Practice (2nd ed.). Morgan & Claypool Publishers.
21. Ceri, S., Fraternali, P., & Bongio, A. (2000). "Web Modeling Language (WebML): A Modeling Language for Designing Web Sites." Computer Networks, 33(1–6), 137–157.
22. Kappel, G., Proll, B., Reich, S., & Retschitzegger, W. (2006). Web Engineering: The Discipline of Systematic Development of Web Applications. John Wiley & Sons.
23. Dolog, P., & Schäfer, M. (2010). "A Framework for Adaptable Web Applications Using Semantic Web Techniques." Web Engineering, 6189, 40–54.

24. Rossi, G., Pastor, O., Schwabe, D., & Olsina, L. (2007). Web Engineering: Modelling and Implementing Web Applications. Springer.
25. Lang, M., & Fitzgerald, B. (2005). "Hypermedia Affordances and the Role of Metadata in Web Application Development." Journal of Web Engineering, 4(1), 47–65.
26. Beck, K., & Andres, C. (2004). Extreme Programming Explained: Embrace Change (2nd ed.). Addison-Wesley.
27. Kumar, P., & Jaiswal, R. (2020). "Survey on Code Generation Techniques for Web Application Development." International Journal of Advanced Computer Science and Applications, 11(10), 156–163.
28. Nguyen, T., & Tran, D. (2018). "Automated Web Application Generation Using Domain-Specific Languages." Software and Systems Modeling, 17(2), 1–10.
29. Ahmed, S., & Khan, M. (2019). "A Survey on Model-Driven Web Application Development Approaches." ACM Computing Surveys, 52(3), 1–35.
30. Patel, R., & Shah, K. (2020). "Automated Code Generation for Web Applications Using XML and XSLT." International Journal of Advanced Computer Science and Applications, 11(5), 1–7.
31. Singh, P., & Verma, A. (2017). "Model-Driven Development of Web Applications: A Review." Journal of Software Engineering and Applications, 10(3), 1–10.
32. G. Zambon and M. Sekler, Beginning JSPTM, JSFTM, and Tomcat Web Development: From Novice to Professional, New York: Apress, 2007.
33. HTML & CSS: design and build websites, Indianapolis: John Wiley & Sons, Inc., 2014.
34. Ruby on Rails: Scaffolding and Code Generation. (n.d.). Retrieved from https://rubyonrails.org/

35. Django: The Web Framework for Perfectionists with Deadlines. (n.d.). Retrieved from https://www.djangoproject.com/
36. Laravel: The PHP Framework for Web Artisans. (n.d.). Retrieved from https://laravel.com/
37. ASP.NET Scaffolding: Code Generation Framework for ASP.NET Applications. (n.d.). Retrieved from https://docs.microsoft.com/en-us/aspnet/scaffolding/overview
38. N. S. Williams, Professional Java for Web Applications, John Wiley & Sons, Incorporated, February 2014.
39. J. Duckeet, JavaScript & JQuery internactive front-end web development, Indianapolis: John Wiley & Sons, Inc, 2014.

ДОДАТКИ

Додаток А

Конфігурація Tomcat

```
</developers>
<properties>
    <java-version>1.8</java-version>
    <jetty.version>8.1.15.v20140411</jetty.version>
    <org.springframework-version>4.0.3.RELEASE</org.springframework-version>
    <org.aspectj-version>1.7.4</org.aspectj-version>
    <org.slf4j-version>1.7.5</org.slf4j-version>
    <hibernate.version>4.3.5.Final</hibernate.version>
    <jackson.version>2.6.3</jackson.version>
    <activiti.version>5.19.0</activiti.version>
    <guava.version>17.0</guava.version>
    <commons-lang3.version>3.3.2</commons-lang3.version>
    <joda-time.version>2.1</joda-time.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<repositories>
    <repository>
        <id>Activiti</id>
        <url>https://maven.alfresco.com/nexus/content/groups/public</url>
    </repository>
    <repository>
        <id>Maven Central</id>
        <url>http://repo.maven.apache.org/maven2/</url>
    </repository>
    <repository>
        <id>nexus-osc</id>
        <url>http://maven.oschina.net/content/groups/public/</url>
    </repository>
    <repository>
        <id>nexus-osc-thirdparty</id>
        <url>http://maven.oschina.net/content/repositories/thirdparty/</url>
    </repository>
</repositories>
<build>
    <finalName>${project.artifactId}</finalName>
    <resources>
        <resource>
            <directory>src/main/resources</directory>
            <filtering>true</filtering>
        </resource>
    </resources>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.1</version>
            <configuration>
                <source>${java-version}</source>
                <target>${java-version}</target>
                <showWarnings>true</showWarnings>
            </configuration>
```

```
        <plugin>
            <groupId>org.apache.tomcat.maven</groupId>
            <artifactId>tomcat7-maven-plugin</artifactId>
            <version>2.2</version>
            <configuration>
                <server>apache-tomcat-7</server>
                <path>/${project.artifactId}</path>
                <url>http://localhost:8080/onlinewebsitegenerator</url>
                <systemProperties>
                    <JAVA_OPTS>-Xms1024m -Xmx2048m -XX:PermSize=512m -XX:MaxPermSize=1
                </systemProperties>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-war-plugin</artifactId>
            <version>2.4</version>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-eclipse-plugin</artifactId>
            <version>2.9</version>
            <configuration>
                <wtpversion>2.0</wtpversion>
                <additionalProjectnatures>
                    <projectnature>org.springframework.ide.eclipse.core.springnature</
                </additionalProjectnatures>
                <additionalBuildcommands>
                    <buildcommand>org.springframework.ide.eclipse.core.springbuilder</
                </additionalBuildcommands>
                <downloadSources>true</downloadSources>
                <downloadJavadocs>true</downloadJavadocs>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.mortbay.jetty</groupId>
            <artifactId>jetty-maven-plugin</artifactId>
            <version>${jetty.version}</version>
            <configuration>
                <webApp>
                    <contextPath>/${project.artifactId}</contextPath>
                </webApp>
                <stopKey>${project.artifactId}</stopKey>
                <stopPort>9999</stopPort>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-resources-plugin</artifactId>
            <version>2.6</version>
            <configuration>
                <nonFilteredFileExtensions>
                    <nonFilteredFileExtension>zip</nonFilteredFileExtension>
                    <nonFilteredFileExtension>bar</nonFilteredFileExtension>
                    <nonFilteredFileExtension>png</nonFilteredFileExtension>
                    <nonFilteredFileExtension>bpmn</nonFilteredFileExtension>
                </nonFilteredFileExtensions>
                <encoding>${project.build.sourceEncoding}</encoding>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Конфігурація веб-додатку

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
    <display-name>Online Website Generator Application</display-name>
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listene
    </listener>
    <listener>
        <listener-class>org.springframework.web.util.Log4jConfigListener</listener-cla
    </listener>
    <listener>
        <listener-class>com.arvin.sessionlistener.SessionListener</listener-class>
    </listener>
    <session-config>
        <session-timeout>30</session-timeout>
    </session-config>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            /WEB-INF/spring-dao.xml,
            /WEB-INF/spring-service.xml,
            /WEB-INF/spring-bpmn.xml
        </param-value>
    </context-param>
    <servlet>
        <servlet-name>appServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-clas
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring-servlet.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>appServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
    <context-param>
        <param-name>log4jConfigLocation</param-name>
        <param-value>/WEB-INF/log4j.properties</param-value>
    </context-param>
    <context-param>
        <param-name>log4jRefreshInterval</param-name>
        <param-value>60000</param-value>
    </context-param>
    <filter>
        <filter-name>CharacterEncodingFilter</filter-name>
        <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-c
        <init-param>
            <param-name>encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
        <init-param>
            <param-name>forceEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>
    <filter-mapping>
```

```
        <filter-name>CharacterEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <filter>
        <filter-name>springSecurityFilterChain</filter-name>
        <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-cla
    </filter>
    <filter-mapping>
        <filter-name>springSecurityFilterChain</filter-name>
        <url-pattern>/rest/*</url-pattern>
        <dispatcher>ERROR</dispatcher>
        <dispatcher>REQUEST</dispatcher>
    </filter-mapping>
    <filter>
        <filter-name>SessionFilter</filter-name>
        <filter-class>com.arvin.sessionlistener.SessionFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>SessionFilter</filter-name>
        <url-pattern>/*</url-pattern>
        <dispatcher>FORWARD</dispatcher>
        <dispatcher>REQUEST</dispatcher>
    </filter-mapping>
    <error-page>
        <error-code>404</error-code>
        <location>/WEB-INF/pages/page404.jsp</location>
    </error-page>
    <error-page>
        <error-code>500</error-code>
        <location>/WEB-INF/pages/page500.jsp</location>
    </error-page>
</web-app>
```

Конфігурація BPMN

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">
    <context:component-scan base-package="org.activiti.conf,org.activiti.rest.editor,c
        <context:exclude-filter type="annotation" expression="org.springframework.ster
    </context:component-scan>
    <bean id="objectMapper" class="com.fasterxml.jackson.databind.ObjectMapper"/>
    <bean id="uuidGenerator" class="org.activiti.engine.impl.persistence.StrongUuidGer
    <bean id="processEngine" class="org.activiti.spring.ProcessEngineFactoryBean">
        <property name="processEngineConfiguration" ref="processEngineConfiguration" /
    </bean>
    <bean id="processEngineConfiguration" class="org.activiti.spring.SpringProcessEngi
        <property name="dataSource" ref="dataSource" />
        <property name="databaseSchemaUpdate" value="true" />
        <property name="jobExecutorActivate" value="false" />
        <property name="history" value="full" />
        <property name="transactionManager" ref="transactionManager" />
        <property name="processDefinitionCacheLimit" value="10" />
    </bean>
    <bean id="repositoryService" factory-bean="processEngine" factory-method="getRepos
    <bean id="runtimeService" factory-bean="processEngine" factory-method="getRuntimeS
    <bean id="taskService" factory-bean="processEngine" factory-method="getTaskService
    <bean id="historyService" factory-bean="processEngine" factory-method="getHistoryS
    <bean id="managementService" factory-bean="processEngine" factory-method="getManag
    <bean id="IdentityService" factory-bean="processEngine" factory-method="getIdentit
    <bean id="formService" factory-bean="processEngine" factory-method="getFormService
    <bean id="restResponseFactory" class="org.activiti.rest.service.api.RestResponseFa
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: “ Розробка методології автоматизації процесу генерації веб-аплікацій та сторінок ”

Обсяг пояснювальної записки: __75__ аркушів.

Дата закінчення роботи: 10 червня 2025 р.

Підпис студента _____