Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

ВЕБ-ЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ ТА ОЦІНЮВАННЯ НАВЧАЛЬНОГО ПРОЦЕСУ З ВИКОРИСТАННЯМ ХМАРНИХ СЕРВІСІВ

Виконав: студент 2 курсу, групи 2КІ-22м
спеціальності 123 — «Комп'ютерна інженерія»
_____ Терещук Е. П.

Керівник: к.т.н., доц.каф. ОТ
_____ Тарновський М. Г.
«08» 12 2023 р.

Опонент: к.т.н., доц. каф. ПЗ
_____ Ткаченко О. М.
«11» 12 2023 р.

Допущено до захисту

Завідувач кафедри ОТ
_____д.т.н., проф. Азаров О. Д.
«14» 12 2023 р

ВНТУ 2023

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань — Інформаційні технології
Освітній рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія
Освітньо-професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

О.Д. Азаров

"26" вересня 2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Терещуку Едуарду Петровичу

1 Тема роботи «Веб-застосунок для організації та оцінювання навчального процесу з використанням хмарних сервісів» керівник роботи Тарновський Микола Геннадійович к.т.н., доцент, затверджено наказом вищого навчального закладу від **18.09.23** року **№ 247.**

2 Строк подання студентом роботи **13.12.23**.

3 Вихідні дані до роботи: призначення керування даними користувача в мікросервісному середовищі, засоби — середовище програмування IntelleJ IDEA, VS Code, мови програмування Java, JavaScript, фреймворк Spring та бібліотеки Junit, Hibernate, React.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз сучасних методів та технологій, аналіз та обгрунтування вибору технологій, розробка веб-застосунку, тестування веб-застосунку.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): блок-схема архітектури застосунку, структурна схема бази даних.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1— Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Тарновський Микола Геннадійович к.т.н., доцент	19.09.2023р	13.12.2023р
5	Небава Микола Іванович проф., к.е.н	1.11.2023	13.12.2023

7 Дата видачі завдання **19.09.2023** .

8 Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Підпис
1	Постановка задачі	**19.09.23**	
2	Огляд існуючих рішень	**21.09.23**	
3	Розробка блок-схеми	**27.09.23**	
5	Розрахунок аналогової частини	**4.10.23**	
6	Вибір ПЗ для розробки	**13.10.23**	
7	Розробка роботи застосунку	**28.10.23**	
8	Розрахунок економічної частини	**2.11.23**	
9	Оформлення пояснювальної записки та ілюстративного матеріалу	**14.11.23**	
10	Виконання магістерської кваліфікаційної роботи	**26.11.23**	
11	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	**5.12.23**	
12	Підписи супроводжувальних документів у керівника, опонента, нормоконтролера	**9.12.23**	
13	Перевірка «антиплагіат»	**12.12.23**	
14	Попередній захист	**14.12.23**	

Студент Терещук Едуард Петрович

Керівник к.т.н., доц. Тарновський Микола Геннадійович

АНОТАЦІЯ

УДК 004

Терещук Е. П. Веб-застосунок для організації та оцінювання навчального процесу з використанням хмарних сервісів. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп’ютерна Інженерія, Вінниця: ВНТУ, 2023 — 124 с. На укр. мові. Бібліогр.: 13 назв; рис.: 37; табл. 7.

Дана магістерська робота присвячена розробці веб-застосунку для організації та оцінювання навчального процесу на основі хмарних технологій. В роботі був проведений аналіз відомих технологій та методів створення програмного забезпечення, розроблено архітектуру веб-застосунку, базу даних та проаналізовано взаємодію веб-застосунку з хмарними сервісами.

Ключові слова: веб-застосунок, навчальний процес, технології, онлайн-освіта, хмарні технології, хмарні сервіси.

ABSTRACT

УДК 004

Tereshchuk E. P. Web Application for Organization and Evaluation of the Educational Process Using Cloud Services. Master's Qualification Paper in the field of 123 — Computer Engineering, Vinnytsia: VNTU, 2023 — 124 p. In Ukrainian. Bibliography: 13 titles; figures: 37; tables: 7.

This master's thesis is dedicated to the development of a web application for organizing and evaluating the educational process based on cloud technologies. The paper includes an analysis of known technologies and methods for software development, the design of the web application architecture, the database, and an analysis of the interaction of the web application with cloud services.

Keywords: web application, educational process, technologies, online education, cloud technologies, cloud services.

ЗМІСТ

ВСТУП

Сучасна освіта переживає значний розвиток і зміни під впливом технологій та вимог сучасного світу. Характерною ознакою сьогодення стало активне впровадження цифрових інструментів та хмарних сервісів у навчальний процес. Відповідно до цього сучасні цифрові технології вносять значні корективи у методики та методи навчання, створюючи нове інтерактивне освітнє середовище, насичене різноманітною електронною інформацією.

Останні тенденції інноваційних змін у сфері освіти пов’язані з використанням хмарних технологій, які дозволяють зменшити витрати на обладнання та інфраструктуру, а також забезпечують безпеку та резервне копіювання даних. Саме хмарні технології надають можливості повноцінного доступу до високотехнологічної освіти та дозволяють вирішити проблему зберігання даних. Інформація, що зберігається на хмарних серверах, є доступною з будь-якого пристрою та з будь-якої точки світу.

Актуальність теми роботи полягає в тому, що набуває особливого значення пошук шляхів вдосконалення існуючих засобів для організації, інформаційної підтримки та інформаційного забезпечення навчального процесу, підвищення ефективності навчання, полегшення взаємодії між викладачами та студентами, а також збільшення доступності навчальних ресурсів завдяки використанню хмарних сервісів.

У результаті отримується можливість надати викладачам ефективні інструменти для створення та спільно редагування навчальних матеріалів, встановлення завдань для студентів, здійснення контролю за їх виконанням, оцінювання їх результатів. Студенти отримують можливість доступу до навчальних матеріалів і завдань з будь-якого місця у будь-який зручний час, а також засобі для взаємодії з викладачами та однокурсниками.

Забезпечення підтримки хмарних технологій, які відкривають широкі можливості для забезпечення доступності та ефективності освітнього

процесу, дозволяє зберігати, обмінюватися та обробляти дані в онлайн-режимі, забезпечити велику гнучкість та масштабованість системи тощо. Крім того, інтеграція з хмарними сервісами дозволить швидко впроваджувати оновлення та вдосконалення системи без значних затрат часу та ресурсів. Це наддасть можливість підтримувати систему на актуальному рівні та реагувати на потреби користувачів.

Дана робота розглядається як важливий внесок у сучасну освіту, що сприятиме підвищенню якості навчання та розвитку учасників освітнього процесу. Прогнозується, що у надалі отримані напрацювання можуть стати платформою для подальших досліджень та розробок у галузі інноваційних засобів навчання.

Об'єктом дослідження є процес організації навчання в хмарних середовищах.

Предметом дослідження є архітектура програмного забезпечення для використання в хмарних середовищах.

Метою магістерської кваліфікаційної роботи є вдосконалення веб-застосунку для організації та оцінювання навчального процесу за рахунок використання хмарних технологій, які автоматично масштабують застосунок в залежності від навантаження на нього, що дозволить забезпечити оптимальну продуктивність та доступність для користувачів навіть у періоди збільшеного навантаження.

Для досягнення поставленої мети у роботі були розв'язані такі задачі:

— проведено аналіз аналогічних застосунків;

— сформульовані основні вимоги до веб-застосунку;

— обрано архітектуру, технології та сервіси для розробки веб-застосунку;

— спроектовано інтерфейс та функціонал.

Новизна роботи полягає у використанні замість традиційної мікросервісної архітектури, що надає можливість створювати окремі

мікросервіси, кожен з яких відповідає за конкретні функції, які взаємодіють між собою за допомогою постачальників повідомлень.

Практична цінність роботи полягає у тому, що вона може бути використана для розміщення у хмарних середовищах програмного забезпечення усіх навчальних закладах.

Апробація результатів роботи здійснена у доповідях на LI Науково-технічній конференції підрозділів Вінницького національного технічного університету (2022) та Молодь в науці: дослідження, проблеми, перспективи підрозділів Вінницького національного технічного університету (2023) [1].

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ТА ТЕХНОЛОГІЙ

1.1 Визначення веб-застосунку та навчальних платформ

Веб-застосунок — це програмне забезпечення, яке доступне через веб-браузер і фокусується на виконанні певних функцій або завдань. Він може бути частиною навчальної платформи, або окремим інструментом. В свою чергу платформа онлайн-навчання представляє з себе веб-простір, або портал для освітнього контенту та ресурсів, який пропонує студенту все необхідне в одному місці: лекції, ресурси, можливості зустрітися та поспілкуватися з іншими студентами тощо. Водночас з цим навчальна платформа є також чудовим способом для викладача контролювати прогрес студентів [2].

Ця динамічна електронна система спрямована на покращення якості навчання та доступності освіти в сучасному світі. Її визначають як інтегроване середовище, яке об'єднує в собі навчальний контент, інтерактивні можливості та інструменти для взаємодії між учасниками освітнього процесу [2].

Цілком зрозуміло, що навчальна платформа та веб-застосунок в свою чергу мають на увазі не лише технічний аспект, але й важливий соціальний аспект. Вони створюють віртуальну спільноту студентів та викладачів, яка може забезпечити підтримку, обмін знаннями та співпрацю навіть на віддалі.

Кожному, хто хоче створювати та проводити онлайн-курси або використовувати eLearning для освіти та навчання, потрібна навчальна платформа, або веб-застосунок для організації навчального процесу. Сюди входять викладачі, інструктори, корпоративні відділи кадрів і навіть малі підприємства, які хочуть запропонувати деяку базову електронну освіту своїм співробітникам [2].

Сучасне освітнє середовище характеризується швидкими змінами та високими вимогами до якості навчання. У цьому контексті навчальні платформи виконують низку важливих ролей, а саме: доступність освіти, підтримка різних видів навчання, підвищення якості навчання, підтримка

самонавчання та забезпечення доступу до актуальної інформації. Розглянемо кожну з них детальніше.

Доступність освіти. Вони допомагають подолати географічні обмеження та роблять навчання доступним для студентів з різних куточків світу;

Підтримка різних видів навчання. Вони адаптовані для різних форм навчання, включаючи дистанційне навчання, онлайн-курси, бліндоване навчання та інші. Це робить їх універсальним інструментом для навчання на будь-якому рівні та в будь-якому форматі.

Підвищення якості навчання. навчальні платформи можуть включати інтерактивні елементи, мультимедійний контент та інструменти для залучення студентів. Це сприяє активному навчанню, покращенню розуміння та засвоєння матеріалу.

Підтримка самонавчання. Вони надають можливість студентам самостійно керувати своїм навчанням, вибирати теми, які їх цікавлять, та працювати в зручному для них темпі.

Забезпечення доступу до актуальної інформації. Навчальні платформи часто оновлюються з огляду на останні тенденції та вимоги ринку праці, що допомагає студентам набувати актуальні знання та навички.

Усі ці аспекти демонструють важливість навчальних платформ у сучасному освітньому ландшафті.

1.2 Типи та класифікація навчальних платформ

Розглянемо різні типи навчальних платформ та їх класифікацію з урахуванням призначення, функціональності та специфіки.

Навчальні платформи за призначенням розділяють на платформи для вищої освіти та корпоративного навчання. Навчальні платформи, що націлені для вищої освіти спеціалізуються на підтримці навчання в університетах, коледжах та інших вищих навчальних закладах. Вони надають можливість викладачам створювати та керувати курсами для студентів, ведення

реєстрації та оцінювання. В той час коли платформи для корпоративного навчання призначені для навчання співробітників в корпоративних середовищах, вони забезпечують навчання згідно з потребами компанії, надають інструменти для створення корпоративних курсів та відстеження прогресу працівників.

За функціональністю навчальні платформи поділяють на лекційні платформи, що спрямовані на представлення навчального матеріалу у формі відеолекцій, текстових матеріалів та тестів, що часто використовуються в університетах та онлайн-курсах та на інтерактивні навчальні платформи, що надають можливість створювати інтерактивні завдання, відкриті дискусії, взаємодію між учасниками та збір та аналіз даних про навчальний процес [2].

Також за своєю специфікою платформи поділяють на мовні, технічні та платформи для осіб з особливими потребами.

Мовні навчальні платформи, це платформи, що спеціалізуються на навчанні певної мови або групи мов, вони можуть надавати вправи для практики граматики, словникового запасу, навичок мовлення.

Технічні навчальні платформи призначені для навчання технічних навичок та професій, також можуть включати в себе симуляції, віртуальні лабораторії та інші інтерактивні інструменти для навчання.

В той час, навчальні платформи для осіб з особливими потребами спрямовані на надання доступної освіти для людей з різними видами обмежень або особливими потребами.

Класифікація навчальних платформ за цими критеріями допомагає визначити, яка платформа найкраще відповідає конкретним потребам та цілям освіти чи навчання.

1.3 Переваги та недоліки використання навчальних платформ

Навчальні платформи пропонують численні переваги для студентів, викладачів та навчальних установ, а саме:

— глобальний доступ;

— гнучкість у часі;
— персоналізація;
— самонавчання;
— залучення;
— ефективність навчання;
— моніторинг прогресу;
— доступ до різноманітних ресурсів;
— актуальність інформації.

Розглянемо кожну із переваг детальніше.

Навчальні платформи забезпечують доступ до освіти для студентів з усього світу, долаючи географічні обмеження, що є особливо корисно для дистанційного та міжнародного навчання. Студенти можуть навчатися у зручний для них час, дозволяючи працювати або вирішувати інші обов'язки та одночасно отримувати освіту [2].

Навчальні платформи дозволяють студентам вибирати курси та матеріали, які відповідають їхнім інтересам і потребам. Вони можуть вибирати теми, темп та стиль навчання. За допомогою навчальних платформ, студенти розвивають навички самостійного навчання, що важливо в сучасному світі з постійними змінами та вимогами до навчання.

Також використання мультимедійних ресурсів та інтерактивних інструментів допомагає залучити студентів до навчального процесу та покращити їхнє розуміння матеріалу. Платформи дозволяють викладачам створювати та адаптувати навчальний матеріал для підвищення ефективності навчання та зрозуміння студентами.

До переваг моніторингу належить можливість викладачів стежити за прогресом студентів, надавати зворотний зв'язок та адаптувати навчальний процес для кращих результатів.

Переваги навчальних платформ роблять їх потужним інструментом для забезпечення якісної освіти та сприяють активному розвитку навчальних процесів, але водночас вони мають і ряд мінусів серед яких:

— відсутність фізичного контакту;
— потреба у самодисципліні;
— технічні проблеми;
— відсутність особистого підходу до студента.

1.4 Тенденції та інновації у розвитку навчальних платформ та веб-застосунків

Сфера електронного навчання стала свідком видатної еволюції, спричиненої постійним прогресом технологій і зміною уподобань учнів. Від перших днів базових онлайн-курсів до інтеграції мультимедійних елементів, інтерактивного моделювання та гейміфікації, електронне навчання стало більш привабливим і захоплюючим. Мобільне навчання та мікронавчання також набули популярності, дозволяючи учням отримувати доступ до невеликого вмісту на ходу [3].

Крім того, розвиток штучного інтелекту, віртуальної реальності та доповненої реальності революціонізував процес навчання, надаючи персоналізовані та інтерактивні освітні можливості.

На даний момент у світі досить інноваційним підходом у багатьох сферах стало використання штучного інтелекту (AI) та машинного навчання (ML), і ця тенденція не оминула й сферу онлайн-навчання. Використання AI та ML в навчальних веб-застосунках революціонізує спосіб, яким студенти отримують освіту та як викладачі керують навчанням .

Основні переваги використання штучного інтелекту в онлайн-навчанні включають можливість створювати персоналізовані програми навчання, які відповідають індивідуальним потребам та рівню підготовки студентів. AI може аналізувати дані про вивчений матеріал, виконані завдання та результати тестів для визначення слабких місць та надавати рекомендації щодо подальшого навчання.

Крім того, веб-застосунки, які використовують AI, допомагають викладачам автоматизувати процес оцінювання та звітності. Вони можуть

автоматично генерувати звіти про прогрес студентів, відстежувати їхні досягнення та вчасно ідентифікувати проблеми [4].

Машинне навчання також дозволяє передбачати потреби студентів і підтримувати їхнє навчання в реальному часі. Завдяки аналізу великих обсягів даних, AI може прогнозувати, які ресурси та підходи до навчання найбільше підходять кожному студенту [4].

Отже, використання штучного інтелекту та машинного навчання у веб-застосунках для онлайн-навчання розширює можливості навчання, роблячи його більш ефективним, інтерактивним та індивідуалізованим для студентів.

Наступною іновацією у сфері онлайн навчання виступає розширена реальність (AR) та віртуальна реальність (VR). AR та VR технології використовуються для створення іммерсивних навчальних середовищ, які дозволяють студентам взаємодіяти з матеріалом іншим способом. Вони особливо корисні для навчання у віртуальних лабораторіях, музеях або історичних симуляціях [5].

AR доповнює реальний світ віртуальними об'єктами та інформацією, яка відображається на екрані смартфона, планшета або спеціальних AR-пристроях. У контексті навчання AR може створювати інтерактивні сценарії, де студенти можуть переглядати і взаємодіяти з віртуальними об'єктами, діаграмами або 3D-моделями, щоб краще зрозуміти складні концепції.

Наприклад, в анатомічному навчанні AR може дозволити студентам досліджувати внутрішні органи, переміщаючись навколо їхніх віртуальних моделей та отримуючи детальну інформацію [6].

В свою чергу, VR занурює користувачів у повністю віртуальну обстановку, яка може бути створена для навчання в будь-якому контексті. Це особливо корисно для симуляцій та навчання, які потребують взаємодії в умовах, які важко відтворити в реальному світі.

Наприклад, у медичному навчанні VR може надати можливість студентам виконувати хірургічні операції у віртуальному середовищі, де

вони можуть вправно вчитися та вдосконалювати свої навички до того, як вони зустрінуться з реальними пацієнтами.

Обидві ці технології допомагають зробити навчання більш доступним, цікавим та практичним. Вони сприяють активній участі студентів, дозволяючи їм навчатися шляхом дослідження та експериментування, а також відкривають нові можливості для навчання у віртуальних лабораторіях, історичних симуляціях та інших іммерсивних середовищах [6].

1.5 Впровадження хмарних сервісів у розробці веб-застосунків

Хмарні обчислення — це новий метод додавання можливостей до комп'ютера без ліцензування нового програмного забезпечення, інвестування в нове обладнання чи інфраструктуру чи навчання нового персоналу. Програми купуються, ліцензуються та запускаються через мережу замість робочого столу користувача. Він надає звичайні онлайн-програми для бізнесу, доступ до яких здійснюється з веб-браузера, а програмне забезпечення та дані зберігаються на серверах [7].

Мережа — це глобальна гіпертекстова система, яка перетворилася на розподілену платформу додатків, у якій логіка програми та інтерфейс користувача є двома окремими сутностями. У міру просування до веб-платформи хмарних обчислень основна частина програм для даних користувачів буде знаходитися в мережевій хмарі. Веб-додатки є критично важливою частиною інтернет-інфраструктури та використовуються для банківських операцій, електронної пошти, управління фінансами, онлайн покупки, аукціони, соціальні мережі та такі корпорації, як Google, Microsoft і yahoo, докладають значних зусиль, щоб не відставати від зростаючого попиту на комунікаційні інтернет-послуги, які потребують обміну зображеннями та відео, соціальних мереж і пошуку. Відбувається перехід від відображення інформації за допомогою локально встановлених програм до відображення інформації в браузері [7].

Впровадження хмарних сервісів у розробці веб-застосунків є ключовим елементом сучасної технологічної інфраструктури, який відкриває безліч можливостей для розширення функціональності, забезпечення доступності та зберігання даних. Деякими з ключових аспектів впровадження хмарних сервісів у розробці веб-застосунків є:

— зберігання та обробка даних;
— масштабованість;
— доступність та мобільність;
— забезпечення безпеки;
— інтеграція та розширюваність;
— зниження витрат;
— швидкість розробки.

Хмарні сервіси надають можливість зберігати дані в розподіленому середовищі, що забезпечує надійність та доступність. Вони також дозволяють використовувати потужності хмарної інфраструктури для обробки та аналізу даних, що полегшує роботу з великими обсягами інформації.

Хмарні сервіси можуть легко масштабуватися вгору або вниз, в залежності від потреб веб-застосунку. Це дозволяє забезпечити високу доступність та швидкодію навіть у разі збільшення навантаження. Доступність та мобільність, що за допомогою хмарних сервісів можна забезпечити доступ до веб-застосунку з будь-якого пристрою та з будь-якого місця, де є Інтернет. Це особливо важливо для мобільних додатків та сервісів.

Також багато провайдерів хмарних сервісів вкладають значні зусилля в забезпечення безпеки даних, що надають інструменти для шифрування даних, аутентифікації користувачів та моніторингу безпеки. Інтеграція та розширюваність, які дозволяють легко інтегрувати їх з іншими сервісами, додатками та дозволяє розширювати функціональність веб-застосунку шляхом підключення сторонніх сервісів.

Використання хмарних сервісів може зменшити витрати на обладнання та інфраструктуру, оскільки ресурси зберігаються у хмарі. Крім того, більшість хмарних сервісів працюють на основі оплати за використання, що дозволяє ефективно управляти витратами. Використання хмарних платформ та сервісів може значно прискорити розробку веб-застосунків, оскільки багато компонентів та інфраструктура вже наявні та готові до використання.

Також хмарні сервіси можна класифікувати за рівнем доступу до інфраструктури та рівнем керованості. Серед найпоширеніших моделей схарних сервісів виділяють SaaS, PaaS і IaaS.

SaaS (Software as a Service) — це модель, в якій користувачам надається доступ до вже готових програмних додатків та сервісів через Інтернет. У даній моделі користувачі не повинні володіти або управляти інфраструктурою, серверами або платформами, вони просто використовують програми, які надаються у хмарі. Приклади SaaS-послуг включають Google Workspace, Microsoft 365, Salesforce та багато інших [8].

PaaS (Platform as a Service) — це модель, в якій користувачам надаються інструменти та середовище для розробки, тестування та розгортання власних програмних додатків. Користувачі можуть розробляти свої власні застосунки без необхідності управління інфраструктурою або операційною системою. Приклади PaaS-платформ включають Heroku, Microsoft Azure App Service, Google App Engine тощо [8].

IaaS (Infrastructure as a Service) — це модель, в якій користувачам надається доступ до обчислювальних ресурсів (сервери, мережі, сховища) у віртуальному середовищі, де користувачі можуть керувати цими ресурсами, встановлювати операційні системи та застосунки за своїми потребами. Приклади IaaS-постачальників включають Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP) та інших [8].

Отже виходячи з вище наведених даним ми можемо описати контроль над кожним компонентом системи як показано на рисунку 1.1.

Кожен з цих типів схарних сервісів має свої переваги та використовується для різних цілей. SaaS відповідає за найбільший рівень абстракції та надає готові додатки, PaaS вимагає меншого рівня управління інфраструктурою і спрощує розробку власних застосунків, а IaaS надає повний контроль над інфраструктурою для великих і складних проектів. Вибір конкретної моделі залежить від конкретних потреб і вимог проекту.

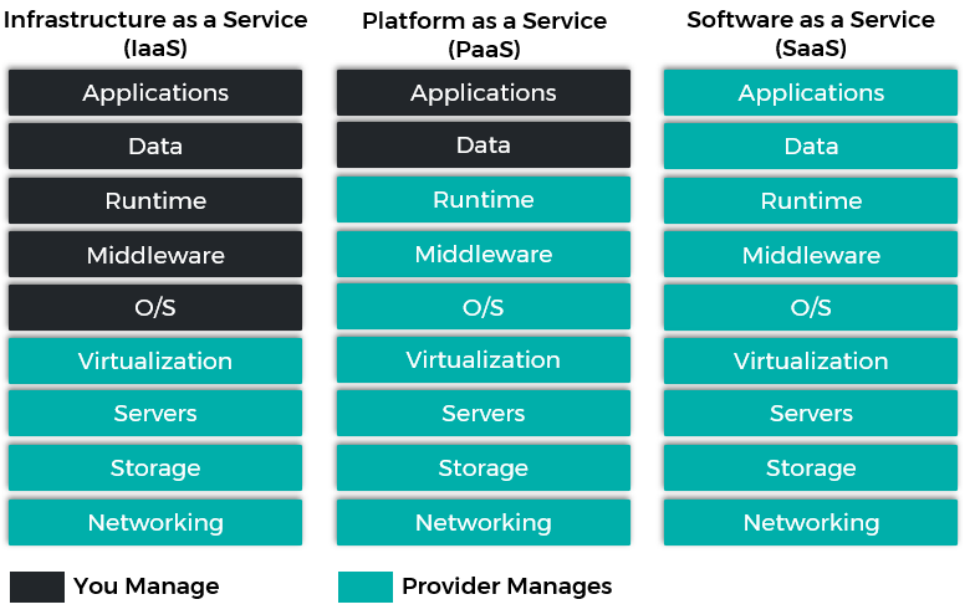

Рисунок 1.1 — Менеджмент контроль над хмарними сервісами

Загалом, впровадження хмарних сервісів у розробці веб-застосунків сприяє покращенню продуктивності, доступності, безпеки та масштабованості додатків. Воно робить можливим створення високопродуктивних та доступних веб-застосунків, які задовольняють сучасні потреби користувачів.

1.6 Взаємодія веб-застосунку з хмарними сервісами

Створення чудового веб-застосунку з використанням хмарних технологій багато в чому схоже на створення будь-якого іншого продукту чи послуг, потрібно скласти архітектурний план за стосунку, обрати платформу хмарних сервісів, систему баз даних та імплементувати це все між собою.

Планування архітектури застосунку передбачає представлення каркаса або макета, який допоможе візуалізувати, як застосунок виглядатиме та функціонуватиме. Після цього можете перейти до створення історій користувачів, які описуватимуть, що ваші користувачі будуть робити на кожному екрані застосунку.

Також важливо мати на увазі, що застосунок доведеться оновити або змінити в майбутньому. Через це гарною ідеєю є створення застосунку з використанням архітектури, яка дозволяє легко оновлювати та покращувати шлях розробки.

Після того як було визначено архітектуру застосунку настає час вибрати платформу. Хмарна система є ідеальною, оскільки вона дозволяє легко збільшувати або зменшувати масштаб залежно від попиту. Це також дозволяє легко додавати додаткові функції в майбутньому, не турбуючись про витрати на сервер або проблеми з інфраструктурою.

Найпопулярнішими хмарними платформами є Amazon Web Services, Google Cloud Platform і Microsoft Azure. Прикладом сервісів, що надають дані платформи наведені на рисунку 1.2 [9].

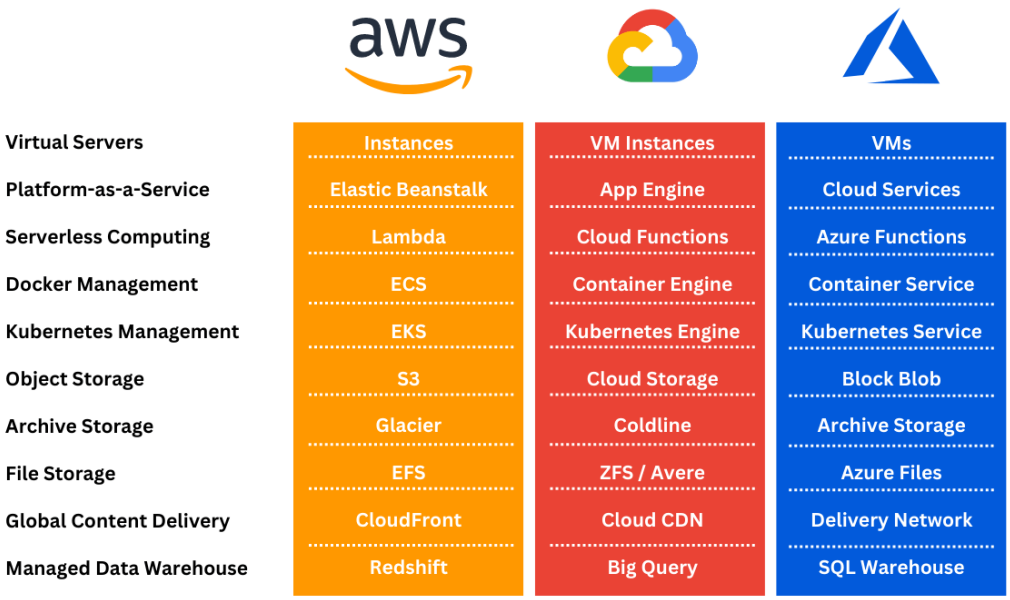

Рисунок 1.2 — Сервіси, що надають хмарні платформи

Кожна з цих послуг пропонує широкий спектр функцій, включаючи простір для хмарного сховища, обробку даних і розміщення серверів, тому важливо визначити, які з них будуть найбільш вигідними у розробці застосунку.

Окрім цих служб, доступно також багато інструментів сторонніх розробників, які можуть допомогти спростити процес створення застосунку. Кожна з цих служб розроблена, щоб полегшити розробникам створення, розгортання та керування програмами в хмарі. Вони також корисні для тих, хто не має ресурсів або досвіду для створення власної хмарної інфраструктури [10].

Наступним кроком є вибір системи бази даних, яка може підтримувати за стосунок, що розробляється. На ринку існує багато різних типів баз даних, але не всі вони підходять для конкретних потреб. Для деяких програм може знадобитися реляційна база даних, тоді як інші можуть краще обслуговуватися за допомогою NoSQL, або хмарних рішень, таких як Amazon DynamoDB і Google Cloud Datastore [10].

Наприклад, якщо ваша застосунок потребує складних запитів, то найкращим варіантом може бути реляційна база даних, що використовуватиметься у даній роботі. Однак якщо розробляється хмарна програма, яка зосереджена на простому зберіганні та пошуку даних, наприклад, блог або вікі-сайт, тоді бази даних NoSQL, як правило, ефективніші.

Також у ході розробки повинна враховуватись вартість кожного рішення, перш ніж зробити остаточний вибір. Багато постачальників хмарних баз даних пропонують безкоштовні плани, які можна використовувати для початку роботи, але вони зазвичай певним чином обмежені. Наприклад, Amazon DynamoDB обмежує кількість даних, які ви можете зберігати, і обслуговує лише один запит на секунду. Це може бути проблемою, якщо програма вимагає високого рівня продуктивності або масштабованості. Якщо потрібно швидко збільшити масштаб, тоді доведеться заплатити за послугу.

1.7 Порівняльна характеристика аналогів

З великої кількості доступних веб-застосунків для оцінювання та організації навчального процесу було обрано два найпопулярніші сервіси: Google Classroom та NEO.

Google Classroom є веб-сервісом, створеним компанією Google у 2014 році для навчальних установ. Цей інструмент спрямований на спрощення процесів створення, розповсюдження та категоризації навчальних завдань, уникнення використання паперу та забезпечення зручної роботи в онлайн-середовищі. (рисунок 1.3) [10].

Рисунок 1.3 — Логотип Google Classroom

NEO LMS — це система дистанційного навчання, що представляє собою хмарну інтелектуальну освітню платформу, розроблену для використання у школах та університетах. (рисунок 1.4) [11].

Рисунок 1.4 — Логотип NEO

Звісно, обидва сервіси призначені для спрощення навчального процесу, проте вони мають відмінності, які можуть бути і їхніми перевагами, і недоліками. Давайте розглянемо кілька з них.

Для їхнього порівняння оберемо декілька категорій, серед яких:

— інтерфейс взаємодії із системою;

— простота використання;

— персональний підхід;

— інтеграція з іншими системами;

— налаштування системи;

— комунікація між користувачами;

— вартість.

Інтерфейс взаємодії з системою відрізняється в NEO та Google Classroom. У NEO дизайн інтерфейсу є адаптивним і автоматично налаштовується під різні типи пристроїв, надаючи користувачам інформаційні панелі на основі плиток для студентів, викладачів та адміністраторів. Користувачі можуть переглядати завдання, події, групи, календарі й інше.

Інтерфейс інформаційної панелі Google Classroom є простішим, але менше графічним. Викладачі та студенти бачать лише курси, в яких вони беруть участь або які вони викладають.

NEO пропонує зручну спливаючу навігаційну систему з ярликами та легким доступом до всіх розділів сайту, незалежно від того, де ви перебуваєте на платформі. Користувачі можуть отримати доступ до класів, груп, налаштувань тощо через спливаюче меню на лівій панелі, не переходячи на інші сторінки.

У Google Classroom головне меню можна приховати або показати, але відсутня спливаюча навігація для основних пунктів у меню. Усі класи відображаються у лівому меню, що може призвести до необхідності прокручувати список класів, якщо їх багато (рисунок 1.5).

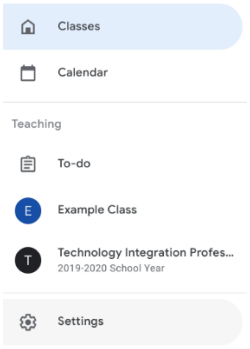

Рисунок 1.5 — Навігаційна панель в Google Classroom

Макет класу NEO є дуже інтуїтивним і пропонує користувачам вибір між графічним виглядом плитки або рядковим (рисунок 1.6). Кожна плитка містить ключову інформацію, таку як значки, бали та сертифікати, які стосуються кожного уроку. Учні можуть відстежувати свій прогрес у класі та на кожному уроці за допомогою піктограм, розміщених на кожній плитці.

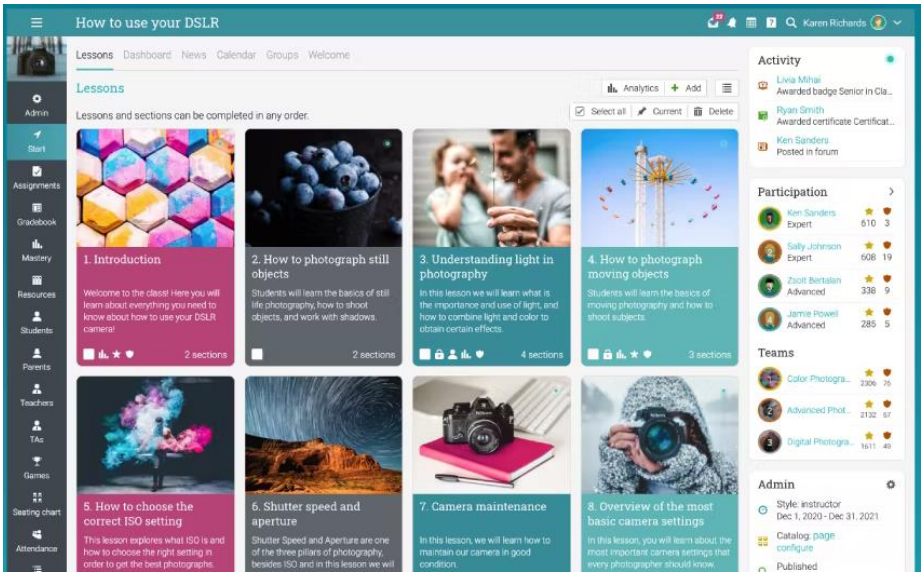

Рисунок 1.6 — Сторінка навчального курсу в NEO

Простота використання є однією з ключових переваг NEO, який пропонує інтуїтивний та легкий у використанні інтерфейс. Крім того, ця платформа має онлайн-довідковий центр з відеоуроками та посібниками для новачків, а також швидкозагальний форум підтримки, де на запитання відповідають протягом короткого часу.

Google Classroom, з іншого боку, простіший у сприйнятті та використанні, але має менший обсяг підтримки користувачів. Вона також надає довідковий центр з посібниками та форум, але рівень підтримки менший порівняно з NEO.

У NEO є широкий спектр функцій, які полегшують роботу вчителям та роблять процес навчання цікавішим для учнів. У той час, коли Google Classroom має обмежений функціонал, відсутність деяких інструментів, таких як вбудований підручник, автоматизація, і обмежений набір типів завдань, порівняно з NEO.

NEO надає можливість використовувати 15 різних типів завдань, включаючи вікторини, есе, командні завдання, дебати, дискусії та опитування. Усі доступні типи завдань можна побачити на рисунку 1.7. У Google Classroom функціонал є досить обмеженим, оскільки вчителі можуть створювати завдання, додавати питання (для обговорень) та прикріплювати файли з Google Drive. Щоб створити вікторину, вчителям потрібно використовувати Google Forms [11].

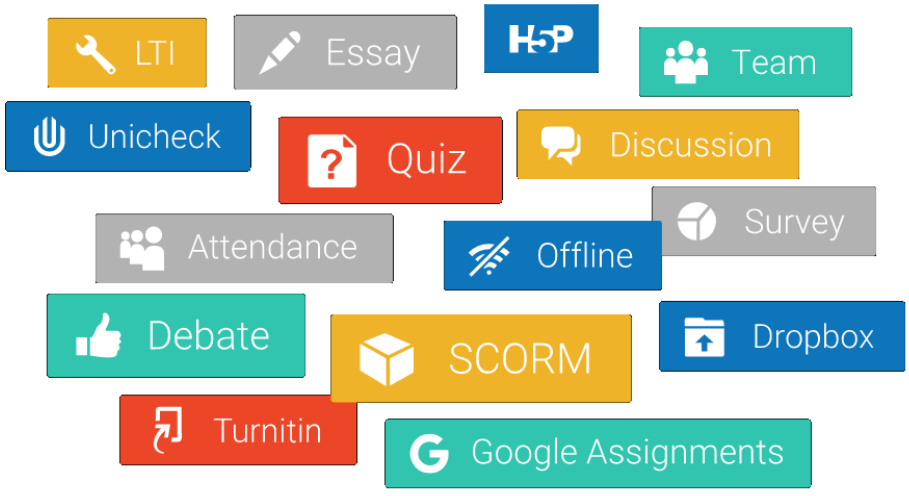

Рисунок 1.7 — Перелік всіх можливих типів завдань у NEO

NEO має потужний інструмент для узагальнення оцінок, де користувачі можуть централізовано переглядати та редагувати усі оцінки, додавати результати та вносити зміни, що забезпечує швидкість та ефективність у роботі. У Google Classroom відсутня можливість вести журнал оцінок.

У NEO вчителі можуть налаштовувати дії, які відбуваються при завершенні учнями занять, вступі у групи, або відставанні в їхньому навчанні. Це дозволяє зекономити час і персоналізувати навчання. У Google Classroom відсутні інструменти для персоналізації навчання.

Щодо інтеграції з іншими сервісами, NEO взаємодіє з популярними освітніми інструментами, такими як Zapier, iCal, xAPI, Microsoft 365, OneDrive та Zoom, а також інтегрується з Google Drive, Google Workspace і багатьма іншими сервісами.У Google Classroom є можливість

використовувати лише такі інструменти, як Google Drive або інструменти, які є частиною G Suite for Education.

Систему NEO можна налаштовувати у різних напрямках, таких як колірну схему, шрифти, термінологію тощо, надаючи школам велику свободу налаштування. У порівнянні з цим, Google Classroom не має таких розширених можливостей налаштування, крім невеликих змін, наприклад, теми уроків. Це може бути обмеженням для шкіл, які хочуть налаштувати особистий портал для відвідувачів, логотип, кольорову схему чи створити свій унікальний вигляд порталу.

У NEO користувачі можуть спілкуватися через різні інструменти, такі як чат, повідомлення, групи, команди, соціальні мережі, стрічки новин, вікі, блоги та каталог ресурсів для обміну матеріалами. У Google Classroom функціонал спілкування обмежений, відсутній чат, вбудовані повідомлення, групи та соціальні мережі.

Щодо вартості, NEO пропонує безкоштовний план з повним функціоналом для управління навчальним процесом та платний преміум-план із розширеними можливостями для шкіл. Крім того, немає додаткових оплат за налаштування системи чи прихованих витрат, хоча облікові записи вчителів, викладачів та батьків можуть бути окремо оплачені.

Google Classroom є безкоштовним для будь-якого користувача з обліковим записом Google, який використовує його для особистих потреб. Однак для шкіл або університетів, яким потрібні розширені функції, такі як спільні календарі та необмежене хмарне сховище, доступний пакет G Suite Enterprise for Education за 4 долари США на користувача щомісяця для викладачів, співробітників і студентів.

Після порівняння переваг та недоліків подібних систем розробник вирішив, яким має бути розроблений веб-застосунок. Він повинен мати зручний та інтуїтивний інтерфейс, легко редаговані компоненти, а також потужність для обробки великого обсягу серверних запитів та базу даних, що зможе ефективно зберігати всі необхідні ресурси.

2 АНАЛІЗ ТА ОБГРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ

2.1 Архітектура веб-застосунку

Архітектура програмного забезпечення відіграє вирішальну роль у здатності сервісу масштабуватися та задовольняти потреби користувачів з часом. Недоліки будь-якого програмного забезпечення мають значний вплив на працездатність системи. Основною причиною будь-якого збою програмного забезпечення може бути вибір неправильної архітектури побудови сервісу.

Архітектура веб-застосунку — це модель взаємодії між компонентами веб-застосунку. Конкретний вид архітектури для веб-додатків строго залежить від способу розподілу логіки додатка між клієнтською та серверною сторонами. Технічно, це каркас веб-застосунку, у який включається його елементи, бази даних, системи, сервери, інтерфейси та вся комунікація, що відбувається між ними. В більш абстрактних термінах це вказує на логіку відповідей на запити клієнта та сервера [12].

Сучасна архітектура веб-застосунків зазвичай складається з кількох рівнів, які працюють разом, щоб забезпечити надійну, масштабовану та придатну для обслуговування систему. Кожен рівень служить певній меті, і разом вони утворюють цілісне та інтегроване рішення, яке може виконувати різноманітні завдання.

Поділяючи веб-застосунок на ці рівні, дана архітектура гарантує, що кожен із них можна розробляти та обробляти незалежно від інших. Цей підхід полегшує модифікацію та покращення програми, оскільки вимоги змінюються з часом.

На рисунку 2.1 зображено клієнтський і серверний рівні архітектури веб-застосунку. Розглянемо кожен з наведених рівнів детальніше. Презентаційний рівень — це рівень, що відповідає за інтерфейс користувача веб-застосунку. Він займається відображенням інформації, збором даних і визначенням взаємодії користувача. Рівень презентації зазвичай створюється за

допомогою HTML, CSS і JavaScript, які обробляють макет, стиль і інтерактивність інтерфейсу користувача. Часто використовуються такі сучасні фреймворки, як React, Angular або Vue.js.

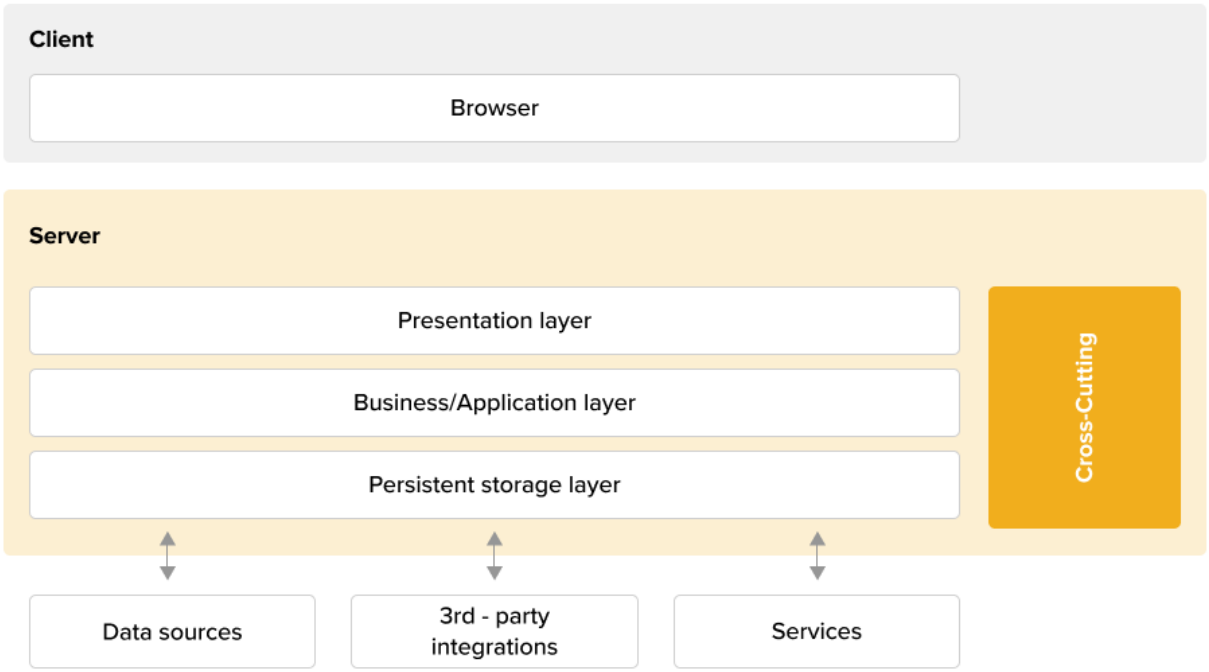

Рисунок 2.1 — Рівні архітектури веб-застосунку

Наступним є рівень бізнес-логіки (business logical level — BLL), що містить бізнес-логіку програми та функціональність веб-застосунку. Він обробляє запити користувачів, обробляє дані та виконує операції на основі визначених правил і алгоритмів. Рівень бізнес-логіки забезпечує дійсність і точність оброблених даних. Цей рівень часто використовує такі мови програмування, як Python, Java або C#, а також фреймворки та бібліотеки, які полегшують розробку додатків і надають інструменти для маршрутизації, перевірки та маніпулювання даними.

Далі йде рівень доступу до даних (data access level), що відповідає за керування зберіганням і отриманням даних, які використовуються програмою. Він обробляє всі зв’язки з базою даних, а також збереження даних і маніпулювання ними, забезпечуючи цілісність і послідовність інформації. Зазвичай це стосується бази даних або комбінації баз даних, таких як реляційні

бази даних (наприклад, MySQL, PostgreSQL) або бази даних NoSQL (наприклад, MongoDB, Cassandra).

Рівень служб даних, або ж data service layer (DSL) забезпечує інтерфейс служби для інших рівнів додатків. Він розкриває функціональні можливості рівня доступу до даних і забезпечує послідовне отримання та збереження даних. Він об’єднує прикладний рівень та інші компоненти, забезпечуючи зв’язок і обмін даними із зовнішніми системами чи службами. Він забезпечує інтеграцію зі сторонніми API, базами даних або мікросервісами, дозволяючи веб-програмі отримувати доступ і маніпулювати даними з різних джерел.

У більш складних архітектурах між BLL і DSL може існувати додатковий рівень послуг. Цей рівень абстрагує конкретні функціональні можливості в модульні служби, забезпечуючи більш відокремлену та масштабовану систему. Сервіси можуть бути розроблені для виконання таких завдань, як автентифікація, авторизація, зберігання файлів, сповіщення електронною поштою або обробка платежів, і їх можна незалежно розробляти, тестувати та розгортати.

Також існують різні типи архітектури веб-застосунків, кожна з яких має свої сильні та слабкі сторони.

Найпоширеніші типи включають монолітну архітектуру, мікросервіси, прогресивні веб-застосунки (progressive web apps) та serverless архітектуру. Розуміючи їх відмінності, ви можете прийняти набагато більш глибоке рішення про те, який підхід підходить найбільше у розроблюваному застосунку.

Моноліт — це традиційний підхід до створення веб-додатків, коли всі компоненти (інтерфейс, бек-енд, база даних) тісно поєднані та упаковані в єдину програму. Цю методологію легко розробити та розгорнути, але її використання та масштабування можуть ставати дедалі складнішими у міру зростання програми.

На відміну від монолітної архітектури, мікросервіси розбивають додаток на невеликі незалежні сервіси, які можна розробляти, розгортати та масштабувати незалежно один від одного. Хоча цей підхід забезпечує більшу гнучкість і масштабованість, він також може бути набагато складнішим у

створенні та управлінні [13]. На рисунку 2.2 зображено відмінність між молітною та мікросервісною архітектурою.

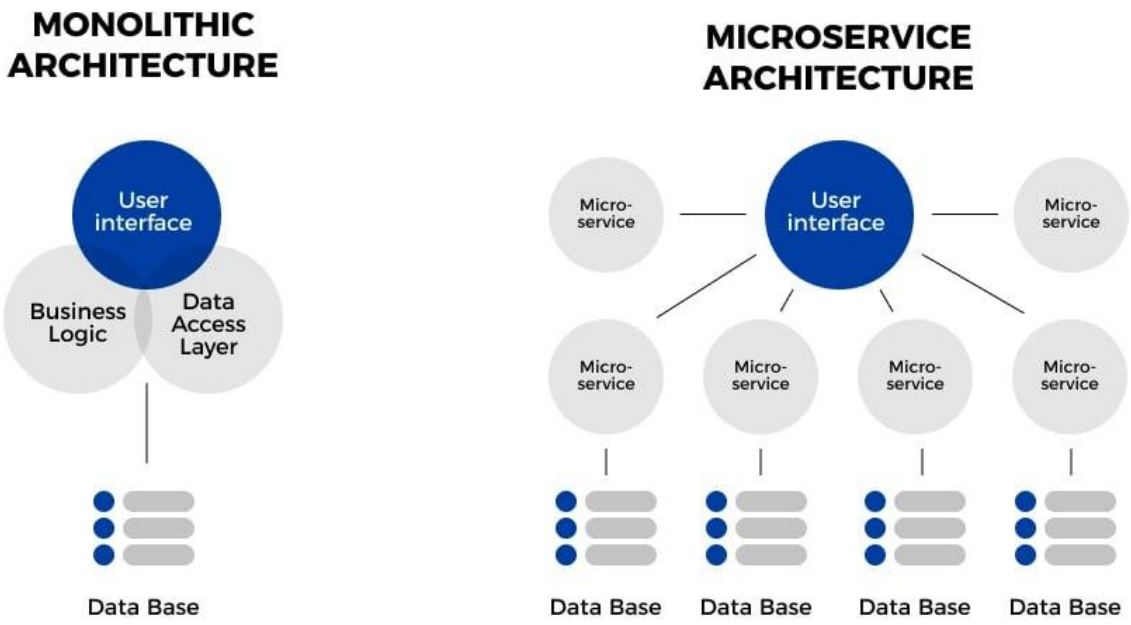

Рисунок 2.2 — Відмінність монолітної архітектури від мікросервісної

Прогресивні веб-програми (PWA) — це веб-програми, розроблені для роботи як рідні мобільні програми, пропонуючи такі функції, як офлайн-функціональність, push-сповіщення та доступ до апаратного забезпечення пристрою. PWA створено з використанням веб-технологій, які можна розгорнути на будь-якому пристрої з веб-браузером [14].

Serverless архітектура, належить до Function-as-a-Service (FaaS) — це новий підхід до архітектури веб-додатків, який покладається на хмарних провайдерів для керування основною інфраструктурою. Тут розробники пишуть функції, які виконуються у відповідь на події (наприклад, запити користувачів) без необхідності керувати серверами чи інфраструктурою.

Ознайомившись із доступними типами архітектур за якими ми можемо побудувати даний застосунок, можна прослідкувати, що найкраще підходять монолітна та мікросервісна архітектури. Зважаючи на те, що даний веб-застосунок на даний момент не потребує інтеграції із іншими необхідними компонентами функціональності, які можуть стати у нагоді, найбільш

релевантною є розробка монолітного застосунку, що дозволить спростити його розробку на початковому рівні, зменшить ресурсозатратність, та дозволить всім компонентам взаємодіяти без використання мережі, що може сприяти простоті управління внутрішніми системами.

Розглянемо більш детально взаємодію компонентів застосунку між собою (рисунок 2.3).

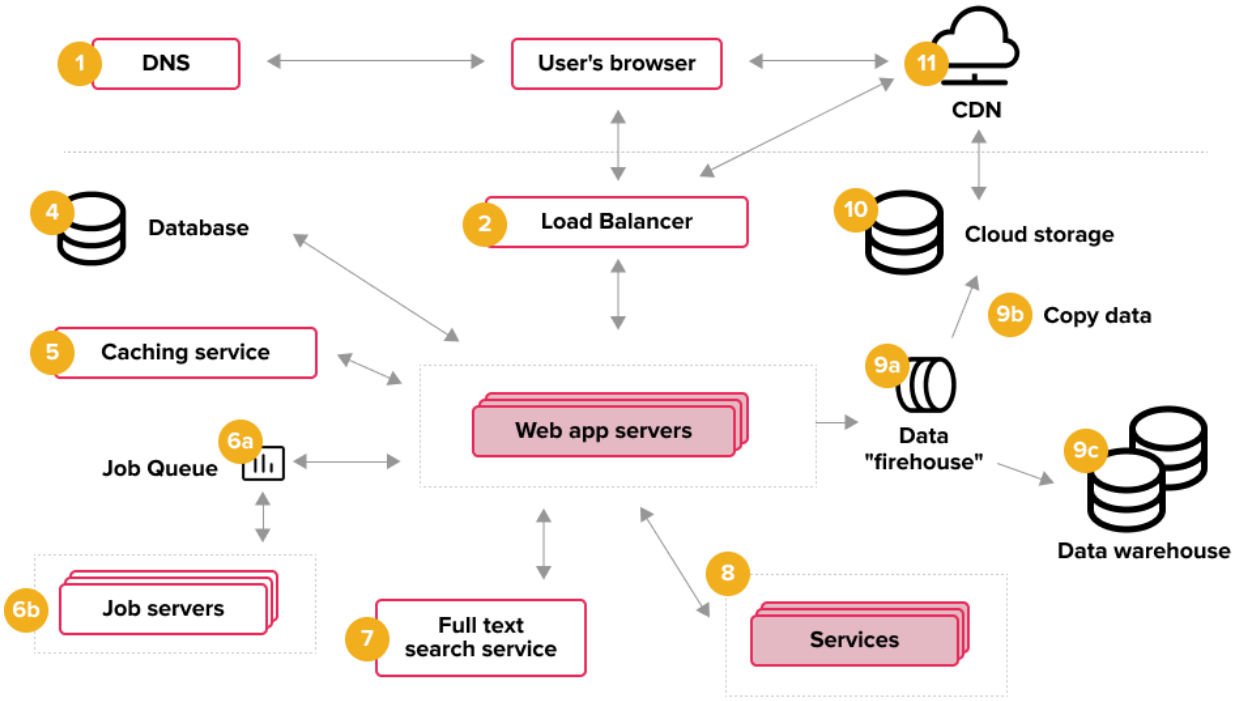

Рисунок 2.3 — взаємодія компонентів застосунку між собою

На вище наведеному рисунку зображено діаграму, що показує взаємодію між компонентами веб-застосунку. Дана діаграма включає в себе такі компоненти як:

— DNS (Domain Name System);
— сервіси веб-застосунку;
— load balancer;
— бази даних
— сервіси кешування
— сховища даних;
— мережа доставки контенту (CDN).

Розглянемо кожен із компонентів застосунку та їхню взаємодію між собою. Першим компонентом виступаю система доменних імен, або ж DNS (Domain Name System), що перетворює доменні імена в IP-адреси, що дозволяє користувачам отримувати доступ до веб-сайтів, використовуючи зрозумілі людині імена. Служба DNS забезпечує безпроблемний доступ користувачів до розроблюваного веб-застосунку.

Сервери веб-додатків відповідають за обробку запитів від клієнтів і створення відповідей. Ці сервери запускають код програми та виконують завдання автентифікації, авторизації та керування сеансами. До поширених серверів веб-додатків належать Apache, Nginx і IIS.

Load balancer, або ж балансувальник навантаження розподіляє трафік між декількома серверами, щоб програма могла обробляти великі запити. Це також допомагає запобігти перевантаженню сервера, покращуючи загальну продуктивність програми.

Бази даних зберігають і керують даними, які використовує веб-застосунок. Тип використовуваної бази даних залежить від конкретних вимог, що поставлені перед веб-застосунком. В свою чергу служби кешування покращують продуктивність програми, зберігаючи в її пам’яті дані, до яких часто звертаються. Їх використання допомагає зменшити кількість запитів до бази даних і покращити час відповіді.

Зберігання інформації веб-застосунку в хмарі замість локального сервера робить дані більш доступними незалежно від місця розташування користувачів. Більшість постачальників хмарних послуг пропонують кілька планів передплати з різним обсягом і навантаженням трафіку. Вони гарантують, що інформація зберігатиметься в безпеці, що є неоціненною перевагою з боку бізнес-аналізу.

Хмарне сховище також дозволяє розробникам оптимізувати час доступу для користувачів у цільових географічних регіонах. Таким чином користувачі веб-застосунку відчуватимуть менше затримок. Після релізу веб-застосунку розробники можуть додавати нові одиниці хмарного сховища, або видаляти існуючі та змінювати план підписки на обслуговування. Це значно покращує масштабованість і оптимізує витрати на застосунок [15].

Метою сховищ даних є зберігання та аналіз великих обсягів даних. Він забезпечує центральне розташування для зберігання даних із багатьох джерел і забезпечує розширену аналітику даних.

Та мережа доставки вмісту, широко відома як CDN (Content Delivery Network), — це розподілена мережа серверів, яка зберігає кешовані копії статичного вмісту. Це допомагає покращити продуктивність програми, зменшуючи затримку, пов’язану з наданням вмісту з одного сервера.

В свою чергу «хмарна» частина застосунку буде описана в одному із наступних розділів, у якому буде представлено більш детальний опис взаємодії хмарних сервісів із внутрішніми сервісами застосунку, що розробляється.

Таким чином ми визначилися із архітектурою за якою розроблятиметься застосунок та які вимоги перед ним поставленні. Після чого потрібно обрати підбільш підходящий стек технологій для розробки застосунку. Для цього потрібно обрати технології та інструменти для розробки серверної та клієнтської частини.

2.2 Технології створення серверної частини застосунку

2.2.1 Огляд мови програмування та фреймворку

Серед мов програмування, які дозволяють розробити серверну частину веб-застосунку відмічають наступні мови: C#, Go, Java, JavaScript, PHP, Python, Ruby. Розглянемо детальніше кожну із вище перелічених мов програмування, що вони пропоную для розробки і яка із перелічених мов найкраще підходить для реалізації поставленої мети.

C# — це мова, яка може працювати на різних типах комп'ютерів. Програмісти використовують цю мову разом із такими фреймворками, як .NET, щоб розробити внутрішню структуру веб-сторінок і зв’язати взаємодію клієнта з взаємодією сервера для повноцінної роботи програми. Універсальність і стабільність C# і пов’язаних з ним фреймворків роблять

його надійним вибором для веб-розробки та розробки програмного забезпечення.

C# схожа на C і C++, які є корисними мовами програмування, знаючи основи яких дозволить легко вивчити C#. Дана мова також є зручною для початківців, оскільки його захищені від помилок команди попереджають про проблеми перед тестуванням програми, що спрощує пошук несправностей. Також одним із плюсів C# є те, що ця мова є мовою компілювання, що означає, що код, який зберігається на загальнодоступному сервері, має двійкову форму. Якщо сервер буде зламано, хакер автоматично не матиме доступу до вихідного коду. З іншими поширеними мовами, такими як PHP, хакер отримує доступ до вихідного коду, який потім може надати йому доступ до паролів бази даних. З C# в свою чергу хакер повинен декомпілювати або «зламати» програмне забезпечення, перш ніж він зможе побачити критичні компоненти [16].

Мова Go (Golang) — це мова програмування з відкритим вихідним кодом, створена Google для корпоративних потреб. Це мова компілювання, із статичною типізацією, яка набула великої популярності з моменту випуску наприкінці 2000-х. Go поєднує в собі найкращі функції кількох інших мов програмування. Наприклад, він має високу продуктивність, як C/C++ і Java, і підходить для різних цілей, наприклад як Python. Він пропонує паралелізм, динамічно форматовані інтерфейси, покращену безпеку пам’яті та чудову масштабованість. Хоча Go має багато переваг, включаючи ефективність, швидкодію та простоту, є деякі аспекти, які можуть розглядатися як його недоліки, а саме система бібліотек Go постійно росте, вона все ще може бути менш розвиненою порівняно з іншими мовами, такими як Java або Python, оскільки Go є молодою мовою програмування порівняно з іншими, що може вплинути на кількість доступних інструментів, фреймворків або навіть рішень для деяких конкретних завдань. Також одним із мінусів Go є відсутність всіх функцій традиційного об'єктно-орієнтованого

програмування, що може ускладнити роботу з деякими паттернами або аспектами розробки [17].

Як найпопулярніша мова програмування, JavaScript також є однією з найбільш універсальних мов розробки програмного забезпечення. Традиційно використовувана як інструмент розробки веб-інтерфейсу, вона також стала основним крос-платформним інструментом мобільної розробки та базовою технологією для великої кількості платформ, таких як Apache Cordova/PhoneGap, React Native, NativeScript і Appcelerator Titanium [18].

Серед таких переваг JavaScript як мінімізація роз'єднання між собою back-end та front-end частин застосунку, спрощення процесу розробки, легке повторне використання коду та можливість розробки різних за призначенням програмних модулів за допомогою Node.js, що є великою перевагою перед іншими мовами, такими як PHP або Go, незважаючи на це JavaScript має перелік мінусів, що є критичними при виборі її як мови для розробки back-end частини веб-застосунку, а саме: однопоточність, масштабованість, потреба у великій кількості модулів, швидкість, обробка помилок.

JavaScript має однопоточну модель виконання, що може стати обмеженням для завдань, які потребують багатопоточного оброблення, таких як паралельні обчислення або робота з великим обсягом обчислень

Масштабованість: незважаючи на те, що Node.js добре масштабується, JavaScript може мати деякі обмеження в масштабованості для дуже великих та складних проектів порівняно з іншими мовами. У деяких випадках, для досягнення певного функціоналу або операцій, може знадобитися велика кількість модулів та бібліотек. Це може зробити код більш складним для управління [19].

У порівнянні з деякими іншими мовами програмування (наприклад, Go), швидкодія JavaScript може бути меншою у деяких випадках. Однак, це може бути менш важливим при використанні хмарних сервісів, оскільки вони можуть компенсувати деякі обмеження.

Обробка помилок в асинхронному коді JavaScript може бути складнішою порівняно зі звичайними синхронними мовами, що може призвести до складнощів у виявленні та виправленні помилок [20].

PHP — це абревіатура гіпертекстового препроцесора. Це мова сценаріїв, що означає, що код, який пишиться цією мовою, проходить через інтерпретатор для читання комп'ютером, а не через компілятор, де він перекладатиметься в машиночитаний код. Розробники використовують PHP для роботи на стороні сервера, і це спрощує розгортання або надсилання та підтвердження вашого коду завдяки інструментам і можливостям розгортання. PHP корисний у різних сферах розробки, оскільки ви можете вставляти код PHP у HTML, який є зовнішнім інструментом для розробки веб-додатків і сторінок [21]. Але PHP має кілька недоліків для розробки back-end частини застосунків. По-перше, він часто критикується за свою консистентність та якість коду через історичні особливості. Неодноразово змінювана архітектура та стандарти можуть призвести до нестабільності та непередбачуваності роботи. Також, швидкодія PHP може бути меншою порівняно з деякими іншими мовами, особливо при високих навантаженнях та складних обчисленнях. Крім того, PHP не завжди демонструє високу продуктивність при роботі з багатошаровими та розширеними аплікаціями [22].

Іншим аспектом є обмежена масштабованість PHP для дуже великих проектів. Хоча з'явилися фреймворки, які спрощують цей процес, PHP може виявитися менш ефективним у порівнянні з іншими мовами програмування при створенні складних систем. Наостанок, PHP, не дуже добре підходить для багатопоточних завдань через свою специфіку виконання та історичні обмеження.

Python — це об'єктно-орієнтована мова, яка фокусується на маніпулюванні об'єктами, що містять дані. Це проста мова з простим синтаксисом, який полегшує читання та налагодження програми. За допомогою Python веб-розробники можуть використовувати фреймворк

Django з відкритим вихідним кодом для створення масштабованого програмного забезпечення, яке можна легко оновлювати [23].

Python славиться своєю простотою та лаконічністю синтаксису, що робить код більш читабельним і зрозумілим для розробників. Це полегшує роботу над проектами в команді. Також Python добре поєднується з іншими мовами та технологіями, що робить його привабливим для інтеграції з різними системами та бібліотеками, має власну велику кількість бібліотек та фреймворків, які сприяють швидкому розвитку додатків. Серед яких є такі бібліотеки як Django та Flask, що є популярними фреймворками для розробки back-end частини веб-застосунків [24]. Водночас порівняно з іншими мовами, такими як Go чи Java, Python може бути повільнішим через свою інтерпретованість. Це може вплинути на швидкодію додатків у випадку великих обчислень або високих навантажень. У випадку дуже великих проектів, Python може виявитися менш ефективним через його обмежену масштабованість порівняно з іншими мовами, особливо при роботі з великим обсягом даних або підвищеному навантаженні. Також Python володіє таким недоліком як Global Interpreter Lock (GIL), це обмеження, яке призводить до того, що в один момент часу може виконуватися тільки один потік, що ускладнює багатопоточність та паралелізм в деяких сценаріях [25].

Ruby — це мова програмування, яку можна використовувати з Ruby on Rails (RoR), фреймворком, створеним спеціально для Ruby, який дозволяє створювати та виконувати завдання, використовуючи менше коду. Фреймворки часто є чудовими інструментами для архітекторів програмного забезпечення, які будують основу програми або програми, використовуючи бібліотеки коду фреймворку, вбудовані інструменти та спеціальні функції, які спрощують процес написання коду [26].

Ruby часто є гарним вибором для початківців, щоб вивчати та практикувати бекенд-розробку. Але при розробці комплексного веб-застосунку є можливість зіштовхнутися з обмеженнями даної мови, а саме з продуктивністю та швидкодією, потребою у ресурсах, що може призвести до

вищих витрат на хмарні обчислення для підтримки великих обсягів даних та обробки, складністю з масштабуванням застосунку та обмеженою підтримкою для деяких хмарних сервісів, підтримка Ruby на хмарних платформах може бути менш розвиненою порівняно з іншими мовами, такими як Java або Python [27].

Java — це мова для розробників, які хочуть створювати великі, надійні веб-програми, які вимагають високих заходів безпеки для збереження даних. Це універсальна мова, яку можна використовувати з багатьма цифровими платформами, включно з мобільними пристроями та комп'ютерами, для створення веб-, мобільних і настільних програм та інструментів. Java працює на віртуальній машині Java (JVM), яка стандартизує машину, на якій програмісти запускають код, а не дозволяє запускати її на окремій машині кожного програміста [28]. Саме Java Virtual Machine є одним із найбільших переваг Java перед іншими мовами програмування, оскільки вона дозволяє запустити одноразово написаний код на різних пристроях без зайвих затрат в плані коштів та часу. Технічно кажучи, JVM являє собою строго визначений набір інструкцій і правил, описаних у специфікації програмного забезпечення. Ця специфікація є посібником для розробників, що описує, як машина має інтерпретувати байт-код, який генерується компілятором Java. Вона містить докладні технічні вказівки, що описують структуру даних, формат інструкцій і очікувану поведінку JVM під час виконання програми [29].

Java надає високий рівень надійності та стабільності, на це вказує, що її використовують у багатьох критичних системах, оскільки має добре побудований стандартний стек технологій та відмінну систему управління пам'яттю. Програми розроблювальні на мові Java добре масштабуються до великих проектів та що було раніше згадано є кросплатформенними, тобто не залежать від платформи на якій вона запускається. Також Java має великий екосистему інструментів та бібліотек, які полегшують розробку, тестування, підтримку додатків та є чудовим рішенням для реалізації великих

ентерпрайз-рішень. Звичайно, як і решта мов програмування, Java має власний перелік недоліків, до якого належать високі вимоги до ресурсів, повільний запуск на слабких пристроях, недостатня асинхронність порівняно з деякими іншими мовами програмування та низька продуктивність у низькорівневих завданнях [30].

Але зважаючи на вимову, які поставленні перед нами та проаналізувавши вищенаведені мови програмування, можна дійти висновку, що мова програмування Java підходить якнайкраще для поставленої задачі. Окремо підкреслити можна те, що хоч Java і володіє певними недоліками, але вони по більшій мірі відносяться лише для невеликих програм, а у випадку розробки веб-застосунку перелік плюсів є неабиякою перевагою перед іншими мовами програмування. Однією із найвагоміших переваг є саме використання Spring фреймворку для розробки серверної частини веб-застосунку, про плюси якого буде описано нижче.

Spring Framework (рисунок 2.4) — це великий і популярний фреймворк для розробки програмного забезпечення на платформі Java. Він надає комплексний набір інструментів та інфраструктури для спрощення розробки і підтримки Java-додатків. Spring дозволяє розробникам створювати різноманітні типи програм, включаючи веб-додатки, корпоративні рішення та сервіси [32].

Рисунок 2.4 — Логотип Spring Framework

Spring дозволяє створити обробник повідомлень з POJO (Plain Old Java Object), що представляють з себе об’єкти, які не реалізовують жодні

специфічні класи, не маючи роботи з JMS API, за допомогою цього реалізовується спрощене написання додатків і легке їх налаштування.

Також Spring Framework містить близько 20 модулів, які підтримують проекти розробки Java. Ці модулі можна згрупувати базуючись на їхніх основних функцій у такі групи як Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation і Test. Ці групи показано на рисунку 2.5.

Розглянемо кожен із модулів детальніше, який знадобиться у ході розробки веб-застосунку, а саме:

— Spring Core
— Spring Boot
— Spring MVC
— Spring Security
— Spring Data

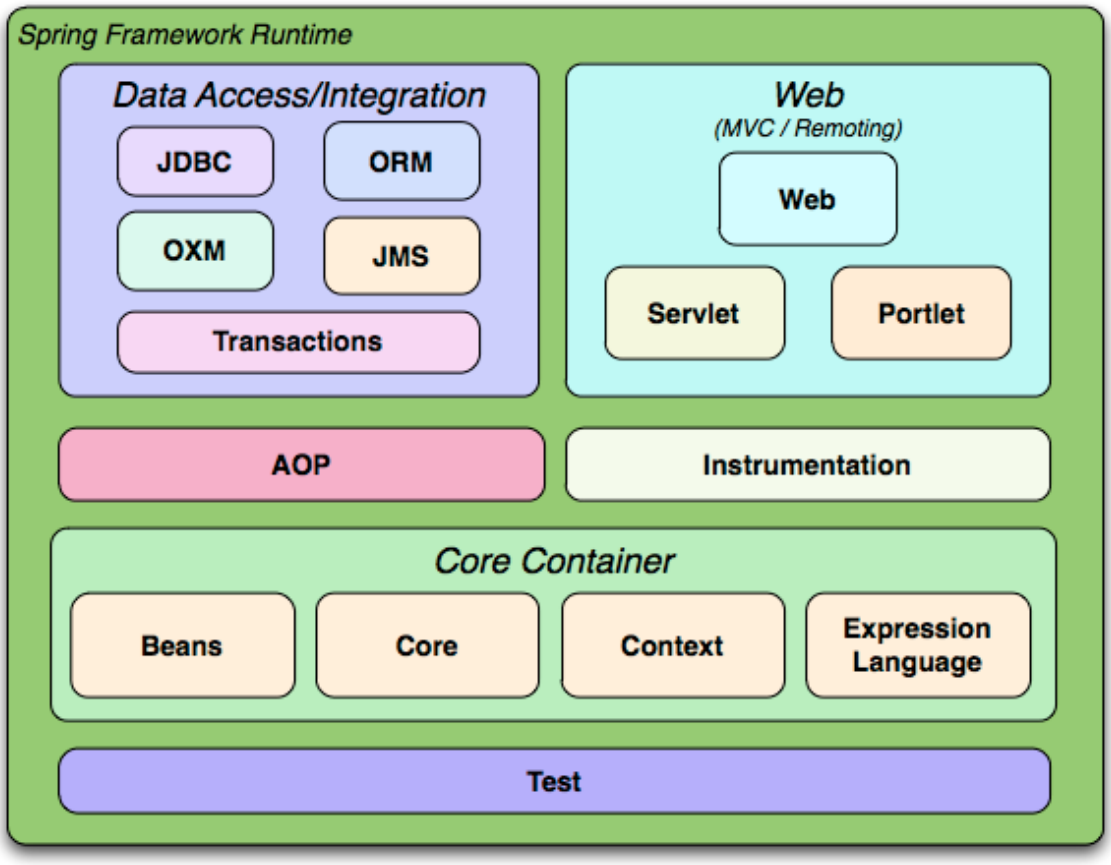

Рисунок 2.5 — Діаграма модулів Spring Framework

Spring Core є одним з основних модулів у Spring Framework. Цей модуль містить базовий функціонал для управління компонентами додатку та реалізацію інверсії управління (Inversion of Control — IoC) та внесенням залежностей (Dependency Injection — DI). Основні функції Spring Core включає в себе IoC контейнер, який керує створенням об'єктів та їх залежностями. Замість того, щоб вручну створювати об'єкти, розробник описує їх в конфігураційних файлах, і контейнер Spring бере на себе їх створення та керування. Dependency Injection — це концепція, за якою об'єкти не створюють свої залежності самостійно, а отримують їх зовні, Spring дозволяє визначати залежності об'єктів через конструктори, методи або поля [32].

Наступним елементом Spring Core є Bean Factory, що представляє з себе основну реалізацію контейнера Spring, яка створює та керує бінами (компонентами) за допомогою інформації з конфігураційних файлів. Біни — це звичайні об’єкти класів, що створені за допомогою Spring Framework. Та Resource Management, що надає Spring Core для керування ресурсами, такими як рядки, файлові ресурси, URL-адреси тощо.

Spring Core є основою для роботи з іншими модулями та функціоналом Spring Framework. Він надає основні можливості для роботи з IoC та DI, що спрощує розробку додатків та полегшує їх тестування та супровід.

Незважаючи на потужність і комплексність Spring Framework, він все одно потребує значного часу та знань для налаштування та розгортання додатків написаних за допомогою Spring. Spring Boot зменшує ці зусилля за допомогою трьох важливих можливостей:

— автоконфігурації;

— opinionated підходу;

— автономності (standalone) застосунку.

Автоконфігурація означає, що програми ініціалізуються з попередньо встановленими залежностями, які не потрібно налаштовувати вручну. Оскільки Java Spring Boot має вбудовані можливості автоконфігурації, він

автоматично налаштовує базову Spring Framework і пакети сторонніх розробників на основі налаштувань, що задав розробник (і на основі найкращих практик, що допомагає уникнути помилок). Незважаючи на те, що розробник може змінити ці значення за замовчуванням після завершення ініціалізації, функція автоконфігурації Java Spring Boot дає змогу швидко розпочати розробку програм на основі Spring і зменшує ймовірність людських помилок [33].

Spring Boot використовує “opinionated” підхід до додавання та налаштування початкових залежностей на основі потреб проекту. Дотримуючись власного рішення, Spring Boot вибирає, які пакети встановити та які значення за замовчуванням використовувати, замість того, щоб вимагали від розробника самостійно приймати всі ці рішення та налаштовувати все вручну.

Ви можете визначити потреби проекту під час процесу ініціалізації, під час якого можна вибрати одну з кількох початкових залежностей, які називаються Spring Starters, і охоплюють типові випадки використання. Запускаючи Spring Boot Initializr розробнику залишається лише заповнити просту веб-форму без зайвого написання коду, щоб описати стартовий набір модулів, бібліотек та інших елементів, що необхідні у проекті.

Наприклад, початкова залежність «Spring Web» дозволяє створювати веб-програми на основі Spring з мінімальною конфігурацією, додаючи всі необхідні залежності, наприклад веб-сервер Apache Tomcat, до вашого проекту. «Spring Security» — ще одна популярна початкова залежність, яка автоматично додає до вашої програми функції автентифікації та контролю доступу [33].

Spring Boot містить понад 50 Spring стартерів і містить доступ до багатьох інших сторонніх програм.

Також Spring Boot допомагає розробникам створювати програми, які просто запускаються. Зокрема, це дозволяє створювати автономні (standalone) програми, які запускаються самі по собі, не покладаючись на

зовнішній веб-сервер, вбудовуючи веб-сервер, наприклад Tomcat або Netty, у вашу програму під час процесу ініціалізації. У результаті розробник може запустити свою програму на будь-якій платформі, просто натиснувши команду «Виконати».

Знову ж таки, найбільшими перевагами використання Spring Boot порівняно з Spring Framework є простота використання та швидша розробка. Теоретично це відбувається за рахунок більшої гнучкості, яку ви отримуєте від безпосередньої роботи з Spring Framework.

Але на практиці, якщо не потрібно, або потрібно реалізувати дуже унікальну конфігурацію, використання Spring Booth варте компромісу. У такому випадку розробники все ще можуть використовувати дуже популярну систему анотацій Spring Framework, яка дозволяє легко вводити додаткові залежності (не охоплені Spring стартерами) у програму.

Наступним необхідним модулем у розробці веб-застосунку є Spring MVC. Як випливає з назви, це модуль фреймворку Spring, який працює з шаблоном Model-View-Controller, або скорочено MVC. Він поєднує в собі всі переваги шаблону MVC із зручністю Spring фреймворку.

Spring реалізує MVC із шаблоном front-контролер за допомогою свого DispatcherServlet. DispatcherServlet діє як головний контролер для маршрутизації запитів до їхнього призначення [34]. Model — це не що інше, як дані нашої програми, а View представляє візуальні частину додатку за допомогою якого користувач взаємодіє із застосунком.

Розглядаючи основну концепцію MVC моделі можна виділити головні аспекти за які вона відповідає, а саме:

— перехоплення вхідних запитів;

— перенаправлення запиту на потрібен контролер;

— надсилання даних моделі для подальшої обробки;

— отримання оброблених даних із моделі та переміщення цих даних у view для візуалізації.

Діаграма виконання кожного із вищенаведених пунктів зображено на рисунку 2.6.

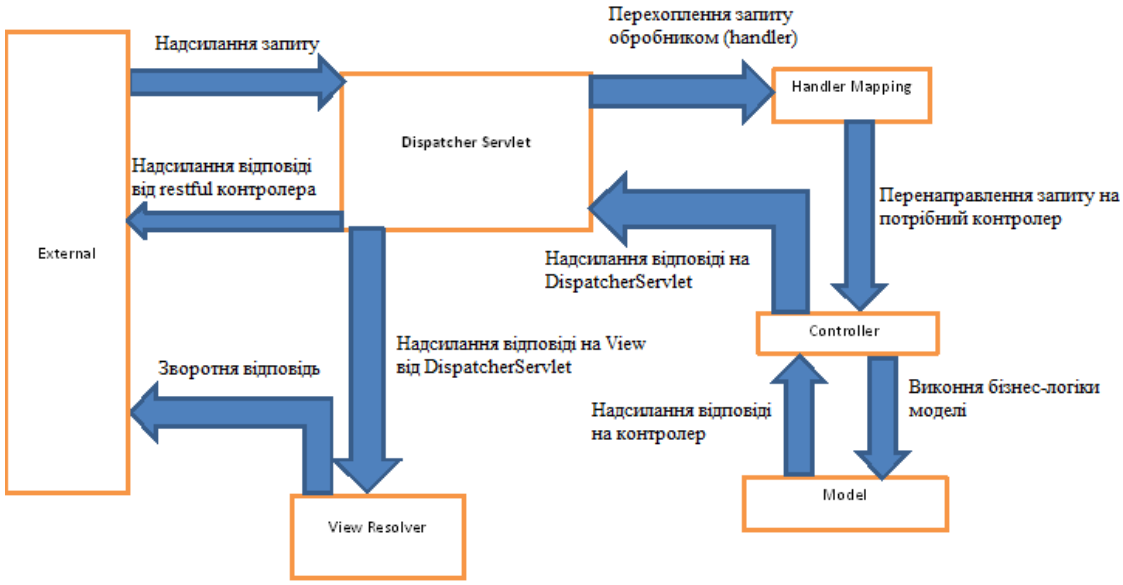

Рисунок 2.6 — Взаємодія компонентів MVC моделі між собою

Spring Data — це окремий модуль, керований Spring, який спрямований на забезпечення узгодженого рівня доступу до даних для різних сховищ даних, починаючи від реляційних баз даних і баз даних no-SQL. Spring Data надає нам специфікацію Spring Data JPA (Java Persistence API), яка представляє інструкції взаємодії з даними [34].

Завдяки Spring Data JPA нам не потрібно писати рівень доступу до даних, або писати будь-який оператор SQL взагалі. Базуючись на специфікації JPA, базова реалізація JPA дає змогу відображати об’єкти Entity та їхні метадані. Це також дозволяє перекласти відповідальність за збереження та отримування об’єктів із бази даних на менеджера об’єктів.

Spring Data JPA визначає інтерфейси сховища, які мають методи для зберігання та пошуку компонентів сутності в базі даних. У той час як інтерфейс визначає лише методи запиту, Spring під час виконання забезпечує реалізацію проксі для визначених інтерфейсів.

У фоновому режимі Spring використовує такі елементи JPA, як сутності та менеджери сутностей, але зберігає це від прямого доступу для розробників

і не дозволяє розробникам мати справу з ними напряму. Найпопулярнішою реалізацією даної специфікації є Hibernate.

Hibernate — це інструмент реляційного відображення об'єктів (ORM) з відкритим вихідним кодом, який надає структуру для відображення об'єктно-орієнтованих моделей домену в реляційні бази даних для веб-додатків.

Об'єктно-реляційне відображення базується на контейнеризації об'єктів і абстракції, яка забезпечує цю здатність. Абстракція дає змогу звертатися до об'єктів, отримувати доступ до них і маніпулювати ними, не враховуючи, як вони пов'язані зі своїми джерелами даних.

Інфраструктура Hibernate ORM керує зіставленням класів Java з таблицями бази даних і типів даних Java з типами даних SQL і забезпечує запити та пошук.

Найбільшою перевагою Hibernate виступає те, що будь-які внесені зміни інкапсулюються в самому джерелі даних, тому, коли ці джерела або їхні програмні інтерфейси (API) змінюються, програмам, які використовують ORM, не потрібно вносити зміни або навіть знати цю інформацію. Подібним чином програмісти можуть мати узгоджене уявлення про об'єкти з часом, хоча джерела, які їх доставляють, та приймачі, які їх отримують, і програми, які до них вони звертаються, можуть змінюватися.

Hibernate доступний для завантаження у вільному доступі та має ліцензію з відкритим вихідним кодом GNU Lesser General Public License (LGPL).

Наступним необхідним модулей є Spring Security. Spring Security — це потужна структура автентифікації та контролю доступу, яка легко налаштовується. Spring Security — це структура, яка зосереджена на забезпеченні як автентифікації, так і авторизації програм Java. Як і в усіх проектах Spring, справжня сила Spring Security полягає в тому, наскільки легко його можна розширити, щоб відповідати користувацьким вимогам [39].

Spring Security має численні переваги, ось деякі з них:

— комплексна підтримка автентифікації та авторизації;

— захист від типових загроз;
— інтеграція з Servlet API;
— інтеграція з Spring MVC;
— портативність;
— CSRF захист;
— підтримка конфігурації Java.

Основна функціональність Spring Security включає в себе фільтри безпеки, які обробляють кожен запит у веб-застосунку. Вони виконують різні завдання, такі як перевірка аутентичності користувача, авторизація на основі ролей, керування сесіями та безпекою переповідомлень між сторінками.

Spring Security дозволяє використовувати різні види аутентифікації, такі як форма входу, базова аутентифікація, аутентифікація з використанням зовнішніх постачальників (наприклад, OAuth, LDAP, SAML) та багато інших. Він також дозволяє налаштовувати права доступу користувачів на основі їх ролей або власних правил безпеки.

Один з ключових аспектів Spring Security — це можливість налаштовувати його для виконання на різних рівнях — від рівня веб-сервера до рівня методів контролерів. Це дозволяє забезпечити повний контроль над захистом веб-застосунків на будь-якому рівні інфраструктури застосунку.

Визначившись із мовою програмування, фреймворком та його модулями, які знадобляться у розробці веб-застосунку, наступним етапом виступає вибір вендора хмарних сервісів.

2.2.2 Вибір хмарної платформи

Постачальник хмарних послуг, або ж скорочено CSP — це IT-компанія, яка надає масштабовані обчислювальні ресурси на вимогу, наприклад обчислювальну потужність, сховище даних або програми через Інтернет. Як правило, моделі хмарних послуг визначаються як IaaS (інфраструктура як сервіс), PaaS (платформа як сервіс) або SaaS (програмне забезпечення як сервіс). Одними із найпопулярніших постачальників

хмарних послуг є компанії Amazon, Microsoft та Google, із своїми платформами Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP) відповідно.

Amazon, Microsoft та Google займають лідируючі позиції на ринку хмарних провайдерів, що надають безпечні, гнучкі та надійні хмарні послуги. Їх відповідні хмарні платформи, AWS, Azure та GCP, надають користувачам доступ до обчислюваних ресурсів та аналітичних інструментів, а також до зберігання даних, серверів, ПО тощо [35].

AWS зараз домінує в інфраструктурі, включаючи масштабоване сховище, мережеві рішення, сервери, мобільні розробки та рішення для кібербезпеки. Microsoft Azure, його головний конкурент, пропонує одні з найбільш масштабованих і ефективних програмних рішень. GCP в свою чергу пропонує високоякісні рішення для аналізу великих даних і дозволяє легко взаємодіяти з продуктами інших постачальників.

Перейдемо до порівняння вищенаведених хмарних платформ.

AWS, Microsoft Azure та Google Cloud Platform охоплюють новий цифровий світ новим стеком технологій на основі віддалених серверів. На ринку публічних хмар існує жорстка конкуренція, і ось що відрізняє кожну платформу (таблиця 2.1)

При виборі хмарного постачальника в першу чергу слід враховувати підтримувані регіони та доступність. Оскільки такі проблеми, як затримка та правила відповідності, особливо під час роботи з даними, безпосередньо впливають на продуктивність хмари. Наведемо детальні інформацію по даному аспекту для кожного хмарного постачальника. Amazon Web Service поділено на 22 географічні регіони та 14 центрів обробки даних, також є понад 114 мережевих місць та 12 регіональних мережевих кешів. Microsoft Azure працює в 54 регіонах, кожний з яких має щонайменше три зони доступності та 116 мережевих місць. Та Google Cloud Platform, що складається з 34 хмарних регіонів, 103 зон та понад 200 мережевих місць.

Таблиця 2.1 — Особливості хмарних платформ

Особливості	Amazon (AWS)	Microsoft Azure	Google Cloud
Обчислення	EC2 (Elastic Compute Cloud) - управління обчисленнями через віртуальні машини, які можуть бути налаштовані користувачем або мати готові налаштування	Створення віртуальних машин та масштабовані набори для віртуальних машин	GCE (Google Compute Engine) - віртуальні машини та їх управління
Сховище	Пропонує розподілене тимчасове зберігання, де екземпляри знищуються в кінці свого життєвого циклу	Використання ID-накопичувачів, Page Blobs для обсягів на основі VM, а також опції Block Storage для об'єктного сховища	Пропонує як тимчасове, так і постійне сховище, включаючи Google Cloud Storage для об'єктного сховища

Наступним компонентом аналізу хмарних платформ є їхня вартість, що зображено у таблиці 2.2 [36].

Порівнявши основні характеристики можемо перейти до переваг та недоліків, які присутні у кожної хмарної платформи.

Переваги AWS: надає найбільшу кількість послуг; вважається найкращим щодо надійності та безпеки; більша обчислювальна потужність, ніж у Azure та GCP; вважається більш досвідченішим провайдером хмарних послуг, ніж інші провайдери. Недоліки AWS: всі великі постачальники програмного забезпечення, які роблять свої програми доступними на AWS Dev/Enterprise, мають підтримку, за яку потрібно платити; велика кількість

послуг та опцій може бути пригнічувальною для новачків; відносно мало альтернатив гібридного хмарного середовища.

Таблиця 2.2 — Порівняння вартості використання хмарних платформ

Тип машини	AWS	Azure	GCP
Найменший екземпляр	AWS вимагає приблизно 69 доларів США за місяць за основний екземпляр із двома віртуальними процесорами та восьмома гігабайтами оперативної пам'яті.	У Azure, такий самий тип екземпляра, тобто екземпляр із 2 процесорами та 8 ГБ ОЗП, коштує приблизно 70 доларів США на місяць.	Порівняно з AWS, GCP надасть найбазовіший екземпляр із двома віртуальними процесорами та восьмома гігабайтами оперативної пам'яті за 25% менше вартості. Це обійдеться приблизно в 52 долари США щомісяця.
Найбільший екземпляр	Найбільш дорогий екземпляр AWS із 3,84 ТБ оперативної пам'яті та 128 процесорами обійдеться вам приблизно 3,97 доларів США за годину.	Найбільший екземпляр Azure має 3,89 ТБ оперативної пам'яті та 128 процесорів. Вартість становить близько 6,79 доларів США за годину.	GCP лідирує завдяки своєму найбільшому екземпляру із 3,75 ТБ оперативної пам'яті та 160 процесорами. Це обійдеться вам приблизно в 5,32 долара США за годину.

Переваги Microsoft Azure: інтеграція та міграція поточних послуг Microsoft є простими; багато доступних опцій, включаючи найкращі сервіси штучного інтелекту, машинного навчання та аналітики; більшість послуг мають меншу вартість у порівнянні з AWS та GCP; є великий рівень підтримки для стратегій гібридного хмару. Недоліки використання Microsoft

Azure: менше варіантів послуг порівняно з AWS; спеціально призначений для корпоративних клієнтів.

Перевагами GCP: добре працює з іншими сервісами та продуктами Google; відмінна підтримка контейнеризованих робочих навантажень. Недоліки GCP: обмежені послуги порівняно з AWS та Azure; обмежена підтримка випадків використання для підприємств.

Зважаючи на перераховані переваги та недоліки кожної із платформ, можна дійти висновку, що найкращою хмарною платформою сервісів виступає Amazon Web Services. Оскільки AWS пропонує широкий набір послуг, включаючи обчислення, зберігання, бази даних, машинне навчання та інші, що дає можливість обирати сервіси, які відповідають конкретним потребам веб-застосунку. Крім того, AWS відомий своєю надійністю та безпекою завдяки мережевим та дата-центрам, спроектованим для мінімізації можливих відмов. Платформа також надає гнучкість масштабування, дозволяючи змінювати ресурси в залежності від навантаження застосунку, що допомагає оптимізувати витрати.

2.2.3 Вибір хмарних послуг та ресурсів

Визначившись із провайдером хмарних сервісів, потрібно виокремити конкретні послуги та ресурси, які використовуватимуться в розроблювальному веб-застосунку, а саме:

— Amazon S3;

— Amazon EC2;

— Amazon RDS.

Кожний із вище перелічених сервісів призначений для розв’язання конкретного задання, отже розглянемо детальніше кожен із них.

Amazon Simple Storage Service (Amazon S3) — це служба зберігання об’єктів, яка пропонує найкращі в галузі масштабованість, доступність даних, безпеку та продуктивність. Клієнти будь-якого розміру та галузі можуть використовувати Amazon S3 для зберігання та захисту будь-яких

обсягів даних для різноманітних випадків використання, веб-сайти, мобільні програми, резервне копіювання та відновлення, архівування, корпоративні програми. Amazon S3 надає функції керування, щоб оптимізувати, упорядковувати та налаштовувати доступ до даних відповідно до конкретних бізнес-вимог, організаційних вимог і вимог відповідності [37].

Amazon S3 має функції керування сховищем, за допомогою яких можна керувати витратами, відповідати нормативним вимогам, зменшувати затримку та зберігати кілька окремих копій даних.

Також Amazon S3 надає функції для аудиту та керування доступом до створеного бакету (bucket), що представляє з себе загальну сегмент, який зберігає в собі дані, і сласне об’єктів, що знаходяться в нього. За замовчуванням сегменти S3 та об’єкти в них є приватними. Розробник має доступ лише до створених ним ресурсів S3. Щоб надати детальні дозволи на ресурси, або перевірити дозволи ваших ресурсів Amazon S3, використовуються такі функції як S3 Block Public Access та AWS Identity and Access Management (IAM).

Структуру сервісу S3 можемо спостерігати на рисунку 2.7.

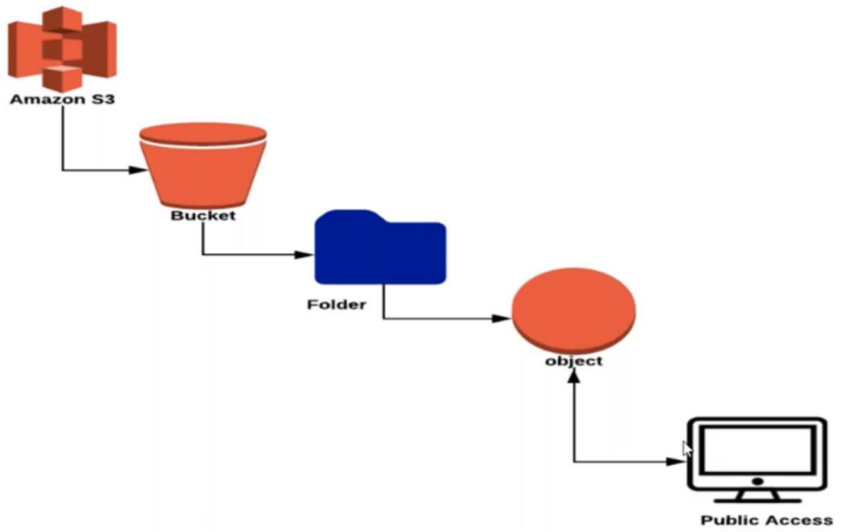

Рисунок 2.7 — Структура Amazon S3

Amazon RDS (Relational Database Service) — це керований сервіс баз даних, спрямований на спрощення налаштування, експлуатації та

масштабування реляційних баз даних у хмарному середовищі. Він надає можливість легко створювати та керувати реляційними базами даних, такими як MySQL, PostgreSQL, Oracle, SQL Server і Amazon Aurora, без необхідності в адмініструванні апаратного забезпечення або встановленні та налаштуванні баз даних.

RDS дозволяє розгортати бази даних в хмарному середовищі за кілька хвилин. Він автоматично керує резервними копіями, патчами програмного забезпечення та розширенням ресурсів, що робить процес адміністрування баз даних більш простим та менш часо- та ресурсозатратним.

Amazon RDS також забезпечує високий рівень захисту даних, використовуючи шифрування для даних. Крім того, він надає інструменти для моніторингу та збору логів, що дозволяє вам відстежувати продуктивність та виконання запитів до бази даних.

Для користувачів з великими обсягами даних Amazon RDS може бути вигідним завдяки можливості використання різних типів сховищ (наприклад, SSD або HDD), що дозволяє оптимізувати швидкодію та вартість зберігання даних відповідно до потреб проєкту [38].

Структура взаємодії з Amazon RDS зображено на рисунку 2.8.

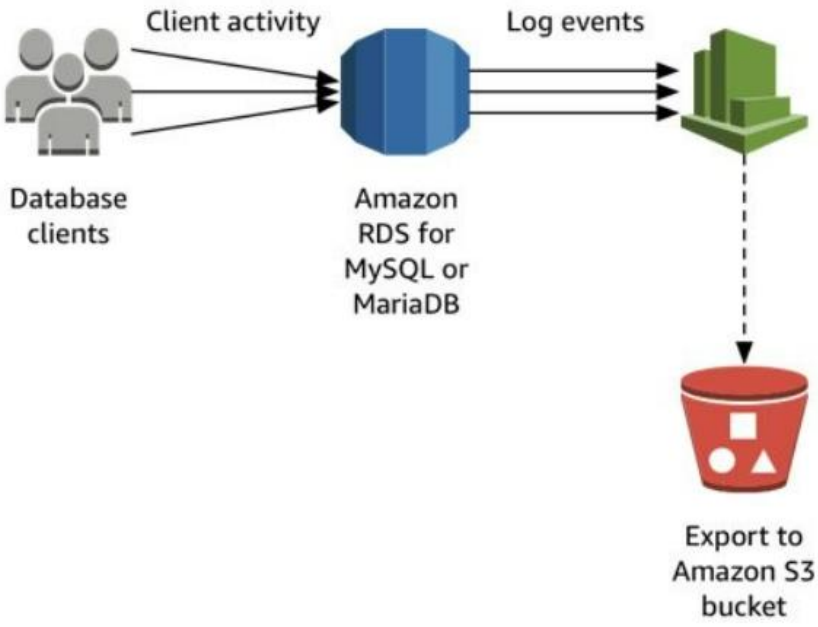

Рисунок 2.8 — Структура взаємодії з Amazon RDS

Amazon EC2 (Elastic Compute Cloud) — це веб-сервіс, що надає можливість орендувати віртуальні сервери та обчислювальні ресурси в хмарному середовищі AWS. Цей сервіс дозволяє користувачам виробляти, розгортати та масштабувати обчислювальні ресурси на основі їхніх потреб та вимог проекту, не маючи необхідності власного обладнання або фізичного серверного приміщення.

EC2 пропонує широкий вибір конфігурацій обчислювальних ресурсів, включаючи різні типи процесорів, обсяги пам'яті, потужності обчислень та типи зберігання даних. Це дозволяє користувачам налаштовувати серверні ресурси для підтримки різних видів додатків та завдань.

Однією з основних переваг EC2 є гнучкість масштабування. Користувачі можуть змінювати розмір своїх серверів, додавати або зменшувати кількість віртуальних машин відповідно до потреб їхніх проєктів. Це дозволяє ефективно керувати обчислювальними витратами та забезпечує підтримку підвищеної навантаженості під час пікових навантажень

EC2 також надає широкий спектр інструментів для керування та моніторингу обчислювальними ресурсами. Користувачі можуть використовувати AWS Management Console для управління своїми віртуальними машинами, автоматизувати розгортання та керування засобами інфраструктури за допомогою інструментів командного рядка або API.

EC2 забезпечує високий рівень безпеки та надійності. AWS надає інструменти для захисту даних, включаючи можливість шифрування та контролю доступу до віртуальних серверів. Крім того, сервіс гарантує високу доступність та резервне копіювання, що забезпечує надійність і безперебійну роботу серверів.

Еластичні можливості оплати є ще однією перевагою EC2. Користувачі мають можливість сплачувати лише за використані обчислювальні ресурси за фактичний час їхнього використання, що дозволяє ефективно управляти витратами на інфраструктуру [39].

Використовуючи дані хмарні сервіси буде реалізовано серверну частину веб-застосунку.

2.4 Технології створення клієнтської частини застосунку

Розробивши серверну частину застосунку потрібен інтерфейс за допомогою якого користувачі зможуть взаємодіяти із застосунком. Для цієї задачі потрібно реалізувати клієнтську частину застосунку, для цього існує велика кількість технологій, серед яких є:

— Angular;

— Vue.js;

— React.

Розглянемо кожен із доступних інструментів, щоб обрати той що найкраще буде відповідати вимогам веб-застосунку. Для цього також розглянемо деяку статистичну інформацію про дані інструменти. Найпростіше різниця між ними, що React — це бібліотека інтерфейсу користувача, а Vue — “progressive” фреймворк, що представляє з себе

структуру реалізовану як додаткову розмітку до HTML. Однак Angular — це повноцінний інтерфейсний фреймворк. Згідно з опитуванням StackOverflow 2022, React є улюбленим фреймворком 40,14% розробників, Angular — 22,96%, а Vue — 18,97% розробників [39].

Angular — це повноцінний фреймворк веб-додатків на основі TypeScript, безкоштовний і відкритий. Він широко використовується для створення односторінкових додатків. Він був спочатку запущений Google у 2016 році як продовження AngularJS, який був випущений у 2010 році.

На початкових етапах, коли Angular вийшов на ринок, він привернув увагу багатьох розробників завдяки своїм чудовим характеристикам, а також його підтримує технічний гігант Google. Багато людей плутають Angular з AngularJS, який пізніше став першою версією Angular, запущеною Google. Angular використовує TypeScript для розробки, тоді як AngularJS використовує JavaScript. Різкий перехід від JavaScript до TypeScript розчарував багатьох людей, оскільки JavaScript є більш відомою мовою, ніж TypeScript.

Серед переваг Angular відмічають його надійність, оскільки Angular підтримується Google, розробники мають величезну довіру для створення великомасштабних програм, знаючи, що це буде підтримуватися в довгостроковій перспективі. Документація Angular є детальною та дуже добре пояснює функціональність і роботу Angular. Та легше масштабування проекту, включають розробку у великім команді розробників.

Але серед переваг знаходяться і обмеження Angular: Angular вимагає навичок роботи з TypeScript, якому, згідно з опитуванням StackOverflow, віддають перевагу 30% розробників, тому вивчення Angular вимагає більше зусиль, ніж інші фреймворки.

React — це інтерфейсна бібліотека JavaScript з відкритим кодом, яка використовується для розробки інтерфейсів користувача на основі компонентів. Він був розроблений Facebook у 2013 році і зараз підтримується спільнотою з відкритим кодом і Facebook.

Це найпоширеніша бібліотека на основі JavaScript через легкість створення веб-додатків. Крім того, окрім створення інтерфейсу користувача, він пропонує кілька функцій, таких як Flux і React Native.

Одна з найбільших переваг використання React полягає в тому, що ви можете створювати як веб-, так і мобільні додатки, використовуючи ту саму кодову базу з фреймворком під назвою React Native.

Кожен компонент у додатку React служить будівельним блоком, що робить додаток придатним для повторного використання. Повторно використовувані компоненти в JavaScript скорочують час і складність розробки.

React часто є найпопулярнішим вибором серед розробників. Однак перш ніж вибрати React, давайте вивчимо переваги та недоліки його використання. Переваги використання React: багаторазові компоненти, продуктивність.

Під час розробки додатків за допомогою React складність та сама кількість кодування зменшується через багаторазові компоненти в React. Це принесло більше функціональності та чітку кодову базу, яку також легко підтримувати. React не залежить від традиційної DOM і використовує структуру JavaScript, віртуальну DOM. Використання віртуального DOM покращує продуктивність і збільшує швидкість програм.

До наявних недоліків React можна віднести аспект із його постійним оновленням, що спонукає розробникам заново вивчати нові концепції та адаптуватися до темпу React.

Vue — це прогресивний зовнішній фреймворк із відкритим вихідним кодом, який набув широкої популярності для створення односторінкових програм (SPA). SPA — це веб-програми, які мають лише один файл HTML, у якому відображається весь код. Він був випущений у 2014 році та привернув увагу розробників завдяки своєму простому інтерфейсу та легкому навчанню.

Вивчити Vue здавалося легшим, ніж більшість існуючих на той час технологій. Angular надихнув творців Vue відкрити Vue, заснований на об’єднанні всіх найкращих частин Angular під одним дахом і нехтуванні всіма обмеженнями Angular для створення високоефективного інструменту. Vue зосереджується на розробниках-початківцях, допомагаючи їм створювати динамічні веб-програми, не потребуючи попереднього виснажливого навчання.

До переваг Vue можна навести те, що для його вивчення достатньо не великого проміжку часу, коли у React і Angular ви повинні володіти TypeScript і JavaScript відповідно. Однак Vue зручний для початківців і не вимагає жодних попередніх навичок. Але попри це Vue.js має деякі недоліки, серед яких: менша підтримка порівняно з іншими популярними фреймворками, обмеженіший набір інструментів у порівнянні з React або Angular, проблеми з масштабованістю у складних проєктах, можливі проблеми з управлінням структурою проєкту через відсутність строгих правил та меншу кількість ресурсів для новачків.

Порівнявши Angular, React та Vue.js можна дійти висновку, що найкраще для реалізації клієнтської частини підходить React. Оскільки Vue.js володіє обмеженим набором інструментів, а написання більш масштабного застосунку, який розробляється, написання компонентів за допомогою Angular може принести ускладнення у розробці.

3 РОЗРОБКА ВЕБ-ЗАСТОСУНКУ

3.1 Загальна структура веб-застосунку

Згідно із вимогами користувачів, архітектура веб-застосунку для навчального процесу включає 8 основних сторінок, кожна з яких має свою власну функціональність для користувачів з різними ролями (викладачем або студентом). Блок схема структури веб-застосунку представлена у додатку Б.

Перша сторінка у системі — це "Авторизація". Це місце, де користувач може увійти до свого особистого облікового запису у системі оцінювання, використовуючи свою електронну адресу та пароль. Якщо користувач не має облікового запису у системі, він може перейти на сторінку реєстрації.

Друга сторінка — "Реєстрація", де користувач може заповнити необхідні поля для створення свого облікового запису. Ця сторінка, так само як і сторінка авторизації, містить валідацію полів вводу, тобто перевірку даних, введених користувачем. Якщо дані введено некоректно, сторінка буде оновлена, а користувач буде повідомлений про помилку.

Після успішної реєстрації користувача в системі його буде перенаправлено на сторінку авторизації для входу у систему.

Друга сторінка — «Сторінка користувача». Після введення коректних даних для входу, користувач буде знаходитись на власній сторінці, за допомогою якої можна редагувати власні дані як користувача системи, та яка дозволить обрати роль, виходячи з якої буде визначено, яка функціональність буде доступна. Залежно від цієї ролі користувача буде перенаправлено на відповідну сторінку. Якщо користувач має роль "Викладач", його перенаправлять на сторінку для викладачів; якщо ж він є "Студентом", то на сторінку для студентів. Користувач водночас може бути і студентом і викладачем, але у різних навчальних курсах. Випадок, коли користувач є вчителем та студентом в одному і тому ж курсі відсутній.

Четверта сторінка — "Сторінка викладач/студента". Залежно від ролі, яку обрав користувач на власній сторінці дана сторінка відрізнятиметься для

кожної ролі. Якщо користувач має роль "Викладача", він може створювати нові навчальні курси та переглядати вже створені. У випадку ролі "Студента", він може переглядати курси, до яких приєднався, та знаходити необхідний навчальний курс за допомогою пошукової стрічки "Search", яка знаходить збіги за назвою курсу.

Наступні сторінки будуть відокремлені для кожної ролі користувача. П’ята сторінка — "Навчальний курс". Ця сторінка не суттєво відрізняється для користувача з ролями "Студент" та "Викладач". В обох випадках користувачі бачать список завдань, пов'язаних з курсом, які можна переглянути, натиснувши на назву завдання, і вони мають можливість переглядати результати. Також на цій сторінці для "викладача" відкрита можливість створювати та змінювати завдання курсу. Дії (створення та редагування) доступні на окремих сторінках, які мають відповідну функціональність для цих дій.

Шоста сторінка — "Результати". Ця сторінка значно відрізняється для "студента" та "викладача": "студент" має доступ лише до власних результатів, а "викладач" бачить результати всіх студентів з можливістю їх редагування.

Сьома сторінка — "Завдання". Дана сторінка може приймати різний вигляд виходячи з типу завдання, який був обраний раніше. Завдання може приймати один із наступних типів: «Завдання», «Тест» та «Корисні матеріали». «Завдання» представляє з себе умову, яку потрібно виконати, прикріплені корисні матеріали для поставленої задачі та поле у яке потрібно вказати правильну відповідь, або прикріпити файл з розв’язанням завдання. «Тест» в свою чергу передбачає з себе форму, яка містить перелік запитань (завдань) на які потрібно дати правильну відповідь обравши із запропонованого переліку відповідей. «Корисні матеріали» містить матеріали, які можуть відноситись до навчальному курсу, або ж кількох завдань водночас. Даний тип «завдання» досить зручний для зберігання файлів, які можуть нести користь під час проходження курсу.

Восьма сторінка — "Відповіді". Ця сторінка доступна лише користувачам з роллю "Викладач". Вона доступна зі сторінки "Завдання" і дозволяє переглядати відповіді, які надали студенти для вирішення завдання. Відповідно до типу завдання дана сторінка може приймати різний вигляд. Якщо обране завдання є тестом, то викладачу буде доступний перегляд відповідей на кожне із завдань, а оцінка до даного завдання буде обраховано автоматично, якщо попередньо було визначено відповіді до кожного із запитань, яке представлене у тесті. У випадку, якщо обране завдання є звичайним завданням, яке очікує короткої відповіді, або ж очікує значного розв’язку, який представлений у файлі, який прикріплює студент, то викладач має можливість переглянути даний розв’язок у браузері, або ж завантажити на власний пристрій, та поставити потрібну оцінку конкретному студенту.

Розглянувши вимови та потреби веб-застосунку для навчання та аналізу навчального процесу перейдемо до безпосередньої його реалізації.

3.2 Розробка клієнтської частини

Розробка клієнтської частини веб-застосунку включає створення користувацького інтерфейсу, з яким користувачі будуть взаємодіяти. Це вимагає розробки функціональних компонентів, які відображають дані та реагують на дії користувачів. Під час розробки клієнтської частини враховуються дизайнерські концепції та потреби користувачів для створення інтуїтивно зрозумілого та зручного у використанні інтерфейсу. У додатку В можемо спостерігати зв’язок клієнтської та серверної частин веб-застосунку

Також у процесі розробки важливо забезпечити взаємодію з сервером, використовуючи API для отримання та відправки даних. Комунікація з сервером здійснюється за допомогою запитів HTTP, які обмінюються інформацією між клієнтом та сервером.

У розробці клієнтської частини було використано середовище програмування Visual Studio Code. В контексті розробки клієнтської частини

веб-застосунку VC Code надає величезні можливості та зручності роботи з React. Це інтегроване середовище розробки пропонує розширені інструменти для написання коду, включаючи автодоповнення, підсвічування синтаксису та швидку навігацію між файлами.

Особлива перевага полягає у можливості використання розширень, спеціально розроблених для React, таких як React Developer Tools. Ці додатки спрощують відлагодження та перегляд React-компонентів, що дозволяє зосередитися на самому процесі розробки.

Варто відзначити також використання ESLint, як інструменту статичного аналізу коду для виявлення проблемних шаблонів, та react-хуків. Цей інструмент дозволяє перевіряти відповідність стандартам, виявляти синтаксичні помилки та потенційні проблеми. Важлива можливість — налаштування правил лінтингу з урахуванням потреб проекту чи команди розробників. ESLint виявляє неправильне використання змінних, синтаксичні помилки та інші потенційні проблеми в коді.

В свою чергу, хуки в React — це функції, які дозволяють використовувати стан та інші можливості React в функціональних компонентах. Основна перевага хуків — вони дозволяють використовувати стан та інші функціональності React без необхідності використання класових компонентів. Наприклад, useState — це хук, що приймає початкове значення стану та повертає масив, що містить поточне значення стану та функцію для його зміни.

Хуки також дозволяють використовувати ефекти, такі як `useEffect`, для роботи з побічними ефектами у функціональних компонентах. Наприклад, виконати певні дії при завантаженні компонента, або при зміні певних значень. Окрім цього, React має інші вбудовані хуки, такі як «useContext» для роботи з контекстом, та «useReducer» для керування складним станом компонентів та інші.

Завдяки хукам React стає більш гнучким та зручним для роботи зі станом, ефектами та іншими функціональностями, що раніше були доступні

лише через класові компоненти. У розробці клієнтської частини використання веб-застосунку можна прослідкувати постійне використання react хуків.

Визначений інтерфейс через який користувачі зможуть взаємодіяти із веб-застосунком і початковим етапом є саме авторизація, аутентифікація та реєстрація користувача у систему. Використовуючи React бібліотеку визначимо компонент, який дозволить представити необхідну форму для поставленої задачі. Даний компонент реєстрації має містити поля: ім’я користувача, прізвище, електрону пошту та пароль до неї. Сутність, яка містить дані поля представлена у листингу 3.1.

Лістинг 3.1 — Компонент для збереження даних користувача

```
interface User {
   id: number | null;
   firstName: string;
   lastName: string;
   email: string;
   password: string;
}
```

Підключивши css-стилі до компонентну реєстрації отримуємо наступну форму (рисунок 3.1).

Рисунок 3.1 — Форма реєстрації

Після ведення коректних даних у форму та натискання на кнопку «Register», викликається метод handleRegister() за допомогою якого буде надіслано запит на потрібний ендпоінт та збережено дані користувача у базі даних (лістинг 3.2).

Лістинг 3.2 — Надсилання запиту на реєстрацію користувача

```
const handleRegister = async () => {
  try {
    const response = await axios.post(`/api/users`, userData);
    window.location.href = response.headers.location;
  } catch (error) {
    console.error('Registration failed:', error);
  }
};
```

Після успішної реєстрації користувача у систему його буде перенаправлено на сторінку авторизації з необхідною формою (рисунок 3.2).

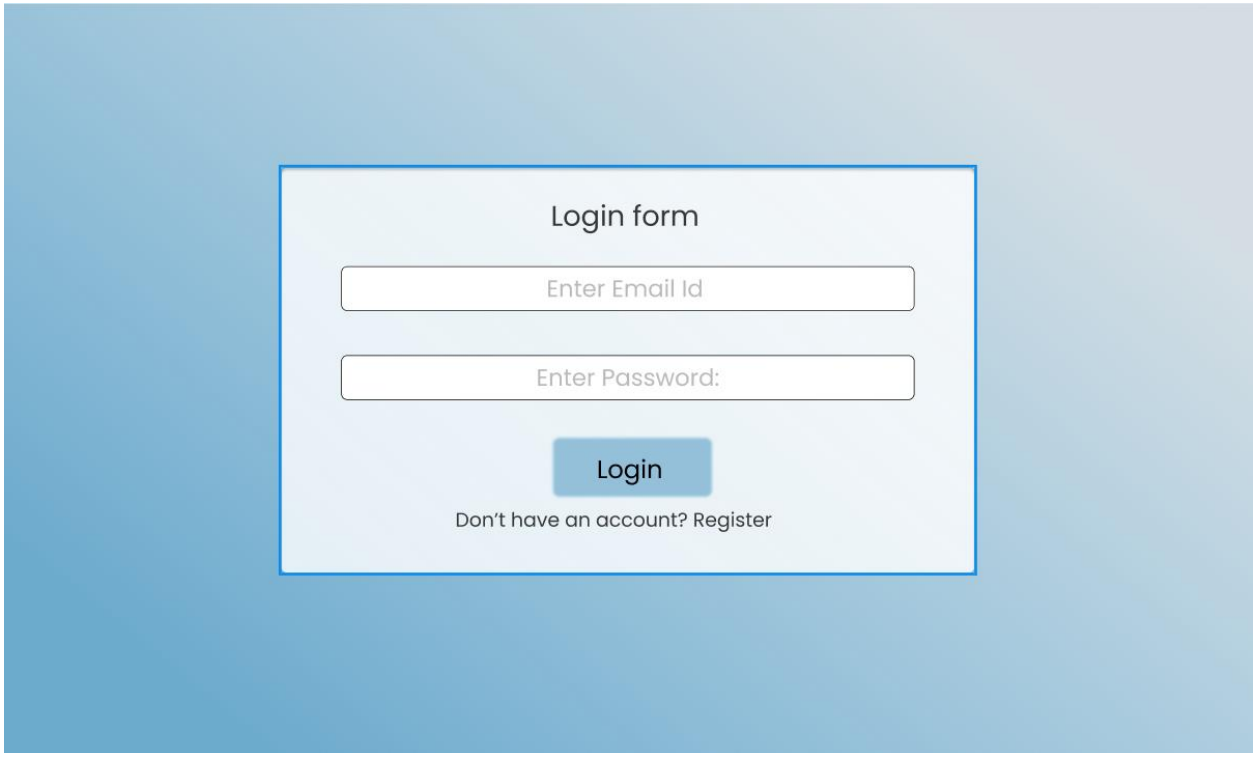

Рисунок 3.2 — Форма авторизації користувача у систему

Наступним етапом після авторизації, користувачу стає доступна можливість створювати власні курси навчання, маючи при цьому роль «Викладач», або проходити курси, які були створенні іншими користувача, переймаючи роль «Студента».

3.2 Розробка серверної частини

Розробка серверної частини веб-застосунку включає у себе створення java-додатку, який буде відігравати роль сервера на який будуть приходити запити, створення бази даних, підключення сховища для зберігання файлів користувачів та розгортання даного застосунку у хмарі. У додатку Г представлено схему взаємодії хмарних сервісів між собою.

Скориставшись Spring Boot Initializr, що представляє з себе легкий, швидкий спосіб створення нового додатку з автоматичним налаштуванням компонентів, створюємо java-додаток (рисунок 3.3).

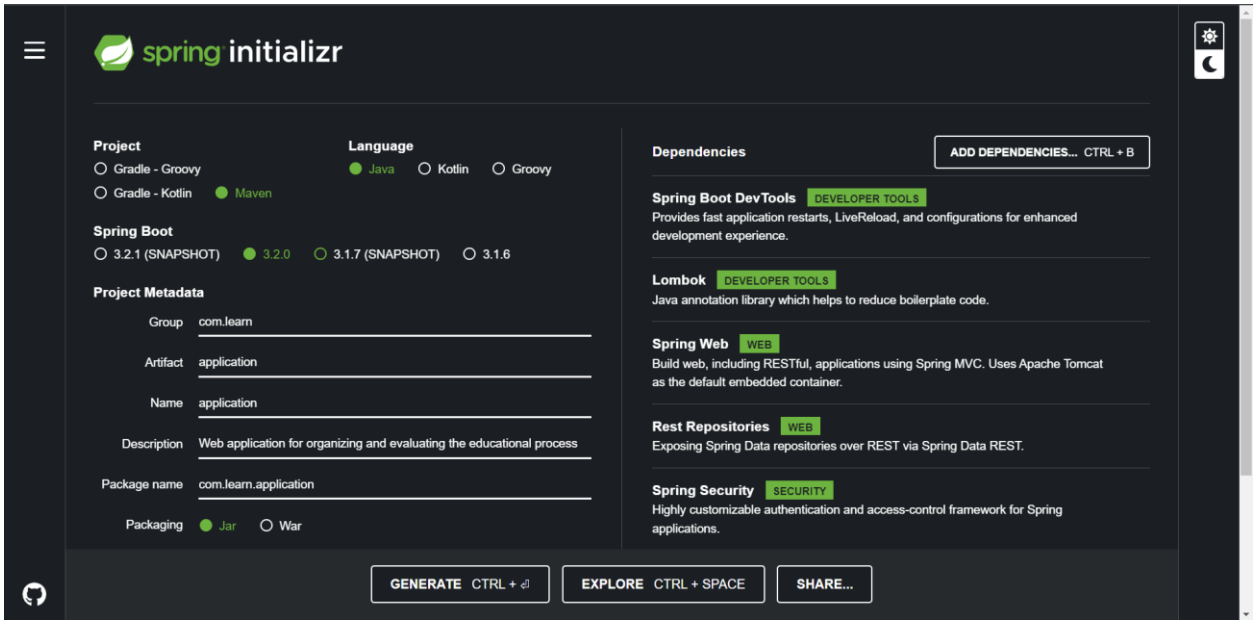

Рисунок 3.3 — Створення java-додатку

На рисунку 3.3 можемо побачити частину залежностей, які необхідні у розробці веб-застосунку. Оскільки використовується Maven, то головним файлом у нашому проекті є файл *pom.xml*. Цей файл відповідає за об’єктну

модель проекту, тобто він створює відповідні «залежності» («dependencies») між різними модулями проекту для їхньої коректної роботи. Нижче приведено лістинг, що містить частину залежностей (лістинг 3.3).

Лістинг 3.3 — Залежності між модулями додатку

```
<dependency>
   <groupId>software.amazon.awssdk</groupId>
   <artifactId>s3</artifactId>
   <version>2.20.52</version>
</dependency>
<dependency>
   <groupId>com.amazonaws</groupId>
   <artifactId>aws-java-sdk</artifactId>
   <version>1.0.12</version>
</dependency>
<dependency>
   <groupId>com.amazonaws</groupId>
   <artifactId>aws-java-sdk-s3</artifactId>
   <version>1.12.537</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
 </dependency>
```

Весь код pom.xml наведено у додатку Д.

Після налаштування необхідних залежностей переходимо до етапу написання коду. Проект розділений на різні логічні модулі, кожен з яких відповідає за певну частину функціоналу та зберігається у відповідній папці.

Це створює структуру каталогів, яка ілюструється на рисунку 3.4. Лістинг серверної частини веб-застосунку зображено у додатку Е.

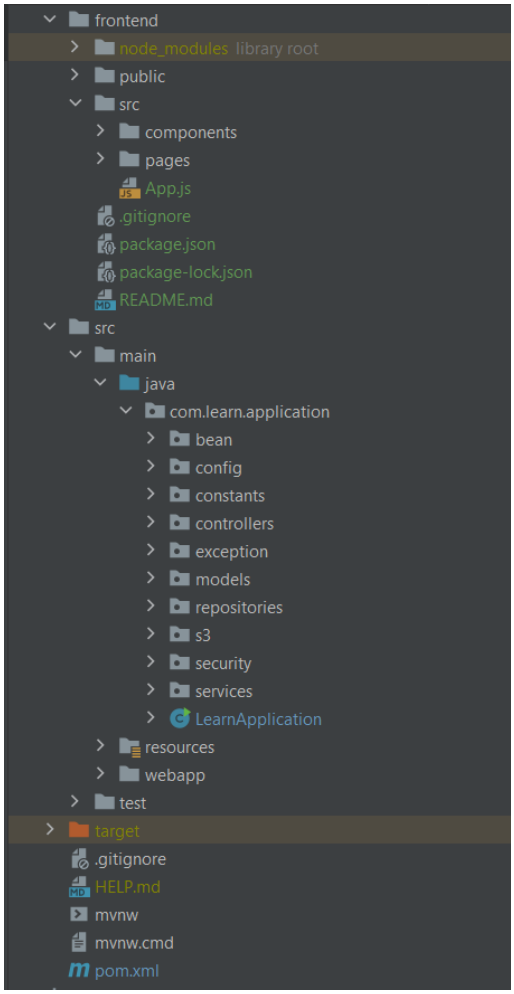

Рисунок 3.4 — Папкова структура застосунку

Кожен каталог містить файли, які відповідають за певну функціональну частину застосунку:

— frontend/src містить код React, розділений на компоненти, сторінки та інші файл;

— bean включає код, який відповідає за перевірку правильності завантаження файлів та роботу з сесією;

— config містить конфігурацію системи безпеки;

— controllers містить контролери, що визначають, яке "view" повернути користувачу;

— exception включає класи, які обробляють виняткові ситуації, що можуть виникнути під час роботи;

— models містить класи, які представляють сутності бази даних;

— s3 каталог представляє класи та інтерфейси для роботи з хмарним сервісом Amazon S3;

— repositories розміщує в собі класи репозиторіїв, які дозволяють взаємодіяти з базою даних;

— security каталог містить налаштування процесу авторизації та аутентифікації;

— services містить класи сервісів для роботи з даними, що надходять з репозиторія.

Після реалізації класів, що виконують вище згадані функції залишається налаштувати роботу із хмарними сервісами, а саме з Amazon S3, Amazon RDS та Amazon EC2.

Для зв’язування застосунку з хмарним сховищем S3 потрібно внести необхідні залежності у pom.xml, що було зроблено при створенні самого додатку, створити bucket у якому будуть зберігатися файли та підключити S3 сховище до проекту.

Для створення бакету потрібно авторизуватися у систему Amazon та обрати сервіс Amazon S3 (рисунок 3.5)

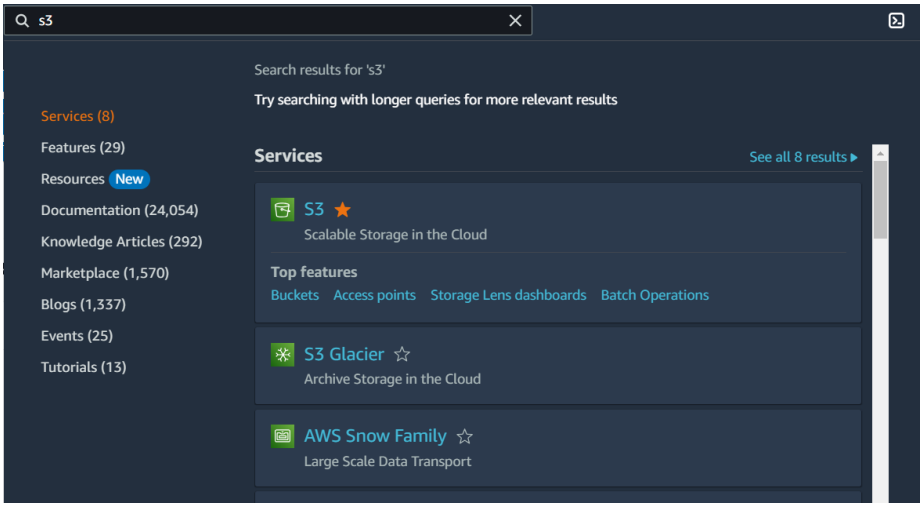

Рисунок 3.5 — Вікно пошуку Amazon сервісів

Після того як нас буде перенаправлено у сервіс Amazon S3 з'являється можливість створити необхідний bucket при натисканні на відповідну кнопку (“Create Bucket”). Після цього буде представлена форма за допомогою якої ми можемо створити bucket (рисунок 3.6).

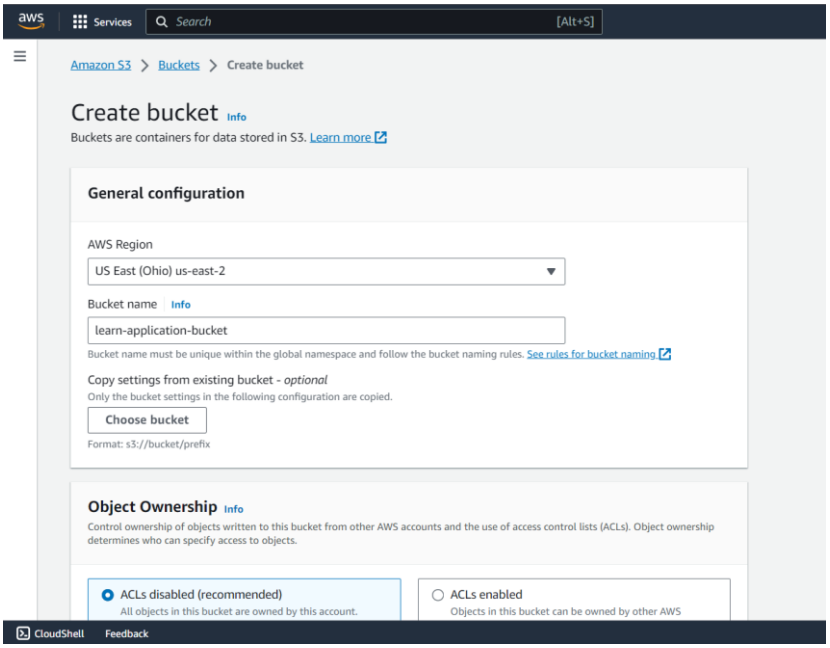

Рисунок 3.6 — Вікно створення Amazon S3 бакету

Після введення назви бакету, налаштування доступу до нього та інших налаштування, бакет буде створено (рисунок 3.7) та залишається під’єднати його до java додатку, щоб забезпечити роботи з ним.

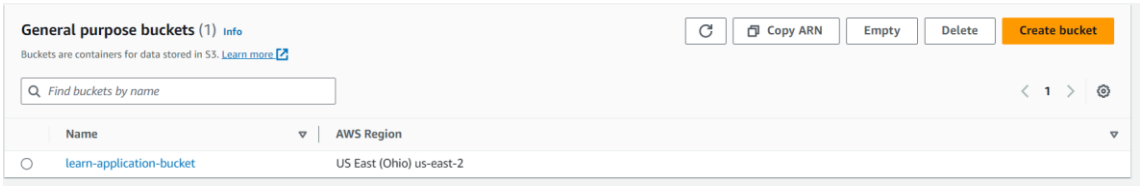

Рисунок 3.7 — Бакет для збереження файлів користувачів застосунку

Для початку потрібно створити підключення клієнта для доступу до веб-служби Amazon S3. Для цього використовується інтерфейс AmazonS3, а саме його реалізації BasicAWSCredentials — визначення введення для входу у

систему Amazon (’AWS accesskey’ та ’AWS secretkey’), та AmazonS3ClientBuilder — реалізація доступу до сервісу S3 (лістинг 3.4).

Лістинг 3.4 — Підключення S3 сервісу

```
AmazonS3 s3client = AmazonS3ClientBuilder
 .standard()
 .withCredentials(new AWSStaticCredentialsProvider(credentials))
 .withRegion(Regions.US_EAST_2)
 .build();
```

Після підключення S3 сервісу за допомогою java-коду з’являється можливість роботи з необхідним бакетом, а саме додавання нових об’єктів у сховище, видалення, редагування та видобування необхідних файлів з сховища у проект для потреб користувачів.

Наступним етапом реалізації веб-застосунку є підключення сервісу Amazon RDS. За допомогою Amazon RDS сервісу ми можемо визначити структуру бази даних даного веб-застосунку та керувати її сутностями для роботи з персональними даними користувачів. На рисунку 3.8 зображено створення бази даних за допомогою сервісу Amazon RDS.

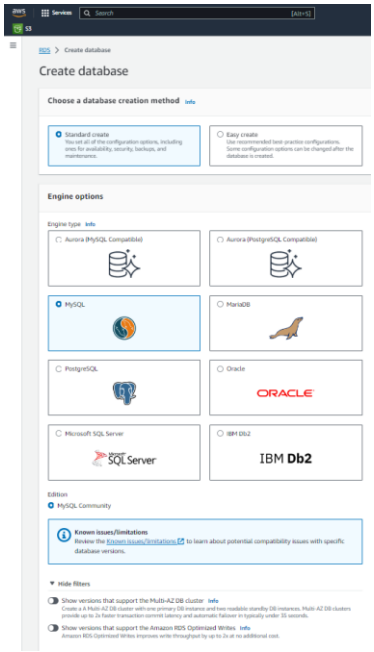

Рисунок 3.8 — Створення бази даних за допомогою Amazon RDS

Після необхідних налаштувань база даних буде створена (рисунок 3.9).

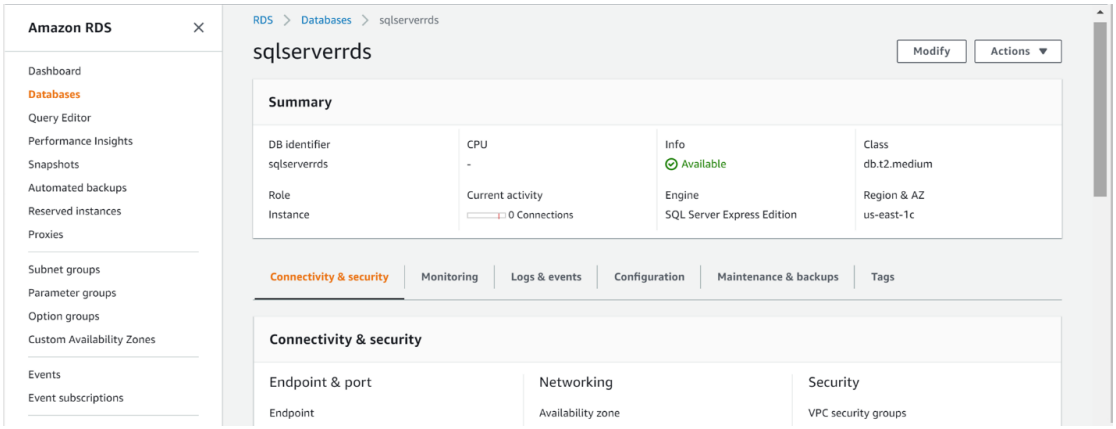

Рисунок 3.9 — Створена база даних в сервісі Amazon RDS

Створивши базу даних у сервісі Amazon RDS необхідно визначити її структуру. Структурна схема бази даних зображено у додатку Ж. Відповідно до визначеної структури створюємо необхідні таблиці. Після створення необхідних таблиць потрібно зв’язати базу даних з веб-застосунком за допомогою java-коду. По аналогії з сервісом Amazon S3, підключення сервісу RDS потребує необхідної залежності в файлі pom.xml, а саме “software.amazon.awssdk”. Наступним кроком є підключення сервісу за допомогою клієнта, що представлено у лістингу 3.5.

Лістинг 3.5 — Підключення Amazon RDS сервісу за допомогою коду

```
RdsClient rdsClient = RdsClient.builder()
    .region(Region.AP_SOUTHEAST_2)
    .credentialsProvider(ProfileCredentialsProvider.create("default"))
    .build();
Properties prop = new Properties();
InputStream input =
AwsRds.class.getClassLoader().getResourceAsStream("db.properties");
prop.load(input);
String db_hostname = prop.getProperty("db_hostname");
String db_username = prop.getProperty("db_username");
```

```
String db_password = prop.getProperty("db_password");
String db_database = prop.getProperty("db_database");
Connection conn = DriverManager.getConnection(jdbc_url, db_username, db_password);
```

Виконавши підключення необхідних хмарних сервісів до застосунку, залишається його розгорнути на потрібному сервері за допомогою сервісу Amazon EC2. Обравши даних сервіс та написнувши на кнопку «Create New Instance» відкриється форма для створення екземпляру серверу на якому розгортатиметься веб-застосунок. На рисунку 3.10 можемо спостерігати налаштування даного серверу.

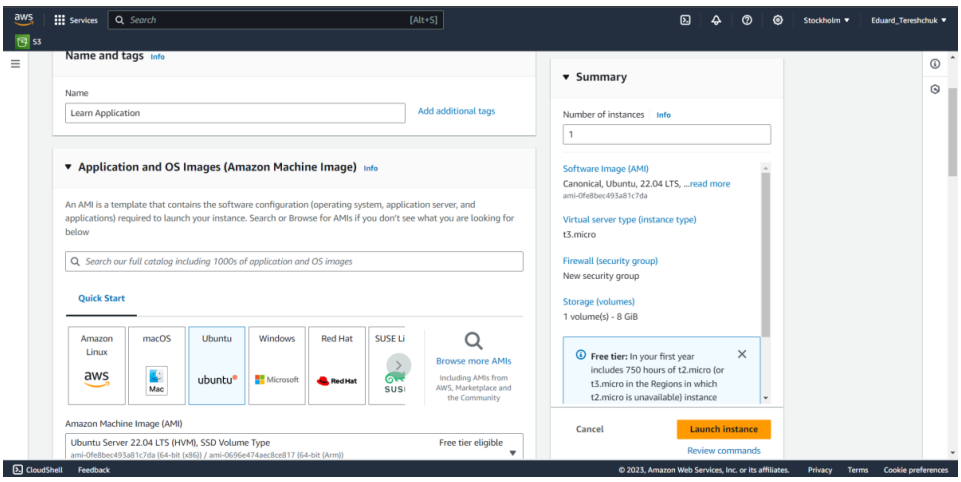

Рисунок 3.10 — Налаштування серверу за допомогою сервісу Amazon EC2

Ввівши потрібні налаштування натискаємо кнопку «Launch instance» і сервер буде створено (рисунок 3.11).

Рисунок 3.11 — Успішне створення серверу для розгортання веб-застосунку

Оскільки серверна частина розроблювального застосунку написана за допомогою мови програмування java, то з’являється необхідність, щоб сервер містив віртуальну машину java (JVM). Щоб перевірити наявність віртуальної машини java потрібно ввести команду “sudo update-alternatives --config java” (рисунок 3.12)

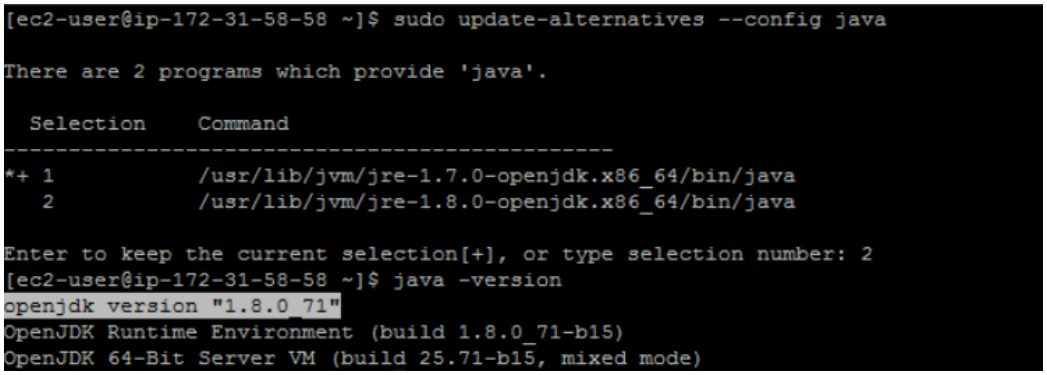

Рисунок 3.12 — Наявність JVM на EC2 сервері

Після успішного встановлення JVM слід перенести застосунок на машину ec2 у вигляді .war пакету. Для цього знову скористайтеся командою SCP: “scp -i aTest.pem learn-application-1.0.war ubuntu@ec2 — compute-1.amazonaws.com:/home/ubuntu/ learn-application-1.0.war”. Після чого застосунок буде успішно розгорнуто у хмарі (рисунок 3.13).

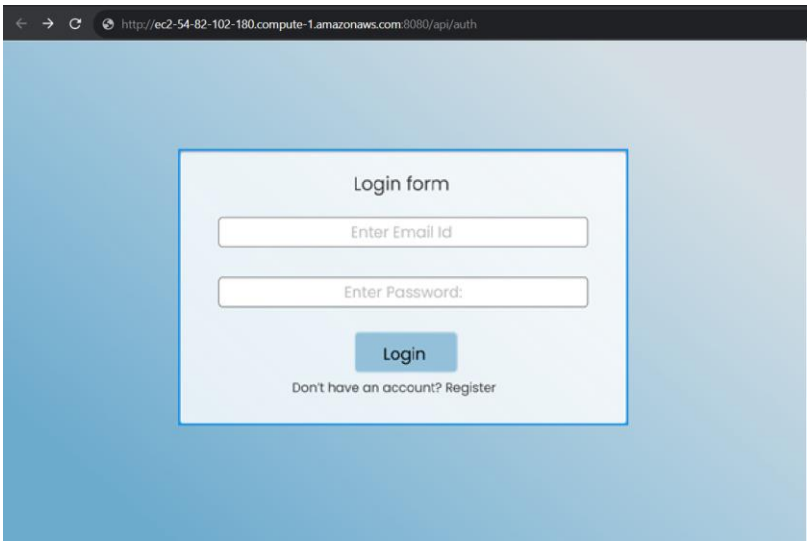

Рисунок 3.13 — Успішне розгортання веб-застосунку у хмарному сервісі

Перейшовши у потрібний instance, скопіювавши значення Public DNS(IPv4) та ввівши це значення у рядок браузера, можемо переконатися у працездатності нашого веб-застосунку, отримавши перенаправлення на сторінка авторизації.

4 ТЕСТУВАННЯ ВЕБ-ЗАСТОСУНКУ

4.1 Валідація серверної частини веб-застосунку

Під час розробки веб-застосунків, важливо звертати увагу на перевірку правильності введених даних та валідацію їх формату. Перевірка на правильність введених даних полягає у перевірці коректності і належності даних, що надходять на сервер. Це включає у себе перехоплення та обробку можливих атак, таких як SQL-ін’єкції чи XSS-атаки, щоб уникнути вразливостей системи перед обробкою цих даних. Цей етап дозволяє гарантувати, що вхідні дані не містять шкідливих скриптів чи команд, що можуть пошкодити систему чи зловмисником використати їх для несанкціонованого доступу.

На цьому етапі використання інструменту Postman стає ключовим для перевірки правильності введених даних та їх формату. Оскільки Postman надає зручний інтерфейс для відправки HTTP-запитів на сервер та отримання відповідей.

Щоб перевірити чи правильно працює авторизація веб-застосунку спробуємо надіслати кілька запитів через Postman. Спробуємо отримати список навчальних курсів до яких приєднаний користувач за допомогою «/api/student/courses» ендпоінта. На рисунку 4.1 зображено спробу отримати доступ до даної інформації ввівши некоректні дані для авторизації, отримуємо відповідь з кодом 401, що свідчить про некоректну авторизацію користувача.

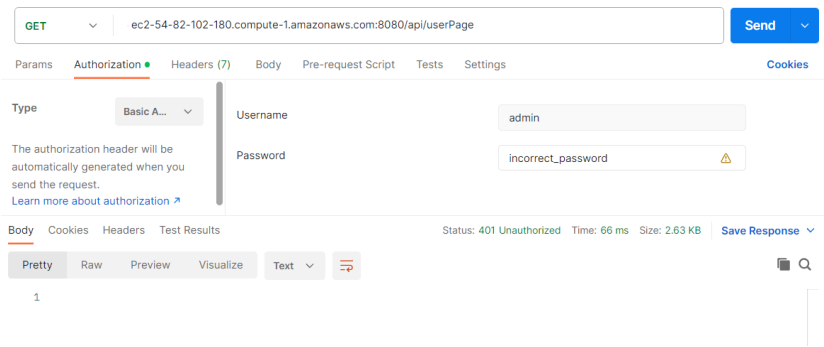

Рисунок 4.1 — Некоректна авторизація у веб-застосунок

Ввівши коректні дані, а саме логін та пароль, користувач отримує відповідь з кодом 200, що свідчить про успішну обробку запиту, який був надісланий на сервер (рисунок 4.2).

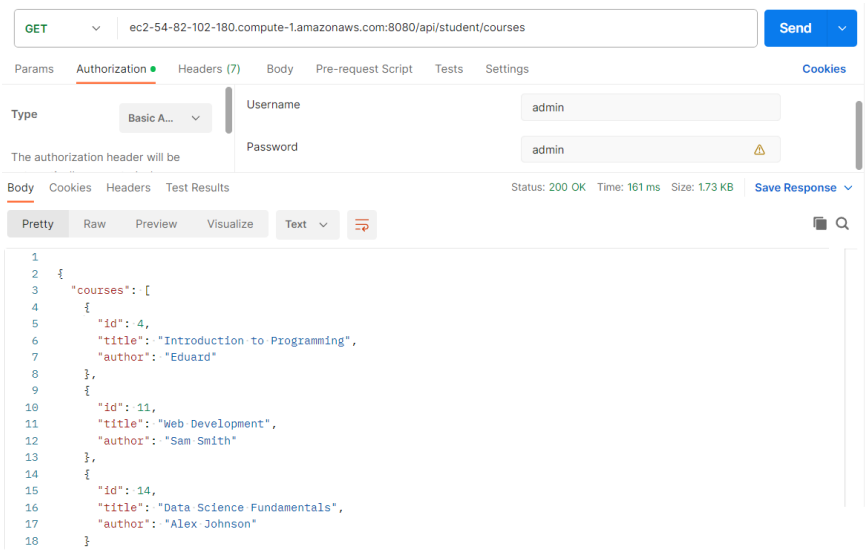

Рисунок 4.2 — Успішна обробка запиту користувача на сервер

Перевіривши можливість користувача отримати необхідну інформацію від сервера, протестуємо запис необхідних даних у систему. Для цього скористуємося ендпоінтом «api/teacher/course» з методом POST, який дозволить користувачу з ролю викладач створити новий навчальний курс (рисунок 4.3).

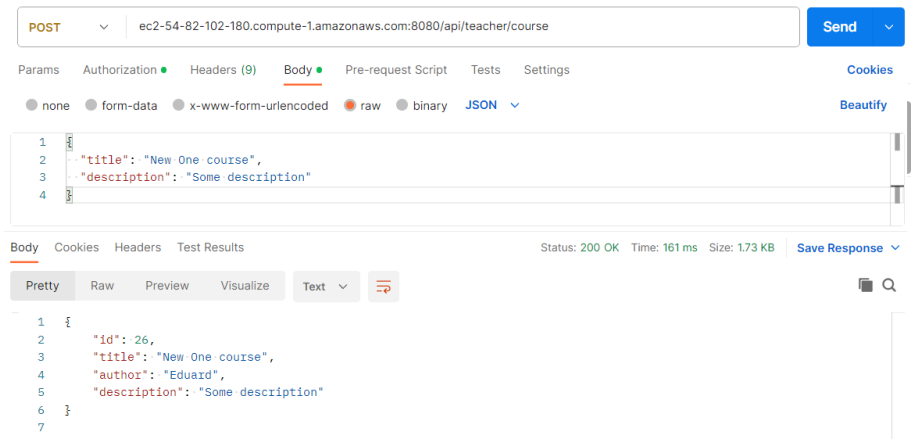

Рисунок 4.3 — Додання нового курсу за допомогою Postman

Як зображено на рисунку 4.3 запит успішно оброблено, а відповідь запиту містить оновлені дані про новостворений навчальний курс. Щоб перевірити валідацію запитій, що надходять на сервер заберемо обов'язкове поле «title» з тіла запиту. На рисунку 4.4 можемо пересвідчитись, що даний запит надійшов на сервер і через невалідність даних він був перехоплений за допомогою обробником запитів та повернуто деталізовану відповідь про некоректність запиту.

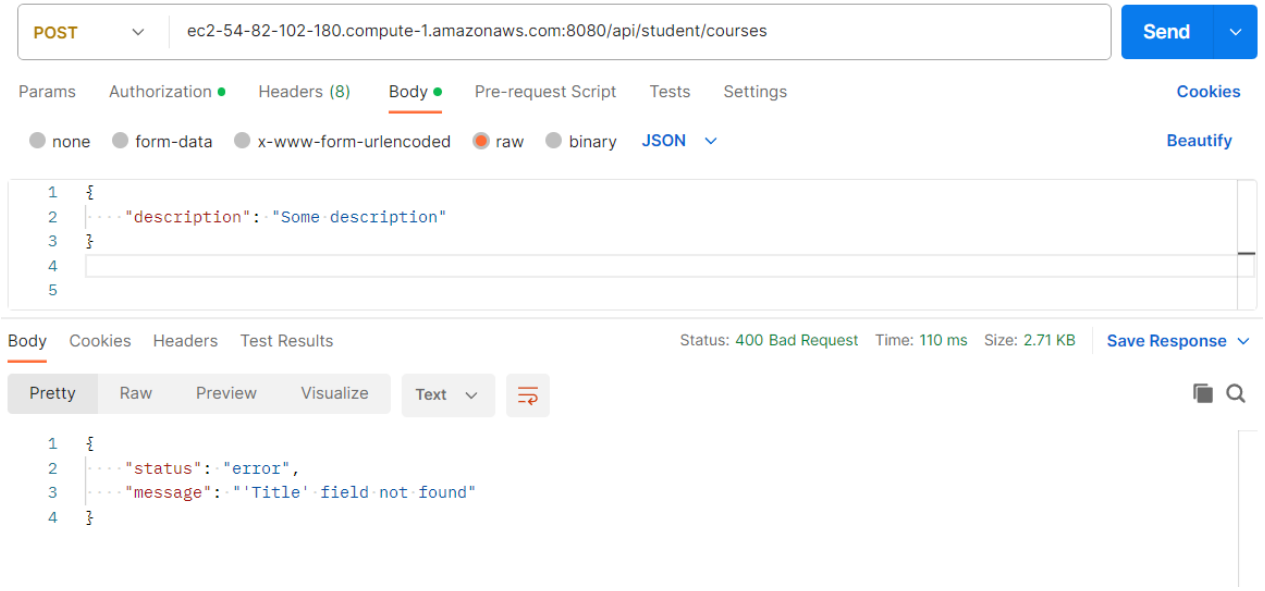

Рисунок 4.4 — Некоректне ведення даних у запиті

Проаналізувавши відповіді на запити, які були надіслані на сервер через Postman, можемо зазначити, що обробка запитів та їхня валідація працює коректно.

4.2 Перевірка функціональності клієнтської частини веб-застосунку

Протестувавши серверну частину веб-застосунку перейдемо до тестування клієнтської частини. Перейшовши по посиланню "http://ec2-54-82-102-180.compute-1.amazonaws.com:8080" нас буде перенаправлено на сторінку авторизації. Спробувавши увійти у систему за допомогою невірних даних ми отримуємо відповідне сповіщення (рисунок 4.5). Після того як ми переконалися, що ввівши не коректні дані система забороняє доступ вводимо

коректні дані авторизації і нас перенаправляє на головну сторінку користувача (рисунок 4.6).

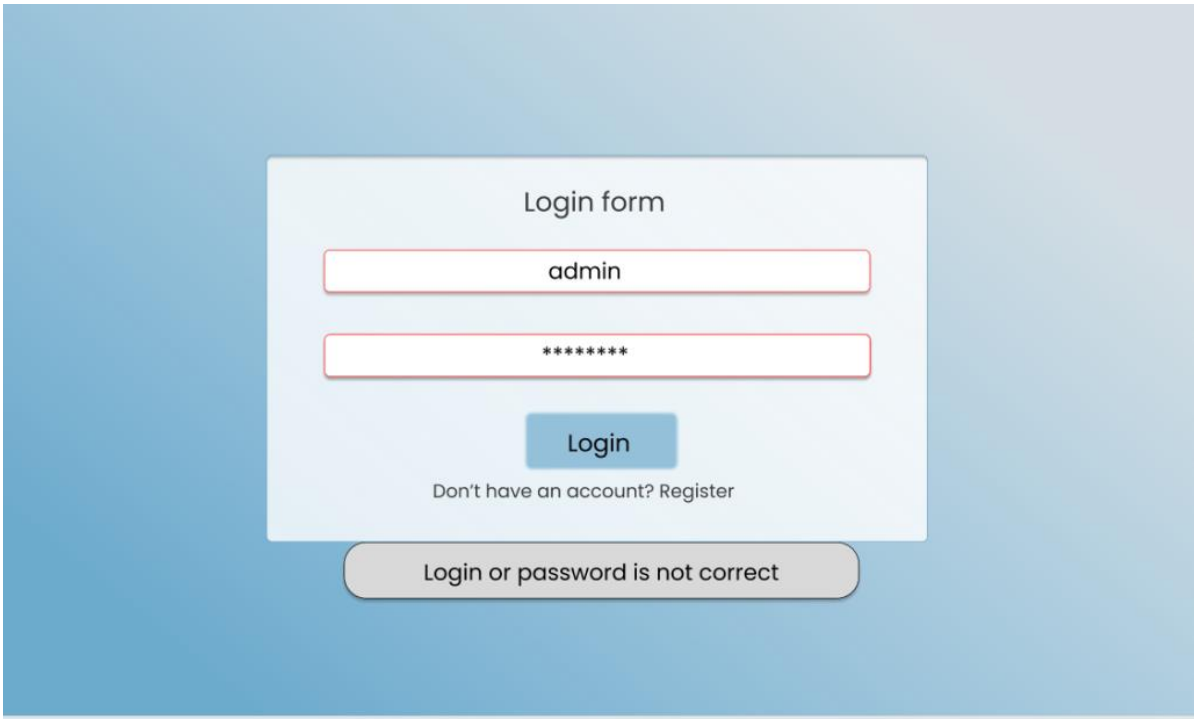

Рисунок 4.5 — Введення некоректний даних авторизації

Рисунок 4.6 — Сторінка користувача

Після того як користувача було переведено на головну сторінку, йому доступна можливість обрати роль під якою від буде далі взаємодіяти із системою. Обравши роль студент користувача буде переведено на сторінку студента із списком курсів, які він розпочав раніше та рядком пошуку “Search” для пошуку нових навчальних курсів (рисунок 4.7).

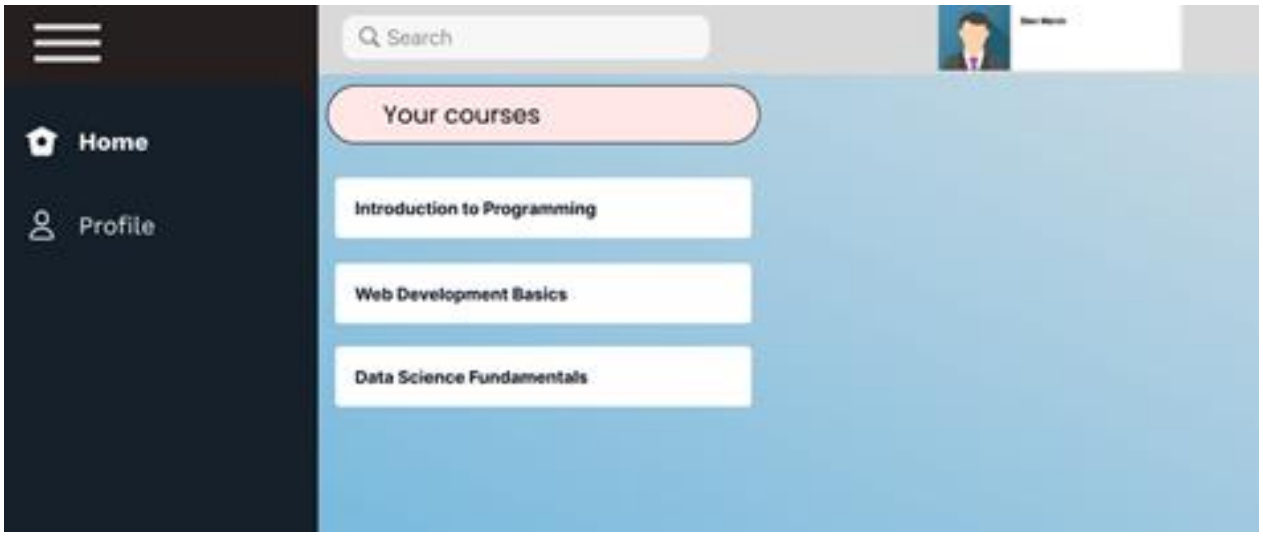

Рисунок 4.7 — Сторінка студента

Обравши один із курсів, які доступні на даних момент ми переходимо на сторінку курсу, де можемо спостерігати список завдань, які потрібно виконати (рисунок 4.8).

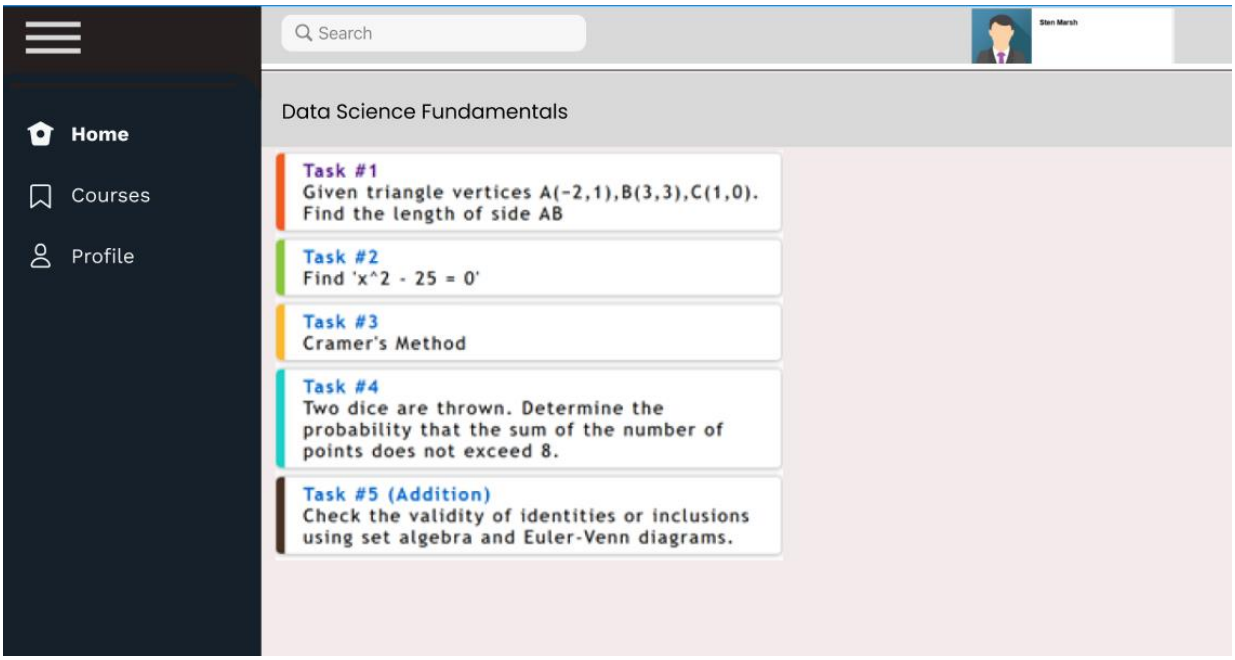

Рисунок 4.8 — Сторінка курсу

Натиснувши на одне із завдань користувач потрапить на сторінку завдання (рисунок 4.9), де йому буде можливо надати відповідь на дане

завдання та переглянути доступні файли, що прикріплені викладачем до даного завдання.

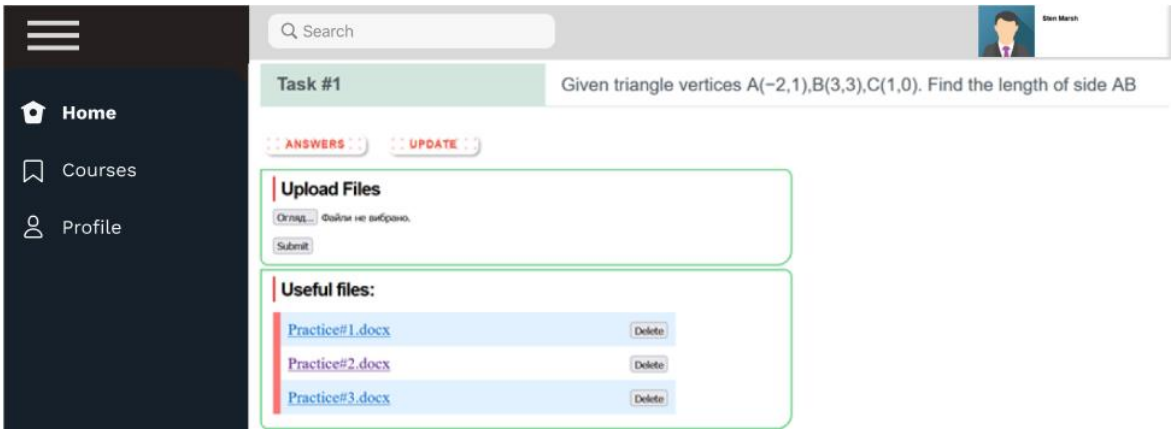

Рисунок 4.9 — Сторінка завдання

Переглянувши вище згадані сторінки, які доступні користувачу можемо дійти висновку, що клієнтська частина працює згідно вимог, які поставлені перед веб-застосунком.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного та технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Магістерська кваліфікаційна робота за темою "Веб-застосунок для організації та оцінювання навчального процесу з використанням хмарних сервісів" полягає в підвищенні ефективності навчання, полегшенні взаємодії між викладачами та студентами.

Проведемо оцінювання комерційного та технічного аудиту даної розробки. Для проведення комерційного та технічного аудиту було залучено 3-х незалежних експертів: к.т.н. доц. кафедри ОТ Тарновський М. Г., к.т.н. доц. кафедри ОТ Городецька О. С., к.т.н. доц. кафедри ОТ Муращенко О. Г. Оцінювання комерційного потенціалу буде здійснене за критеріями, що наведені в таблиці 5.1.

Таблиця 5.1 — Критерії оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогі	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 5.1

4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження таблиці 5.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу експертами розробки зведено в таблицю 5.2.

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище експерта		
	1 — Тарновський	2 — Городецька	3 — Муращенко
	Бали, виставлені експертами:		
1	4	3	3
Ринкові переваги (недоліки):			
2	3	2	2
3	2	4	3
4	2	3	4
5	4	4	3
Ринкові перспективи			

Продовження таблиці 5.2

6	4	4	3
7	3	4	3
Практична здійсненність			
8	4	4	4
9	1	2	2
10	3	4	4
11	3	4	3
12	3	2	3
Сума балів	$СБ_1$=36	$СБ_2$=40	$СБ_3$=37
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ}\frac{\sum_1^3 СБ_i}{3} = \frac{113}{3} = 37.6$		

За даними таблиці 5.2 можна зробити висновок, щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 5.3.

Таблиця 5.3 — Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 — 10	Низький
11— 20	Нижче середнього
21— 30	Середній
31— 40	Вище середнього
41— 48	Високий

Рівень комерційного потенціалу розробки, становить 35,6 балів, що відповідає рівню «вище середнього».

5.2 Розрахунок витрат на здійснення науково-технічної розробки

Витрати на здійснення науково-технічної розробки розраховуються за наступними категоріями: витрати на оплату праці, відрахування на соціальні заходи, матеріали, програмне забезпечення для наукових робіт, накладні (загальновиробничі) витрати та ін. Обрахуємо витрати за кожною категорією.

Основна заробітна плата для розробника-дослідника $З_о$:

$$З_о = \frac{М}{Т_р} \cdot t \text{ [грн]}, \quad (5.1)$$

де М — місячний посадовий оклад , 50000 грн;

$Т_р$ — число робочих днів в місяці, приблизно $Т_р$ = (22) дні;

t — число робочих днів роботи розробника-дослідника , 90.

$$З_о = \frac{30000}{22} \cdot 90 = 204545{,}00 \text{ (грн)}.$$

Результати розрахунків зведемо до таблиці 5.4.

Таблиця 5.4 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Розробник	50000,00	2272,72	90	204545,00
Керівник роботи	17000,00	772,72	10	7727,2
Всього				212272,2

Додаткова заробітна плата $З_д$ розробників розраховується як 10% від основної заробітної плати:

$$З_д = 0{,}10 \cdot 212272{,}2 = 21227{,}22 \text{ (грн)}.$$

Нарахування на заробітну плату $Н_{зп}$ розробника становить:

$$Н_{зп} = (З_о + З_д) \cdot \frac{\beta}{100} \text{ [грн]}, \quad (5.2)$$

де $З_о$ — основна заробітна плата розробника;

$З_д$ — додаткова заробітна плата розробника;

β — ставка єдиного внеску на загальнообов'язкове державне соціальне страхування — 22%.

$$Н_{зп} = (212272,2+ 21227,22) \cdot 0,22 = 51369,87(грн).$$

Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час виконання даного етапу роботи. Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

Амортизація обладнання та приміщення, яке використовувалось для проведення розробки, розраховується за формулою:

$$A = \frac{Ц \cdot H_a}{100} \cdot \frac{T}{12} \text{ [грн]}, \quad (5.3)$$

де Ц — балансова вартість обладнання, грн.;

$Н_а$ — річна норма амортизаційних відрахувань;

Т — термін використання під час розробки, місяців.

Розрахуємо амортизаційні витрати для комп'ютера, балансова вартість якого становить 80000 грн, а термін використання — 3 місяці:

Норма амортизації розраховується за формулою:

$$H_a = \frac{B_п - B_л}{B_п \cdot T_{кв}} \cdot 100 \text{ [грн]}, \quad (5.4)$$

де $В_п$ і $В_л$ — відповідно первісна та ліквідаційна вартість основних фондів;

$Т_{кв}$ — строк корисного використання.

$$H_a = \frac{80000-2000}{80000 \cdot 5} \cdot 100 = 19{,}5\% \text{ (грн).}$$

$$A = \frac{80000 \cdot 19{,}5}{100} \cdot \frac{3}{12} = 3900 \text{ (грн).}$$

Таблиця 5.5 — Амортизаційні відрахування

Найменування	Балансова вартість, грн	Термін використання, р	Фактична трив. використання, міс.	Величина амортизаційних відрахувань, грн
Офісне приміщення	100000	25	3	4500,00
Комп'ютер	80000	5	3	3900,00
Всього				8400,00

Під час розробки програмного продукту використовувались лише безкоштовні програмні засоби.

Витрати на енергію визначаються на основі витрат на одиницю продукції та тарифів на енергію за допомогою формули:

$$B_e = B \cdot П \cdot Ф \cdot K_п \ [грн], \qquad (5.5)$$

де В — вартість 1кВт електроенергії;

П — установлена потужність обладнання, кВт;

Ф — фактична кількість годин роботи комп'ютера при створенні програмного продукту, годин;

$К_п$ — коефіцієнт використання потужності.

Отже, витрати на енергію становлять:

$$В_е = 6.4 \cdot 0{,}6 \cdot 720 \cdot 0{,}6 = 1658.88 \text{ (грн).}$$

Витрати за доступ до Інтернет можна розрухувати за формулою:

$$B_{\partial i} = Ц_{\partial i} \cdot T\,[грн], \tag{5.6}$$

де $Ц_{ді}$ — це ціна доступу за місяць;

Т — кількість місяців використання доступу до мережі.

$$В_{ді} = 400 \cdot 3 = 1200{,}00 \text{ (грн)}.$$

Обчислимо витрати на виконання даної роботи, що являтиме собою суму всіх попередніх витрат.

В результаті сума усіх витрат, що вказані вище дає витрати на виконання даного етапу роботи В:

$$В = З_о + З_д + Н_{зп} + А + В_е + В_{ді}\,[грн],$$

В=212272,2+21227,22+51369,87+8400+1658,88+1200,00=296128,17 (грн)

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}\,[грн], \tag{5.8}$$

де β — коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Так, якщо розробка знаходиться:

— на стадії науково-дослідних робіт, то $\beta \approx 0{,}1$;

— на стадії технічного проектування, то $\beta \approx 0{,}2$;

— на стадії розробки конструкторської документації, то $\beta \approx 0{,}3$;

— на стадії розробки технологій, то $\beta \approx 0{,}4$;

— на стадії розробки дослідного зразка, то $\beta \approx 0{,}5$;

— на стадії розробки промислового зразка, $\beta \approx 0{,}7$;

— на стадії впровадження, то $\beta \approx 0{,}9$.

Отже, підставимо дані в формулу й отримаємо результат:

$$ЗВ = \frac{296128{,}17}{0{,}9} = 329031{,}30 \text{ (грн)}.$$

Витрати на виконання наукової роботи та впровадження її результатів становитиме 329 031,3 грн.

5.3 Розрахунок економічної ефективності та обґрунтування економічної доцільності комерціалізації науково-технічної розробки.

Узагальнюючим позитивним результатом, що отримує інвестор від впровадження результатів тієї чи іншої розробки, є збільшення чистого прибутку. Виконання даної наукової роботи та впровадження її результатів складає трохи більше одного року. Позитивні результати від впровадження розробки очікуються вже в перші місяці після впровадження.

Збільшення чистого прибутку у потенційного інвестора протягом декількох років розраховується за формулою:

$$\Delta П_i = (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}) \qquad (5.9)$$

де $\Delta Ц_o$ — зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році;

N — основний кількісний показник, який визначає величину попиту на аналогічні розробки у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ — основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році, $Ц_o = Ц_б \pm \Delta Ц_o$;

$Ц_б$ — основний якісний показник, який визначає ціну реалізації існуючої науково-технічної розробки у році до впровадження результатів;

ΔN — зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році;

λ — коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. λ =0,8333;

ρ — коефіцієнт, який враховує рентабельність інноваційного продукту. ρ = 0,3;

ϑ — ставка податку на прибуток, який має сплачувати потенційний інвестор, у ϑ = 18%.

В результаті впровадження наукової розробки покращується якість певного продукту, що дозволяє підвищити ціну його реалізації на 8000 грн. Кількість користувачів збільшиться протягом першого року на 1000, другого — 2000, третього — 3000. Реалізація продукції до впровадження результатів наукової розробки складала 1000 користувачів, а її ціна — 50000 грн.

Розрахуємо показник прибутку впродовж трьох років відносно базового.

$$\Delta П_1 = 50000 \cdot 1 + (50000 + 8000) \cdot 1000) \cdot 0{,}8333 \cdot 0{,}3 \cdot (1 - 18/100) = 11\ 852\ 649 \text{ (грн)}.$$

$$\Delta П_1 = (50000 \cdot 1 + (50000 + 8000) \cdot 2000) \cdot 0{,}8333 \cdot 0{,}3 \cdot (1 - 18/100) = 23\ 695\ 089 \text{ (грн)}.$$

$$\Delta П_1 = (50000 \cdot 1 + (50000 + 8000) \cdot 3000) \cdot 0{,}8333 \cdot 0{,}3 \cdot (1 - 18/100) = 35\ 537\ 529 \text{ (грн)}.$$

Далі розрахуємо приведену вартість збільшення чистих прибутків, що їх може отримати потенційний інвестор від впровадження розробки за формулою:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} \qquad (5.10)$$

де T — період часу, протягом якого очікується отримання позитивних результатів від впровадження науково-технічної розробки, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15;

t — період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = \frac{11\ 852\ 649}{(1+0{,}05)^1} + \frac{23\ 695\ 089}{(1+0{,}05)^2} + \frac{35\ 537\ 529}{(1+0{,}05)^3} = 6\ 347\ 900\ (грн).$$

Отже, розрахунки показують, що комерційний ефект від впровадження розробки виражається у значному збільшенні чистого прибутку підприємства.

Основними показниками, які визначають доцільність фінансування наукової розробки інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Якщо $E_{абс} > 0$, то результат від проведення наукових досліджень та їх впровадження принесе прибуток, але це також ще не свідчить про те, що інвестор буде зацікавлений у фінансуванні даного проекту (роботи).

Розрахуємо абсолютну ефективність інвестицій, вкладених у реалізацію проекту. Ставка дисконтування τ дорівнює 0,1. Отримаємо:

Розраховуємо величину початкових інвестицій PV, які розробник (замовник) має вкласти для здійснення науково-технічної розробки.

Для цього можна використати формулу:

$$PV = k_{розр} \cdot ЗВ\ [грн]\ , \qquad (5.11)$$

де розр *k* — коефіцієнт, що враховує витрати розробника (замовника) на впровадження науково-технічної розробки, це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай розр k =2...5, але може бути і більшим;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 4 \cdot 329\,031{,}3 = 1316125{,}2 (грн).$$

Розраховуємо абсолютну ефективність вкладених інвестицій $Е_{абс}$ за формулою:

$$Е_{абс} = (ПП - PV)\ [грн], \qquad (5.12)$$

де ПП — приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.;

PV — теперішня вартість інвестицій PV = ЗВ, грн.

$$Е_{абс} = 6\,347\,900 — 1\,316\,125{,}20 = 5\,031\,774{,}8 (грн).$$

Оскільки $Е_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів розробки є доцільним.

Розраховуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$. Для цього використовуємо формулу:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1 \qquad (5.13)$$

де $Е_{абс}$ — абсолютна ефективність вкладених інвестицій, грн;

PV — теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ — життєвий цикл наукової розробки, роки.

Тоді відносна ефективність вкладних інвестицій в проведення наукових досліджень та впровадження їх результатів складе:

$$E_в = \sqrt[3]{1 + \frac{5\,031\,774{,}8}{1316125{,}2}} - 1 = 0{,}68$$

Далі, розраховану величину $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування τ мін, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть.

У загальному вигляді мінімальна (бар'єрна) ставка дисконтування τ мін визначається за формулою:

$$\tau = \text{d} + \text{f}, \qquad (5.14)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d= (0,14…0,2);

f показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau \text{ мін} = 0{,}5 + 0{,}05 = 0{,}55.$$

Оскільки $E_в$ = 68% > τ мін = 55%, то у інвестора буде зацікавленість вкладати гроші в дану наукову розробку, оскільки значно більші прибутки він отримає від того, що інвестує кошти розробку, а не розмістить гроші на депозиті у комерційному банку.

Розраховуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}. \qquad (5.15)$$

Для нашої розробки термін окупності вкладених у реалізацію проекту інвестицій $Т_{ок}$ складе:

$$T_{ок} = \frac{1}{0{,}68} = 1{,}47\ (року),$$

Термін окупності складає 1,47 року, що свідчить про доцільність фінансування даної наукової розробки.

5.4 Результати економічного аналізу

В даному розділі було проведено комерційний та технологічний аудит розробки із залученням трьох незалежних експертів. Оцінка комерційного потенціалу показала, що рівень комерційного потенціалу розробки є вище середнього.

Згідно із розрахунками всіх статей витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторської-технологічної роботи загальні витрати на розробку складають 329 031,3 грн.

Розрахована абсолютна ефективність вкладених інвестицій в сумі 6 347 900 грн свідчить про отримання прибутку інвестором від комерціалізації програмного продукту.

Відносна ефективність вкладних інвестицій в проведення наукових досліджень та впровадження їх результатів становить 68%, що вище за мінімальну бар'єрну ставку дисконтування, яка складає 55%. Це означає потенційну зацікавленість інвесторів у фінансуванні розробки.

Термін окупності вкладених у реалізацію проекту інвестицій становить 1,47 року, що також свідчить про доцільність фінансування нової розробки.

ВИСНОВКИ

У ході виконання магістерської роботи було розроблено веб-застосунок для організації та оцінювання навчального процесу з використанням хмарних сервісів, що представляє собою значущий крок у вдосконаленні сучасної освіти.

Такий інноваційний інструмент надає вчителям та учням можливість легко взаємодіяти, спільно працювати над завданнями та відстежувати прогрес навчання. Він забезпечує автентифікацію користувачів, створення курсів, завдань та тестів, інтерактивну спільноту для обговорення матеріалів, а також можливість зберігати та отримувати доступ до даних з будь-якого пристрою завдяки хмарному зберіганню.

Також було проаналізовано аналоги даного веб-застосунку, переваги та недоліки використання навчальних платформ, тенденції та інновації у розвитку навчальних платформ та веб за стосунків, впровадження хмарних сервісів у розробці веб-застосунків.

Розроблено загальну структуру та архітектуру роботи веб-застосунку. Реалізовано інтерфейс та функціонал клієнтської та серверної частини. Проведено валідація серверної частини веб-застосунку та перевірено функціональность клієнтської частини.

Отже, можна зробити висновок, що використання хмарних технологій в даному проекті створює основу для сучасної, гнучкої та ефективної системи організації та оцінювання навчального процесу, що відповідає потребам освіти в 21 столітті.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Терещук Е. П., Тарновський М.Г. «Веб-застосунок для організації та оцінювання навчального процесу з використанням хмарних сервісів». Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)», Вінниця, 2022. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/19666/16303

2. Веб-застосунок [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Вебзастосунок

3. Розробка веб-додатків [Електронний ресурс]. Режим доступу: https://en.yeeply.com/blog/6-different-kinds-веб-app-development/#definition

4. Переваги використання веб-застосунків [Електронний ресурс]. Режим доступу: https://www.geeks.ltd.uk/insights/the-benefits-of-using-веб-based-applications

5. The benefits of web applications in todays technological era [Електронний ресурс]. Режим доступу: https://www.kcsitglobal.com/blogs/detail-blog/the-benefits-of-веб-applications-in-todays-technological-era

6. eLearning Trends And Innovations Shaping The Industry [Електронний ресурс]. Режим доступу: https://elearningindustry.com/evolution-of-elearning-trends-and-innovations-shaping-the-industry

7. Implementation of Cloud Computing on Web Application [Електронний ресурс]. Режим доступу: https://www.researchgate.net/publication/44244505_Implementation_of_Cloud_Computing_on_Web_Application

8. SaaS, PaaS, IaaS [Електронний ресурс]. Режим доступу: https://www.it.ua/knowledge-base/architecture-security/cloud-infrastructure-saas-paas-iaas

9. AWS vs Azure vs Google Cloud - Detailed Cloud Comparison [Електронний ресурс]. Режим доступу: https://intellipaat.com/blog/aws-vs-azure-vs-google-cloud/

10. Google Classroom [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Google_Classroom

11. NEO LMS [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/NEO_LMS

12. Guide to Web Application Architecture [Електронний ресурс]. Режим доступу: https://www.intellectsoft.net/blog/web-application-architecture

13. Web Application Architecture: Diagram and 8 Best Practices to Follow [Електронний ресурс]. Режим доступу: https://asperbrothers.com/blog/web-application-architecture/

14. Progressive web apps [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps

15. Guide to Web Application Architecture [Електронний ресурс]. Режим доступу: https://www.intellectsoft.net/blog/web-application-architecture

16. Pros and Cons of Using C# as Your Backend Programming Language [Електронний ресурс]. Режим доступу: https://www.agilites.com/pros-and-cons-of-using-c-as-your-backend-programming-language.html

17. Nodejs Vs. Go [Електронний ресурс]. Режим доступу: https://www.peerbits.com/blog/nodejs-vs-golang.html

18. JavaScript For Backend Development [Електронний ресурс]. Режим доступу: https://blog.hubspot.com/website/java-backend

19. Pros and Cons of JavaScript [Електронний ресурс]. Режим доступу: https://data-flair.training/blogs/advantages-disadvantages-javascript

20. The Good and the Bad of Node.js Web App Programming [Електронний ресурс]. Режим доступу: https://www.altexsoft.com/blog/the-good-and-the-bad-of-node-js-web-app-development/

21. Advantages and Disadvantages of PHP You Should Know When Starting a New Project [Електронний ресурс]. Режим доступу: https://anywhere.epam.com/business/pros-and-cons-of-php

22. Advantages and Disadvantages of PHP [Електронний ресурс]. Режим доступу: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-php/

23. The Beginner's Guide to Python Back-End Development [Електронний ресурс]. Режим доступу:https://blog.hubspot.com/website/python-backend

24. Python vs. Node.js: Making the Right Choice for Backend [Електронний ресурс]. Режим доступу: https://elitex.systems/blog/python-vs-nodejs/

25. Pros and Cons of Python Programming Language [Електронний ресурс]. Режим доступу: https://www.pixelcrayons.com/blog/software-development/python-pros-and-cons

26. Ruby [Електронний ресурс]. Режим доступу: https://www.ruby-lang.org/en/

27. Ruby for backend development [Електронний ресурс]. Режим доступу: https://talent500.co/ruby-for-backend-development/

28. What Is Java: The Beginner's Guide to the Java Programming Language [Електронний ресурс]. Режим доступу: https://blog.hubspot.com/website/javascript-for-backend

29. JVM: що це у світі Java [Електронний ресурс]. Режим доступу: https://foxminded.ua/jvm-tse/

30. Disadvantages of Java [Електронний ресурс]. Режим доступу: https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Java-Disadvantages-Drawbacks-Advantages-17-Speed-21-Performance-8-Oracle-Vendor-Memory-CPU

31. Spring Framework [Електронний ресурс]. Режим доступу: https://spring.io/projects/spring-framework

32. IoC Introduction [Електронний ресурс]. Режим доступу: https://www.tutorialsteacher.com/ioc/introduction

33. Java Spring Boot [Електронний ресурс]. Режим доступу: https://www.ibm.com/topics/java-spring-boot

34. Spring Data [Електронний ресурс]. Режим доступу: https://golden.com/wiki/Spring_Data-ZY9RE4P

35. Spring Data JPA [Електронний ресурс]. Режим доступу: https://spring.io/projects/spring-data-jpa/

36. What's the Difference Between AWS vs. Azure vs. Google Cloud? [Електронний ресурс]. Режим доступу: https://www.coursera.org/articles/aws-vs-azure-vs-google-cloud

37. Amazon S3 [Електронний ресурс]. Режим доступу: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

38. Amazon Relational Database Service [Електронний ресурс]. Режим доступу: https://aws.amazon.com/rds/?nc1=h_ls

39. Amazon EC2 [Електронний ресурс]. Режим доступу: https://aws.amazon.com/ec2/?nc1=h_ls

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н.. Азаров О.Д..

"29" вересня 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

"Веб-застосунок для організації та оцінювання навчального процесу з використанням хмарних сервісів"
08-54.КМКР.044.00.000 ПЗ

Науковий керівник: доцент
к.т.н. каф.ОТ

_____ Тарновський М. Г.

Студент групи 2КІ-22м

_____ Терещук Е. П.

ВНТУ 2023

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність роботи полягає у розробці веб-застосунку спрямованого на огранізацію навчального процесу, що забезпечує користувачам зручний спосіб отримування нових знань, та адаптацією системи до нових вимог та динамічною регуляцією обчислювальних ресурсів з урахуванням навантаження на систему.

1.2 Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета роботи — вдосконалення веб-застосунку для організації та оцінювання навчального процесу.

2.2 Призначення розробки — визначається необхідністю створення веб-застосунку, що надаватиме функціональність для організації навчального процесу шляхом моніторингу та аналізу даних, що надходять від користувачів системи.

3 Вихідні дані для виконання МКР

3.1 Проведення аналізу існуючих методів та рішень.

3.2 Розробка алгоритму системи та веб-застосунку.

3.4 Проведення верифікації та аналізу отриманих результатів.

3.5 Виконання розрахунків для доведення доцільності нової розробки з економічної точки зору.

4 Вимоги до виконання МКР

Головна вимога — веб-застосунок має надавати графічний інтерфейс для огранізації навчального процесу, шляхом створення навчальних курсів та можливістю аналізу та оцінювання прогресу студентів.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз існуючих технологій, огляд аналогів.	19.09.2023	28.09.2023	Розділ 1
2	Визначення архітектури веб-застосунку	5.10.2023	15.10.2023	Розділ 2
3	Розробка структури та функціоналу веб-застосунку	16.10.2023	23.10.2023	Розділ 3
4	Тестування веб-застосунку	23.10.2023	26.10.2023	Розділ 4
5	Підготовка економічної частини та формування додатків	2.11.2023	12.11.2023	Розділ 5, Додатки
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	1.12.2023	11.12.2023	ПЗ, графічний матеріал і презентація
7	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат	11.12.2023	13.12.2023	Оформленні документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і

ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлюванні МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104— 2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ— 03.02.02 П.001.01:21

ДОДАТОК Б

Блок схема архітектури веб-застосунку

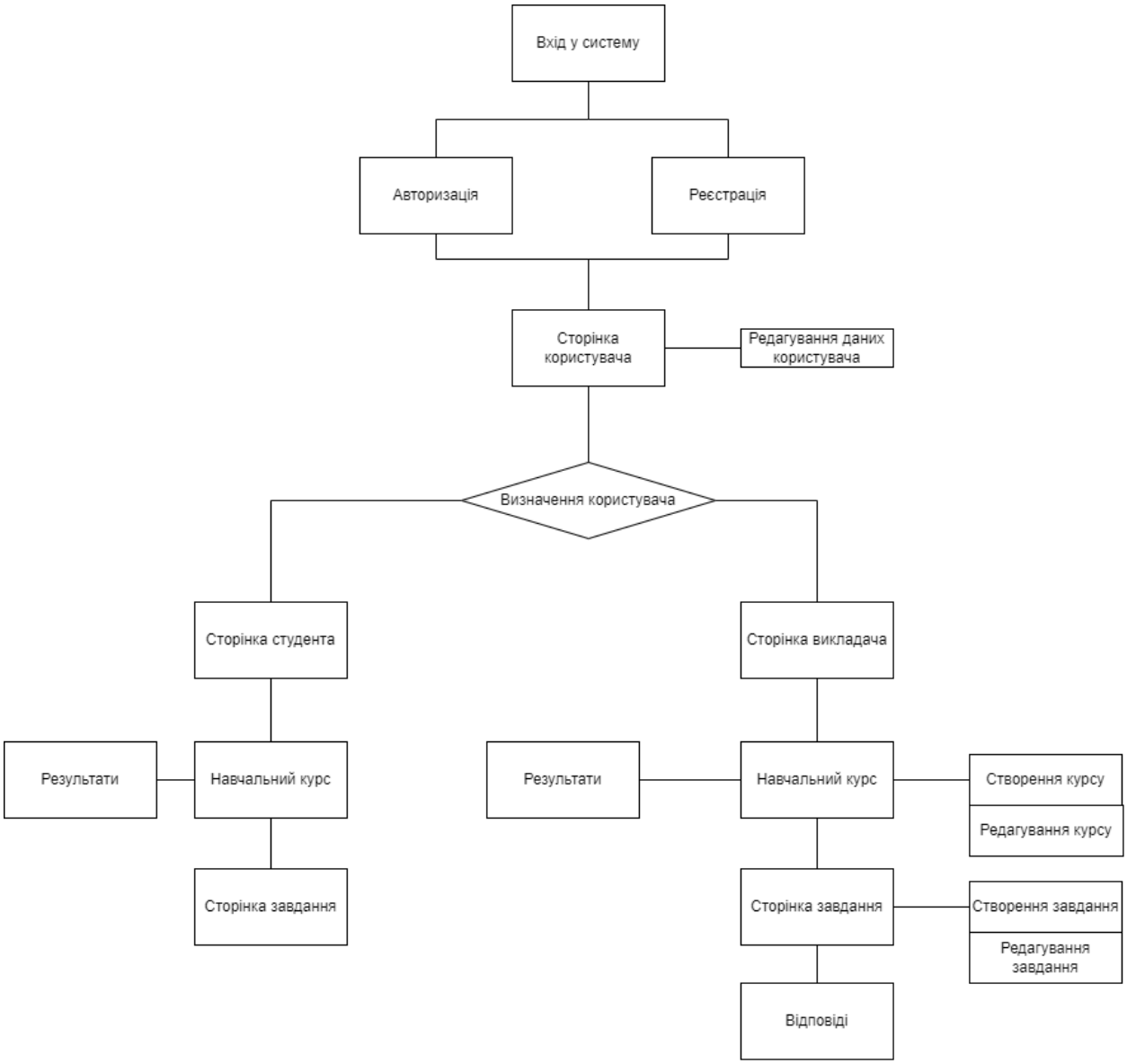

Рисунок Б.1 — Блок схема архітектури веб-застосунку

ДОДАТОК В

Зв’язок клієнтської та серверної частини веб-застосунку

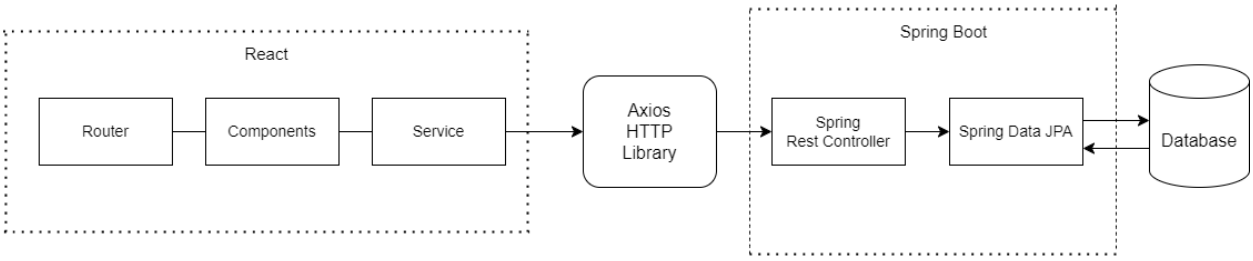

Рисунок В.1 — Зв’язок клієнтської та серверної частини веб-застосунку

ДОДАТОК Г

Схема взаємодії хмарних сервісів між собою

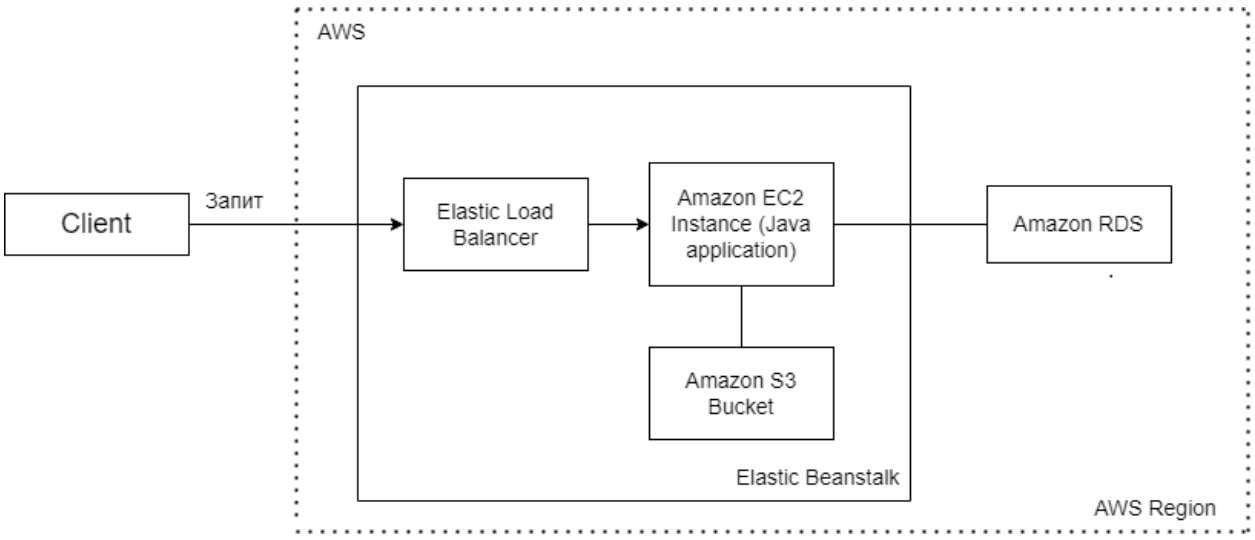

Рисунок Г.1 — Схема взаємодії хмарних сервісів між собою

ДОДАТОК Д

Налаштування конфігураційного файлу “pom.xml”

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.7</version>
        <relativePath/>
    </parent>
    <groupId>com.vntu</groupId>
    <artifactId>Learn.application</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>Learn.application</name>
    <description>Learn.application</description>
    <properties>
        <java.version>11</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-rest</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
```

```
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.hibernate.validator</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.1.6.Final</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
```

```
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>s3</artifactId>
            <version>2.20.52</version>
        </dependency>
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk</artifactId>
            <version>1.0.12</version>
        </dependency>
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-s3</artifactId>
            <version>1.12.537</version>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

ДОДАТОК Е

Реалізація контролерів

Лістинг файлу “src/java/com/learn/application/AuthController.java”

```
package com.vntu.learn.application.controllers;
import com.vntu.learn.application.bean.HttpSessionBean;
import com.vntu.learn.application.models.Course;
import com.vntu.learn.application.models.Role;
import com.vntu.learn.application.services.CourseImpl;
import com.vntu.learn.application.services.CoursesHasStudentsServise;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
@Controller
@RequestMapping("/auth")
public class AuthController {
   private final HttpSessionBean httpSessionBean;
   private final CourseImpl courseService;
   private final CoursesHasStudentsServise coursesHasStudentsServise;
   @Autowired
   public AuthController(HttpSessionBean httpSessionBean, CourseImpl courseService,
CoursesHasStudentsServise coursesHasStudentsServise) {
      this.httpSessionBean = httpSessionBean;
      this.courseService = courseService;
      this.coursesHasStudentsServise = coursesHasStudentsServise;}
   @GetMapping("/login")
   public String getLoginPage() {
      return "login";}
   @GetMapping("/success")
   public String getSuccessPage(Model model) {
      if(httpSessionBean.getUser().getRole().equals(Role.TEACHER)) {
         model.addAttribute("teacher", httpSessionBean.getUser());
System.out.println(courseService.getAllTeachersCourses(httpSessionBean.getUser().getId()));
```

```
            model.addAttribute("courses",
courseService.getAllTeachersCourses(httpSessionBean.getUser().getId()));
            return "teacher/teacherPage";}
        model.addAttribute("student", httpSessionBean.getUser());
        model.addAttribute("course1", new Course());
        model.addAttribute("joinCourses",
coursesHasStudentsServise.getCoursesOfStudentById(httpSessionBean.getUser().getId()));
        return "student/studentPage";
    } }
```

Лістинг файлу “src/java/com/vntu/learn.application/StudentController.java”

```
package com.vntu.learn.application.controllers;
import com.vntu.learn.application.bean.FileLoader;
import com.vntu.learn.application.bean.HttpSessionBean;
import com.vntu.learn.application.bean.MediaTypeUtils;
import com.vntu.learn.application.models.Course;
import com.vntu.learn.application.models.Task;
import com.vntu.learn.application.services.*;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
import javax.servlet.ServletContext;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
```

```
import java.util.Arrays;
import java.util.List;
@Controller
@RequiredArgsConstructor
@RequestMapping("/student")
public class StudentController {
    private final UserService userService;
    private final CourseService courseService;
    private final CoursesHasStudentsServise coursesHasStudentsServise;
    private final TaskService taskService;
    private final FileService fileService;
    private final AnswerService answerService;
    private final HttpSessionBean httpSessionBean;
    @Autowired
    private ServletContext servletContext;
    private static final String DIRECTORY = "src/main/resources/storage/student";
    private static final String DEFAULT_FILE_NAME = "";

    @Value("${upload.path1}")
    private String path;
    @Value("${upload.path}")
    private String teacherPathFiles;
    @GetMapping
    public String getUser(Model model) {
        model.addAttribute("student", httpSessionBean.getUser());
        model.addAttribute("course1", new Course());
        model.addAttribute("joinCourses",
coursesHasStudentsServise.getCoursesOfStudentById(httpSessionBean.getUser().getId()));
        return "student/studentPage";}
    @PostMapping("/searchCourse")
    public String searchCourses(@ModelAttribute("course1") Course course,
                                Model model) {
        List<Course> courseList = courseService.getCoursesByName(course.getCourseName());
        model.addAttribute("student", httpSessionBean.getUser());
        model.addAttribute("courses", courseList);
```

```
        model.addAttribute("joinCourses",
coursesHasStudentsServise.getCoursesOfStudentById(httpSessionBean.getUser().getId()));
        return "student/studentPage";}
    @GetMapping("/all")
    public String getAll(Model model) {
        model.addAttribute("users", userService.getAllStudents());
        return "student/allStudents";}
    @GetMapping("/join/{cid}")
    public String joinCourse(@PathVariable("cid") long cid,
                             Model model) {
coursesHasStudentsServise.addStudentToCourse(httpSessionBean.getUser().getId(), cid);
        answerService.addUserToAnswers(httpSessionBean.getUser().getId());
        model.addAttribute("student", httpSessionBean.getUser());
        model.addAttribute("course1", new Course());
        model.addAttribute("joinCourses",
coursesHasStudentsServise.getCoursesOfStudentById(httpSessionBean.getUser().getId()));
        return "student/studentPage";}
    @GetMapping("/course/{cid}")
    public String coursePage(@PathVariable("cid") long cid, Model model) {
        Course course = courseService.getCourseById(cid);
        model.addAttribute("course", course);
        model.addAttribute("tasks", taskService.getAllTaskOfCourseId(course));
        return "student/coursePage";}
    @GetMapping("/course/{cid}/{tid}")
    public String taskPage(@PathVariable("cid") long cid,
                           @PathVariable("tid") long tid,
                           Model model) {
        model.addAttribute("tasks",
taskService.getAllTaskOfCourseId(courseService.getCourseById(cid)));
        model.addAttribute("task", taskService.getTaskById(tid));
        String p = path + "/" + cid;
        File a = new File(p);
        if(!a.exists()) {
            a.mkdir();
        }
```

```
        p += "/" + tid;
        a = new File(p);
        if(!a.exists()) {
            a.mkdir();
        }
        List<File> files = FileLoader.getAllFilesOfDirectory(p);
        List<File> teachersFiles = FileLoader.getAllFilesOfDirectory(teacherPathFiles + "/" + cid +
"/" + tid);
        model.addAttribute("teacherFiles", teachersFiles);
        model.addAttribute("files", files);
        model.addAttribute("uid", httpSessionBean.getUser().getId());
        return "student/taskPage";}
    @RequestMapping("/{cid}/{tid}/{uid}/download")
    public ResponseEntity<InputStreamResource> downloadFile(
            @PathVariable("cid") long cid,
            @PathVariable("tid") long tid,
            @PathVariable("uid") long uid,
            @RequestParam(defaultValue = DEFAULT_FILE_NAME) String fileName) throws
IOException {
        MediaType mediaType = MediaTypeUtils.getMediaTypeForFileName(this.servletContext,
fileName);
        File file = new File(DIRECTORY + "/" + cid + "/" + tid + "/" + uid + "/" + fileName);
        InputStreamResource resource = new InputStreamResource(new FileInputStream(file));
        return ResponseEntity.ok()
                .header(HttpHeaders.CONTENT_DISPOSITION,
                  "attachment;filename=" + file.getName())
                .contentType(mediaType)
                .contentLength(file.length())
                .body(resource);}
    @PostMapping("/course/{cid}/{tid}/{uid}")
    public String uploadFiles(
            @RequestParam("files") MultipartFile[] files, RedirectAttributes redirectAttributes,
            @PathVariable("cid") long cid,
            @PathVariable("tid") long tid,
            @PathVariable("uid") long uid,
```

```
            Model model) {
        String p = path + "/" + cid;
        File a = new File(p);
        if(!a.exists()) {
            a.mkdir();
        }
        try {
            String finalP = p;
            Arrays.asList(files)
                    .stream()
                    .forEach(file -> fileService.uploadFile(finalP, file));

            redirectAttributes.addFlashAttribute("message",
                    "You successfully uploaded all files!");
        }
        catch (Exception exc) {
            Task task = new Task();
            task.setCourseId(courseService.getCourseById(cid));
            model.addAttribute("task", task);
            return "/student/taskPage";
        }
        return "redirect:/student/course/" + cid + "/" + tid;
    }
}
```

Лістинг файлу “src/java/com/vntu/learn.application/TeacherController.java”

```
package com.vntu.learn.application.controllers;
import com.vntu.learn.application.bean.FileLoader;
import com.vntu.learn.application.bean.HttpSessionBean;
import com.vntu.learn.application.bean.MediaTypeUtils;
import com.vntu.learn.application.models.Answer;
import com.vntu.learn.application.models.Course;
import com.vntu.learn.application.models.Task;
import com.vntu.learn.application.services.*;
```

```
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
import javax.servlet.ServletContext;
import javax.validation.Valid;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.*;
import java.util.stream.Collectors;
@Controller
@RequiredArgsConstructor
@RequestMapping("/teacher")
public class TeacherController {
    private static final String DICT =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    private static final Random random = new Random();
    private final UserService userService;
    private final CourseService courseService;
    private final TaskService taskService;
    private final FileService fileService;
    private final AnswerService answerService;
    private static final String DIRECTORY = "src/main/resources/storage";
    private static final String DEFAULT_FILE_NAME = "";
    @Autowired
```

```
    private ServletContext servletContext;
    private final HttpSessionBean httpSessionBean;
    @Value("${upload.path}")
    private String path;
    @Value("${upload.path1}")
    private String path1;
    @GetMapping
    public String getUser(Model model) {
        model.addAttribute("teacher", httpSessionBean.getUser());
        model.addAttribute("courses",
courseService.getAllTeachersCourses(httpSessionBean.getUser().getId()));
        return "teacher/teacherPage"; }
    @GetMapping("/change")
    public String changeInfoAboutTeacher() {
        return "teacher/";}
    @GetMapping("/all")
    public String getAllTeacher(Model model) {
        model.addAttribute("users", userService.getAllTeachers());
        return "teacher/allTeachers";}
    @GetMapping("/createCourse")
    public String createCourse(@ModelAttribute("course") Course course) {
        return "/teacher/createCourse";    }
    @PostMapping("/createCourse")
    public String createCourse(@ModelAttribute("course") @Valid Course course,
                               BindingResult bindingResult) {
        if(bindingResult.hasErrors()) {
            return "/teacher/createCourse";        }
        course.setUserId(httpSessionBean.getUser());
        courseService.addCourse(course);
        return "redirect:/teacher/";  }
    @GetMapping("/course/{cid}")
    public String takeCourse(@PathVariable("cid") long cid,
                             Model model) {
        Course course = courseService.getCourseById(cid);
        model.addAttribute("course", course);
```

```
        model.addAttribute("tasks", taskService.getAllTaskOfCourseId(course));
        return "/teacher/coursePage"; }
    @GetMapping("course/{cid}/{tid}")
    public String getTask(@PathVariable("cid") long cid,
                          @PathVariable("tid") long tid,
                          Model model) throws IOException {
        Course course = courseService.getCourseById(cid);
        model.addAttribute("task", taskService.getTaskById(tid));
        model.addAttribute("tasks", taskService.getAllTaskOfCourseId(course));
        String p = path + "/" + cid;
        List<File> files = FileLoader.getAllFilesOfDirectory(p);
        model.addAttribute("files", files);
        return "/teacher/taskPage";
    }
    @GetMapping("/course/{cid}/addTask")
    public String addTaskForCourse(
                            @PathVariable("cid") long cid,
                               Model model) {
        Task task = new Task();
        task.setCourseId(courseService.getCourseById(cid));
        model.addAttribute("task", task);
        return "/teacher/createTask" }
@PostMapping("/course/{cid}/addTask")
    public String addTaskForCoursePost(@PathVariable("cid") long cid,
                                @ModelAttribute("task") @Valid Task task,
                                BindingResult bindingResult) {
        task.setCourseId(courseService.getCourseById(cid));
        if(bindingResult.hasErrors()) {
            return "/teacher/createTask";
        }
        taskService.addTask(task);
        Task task1 = taskService.getTaskByNameAndCourseId(task.getTaskName(),
task.getCourseId().getId());
        answerService.addUserToTaskList(task1.getId());
        return "redirect:/teacher/course/" + task.getCourseId().getId();}
```

```
    @PostMapping("/course/{cid}/{tid}")
    public String uploadFiles(
            @RequestParam("files") MultipartFile[] files, RedirectAttributes redirectAttributes,
            @PathVariable("cid") long cid,
            @PathVariable("tid") long tid,
            Model model) {
        String p = path + "/" + cid;
        File a = new File(p);
        if(!a.exists()) {
            a.mkdir();
        }
        try {
            String finalP = p;
            Arrays.asList(files)
                    .stream()
                    .forEach(file -> fileService.uploadFile(finalP, file));
            redirectAttributes.addFlashAttribute("message",
                    "You successfully uploaded all files!");
        }
        catch (Exception exc) {
            Task task = new Task();
            task.setCourseId(courseService.getCourseById(cid));
            model.addAttribute("task", task);
            return "/teacher/taskPage";
        }
        return "redirect:/teacher/course/" + cid + "/" + tid;
    }
    @RequestMapping("/{cid}/{tid}/download")
    public ResponseEntity<InputStreamResource> downloadFile(
            @PathVariable("cid") long cid,
            @PathVariable("tid") long tid,
            @RequestParam(defaultValue = DEFAULT_FILE_NAME) String fileName) throws
IOException {
```

```
        MediaType mediaType = MediaTypeUtils.getMediaTypeForFileName(this.servletContext,
fileName);
        File file = new File(DIRECTORY + "/" + cid + "/" + tid + "/" + fileName);
        InputStreamResource resource = new InputStreamResource(new FileInputStream(file));
        return ResponseEntity.ok()
                .header(HttpHeaders.CONTENT_DISPOSITION, "attachment;filename=" +
file.getName())
                .contentType(mediaType)
                .contentLength(file.length()) //
                .body(resource);
    }
    @GetMapping("/course/{cid}/{tid}/answers")
    public String answersForTask(@PathVariable("cid") long cid,
                                 @PathVariable("tid") long tid,
                                 Model model) {
        Course course = courseService.getCourseById(cid);
        model.addAttribute("tasks", taskService.getAllTaskOfCourseId(course));
        String p = path + "/student" + "/" + cid;
        File a = new File(p);
        if(!a.exists()) {
            a.mkdir();
        }
        List<Answer> answers = answerService.getAnswersByTaskId(tid);
        Map<Long, List<File>> answersAndFilesList = new HashMap<>();
        for(int i = 0; i < answers.size(); i++) {
            String finalP = p;
            String pp = finalP + "/" + answers.get(i).getUser().getId();
            System.out.println(pp);
            File aaa = new File(pp);
            answersAndFilesList.put(answers.get(i).getUser().getId(),
FileLoader.getAllFilesOfDirectory(pp));
            FileLoader.getAllFilesOfDirectory(pp).forEach(System.out::println);
        }
        model.addAttribute("task", taskService.getTaskById(tid));
        model.addAttribute("answers", answers);
```

```
        model.addAttribute("answer1", new Answer());
        model.addAttribute("studentsFiles", answersAndFilesList);
        return "teacher/answersForTask";
    }
    @PostMapping("/course/{cid}/{tid}/{uid}")
    public String setMark(@PathVariable("cid") long cid,
                          @PathVariable("tid") long tid,
                          @PathVariable("uid") long uid,
                          @ModelAttribute("answer1") @Valid Answer answer,
                          Model model) {
if(answer.getMark() > taskService.getTaskById(tid).getMaxMark()) {
            return "redirect:/teacher/course/" + cid + "/" + tid + "/" + "answers";
        }
        answerService.setMarkForTask(answer.getMark(), tid, uid);
        return "redirect:/teacher/course/" + cid + "/" + tid + "/" + "answers";}
    private String makeKeyCourse() {
        StringBuilder key = new StringBuilder();
        for (int i = 0; i < 8; i++ )
            key.append(DICT.charAt(random.nextInt(DICT.length())));
        return key.toString();}
    @RequestMapping("/{cid}/{tid}/{uid}/download")
    public ResponseEntity<InputStreamResource> downloadStudentFile(
            @PathVariable("cid") long cid,
            @PathVariable("tid") long tid,
            @PathVariable("uid") long uid,
            @RequestParam(defaultValue = DEFAULT_FILE_NAME) String fileName) throws
IOException {
        MediaType mediaType = MediaTypeUtils.getMediaTypeForFileName(this.servletContext,
fileName);
        File file = new File(DIRECTORY + "/student"+ "/" + cid + "/" + tid + "/" + uid + "/" +
fileName);
        InputStreamResource resource = new InputStreamResource(new FileInputStream(file));
        return ResponseEntity.ok()
                .header(HttpHeaders.CONTENT_DISPOSITION, "attachment;filename=" +
file.getName())
```

```
                .contentType(mediaType)
                .contentLength(file.length()) //
                .body(resource);}
    @GetMapping("/course/{cid}/resultsPage")
    public String resultsPage(@PathVariable("cid") long cid,
                              Model model) {
        List<Answer> allAnswersList = answerService.getAllAnswersByCourseId(cid);
        Course course = courseService.getCourseById(cid);
        model.addAttribute("tasks1", taskService.getAllTaskOfCourseId(course));
        List<List<Answer>> listAnswers = new ArrayList<>();
        model.addAttribute("answers", answerService.getAllAnswersByCourseId(cid));
        model.addAttribute("tasks", answerService.getAllAnswersByCourseIdDistinct(cid));
        model.addAttribute("firstTask",
answerService.getAllAnswersByCourseIdDistinct(cid).get(0));
answerService.getAllAnswersByCourseId(cid).stream().forEach(System.out::println);
        List<Long> listStudentId = allAnswersList.stream().map(el ->
el.getUser().getId()).distinct().collect(Collectors.toList());
        listStudentId.stream().forEach( el ->
listAnswers.add(answerService.getAnswerByCourseIdAndUserId(cid, el)));
        model.addAttribute("groupStudents", listAnswers);
        return "teacher/resultsPage";
    }}
```

ДОДАТОК Ж

Структурна схема бази даних

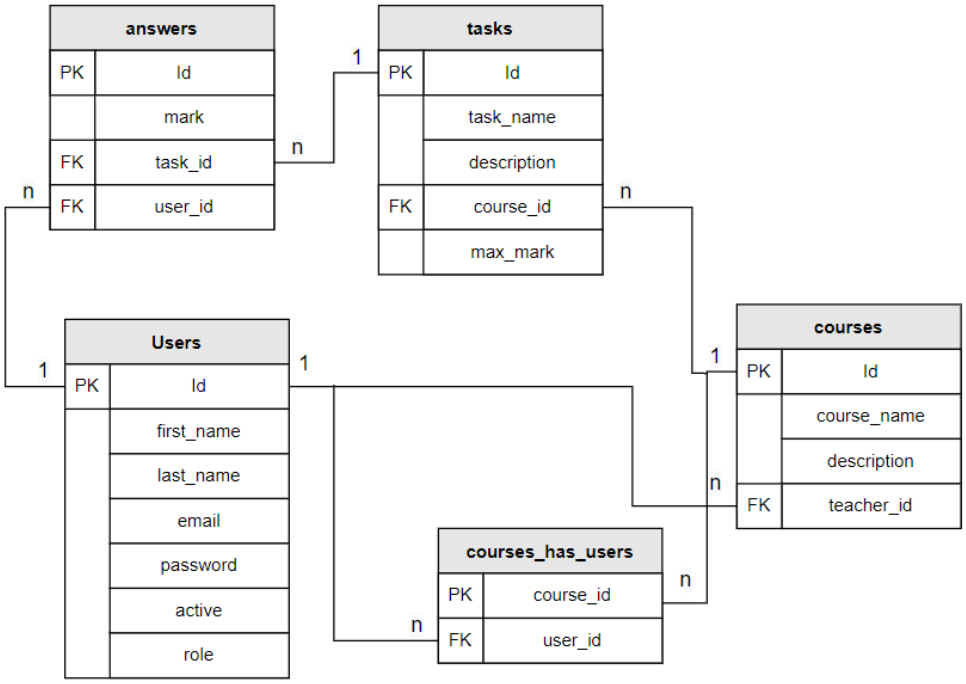

Рисунок Ж.1 — Структурна схема бази даних

ДОДАТОК И

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Веб-застосунок для організації та оцінювання навчального процесу з використанням хмарних сервісів.

Тип роботи: магістерська дипломна робота

Підрозділ кафедра обчислювальної техніки

Показники звіту подібності
Unicheck

Оригінальність 93,1% Схожість 6,9%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Терещук Е. П.

Керівник роботи _____ Тарновський М. Г.