**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: **«ДОСЛІДЖЕННЯ ІНТЕГРОВАНОГО СЕРЕДОВИЩА ВЕБ СЛУЖБ З МЕТОЮ СТВОРЕННЯ ВЕБ-ПЛАТФОРМИ ДЛЯ НАДАННЯ РІЗНОМАНІТНИХ ПРОГРАМНИХ ПОСЛУГ КОРИСТУВАЧАМ»**

на здобуття освітнього ступеня магістр
за спеціальності 123 Комп'ютерна інженерія
(код,найменування спеціальності)
освітньо-професійної програми Комп'ютерні системи та мережі
(назва)

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

_____ Анна КОВАЛЕНКО
(підпис) *(ім'я, ПРІЗВИЩЕ здобувача)*

Виконав: здобувачка вищої освіти гр.КСДМ-62
Анна КОВАЛЕНКО
(ім'я, ПРІЗВИЩЕ)

Керівник: Наталія ЛАЩЕВСЬКА
к.т.н., доцент *(ім'я, ПРІЗВИЩЕ)*

Рецензент: _____
науковий ступінь, вчене звання *(ім'я, ПРІЗВИЩЕ)*

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерної інженерії
Ступінь вищої освіти «Магістр»

Спеціальність 123 Комп'ютерна інженерія
Освітньо-професійна прорама Комп'ютерні системи та мережі

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерної інженерії
Наталія ЛАЩЕВСЬКА
(ім'я, ПРІЗВИЩЕ)
"____" _____2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Коваленко Анні Русланівні
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Дослідження інтегрованого середовища веб-служб з метою створення веб-платформи для надання різноманітних програмних послуг користувачам
керівник роботи Наталія ЛАЩЕВСЬКА к.т.н., доцент
(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від "19" 10 2023 р. №145
2. Строк подання кваліфікаційної роботи
3. Вихідні дані кваліфікаційної роботи:
3.1. Інтегровані середовища веб-служб.
3.2. Веб-платформи для надання програмних послуг.
3.3. Науково-технічна література.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
4.1. Теоретичні основи створення веб-сервісів.
4.2. Проектування веб-платформи для вивчення онлайн курсів.
4.3. Етапи та методи розробки веб-платформи.
4.4. Тестування створеної веб-платформи.

5.Перелік ілюстраційного матеріалу: *презентація*

6. Дата видачі завдання “19” жовтня 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	**Примітка**
1.	Підбір технічної літератури	.2023р. .2023р.	Виконано
2.	Теоретичні основи створення веб-сервісів	.2023р. .2023р.	Виконано
3.	Проектування веб-платформи для вивчення онлайн курсів	.2023р. .2023р.	Виконано
4.	Етапи та методи розробки веб-платформи	.2023р. .2023р.	Виконано
5.	Тестування створеної веб-платформи	.2023р. .2023р.	Виконано
6.	Оформлення роботи, висновки	.2023р. .2023р.	Виконано
7.	Розробка демонстраційного матеріалу, доповідь	.2023р. .2023р.	Виконано

Здобувач вищої освіти Анна КОВАЛЕНКО

(підпис) *(ім’я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи Наталія ЛАЩЕВСЬКА

(підпис) *(ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття ступеня магістр: 81 стор., 25 рис., 3 табл., 24 джерел.

Мета роботи – створення веб-платформи для надання різноманітних програмних послуг користувачам для забезпечення зручного та ефективного доступу до них.

Об'єкт дослідження – процес створення веб-платформи.

Предмет дослідження – інтегроване середовище веб-служб.

Короткий зміст роботи: В цій магістерській роботі розглянуто теоретичні основи створення веб-сервісів, теорію розробки та інструменти веб-серверів.

В дипломній роботі реалізовано веб-платформу для вивчення онлайн курсів, призначений для бажаючих поглибити свої знання з різних областей знань. Веб-додаток дозволяє користувачам переглядати відео, читати освітні матеріали та проходити тести.

Програмний продукт був створений з використанням мікросервісної архітектури. Для серверної частини був використаний додаток MAMP з базою даних MySql. Для клієнтської частини було використано такі мови програмування: JS, HTML, CSS, PHP.

КЛЮЧОВІ СЛОВА: ІНТЕГРОВАНЕ СЕРЕДОВИЩЕ ВЕБ-СЛУЖБ, ВЕБ-ПЛАТФОРМА, ПРОГРАМНІ ПОСЛУГИ, ІНТЕРНЕТ.

ABSTRACT

The text part of the qualification work for obtaining a master's degree: 81 pages, 25 figures, 3 tables, 24 sources.

The purpose of the work is creation of a web platform for providing various software services to users to ensure convenient and efficient access to them.

The object of research is the process of creating a web platform.

The subject of research is integrated web services environment.

Summary of the work: This master's thesis examines the theoretical foundations of creating web services, development theory and web server tools.

The thesis has implemented a web platform for teaching online courses for those wishing to expand their knowledge of various areas of knowledge. The web extension allows students to watch videos, read educational materials, and take tests.

A software product created using a unique microservice architecture. For the server side, we use the MAMP add-on with the MySql database. For the client part, the following programming was used: JS, HTML, CSS, PHP.

KEY WORDS: INTEGRATED ENVIRONMENT OF WEB SERVICES, WEB PLATFORM, SOFTWARE SERVICES, INTERNET.

ЗМІСТ

ВСТУП

Сучасний інтернет є невід'ємною частиною життя людини в багатьох сферах. Завдяки тому, що інтернет поширений практично по всьому світу, користувачі мають можливість вигідно і швидко вирішувати безліч завдань, вирішення яких в звичайних умовах зайняло б набагато більше часу і сил. Однак Інтернет не є однорідним – він об'єднує безліч платформ, заснованих на різних технологіях, а пошук інформації здійснюється з безлічі розрізнених джерел. У зв'язку з цим спостерігається тенденція до створення способу відображення інформації таким чином, щоб вона була зручно організована і придатна для подальшої обробки. Одним з плодів цієї тенденції є концепція Web Services, суть якої полягає в уніфікації та впорядкуванні інформації, а також інтеграції різнорідних систем, що розширюють можливості сервісу. Сучасний Інтернет неможливо уявити без веб-сервісів.

Зростання популярності веб-служб та потреба у наданні різноманітних програмних послуг користувачам створює потребу у розробці інтегрованого середовища та веб-платформи для їх надання. Це дозволить забезпечити зручний та ефективний доступ до програмних послуг, сприяти розвитку електронної комерції та покращенню якості обслуговування користувачів.

Виходячи з вищесказаного, метою даної магістерської роботи є створення веб-платформи для надання різноманітних програмних послуг користувачам для забезпечення зручного та ефективного доступу до них.

Поряд із зростаючими вимогами до використання програмного забезпечення для моделювання моделей і фізичного прогнозування в промисловості, професіонали вважають за краще вибирати ефективний і ефективний спосіб досягнення своїх цілей, використовуючи професійні програмні додатки. Необхідно надати користувачам простий спосіб доступу до програмних послуг на вимогу без необхідності самостійно налаштовувати та підтримувати їх.

Веб-платформи для вивчення онлайн курсів – це один з видів дистанційного навчання, які передбачають застосування спеціалізованих ресурсів для взаємодіяння між викладачем та студентом.

Незалежно від того, чи прагнете ви навчитися деяким новим товарним навичкам, чи просто хочете дослідити тему для розваги, онлайн-платформи для навчання - чудовий та легкодоступний ресурс для навчання за власним розкладом.

Як альтернатива Інтернет-коледжам, ці платформи, як правило, трохи гнучкіші і можуть пропонувати навіть більш специфічні або незвичні заняття, яких ви не знайдете в традиційному коледжі, але важливо, щоб майбутні студенти порівнювали свої варіанти, щоб знайти найкраще для них .

Є дві основні категорії онлайн-курсів:

Перша категорія – це синхронне навчання величезної кількості студентів, для яких викладач проводить лекцію онлайн у відведений ним час. Позитивні ознаки таких курсів – це їх схожість до занять в аудиторіях: студент може поставити запитання лектору й отримувати відповіді у реальному часі.

Друга – це самостійний вибір курсу та виділення часу для нього, тобто ви проходите навчання у будь-який момент, коли маєте бажання.

Тому було вирішено створити просту веб-платформу для вивчення онлайн курсів, які доступні кожному.

1 ТЕОРЕТИЧНІ ОСНОВИ СТВОРЕННЯ ВЕБ-СЕРВІСІВ

1.1 Теорія розробки веб-сервісів

Сьогодні Інтернет є сполучною ланкою між величезною кількістю людей, що живуть на планеті. За допомогою Інтернету люди, що знаходяться на великих відстанях один від одного, можуть здійснювати різні контакти один з одним, суть яких може полягати в спілкуванні, командній роботі, торгових операціях і т.д. Крім різних соціальних функцій, Інтернет також функціонує як величезний резервуар цифрової інформації, який може бути представлений у вигляді тексту, зображень, цифрового аудіо тощо, програми – така функція дозволяє людям зберігати в Інтернеті паспортні дані, фотографії, електронні заощадження тощо.

У розвинених країнах Інтернет вже давно став невід'ємною частиною людської діяльності. Пошук необхідної інформації, перебуваючи вдома, на роботі чи в дорозі, – це лише одна з багатьох можливостей, які надає Інтернет, щоб полегшити життя людям, незалежно від сфери їхньої діяльності. Крім повсякденної інформації, інтернет також зберігає інформацію на більш високому рівні – державні документи, результати досліджень, курси валют – іншими словами, Інтернет відіграє важливу роль не тільки в житті окремої людини, але і в роботі цілих компаній, структур і навіть держав.

Ключовим елементом мережевого простору, без якого неможливе використання Інтернету, є веб-сервіс. Дане поняття має багато різних значень, але найчастіше під веб-сервісом розуміють програмну систему зі стандартизованими інтерфейсами, представлену браузером користувача у вигляді набору HTML-сторінок. Відмінність від веб-сайту полягає в тому, що веб-сервіс служить для надання користувачеві послуги: веб-сервіс дозволяє користувачеві безпосередньо замовити товар або послугу (наприклад, придбати товар з подальшою доставкою додому або майже), в той час як веб-сайт містить лише інформаційну інформацію.

Якісний веб-сервіс відрізняється не тільки рівнем сервісу - дизайн та навігація є однаково важливими показниками такого сервісу. Високий рівень дизайну має на увазі приємне візуальне оформлення веб-сторінок, завдяки чому забезпечується привабливість веб-сервісу, а грамотно організована навігація допомагає користувачеві не тільки добре орієнтуватися при навігації між сторінками, але і більш ефективно користуватися послугами, що надаються веб-сервісом.

Сучасні технології створення веб-сервісів спрямовані в першу чергу на поліпшення процесу взаємодії користувача з системою. На сьогоднішній день існує безліч систем для створення веб-сервісу. Незважаючи на функціональність сучасних систем, для того, щоб створити веб-сервіс, необхідно виконати певну роботу поза електронним середовищем, наприклад, подумати над дизайном. Загалом розробка веб-сервісу ділиться на кілька етапів: планування, впровадження, тестування та публікація.

На етапі планування формується мета веб-сервісу: визначається його призначення, коло користувачів, для яких призначений сервіс, а також інформація, розміщена на його сторінках. В ході планування структури сайту визначається кількість категорій, розділів, підрозділів, а також кількість сторінок всередині них. Планується оформлення веб-сторінок сайту. На початку етапу реалізації готується матеріал, який необхідний для наповнення сайту (текст, зображення). Веб-сторінки розробляються відповідно до їх призначення, сторінки з’єднуються посиланнями відповідно до навігаційної схеми сайту. Проект, визначений на попередньому етапі, реалізується. Версії сайту створюються для пристроїв з різними персональними операційними системами та роздільною здатністю екранів, а при необхідності і версії для людей з обмеженими можливостями.

Тестування - це етап, на якому здійснюється пошук всіляких технічних помилок, пов'язаних з роботою сайту, з подальшою їх установкою. Перевіряється відображення сайту на різних версіях мови програмного забезпечення, на якому

написаний програмний код сайту, так як не всі браузери однаково відображають одну і ту ж веб-сторінку.

Публікація - це завершальний етап розробки, в ході якого сайт розміщується на виділеному сервері, що в майбутньому дозволить користувачеві побачити готовий сайт в інтернеті і користуватися представленими на ньому сервісами.

Більшість сучасних веб-сервісів мають можливість зв'язку між собою і зі сторонніми додатками за допомогою визначених стандартизованих протоколів (SOAP, XML-RPC) і угод (REST). Ця властивість робить будь-який сучасний веб-сервіс невід'ємною частиною так званої сервісно-орієнтованої архітектури . Згідно з визначенням, наведеним у специфікації OASIS RM-SOA 1.0, сервісно-орієнтована архітектура (SOA) – це парадигма організації та використання розподілених інформаційних ресурсів, які можуть належати різним власникам. Сервісно-орієнтована архітектура також є підходом до розробки програмного забезпечення, який використовує модульні компоненти зі стандартизованими інтерфейсами. Модульність цих компонентів полягає в тому, що вони можуть бути розподілені між собою і слабо пов'язані, а при необхідності можуть бути легко замінені іншими компонентами, які взаємодіють за допомогою аналогічного протоколу.

Архітектура програми - це сукупність програмних елементів, їх зовнішніх властивостей і взаємозв'язків. Проектування архітектури програми є одним з найважливіших завдань розробника, так як від її якості залежить безліч параметрів, які в сукупності визначають якість готового програмного продукту. Архітектура програмного забезпечення визначає межі реалізації програмного забезпечення, містить інформацію про первинні проектні рішення та організаційну структуру системи. Крім того, грамотно продумана архітектура програми сприяє більш ретельному плануванню розподілу ресурсів, необхідних для розвитку. Будь-яка програмна система базується на структурах, що містяться в її програмній архітектурі. Однак архітектура дає лише загальне уявлення про програмну систему, показуючи лише взаємозв'язок програмних блоків і принцип їх взаємодії один з одним.

Хоча архітектура програмного забезпечення забезпечує лише абстрактне уявлення про систему, вона впливає на багато факторів, таких як продуктивність програмної системи, її функціональність, надійність тощо. Однак такий параметр, як здатність програмних систем інтегруватися один з одним, стає все більш популярним. Інтеграція програмних систем - це об'єднання двох або більше програмних систем з метою підвищення якості їх роботи. Даний спосіб роботи з програмними системами має на увазі, що інформація, введена користувачем в одну систему, також потрапляє в іншу систему, інтегровану з першою. Інтеграція систем дає можливість розширити можливості обробки інформації, а в ряді випадків і модифікувати існуючу програмну систему.

Існують стандарти та правила, яких дотримуються, щоб забезпечити створення якісної та надійної архітектури програмного забезпечення, тоді як недотримання цієї вимоги, швидше за все, призведе до не ефективного розвитку архітектурної розробки. Виходячи зі стандарту ISO/IEC/IEEE 42010, критерії високоякісної архітектури включають такі параметри, як: ефективність, гнучкість, розширюваність і масштабованість процесу розробки.

Ефективність - це параметр, який відповідає за загальну продуктивність програмної системи. Надійність процесів виконання поставлених завдань, швидкість їх виконання, витримування навантажень - характеристики, за якими оцінюється ефективність архітектури.

Під гнучкістю розуміється здатність системи зазнавати змін з найменшим впливом на інші її елементи, які не потребують змін, і найменшою кількістю помилок, що виникають в результаті цих змін. У певний момент часу розвивається програмна система неминуче потребує якихось точкових змінах своїх елементів, але такі зміни можуть супроводжуватися необхідністю зміни інших елементів програми, пов'язаних з тим, що було змінено спочатку. Гнучкість архітектури програмного забезпечення тим вище, чим менше такі зміни зачіпають інші елементи системи.

Розширюваність означає здатність системи додавати нові функції без шкоди для своєї базової структури. Можливість додавання нових функцій без втручання в основний код програми є одним із принципів об'єктно-орієнтованого програмування (SOLID – single responsibility, open-closed, Liskov substitution, segregation і dependency inversion), який стверджує, що «сутності програми повинні бути відкритими для розширення, але закритими для модифікації».

Масштабованість процесу розробки – це можливість додавати до процесу розробки інших людей. Простіше залучити сторонніх розробників до розробки системи, яка легко зрозуміла іншим людям, і рівномірно розподілити процес розробки між ними.

Формуванням світових стандартів в області інформаційно-комунікаційних технологій займаються кілька міжнародних організацій, до яких відносяться: Консорціум Всесвітньої павутини (WWW), Java Community Process (JCP), Інститут програмної інженерії (SEI), Інститут інженерів з електротехніки та електроніки (IEEE), Object Management Group (OMG) та інші.

Програмною основою сучасних веб-сервісів є сервісно-орієнтована архітектура (SOA), підхід до розробки програмного забезпечення, заснований на використанні сервісів зі стандартизованими інтерфейсами.

Сьогодні еталонною моделлю сервісно-орієнтованої архітектури є SOA-RM (Reference Model for Service-Oriented Architecture), яка належить організації зі стандартизації OASIS. Ця модель була затверджена в жовтні 2006 року як єдина еталонна модель сервісно-орієнтованої архітектури. До цього часу не існувало стандартного визначення SOA. SOA-PM надає абстрактне уявлення про значущі об'єкти програмної системи і методи їх взаємодії один з одним, а також надає інструкції з розробки гармонізованих стандартів і специфікацій, що підтримують сервісно-орієнтоване середовище. Ця модель забезпечує середовище SOA чіткою технічною термінологією, яка потрібна розробникам, а структурована інформація про сервісно-орієнтовану архітектуру дозволяє більш доступно навчати або

пояснювати концепції SOA. Технічний комітет OASIS зазначає, що SOA-RM не пов'язана з технологіями та стандартами інших окремих реалізацій SOA.

Сучасні веб-сервіси засновані на використанні міжнародних інтернет-стандартів. Завдяки тому, що вони не впливають на конкретно способи реалізації SOA-додатків, а лише визначають, як вони працюють, компанії-розробники не можуть впливати на існуючі міжнародні стандарти за допомогою власних розробок. Веб-сервіси базуються на декількох основних стандартах: SOAP (Simple Object Access Protocol) – протокол обміну довільними повідомленнями у форматі XML; WSDL (Web Services Description Language) — заснована на XML мова для опису інтерфейсів програмування веб-сервісів. UDDI (Universal Descrip- tion Discovery and Integration) – інструмент для індексації веб-сервісів.

Зв'язок між веб-сервісом і користувачем здійснюється в форматі XML з використанням протоколу SOAP. Цей протокол забезпечує розширення деяких прикладних протоколів, таких як HTTP, HTTPS, FTP, SMTP і т.д., але SOAP найчастіше використовується по відношенню до HTTP. Обмін інформацією по протоколу SOAP між інформаційними системами відбувається за допомогою обміну структурованими XML-повідомленнями. Він відрізняється від звичайного обміну тим, що інформація, що передається у форматі XML, структурована особливим чином, щоб забезпечити обмін інформацією між системами, які зазвичай не мають засобів для зв'язку один з одним.

Структура SOAP містить три основні елементи: Envelope (конверт), Header (заголовок), і Body (тіло), (рис. 1.1).

Envelope — це кореневий елемент, який визначає повідомлення та простір імен, використані в документі.

Header – містить атрибути повідомлення.

Body — містить повідомлення, призначене для обміну.

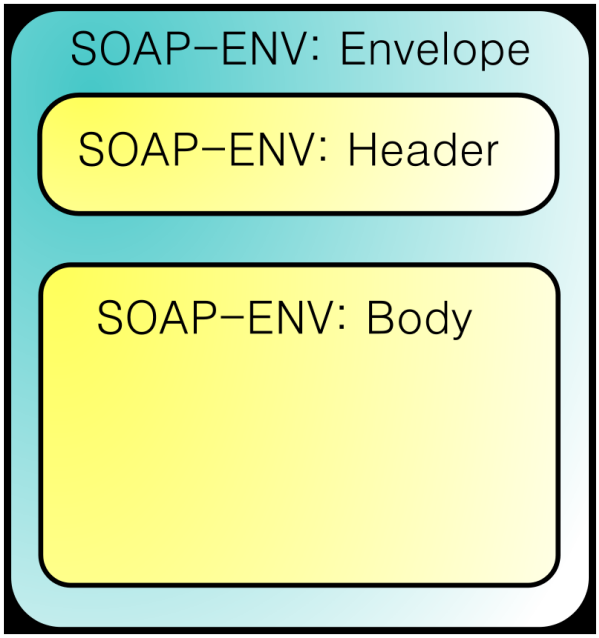


Рисунок 1.1 - Структура повідомлень SOAP1

SOAP є одним з найважливіших компонентів технології веб-сервісів, оскільки він переміщує дані по мережі. SOAP підтримує загальний протокол обміну даними між відправником і одержувачем XML-документів, тим самим забезпечуючи ефективний зв'язок між ними. SOAP забезпечує односпрямоване з'єднання, за допомогою якого дані послідовно передаються від відправника до одержувача. При необхідності на маршруті з'єднання можуть розташовуватися посередники, завдання яких обробляти або доповнювати інформацію, що проходить через них.

Для того, щоб клієнт мав можливість спілкуватися та взаємодіяти з веб-сервісом, сервіс надає WSDL-документ, що містить інформацію та описи, необхідні для роботи з веб-сервісом. WSDL – це формат на основі XML, метою якого є опис синтаксису веб-сервісів за допомогою повідомлень, що містять інструкції щодо доступу до функцій конкретного веб-сервісу. Так як мова WSDL строго формалізована, то в більшості випадків використовується можливість автоматичного створення документів WSDL, яка реалізується за допомогою спеціальних утиліт, які йдуть в комплекті з програмними засобами веб-розробки, але також є можливість написати цей документ самостійно.

Пошук потрібного веб-сервісу є першочерговим завданням для клієнта, який має потребу, але не має уявлення про веб-сервіс, який її задовольнить. Існують програмні системи, які здійснюють пошук веб-сервісів у спеціальних реєстрах, відомих під загальною назвою UDDI. Реєстр UDDI містить контактну та технічну інформацію про веб-сервіси та їх провайдерів, документи WSDL, а також дозволяє здійснювати пошук необхідного веб-сервісу за різними критеріями. Також за допомогою UDDI організації отримують можливість інтегрувати сторонні веб-сервіси в свої системи.

Структура UDDI складається з трьох основних компонентів: білі сторінки, жовті сторінки та зелені сторінки.

Білі сторінки містять контактну інформацію про постачальників веб-сервісу, яка містить назву компанії або опис служби. Скориставшись цією інформацією, можна знайти сервіс з наявною інформацією про нього.

Жовті сторінки містять інформацію про самі веб-сервіси та їх кодові номери, які класифікують їх діяльність. Компанія, яка надає кілька послуг, може містити кілька жовтих сторінок з описом кожної з цих послуг.

Зелені сторінки містять технічну інформацію про веб-сервіси, таку як адреса веб-сервісу, посилання на специфікації інтерфейсу, документи WSDL тощо.

Реєстр UDDI розміщується на виділеному сервері, вміст якого зберігається у вигляді декількох резервних копій у спеціальних вузлах UDDI – серверах, що забезпечують постійну підтримку специфікації UDDI, зареєстрованих в реєстрі UDDI поки він містить принаймні один запис. Примітно, що, як постачальник веб-сервісів для пошуку та індексації послуг, реєстр UDDI сам по собі є веб-сервісом.

Одним з найбільш серйозних недоліків SOA є великий розмір повідомлень на основі XML, що знижує продуктивність системи і споживає багато ресурсів інтернет-трафіку для обміну повідомленнями SOAP. В цьому плані SOA поступається таким програмним системам, як CORBA, RMI, DCOM.

Архітектура SOA заснована на принципах багаторазового використання IT-технологічних функціональних елементів, запобігання повторенню

функціональних можливостей різних програмних продуктів, уніфікації типових програмних процесів. Програмні компоненти, розподілені по мережі, можуть бути незалежними один від одного і слабо пов'язаними, а при необхідності можуть бути легко замінені іншими компонентами, що взаємодіють за аналогічним протоколом. Програма, заснована на сервісно-орієнтованій архітектурі, має елементний інтерфейс, який інкапсулює процес реалізації того чи іншого елемента з інших.

Однією з головних переваг сервісно-орієнтованої архітектури є відносна простота створення складних програмних комплексів шляхом об'єднання невеликих програмних компонентів. Завдяки цьому компанія більш ефективно визначає ресурси для розробки своєї програмної системи та знижує витрати за рахунок оптимізації процесу розробки. Також компанія може визначити рамки для використання платформ, мов розробки та інших інструментів, а масштабованість архітектури дозволяє ефективно розвивати систему в умовах нестабільної (зростаючої) кадрової бази.

Таким чином, найважливішими принципами сервісно-орієнтованої архітектури є: простота реалізації, кросплатформенність, розширюваність, можливість пошуку інших веб-сервісів та інтеграції їх у систему, незалежно від платформи та мови розробки.

1.2 Інструменти розробки веб-сервісів

Незважаючи на те, що існує безліч спеціалізованих програмних засобів, спрямованих на створення веб-сервісів, в основі будь-якої програмної розробки лежить певна мова програмування – система символів і команд, за допомогою яких записується комп'ютерна програма. Архітектура програми залежить від використовуваної мови, так як кожна мова має свої унікальні лексичні, синтаксичні та семантичні правила.

На сьогоднішній день налічується близько декількох тисяч мов програмування, кожна з яких має свою робочу структуру і спрямована на

вирішення певних завдань програмування. Кількість мов програмування зростає з року в рік, кожна нова мова створюється для розробки все більш складних програмних систем. Історія мов програмування дозволяє виділити п'ять поколінь мов: машинні (перше покоління), асемблери (друге покоління), мови високого рівня (третє покоління), об'єктно-орієнтовані мови (четверте покоління), мови штучного інтелекту (п'яте покоління).

Мови першого покоління є апаратно-специфічними, залежно від чого вони використовують двійкові або вісімкові команди у своєму синтаксисі. Складність таких мов визначається необхідністю знати не тільки мову програмування, але і архітектуру робочої машини.

Мови другого покоління називаються асемблерами. Однією з їхніх особливостей є те, що вони використовують свої мнемонічні назви для позначення команд замість двійкових або вісімкових форматів, типових для мов першого покоління, але вони все одно залежні від архітектури ЕВМ. Ці мови використовуються і сьогодні для розробки високопродуктивного програмного забезпечення.

Головною відмінністю мов третього покоління від попередніх є їх незалежність від архітектури машини. Для цих мов характерно широке використання інтерпретації програм, при якій написаний код перед виконанням перетворюється безпосередньо в машинні команди, які виконуються рядок за рядком під час виконання програми до її завершення або виявлення помилки в синтаксисі.

Мови четвертого покоління — це середовища розробки програмного забезпечення, які поєднують основні засоби розробки, до яких належать текстовий редактор, компілятор і налагоджувач (доступність інших інструментів залежить від конкретного середовища розробки). Головна перевага мов четвертого покоління полягає в тому, що вони дозволяють розробнику швидко перемикатися між основними інструментами розробки та уникати різних додаткових кроків, що дозволяє програмісту приділяти більше часу безпосередньо розробці, а

ефективність цього процесу значно зростає. Також ці мови відрізняються об'єктно-орієнтованим підходом до розробки програмного забезпечення, що означає, що вони не є універсальними, а орієнтовані на певну предметну область (веб-розробка, бази даних і т.д.).

Мови п'ятого покоління, як і мови четвертого покоління, використовуються в середовищах розробки, але тенденції нових мов спрямовані на максимізацію процесу програмування шляхом введення інструкцій в середовища розробки, щоб дати можливість користувачеві взаємодіяти з середовищем найбільш зручним способом, наприклад, за допомогою природної мови або графічної візуалізації.

Незважаючи на те, що існує близько 8 000 мов програмування, більшість з них використовуються в певних предметних областях або не використовуються взагалі, тому найпопулярніші мови, будучи найбільш універсальними (щодо певної предметної області) і технічно просунутими, становлять абсолютну меншість серед існуючих мов програмування. Згідно з індексом TIOBE, на сьогоднішній день серед користувачів програмістів найбільшою популярністю користуються такі мови.

Java – це типізована, об'єктно-орієнтована мова програмування, яка є однією з найпопулярніших мов завдяки своїй простоті, надійності та експлуатаційним характеристикам. Головною особливістю цієї мови є використання так званої віртуальної машини Java (JVM - Java Virtual Machine), яка виконує програму, попередньо скомпільовану з текстового формату в байт-код. Ця особливість дає можливість виконувати програми, написані на Java інших мов програмування, шляхом компіляції чужорідного програмного коду в байт-код, який потім буде виконуватися віртуальною машиною Java.

C - це стандартизована мова програмування, спочатку призначена для використання в операційній системі UNIX, але з тих пір була портована на інші операційні системи і стала найпопулярнішою мовою програмування для створення прикладного програмного забезпечення. Завдяки мінімізованій кількості ключових слів і відсутності класів, ця мова зручна у використанні, але не обмежена своєю

простотою. Під впливом цієї мови були розроблені такі мови, як Java, C++, PHP та ін.

C++ – типізована мова програмування загального призначення, спрямована на різні парадигми програмування: об'єктно-орієнтовану, узагальнену, процедурну, функціональну. Мова в основному використовується при розробці програмних додатків, драйверів для зовнішніх апаратних пристроїв, а також програмного забезпечення для серверів. Оскільки C++ є прямим продовженням мови C, синтаксично ці дві мови однакові. Головною відмінністю C++ від C є спрямованість перших на об'єктно-орієнтоване програмування, що має на увазі використання системи класів у процесі написання програмного коду. Серед інших, менш значущих нововведень, можна відзначити додаткові типи даних, віртуальні функції і т.д.

JavaScript — скриптова мова програмування, один із діалектів ECMAScript. Вміє працювати в об'єктно-орієнтованих, функціональних та імперативних парадигмах. Мова розроблялася як полегшена альтернатива Java, більш доступна для новачків, внаслідок чого мова має більш вузьке застосування, ніж Java. Найчастіше використовується як вбудована мова програмування, що використовується у веб-дизайні для створення інтерактивних елементів веб-сторінок, а також для реалізація деяких функцій браузера. Незважаючи на те, що мова була популярна протягом тривалого часу, зараз деякі його можливості застаріли, через що його поступово замінює альтернатива у вигляді п'ятої версії мови HTML.

Python — це високорівнева інтерпретована мова програмування, спрямована на різні парадигми програмування, включаючи об'єктно-орієнтоване програмування. Як і C, Python був розроблений в першу чергу для створення простого, але ефективного синтаксису, який забезпечує виразність програмного коду без шкоди для ефективності та продуктивності вихідної програми. Основна причина мови полягає в тому, що вона націлена на швидке та вільне написання програмного коду, свобода якого виражається в тому, що рішення тієї чи іншої

задачі може бути досягнуто безліччю різних способів, але така свобода відкриває широкий простір для помилок, які може бути допущені недосвідченим програмістом при написанні коду. Крім усього іншого, Python має широку стандартну бібліотеку функцій.

C# — сильно типізована, об'єктно-орієнтована мова програмування. Ця мова була створена Microsoft як основна мова розробки додатків на основі Microsoft .NET Framework. Під впливом C++ та Java, C# має схожий синтаксис, а також широку бібліотеку функцій, яка включає перевантаження операторів, ітератори, класи, атрибути, події, властивості, анонімні функції і т.д.

PHP – це скриптова мова програмування загального призначення. Вона масово використовується для створення веб-додатків, хоча можливе використання мови при розробці додатків GUI (Graphical User Interface) або CLI (Command Line Interface). У середовищі веб-програмування це найпопулярніша мова завдяки своїй простоті, продуктивності та функціональності. Функціональність мови може бути розширена завдяки численним плагінам, доступним у мережі Інтернет.

SQL — декларативна мова програмування для створення, редагування та управління базами даних у реляційних базах даних. Це найпопулярніша мова для виконання різних операцій над базами даних, але більшість найпопулярніших реалізацій цієї мови відрізняються настільки, що практично ніколи не вдається перенести код з однієї СУБД в іншу без внесення в неї істотних змін. Причиною цього є об’ємний і складний стандарт, який не враховує деякі важливі деталі в різних сферах реалізації мови. Крім усього іншого, розробники SQL відійшли від його початкової концепції простої мови управління базами даних, зробивши його інструментом розробника, а не кінцевого користувача, зробивши його більш складним, ніж передбачалося, але все ще найпопулярнішою мовою управління базами даних.

Perl — це інтерпретована, динамічна мова програмування. Головною особливістю мови є широкий спектр можливостей для роботи з текстовою інформацією, а також регулярними виразами. Найчастіше він використовується в

мережевому середовищі для автоматизації деяких повсякденних процесів, хоча мова використовується і в багатьох інших сферах: системному адмініструванні, веб-розробці, біоінформатиці тощо.

Незважаючи на те, що перераховані вище мови програмування розроблялися з середини 20-го століття до кінця 1990-х років, вони все ще популярні і широко використовуються багатьма програмістами та IT-компаніями. Однією з причин цього є те, що всі ці мови програмування все ще підтримуються їхніми розробниками, регулярно отримуючи оновлення, які привносять нові функції в мову та виправлення помилок. Мови підтримуються і сторонніми користувачами, які розробляють для них різні модулі, що розширюють можливості середовищ програмування. Однак головна причина їхньої популярності полягає в тому, що їхній функціонал повністю задовольняє потреби IT-спільноти. У міру зміни потреб компанії-розробники вносять необхідні зміни в функціонал своїх розробок, і якщо з якихось причин вони не в змозі задовольнити поточні потреби своїх клієнтів, то їх напрацювання втратять свою актуальність, а їх місце займуть розробки іншої компанії.

Найскладнішим способом розробки веб-сервісу є його повністю самостійне створення, починаючи з написання та налагодження програмного коду і закінчуючи його інтеграцією в мережеве середовище. У зв'язку з тим, що цей спосіб вимагає дуже великої кількості часу і гранично глибоких знань в області програмування, веб-програмування і мережевого адміністрування, існує метод, при якому розробка веб-сервісу доручається сторонньому розробнику, яким може бути освідчений програміст-фрілансер, або компанії, що займається розробкою програмного забезпечення на замовлення. Найпростішим способом створення робочого веб-сервісу є використання спеціального програмного комплексу, призначеного для створення та управління контентом веб-сервісу, який скорочено позначається як CMS.

CMS (Content management system, система управления контентом) – це інформаційна система, яка надає інструменти для спрощення створення та

управління контентом веб-сервісу. CMS є програмним движком, призначеним для створення веб-сервісу, тому його розробка зводиться до використання готового набору інструментів, що дозволяє управляти текстовим і графічним контентом створюваного веб-сервісу. Найскладнішим в реалізації даного способу створення веб-сервісу є освоєння інструментів системи управління контентом, але ця складність не йде ні в яке порівняння зі складністю самостійного створення веб-сервісу.

З метою економії часу на розробку та реалізацію зовнішнього дизайну веб-сервісу можна скористатися готовим шаблоном з інтернету – спеціальним шаблоном для оформлення веб-сервісу, який дозволяє пропустити графічний етап його розробки. Найчастіше CMS дозволяють зберігати в собі кілька шаблонів і при необхідності замінювати ними поточний шаблон.

Важливою частиною будь-якої CMS є можливість підключення зовнішніх програмних модулів, які розширюють можливості веб-сервісу. Наприклад, до таких розширень можна віднести чат користувача на головній сторінці веб-сервісу або спеціальну транзакційну систему, яка виводить кошти з особистого банківського рахунку користувача під час торгової операції.

Існує багато різних CMS, розроблених окремими програмістами та цілими компаніями-розробниками програмного забезпечення. Кількість існуючих CMS на сьогоднішній день налічує понад десятки тисяч зразків, кожен з яких можна класифікувати за трьома основними принципами:

- платний/безкоштовний;
- спосіб управління контентом сайту;
- За типом керованих даних.

Платна/безкоштовна CMS – це класифікація, яка визначає доступність та спосіб монетизації конкретної CMS. Платне використання програмних розробок, перш за все, має на увазі сплату грошової винагороди, спосіб якої визначається фінансовою політикою компанії-розробника. Програмний продукт може бути придбаний за разову плату або на заздалегідь визначений термін платної підписки,

після чого подальше використання програмного продукту стає можливим тільки після чергової оплати абонентської плати. Також існує політика умовно-безкоштовного програмного продукту, при якій користувачеві доступні тільки найосновніші або демонстраційні можливості програми, а можливість використання додаткових інструментів стає доступною тільки після сплати певної плати. До найпопулярніших платних CMS відносяться: 1С-Бітрікс, Umi, NetCat. При використанні безкоштовної CMS користувач не зобов'язаний платити грошову винагороду за володіння програмою, але все ж є можливість добровільно зробити грошову пожертву на подальший розвиток програмного продукту. Часто розробники вільних програмних продуктів публікують вихідний код своїх розробок з метою забезпечення більш ефективного зворотного зв'язку від користувачів-програмістів, здатних виявити помилки або помилки в опублікованому коді або неефективні методи, допущені при розробці. До найпопулярніших безкоштовних CMS відносяться: WordPress, Joomla, Drupal.

За способом управління контентом сайту – класифікація, в рамках якої визначається метод генерації сторінок, що використовується системою:

- CMS, які генерують сторінки за запитом;
- CMS, які генерують сторінки при редагуванні;
- CMS змішаного типу.

Метод генерації сторінок за запитом має на увазі створення нової сторінки після кожного запиту від користувача. У простому поданні навігація веб-сервісом виглядає як навігація між HTML-сторінками за посиланнями, яка супроводжується генерацією сторінок заново з кожним кліком, що збільшує навантаження на сервер.

Метод генерації сторінок при редагуванні відрізняється від попереднього способом тим, що немає необхідності генерувати сторінки кожен раз при надходженні нових запитів, так як використовується система статичних веб-сторінок, що дозволяє переміщатися по веб-сервісу без створення нових сторінок. Нова веб-сторінка генерується тільки в тому випадку, якщо змінюється матеріал, що наповнює веб-сервіс, в рамках якого відбувається копії блока, що редагується.

CMS змішаного типу поєднують попередні два способи генерації сторінок однак мають різні варіанти реалізації. Перший варіант заснований на використанні спеціального сховища інформації, званого кешем. Копії запитуваних HTML-сторінок відправляються в кеш, а потім CMS завантажує сторінку безпосередньо з кешу, якщо сторінка запитується повторно, усуваючи необхідність генерувати ту саму веб-сторінку знову. Другий варіант - збереження інформації у вигляді блоків, з яких збирається сторінка за бажанням користувача.

За типом керованих даних – оскільки існують CMS, що керують змістом не тільки веб-сервісу, а й інших структур програмного забезпечення, існує класифікація CMS за типами керованих даних. Їх можна розділити на:

- системи управління контентом – CMS, які дозволяють здійснення керування вмістом веб-сайту – редагувати існуючий контент або додавати новий. CMS цього типу призначені для розширення можливостей сайтів, що знаходяться під їх контролем, за допомогою різноманітних плагінів і модулів, вони відрізняються один від одного в основному складністю і набором можливостей;
- системи управління корпоративним контентом – це тип систем, спрямованих на управління контентом веб-сайтів великих компаній, а також впровадження та підтримку різних бізнес-процесів;
- системи документообігу - це системи, які мають деяку технічну схожість з CMS. Найчастіше вони використовуються як доповнення до систем управління контентом для організації управління текстовими файлами всередині CMS, в якості яких можуть виступати різні угоди і документація;
- системи управління цифровими правами – системи, призначені для збереження, редагування та управління інформацією про право власності на частини Інтернет-контенту, якою може бути зображення, аудіо- чи відеозапис, програмний продукт тощо.

В рамках роботи, присвяченої розробці веб-сервісів, виділяються найпопулярніші CMS, які мають певні можливості для вирішення цього завдання: WordPress, Joomla, Drupal.

WordPress – одна з найпопулярніших систем для розробки сайтів. Однією з причин такої популярності є те, що ця CMS безкоштовна, її вихідний код знаходиться у відкритому доступі, а сама система проста у використанні. Для цієї CMS створено безліч різних плагінів і модулів, а також шаблонів дизайну, в які можна вносити свої зміни. Розроблювалась у першому чергу як система створення сайтів-блогів, де є можливість створювати авторські тематичні пости, що складаються з тексту, зображень, аудіо та відео файлів, залишаючи відвідувачам можливість коментувати ці пости. Хоча існують модулі, які можна використовувати для створення сайтів будь-яких цілей, їх надмірне використання негативно позначається на швидкості роботи сайту. Ще одним слабким місцем цієї CMS є відносно високе навантаження на сервер, яке інтенсивно зростає при високій відвідуваності сайту, саме тому виникає потреба у встановленні спеціальних розширень, спрямованих на оптимізацію роботи CMS, а важливість проблеми вибору якісного хостингу значно зростає.

Joomla - це безкоштовна CMS, одна з трьох найпопулярніших безкоштовних систем для створення сайтів. Як і Wordpress, має відкритий вихідний код, а це означає, що в інтернеті існує широкий спектр модулів і шаблонів дизайну, створених сторонніми розробниками. Однією з головних особливостей даної CMS є широка універсальність інструментів системи, що досягається за рахунок установки різноманітних модулів, при цьому не діючи на шкоду простоті використання CMS і її швидкості працювати. Сайти різного призначення і розмірів можуть бути реалізовані на Joomla за допомогою модулів, але система також має широкий набір базових можливостей, що дозволяють ефективно працювати з контентом сайту, таких як різні редактори, контент-менеджери, що дозволяють не тільки редагувати його, але і відстежувати реакцію на нього з боку користувачів і т.д. Розробка на його основі зводиться до використання готових рішень, які не дозволяють створити сайт з унікальними можливостями.

Drupal — ще одна безкоштовна CMS. Як і описані вище системи, він набуває своєї функціональності за рахунок додавання сторонніх модулів, однак ця CMS

відрізняється підвищеною складністю роботи всередині неї, через що недосвідчений користувач, перш ніж створити власний сайт на базі Drupal, змушений витрачати багато часу на вивчення системи. Завдяки високому порогу використання, вона є професійним інструментом розробки, за допомогою якого досягається універсальність і гнучкість процесу розробки сайту.

Вищезгадані системи управління контентом не мають принципових відмінностей один від одного – кожна з них може бути спрямована на вирішення однієї і тієї ж проблеми, але безліч дрібних відмінностей в фундаменті кожної з CMS в сукупності формують причини, за якими одна CMS краще підходить для створення інтернет-магазину, ніж інша, яка, в свою чергу, краще справляється із завданням розробки веб-блогу. Звідси випливає, що вибір CMS здійснюється, перш за все, виходячи з поставлених перед розробником завдань, а потім критерії вибору доповнюються додатковими побажаннями замовника і особистими уподобаннями розробника.

Термін «програмне забезпечення» вперше був використаний Джоном В. Тьюкі в 1958 році, і в наступні десятиліття прикладне програмне забезпечення було розроблено і поступово використовується в багатьох областях. Розвиток програмного забезпечення спричинив значні зміни у способі життя людини та бізнес-моделях. У повсякденному житті люди мають справу зі складними фінансовими формами за допомогою автоматизованого додатку. Вони також діляться відео, фотографіями та статтями через Інтернет із сім'ями та друзями. З іншого боку, в бізнесі програмне забезпечення може допомогти компаніям керувати своїми особистими даними, робити фінансовий прогноз і моделювати процес або модель. Веб-сервіс набуває все більшого поширення в сучасних програмних додатках. Багато веб-сервісів були опубліковані компаніями Amazon, Google і Microsoft. Крім того, численні бізнес-додатки поставляються у вигляді веб-сервісів, таких як планування ресурсів підприємства (ERP) і управління взаємовідносинами з клієнтами (CRM). Веб-сервіс є універсальним за своєю конструкцією, і доступ до нього можна отримати через веб-інтерфейс клієнта. Крім

того, одним веб-сервісом можуть користуватися кілька клієнтів, що знижує вартість.

Поряд зі зростаючими вимогами до використання програмного забезпечення для роботи з моделлю моделювання та фізичним предиктором у промисловій сфері, професіонали вважають за краще вибирати ефективний і дієвий спосіб досягнення поставлених цілей, використовуючи професійні програмні додатки. Висока вартість створення та підтримки програмно-апаратних інфраструктур для надання послуг підприємствам призвела до помітної тенденції до використання сторонніх сервісів, які надають клієнтам обчислювальну потужність та простір для зберігання даних.

Software-as-a-Service (SaaS), модель доставки програмного забезпечення, в якій користувачі можуть отримати доступ до програмного забезпечення та пов'язаних з ним даних за допомогою тонкого клієнта через веб-браузер, пропонує важливі переваги, такі як нижча вартість, віддалений доступ, швидке розгортання та використання хмарних ресурсів. Клієнти можуть підписатися на короткострокову ліцензію, а не на безстрокову ліцензію на використання програмного додатку, щоб заощадити багато початкових витрат, які можуть бути використані для покращення бізнесу. Вартість послуги також залежить від «параметрів використання», таких як групи користувачів і період використання програми. Середовище програмного забезпечення налаштовується та підтримується постачальником SaaS таким чином, щоб розгортання було дуже швидким. Haiqi Liang et al. пропонує підхід до налаштування веб-сервісу за допомогою співпраці між постачальником послуг і споживачем. Вони заявляють: «Оскільки різні споживачі зазвичай мають особливі вимоги до бізнес-додатків з точки зору інтерфейсу користувача, бізнес-процесів і даних, для SaaS-сервісів дуже важливо надавати споживачам послуг легкі можливості персоналізації та налаштування відповідно до їхніх власних вимог». Необхідно побудувати фреймворк для розширення веб-сервісу відповідно до багатьох вимог користувача.

Це призвело до дослідження впровадження SaaS, яке може задовольнити як промислові, так і освітні потреби.

Моделювання, за допомогою симуляційного програмного забезпечення, — це наука про створення статистично точних моделей для представлення поведінки реальних систем з метою піддати їх прогнозним експериментам. Експерименти або сценарії можуть дозволити відповісти на запитання «а що, якщо?» без ризику чи порушення для реальної системи життя. У промисловості компанії використовують програмне забезпечення для моделювання, яке імітує фізичні явища за допомогою набору математичних формул, для підвищення якості продукції та зниження собівартості. Особливо для компаній, що працюють у сегментах пластмас, гуми та передових матеріалів, необхідно знайти спосіб змоделювати процес або передбачити результат, щоб підвищити продуктивність і заощадити початкові витрати. Сервіс програмного забезпечення для моделювання є хорошим рішенням для бізнесу. Наприклад, General Motors використовує паралельні обчислення для моделювання краш-тестів автомобілів. Він стверджує, що може скоротити кількість тестів повнорозмірних краш-автомобілів більш ніж на 85 відсотків при економії коштів у розмірі 500 000 доларів США за тест. Аналогічним чином, суперкомп'ютерне моделювання дозволило знизити витрати, які компанія Goodyear витрачає на прототипи фізичних шин, з 40 до 15 відсотків, одночасно значно скоротивши час на випуск нової шини на ринок. У сфері освіти викладачі, студенти та дослідники повинні отримати доступ до окремої машини, на якій може розміщуватися програмне забезпечення для моделювання, для проведення експериментів, досліджень або розробки продуктів. Як промислові, так і академічні користувачі є експертами в цій галузі, а не комп'ютерними експертами. Їм потрібна система, яка проста у використанні, але дає їм доступ до широкого спектру програмного забезпечення. Крім того, необхідно провести навчальні курси для користувачів симуляційного програмного забезпечення, щоб навчитися користуватися програмним забезпеченням.

Таблиця 1.1 - Промислове та академічне використання

	Промислове використання	Академічне використання
Моделювання	Тестування розробки продукту	Дослідження проекту класу
Візуалізація	Візуалізація та аналіз отриманих результатів	
Навчання	Програмне забезпечення, пов'язане з симуляцією	

І бізнесу, і освіті потрібен такий процес, який займає багато часу та збиває з пантелику користувачів, оскільки вони повинні мати різні сертифікати автентифікації для кожного комп'ютера, на якому розміщено службу. Зокрема, багато промислових процесів піддаються моделюванню, і моделювання цих процесів дозволяє оптимізувати час, вартість, енергію, якість і деякі інші параметри процесу. Однак програмне забезпечення для моделювання дуже дороге, складне у використанні та має поставлятися в комплекті з локальним допоміжним обладнанням. Вартість досить висока, і багато потенційних користувачів не можуть виправдати сплату такої високої початкової вартості за таку невизначену віддачу, тому користувачі хотіли б використовувати відносно дешевий спосіб задоволення потреб симуляції. Крім того, користувачі можуть мати додаткові функціональні вимоги або потребувати інших видів програмних послуг, наприклад, бажання візуалізувати результати симуляції після процесу моделювання. Має існувати спосіб об'єднання кількох програмних служб разом, щоб користувачі могли їх вибрати.

Мета цієї роботи - розробити веб-структуру заходів для надання безлічі програмних послуг з метою задоволення вимог користувачів.

Програмне забезпечення як послуга (SaaS) набирає обертів зі збільшенням кількості вендорів та нещодавніми успіхами провідних гравців на ринку. Для вирішення вищезазначених проблем запропоновано Simulation as a Service (SMaaS) як розширення SaaS, воно забезпечує поєднання інструментів загального призначення та спеціального використання на різних платформах.

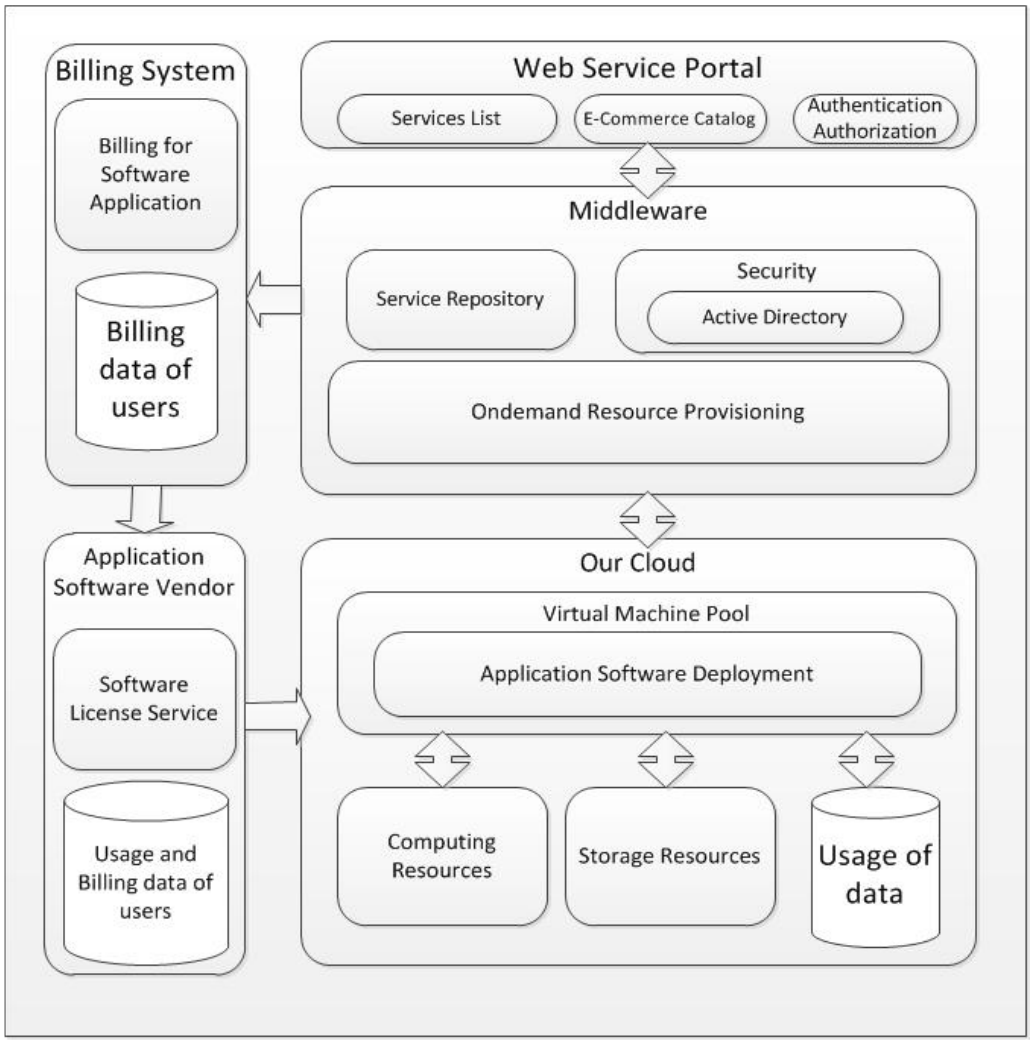


Рисунок 1.2 - Архітектура SMaaS для хмарних обчислень

SMaaS надає інтегрований набір інструментів для наскрізної підтримки симуляції на кількох платформах і побудований з використанням переважно готових компонентів Common-Off-the-Shelf (COTS). До його характеристик можна віднести:

- довготривалі симуляції вимагають як інтерактивних сеансів, так і пакетних завдань;
- Потрібні дорогі сторонні розв'язувачі;
- Потрібні великі набори даних;
- Потрібна обробка високопродуктивних обчислень (HPC)

У системі SMaaS може бути задіяно багато різних груп. Наприклад, є викладачі, які бажають використовувати систему для навчання, студенти, які навчаться користуватися інструментами, розміщеними в системі, постачальники програмного забезпечення, які надають інструменти, і постачальник ресурсів, який повинен забезпечувати та обслуговувати систему. Кожна група має різні і часто суперечливі цілі, і вони повинні мати можливість досягати цих цілей гнучким і безпечним способом. SMaaS прагне обслуговувати ці різноманітні спільноти користувачів, надаючи масштабовану, безпечну та розширювану архітектуру.

Ми маємо намір розробити зручну веб-платформу для надання програмних послуг. Платформа дозволить користувачам купувати короткострокові ліцензії на програмне забезпечення відповідно до їхніх фактичних вимог, а не безстрокову ліцензію, яка є досить високою за вартістю. Каталог електронної комерції буде використовуватися для налаштування вимог користувачів, а це означає, що ми надамо купу програмних послуг. Користувачі зможуть вибрати послуги, якими вони хочуть скористатися. Крім того, користувачі зможуть використовувати віртуальну машину, яка налаштовується та обслуговується постачальником послуг, для доступу до програмного забезпечення, тому користувачі

Не потрібно буде думати про те, як налаштувати та обслуговувати обладнання. Ми також будемо розробляти функції загальної точки входу та єдиного входу для входу користувачів, щоб розрізнені підсистеми могли бути пов'язані разом у єдине ціле, яке матиме сенс для користувача. Після того, як користувач увійде в систему, він зможе використовувати будь-яку частину платформи, для якої він авторизований, без повторної автентифікації. Всі ці функції будуть інтегровані в один веб-фреймворк.

Програмне забезпечення веб-сервісів побудовано на стандартних протоколах і технологіях, таких як HTTP, XML, SOAP, WSDL і UDDI. Web 2.0 – це революція, яка переходить від простого протоколу доступу до об'єктів (SOAP) до зв'язку на основі Representational State Transfer (REST) (таблиця 2.1)

Таблиця 2.1 - Стек технологій веб-сервісів

Сервіс публікації та виявлення послуг (UDDI)
Опис послуги (WSDL)
XML – Повідомлення на основі XML
Служба загальнодоступності та виявлення (TCP/IP, HTTP, FTP)

Люди можуть використовувати різні інтерфейси прикладного програмування (API) для інтеграції кількох веб-сервісів як нового веб-додатку. «Веб-сервіси визначають нову парадигму в сучасній спільній розробці додатків на підприємствах і охоплюють системи, мережі та організації». Наведена нижче діаграма є повністю сервісно-орієнтованою моделлю (рис. 2.1).

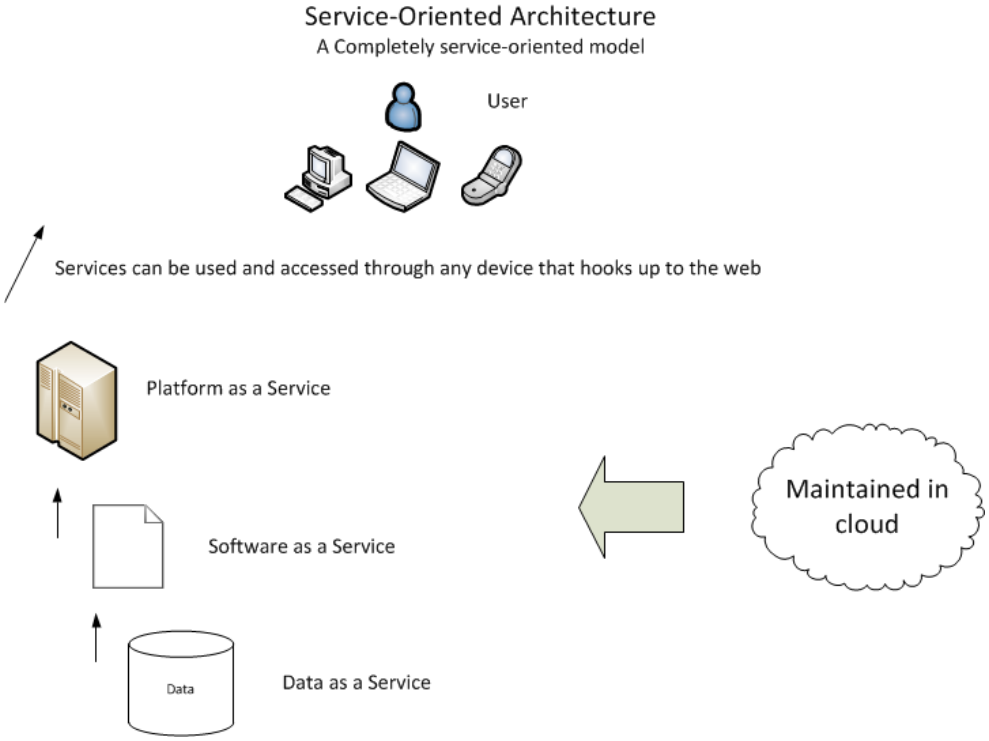


Рисунок 2.1 - Моделі веб-сервісів в сервісно-орієнтованій архітектурі

Найпоширенішим розумінням хмарних обчислень є сервісно-орієнтоване на апаратне забезпечення, від апаратного до прикладного рівня. На наведеній вище діаграмі платформа як послуга (PaaS), програмне забезпечення як послуга (SaaS) і дані як послуга (DaaS) є конкретними реалізаціями хмарних обчислень. Хмарні

обчислення привертають значну увагу, оскільки вони можуть забезпечити швидку доставку обчислювальних ресурсів як утиліт у динамічний, масштабований і віртуалізований спосіб. Існує багато хмарних продуктів як у промисловості, так і в дослідницьких спільнотах, таких як Amazon Elastic Compute Cloud (EC2), Google App Engine (GAE), Microsoft Azure, Salesforce.com's Force.com, VMware, Eucalyptus і Citrix. Хоча кожен з них має свої спеціальності, вони підвищують ефективність IT завдяки своїй масштабованості, величезним обчислювальним ресурсам та підтримці інфраструктури для кількох орендарів. Люди можуть використовувати свою хмарну платформу для виконання власних завдань, і немає необхідності розглядати обчислювальні ресурси або розгортати робочі середовища, технічне обслуговування та планування завдань.

1.3 Веб-сервіс для спільної роботи

Деякі дослідження підтверджують співпрацю веб-сервісів. Zhengxiong Hou et запропонував веб-структуру для надання програмного забезпечення як послуги, заснованої на високопродуктивних обчислювальних ресурсах. Він надає користувачам веб-портал для доступу до прикладного програмного забезпечення в режимі оплати за використання. Це вирішує проблеми традиційного програмного забезпечення, яке не є сервісно-орієнтованим і обмежене ліцензіями на програмне забезпечення. Ця функція електронної комерції схожа на представлений в роботі інтегрований фреймворк, який дозволяє користувачам вибирати бажаний тип ліцензії на програмне забезпечення за різною вартістю. Irfan Awan et al. пропонують нову модель, засновану на веб-сервісах, яка забезпечує різні пріоритети для спільних додатків. Більшість сучасних підходів однаково трактують всю діяльність спільних додатків; однак вони запропонували трирівневу архітектуру, яка обробляє різні обробки для спільних додатків. Представлений в роботі інтегрований фреймворк не реалізує функцію «різної пріоритетності для різних сервісів», тому що всі послуги, які ми надаємо в рамках фреймворку, є

незалежними. Немає необхідності впроваджувати стратегію різних методів лікування. Спільна робота веб-сервісів не може приховати проблеми інтеграції, з якими стикається корпоративна інфраструктура. Обхід брандмауера/NAT і проблеми з безпекою часто є вузьким місцем, оскільки підприємствам може бути незручно запускати критично важливі програми за межами корпоративного брандмауера. У нашому фреймворку, наші сервіси побудовані на системі Linux як кластері, а не розподілених системах. Однак питання співпраці із зовнішніми ресурсами буде враховуватися в майбутньому, тому ця пов'язана робота варта уваги. Feng Liu et al. представляють рішення для інтеграції веб-сервісів на основі проксі-брандмауера/NAT. Це дозволяє веб-сервісу інтегруватися з локальною програмою без конфігурації брандмауера. Важливо керувати платформою веб-сервісу, оскільки все більше і більше таких платформ широко використовуються. Jianzhong Li et al. представляють компонентно-орієнтований підхід для реалізації інтегрованої структури управління платформами веб-сервісів. Фреймворк управління пропонує єдину точку входу, яка зручна для адміністратора для доступу до інструментів управління платформами веб-сервісів. Функція «єдиної точки входу» схожа на наш фреймворк, який також надає користувачам єдину точку входу для входу, щоб користувачі могли безперешкодно мігрувати між різними службами. Крім того, наш фреймворк побудований з різних компонентів, які відповідають за різні функції.

1.4 Конфігурація та кастомізація веб-сервісів

Haiqi Liang et al. впроваджують підхід до налаштування веб-сервісу шляхом співпраці між постачальником послуг і споживачем програмним способом, який полягає в налаштуванні веб-сервісу за допомогою WS-Policy. WS-Policy забезпечує гнучку і розширювану граматику для вираження можливостей, вимог і загальних характеристик сутностей в системі на основі веб-сервісу XML. Він визначає структуру та модель для вираження цих властивостей як політики. На противагу

цьому, в рамках нашої інтегрованої структури, користувачі можуть вибирати бажані послуги у власних каталогах на веб-сторінці, яка є доброзичливою до користувачів. Незважаючи на те, що доступ до SaaS-додатку може отримати велика кількість клієнтів з можливостями мультиоренди, багато клієнтів все одно запитують варіанти функцій відповідно до своїх унікальних бізнес-потреб. Wei Sun et al. пропонують модель компетенцій та методологічну структуру, щоб допомогти постачальникам SaaS планувати та оцінювати можливості та стратегії конфігурації та кастомізації послуг. Модель компетенцій описується діапазоном: від рівнів «Початковий», «Обізнаний» і «Здібний» до «Зрілий» і «Світового класу». Крім того, різні рівні можуть бути використані для визначення конфігурації та налаштування за допомогою порівняльного аналізу з рівнем компетенції ринку. Ця пов'язана робота є корисною для нашого фреймворку, оскільки у нас є різні типи користувачів, починаючи від тих, хто працює в академічних колах і промисловості, щоб використовувати нашу високопродуктивну обчислювальну службу. Вони мають різні вимоги до програмних послуг та обчислювальних ресурсів, тому така модель оцінки корисна, коли ми впроваджуємо наш фреймворк.

1.5 Впровадження SaaS

Jyoti Namjoshi et al. представляють хмарний додаток для бронювання подорожей, заснований на сервіс-орієнтованій архітектурі. Їх рішення забезпечує ефективний процес бронювання, простий у використанні доступ в будь-який час з декількох місць і з гетерогенних технологічних середовищ. Це типова сервісно-орієнтована програма; однак його функціональність не може задовольнити зростаючі вимоги клієнтів. Наш фреймворк має масштабованість, що полегшує нам розширення наших послуг і дозволяє нашим користувачам налаштовувати свої каталоги. Користувачам SaaS потрібно лише придбати підписку у постачальника послуг. Це контрастує з традиційним способом, коли користувачі купують безстрокову або довгострокову ліцензію. В. Чоудхарі показує, що ця властивість

моделі ліцензування SaaS призводить до більшої інвестиції та вищої якості програмного забезпечення. Ми впроваджуємо цю властивість у нашу інтегровану структуру, і користувачі отримують вигоду від її простої у використанні та економії коштів. Для того, щоб зробити програмне забезпечення для моделювання доступним для дослідників та експертів у всіх галузях Інтернету, а не лише на одній машині, Song Guo et al. надає специфікацію симуляційного програмного забезпечення як сервісу та експерименту з сервісно-орієнтованою архітектурою для підтримки автоматичного розгортання послуг моделювання. На противагу цьому, ми попередньо налаштували послуги, що надаються в нашому фреймворку, і немає специфікації для різних програмних сервісів.

1.6 Оцінка веб-порталу

Незважаючи на те, що все більше і більше веб-порталів розробляються для задоволення різних потреб користувачів, користувачі виявляють бажання мати один сайт, який може задовольнити кілька вимог користувачів, а не використовувати різноманітні сайти для конкретних вимог. Mario Christ et al. виміряли використання веб-порталом окремих функцій у проекті HomeNet і розробили демографічні профілі груп з різним рівнем використання порталу. У нашому проекті ми можемо використовувати такого роду методологію для вимірювання використання програмних послуг моделювання, що надаються Polymer Portal для різних типів користувачів, таких як академічні та промислові користувачі, щоб зробити раціональний розподіл обчислювальних ресурсів і ресурсів віртуальних машин. Saeed Nourizadeh Azar et al. пропонують кілька критеріїв для оцінки веб-порталів на основі їх категорій (загальні портали, B2B-портали, корпоративні портали, урядові портали тощо) також перелічує бажані характеристики порталу, а саме: простота у використанні, інтуїтивно зрозуміла класифікація та пошук, спільний обмін інформацією, універсальне підключення до

інформаційних ресурсів, динамічний доступ до інформації, ресурси, інтелектуальна маршрутизація тощо. Ми плануємо використовувати ці критерії для оцінки функцій Polymer Portal and Manifold Flow Predictor, таких як кастомізація та персоналізація каталогу електронної комерції, гнучке надання дозволів на аутентифікацію та авторизацію, а також серверна архітектура для високопродуктивних обчислювальних ресурсів.

2 ПРОЕКТУВАННЯ ВЕБ-ПЛАТФОРМ ДЛЯ ВИВЧЕННЯ ОНЛАЙН КУРСІВ

2.1 Загальні дані про веб-платформи

Сьогодні Інтернет є невід’ємною складовою повсякденного життя. Створення цієї всесвітньої павутини започаткувало рух до зміни життя людей та відкрило велику кількість сучасних можливостей. Після створення Інтернет мав змогу об’єднати тільки два комп’ютери та передавати між ними не дуже велику кількість інформації. На даний момент мережа переповнена величезним обсягом різноманітної інформації, яка доступна усім у кого є Інтернет.

Під час перших показників розвитку Інтернет містив у собі тільки веб- сайти, які отримували та відображали документи від сервера до браузера. Онлайн-сервіси взагалі не мали функціоналу, для кожної людини була однакова інформація. На даний момент існує величезна кількість веб-ресурсів з хорошим функціоналом, який містить у собі автентифікацію, реєстрацію, різний пошук матеріалів і багато іншого.

Веб-платформи в більшості випадків володіють сервером, який зберігає певну інформацію, та користувацької частини для відтворення контенту на сайті.

Велика кількість веб-додатків є звичайними з односторінковими новинами або складними, як масивні онлайн ігри.

Плюси користування веб-платформами:

- Звичайний доступ, який може отримати будь-який користувач, що має комп'ютер та Інтернет.
- Для користування вам потрібен тільки браузер, який з легкістю встановлюється на ваш комп’ютер.
- На даний момент Інтернет має велику кількість спеціалізованих користувачів.

- Безпека мережі на більшості сайтів.

2.2 Огляд архітектурного шаблону програмного забезпечення, що використовується у веб-програмі

Архітектура програмного забезпечення описує основні компоненти системи, взаємозв'язок між ними та спосіб взаємодії між ними. Архітектура програми не має інформації, яка не впливає на певний компонент. Тому архітектура програмного забезпечення є абстракцією системи насамперед. У всіх сучасних системах компоненти взаємодіють між собою за допомогою інтерфейсів, які ділять інформацію про компонент на загальнодоступну інформацію та інформацію, відому лише в межах компонента. Архітектура програмного забезпечення відповідає за розкриття інформації і не описує детальної інформації про внутрішню реалізацію компонентів. Однак, крім інтерфейсів, абстракція архітектури також дозволяє зрозуміти компоненти системи, як організувати , і як переглянути систему. Вони взаємодіють та їх характеристики. Ця абстракція важлива для спрощення роботи із використанням складних систем. Подібним чином, однією з цілей архітектури є визначення вимог, що впливають на структуру програмного забезпечення.

Шаблон архітектурного програмного забезпечення описує основну структуру складної системи. Вони забезпечують набір заздалегідь визначених підсистем, визначають їх унікальні обов'язки та включають правила та принципи взаємовідносин між ними. Шаблон програмного забезпечення для архітектури описує структуру компонентів на макрорівні всього програмного рішення. Шаблони архітектури - це загальні рішення загальних проблем архітектури програмного забезпечення, які можна застосовувати в конкретному контексті. Незліченна кількість програм може реалізувати один і той же шаблон із подібними характеристиками. Використання архітектурних шаблонів дозволяє швидко знайти рішення загальних проблем у розробці програми.

Оглянемо найбільш популяризовані архітектурні методи.

2.2.1 Загальна архітектура

Загальна архітектура - традиційна уніфікована модель для проектування програмного забезпечення. Монолітна, в даному контексті, означає складена все в одному шматку. Згідно з Кембриджським словником, прикметник загальний також означає як занадто великий, так і неможливий для зміни .

Загальне програмне забезпечення розроблено для автономності; компоненти програми взаємопов'язані та взаємозалежні, а не вільно пов'язані, як це відбувається з модульними програмами. У тісно пов'язаній архітектурі кожен компонент та пов'язані з ним компоненти повинні бути присутніми для виконання або компіляції коду.

Крім того, якщо будь-який компонент програми повинен бути оновлений, вся програма повинна бути переписана, тоді як у модульній програмі будь-який окремий модуль (наприклад, мікросервіс) може бути змінений, не впливаючи на інші частини програми. Модульні архітектури зменшують ризик того, що зміна, внесена в один елемент, створить непередбачувані зміни в інших елементах, оскільки модулі відносно незалежні. Модульні програми також піддаються ітераційним процесам легше, ніж загальні програми.

Однак є користь і для загальних архітектур. Загальні програми, як правило, мають кращу пропускну здатність, ніж модульні підходи, такі як архітектура мікросервісу (MSA), і їх можна простіше перевірити та налагодити, оскільки із меншою кількістю елементів стає менше змінних, що вступають у гру.

Приклад схеми загальної архітектури можна побачити на рис. 2.1.

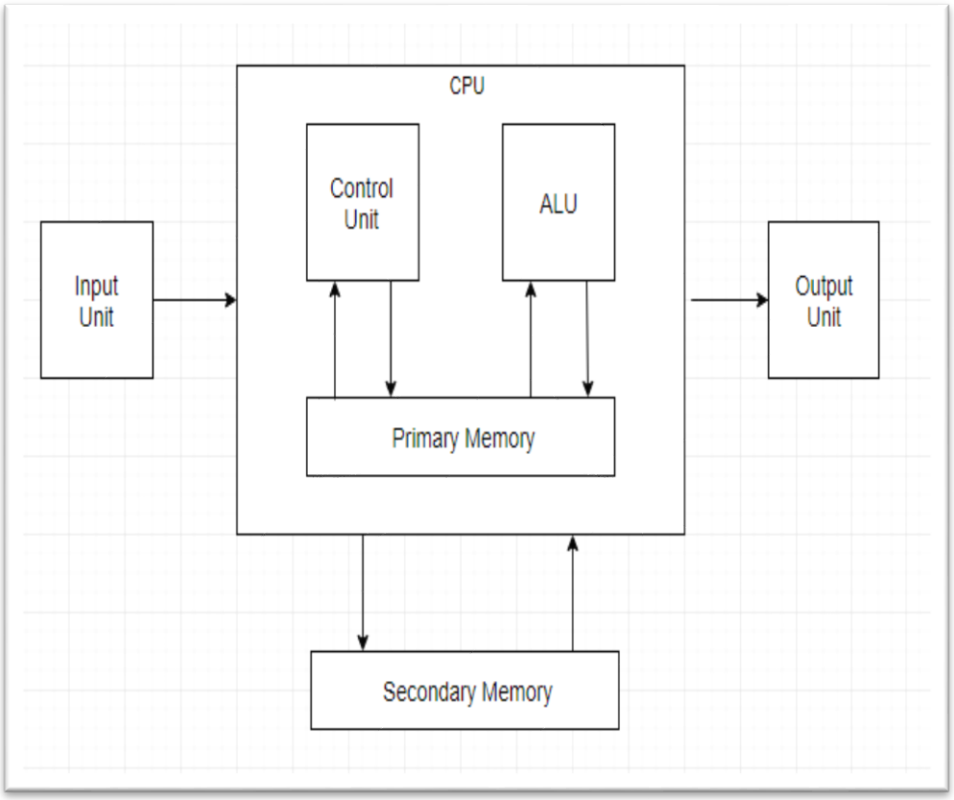


Рисунок 2.1 - Схема загальної архітектури

2.2.2 Дворівнева клієнт-серверна архітектура

У дворівневій архітектурі клієнт знаходиться на першому рівні. Сервер баз даних та сервер веб-додатків розміщені на одному сервері, що є другим рівнем. Цей другий рівень обслуговує дані та виконує ділову логіку для веб- програми. Організації, які віддають перевагу цій архітектурі, зазвичай воліють консолідувати свої можливості програм та можливості сервера баз даних на одному рівні. Другий рівень відповідає за забезпечення доступності, масштабованості та характеристик продуктивності веб-середовища організації [2].

Приклад можемо бачити на рис. 2.2.

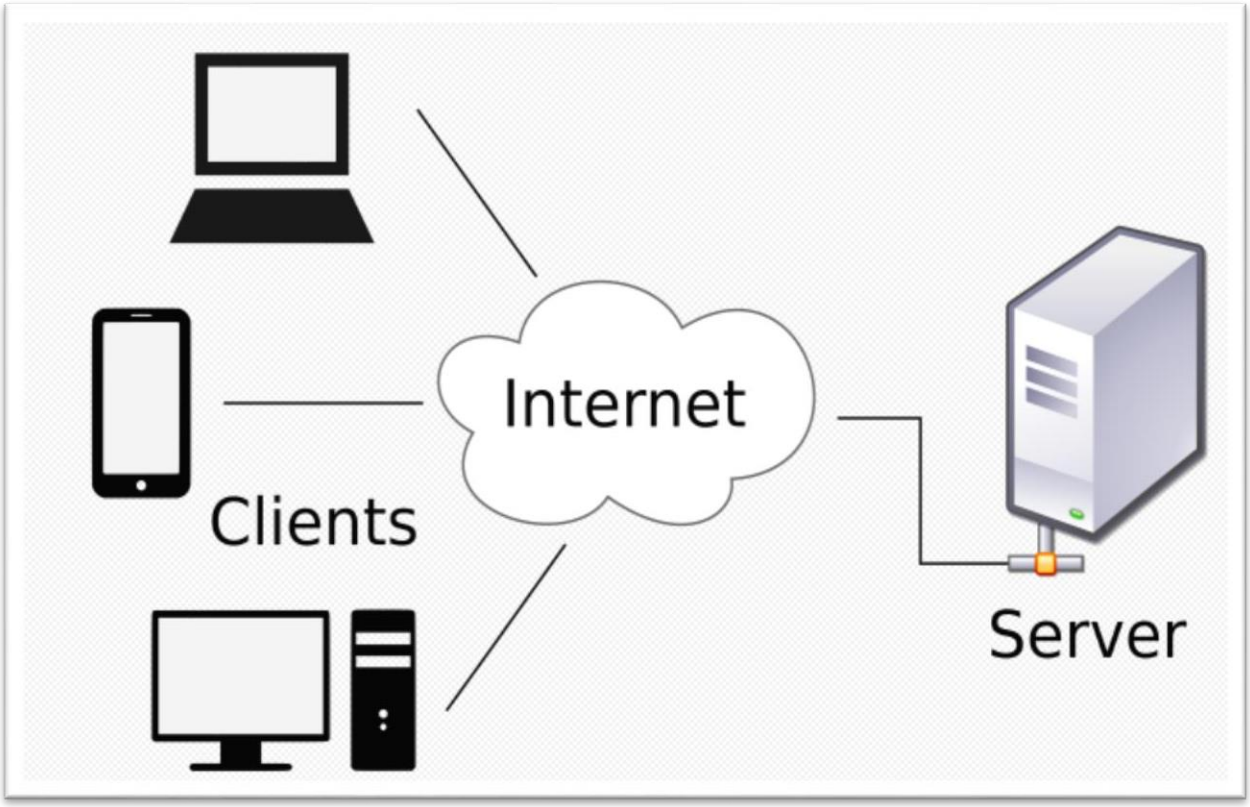


Рисунок 2.2 - Схема дворівневої клієнт-серверної архітектури

2.2.3 N-ярусна архітектура

В n-рівневій архітектурі об'єкти додатків розподіляються на декількох логічних рівнях, як правило, трьох або чотирьох.

У трирівневій архітектурі сервер баз даних не використовує серверну машину із сервером веб-додатків. Клієнт знаходиться на першому рівні, як і в дворівневій архітектурі. На третьому рівні сервер баз даних обслуговує дані. З міркувань продуктивності сервер баз даних зазвичай використовує збережені процедури для обробки певної бізнес-логіки. Сервер додатків знаходиться на другому рівні. Сервер додатків обробляє частину бізнес-логіки, яка не потребує функціональних можливостей, що надаються сервером баз даних. У цьому підході апаратні та програмні компоненти другого та третього рівнів поділяють

відповідальність за доступність, масштабованість та характеристики продуктивності веб-середовища.

У чотирирівневій архітектурі може існувати більше одного логічного рівня в межах середнього рівня або в межах рівня інформаційної системи підприємства. Наприклад:

Середній рівень може складатися з декількох веб-серверів. Крім того, проміжний брандмауер може відокремити веб-сервер від сервера додатків середнього рівня.

Сервер баз даних на третьому рівні може бути джерелом даних для веб-сервера на середньому рівні, а інший сервер баз даних на четвертому рівні є джерелом даних для сервера баз даних на третьому рівні.

Якщо ви оглянете всі веб-програми, які сьогодні доступні, ви знайдете багато варіантів. Наприклад, сервери баз даних можуть працювати на різних платформах, як і клієнти. Дизайнери веб-додатків використовують різні інструменти, які впливають на те, як працюють програми та як вони виглядають. Різні компанії вибирають різні інструменти. Частинки головоломки, що складаються з головоломок однієї компанії, в підсумку відрізняються від головоломок інших компаній.

У багатьох випадках клієнт і сервер для веб-програми знаходяться в різних операційних системах. Наприклад, клієнт може перебувати в операційній системі на базі робочої станції, такі як Windows XP або UNIX. Сервер додатка також може знаходитись на сервері робочої станції або на корпоративному сервері, наприклад z / OS®, як дворівневий зв'язок між клієнтом на базі робочої станції та обома типами серверів.

Браузер використовує протокол передачі гіпертексту (HTTP) для переадресації запитів користувачів на серверну машину другого рівня. (HTTP - це протокол зв'язку, який використовує Інтернет.) Веб-сервер другого рівня викликає локальний сервер баз даних, щоб задовольнити вимоги до даних програми.

У цьому прикладі на середньому рівні встановлено два веб-сервери: HTTP-сервер, такий як IBM® HTTP-сервер, та веб-сервер додатків.

Сервер додатків підтримує різні компоненти, які можуть працювати на середньому рівні (файли JSP, сервлети, EJB та веб-служби). Кожен із цих компонентів виконує функції, що підтримують клієнтські програми.

У середовищі WebSphere Application Server пристрій першого рівня, наприклад браузер, може використовувати HTTP для доступу до HTTP-сервера на середньому рівні. Потім сервер HTTP може рендерити вихідні дані, створені JSP, сервлетами та іншими компонентами, що працюють у середовищі WebSphere Application Server . JSP або сервлети можуть використовувати JDBC, SQLJ або EJB (опосередковано) для доступу до даних на сервері баз даних Db2 третього рівня.

Приклад архітектури можемо бачити нижче на рис. 2.3

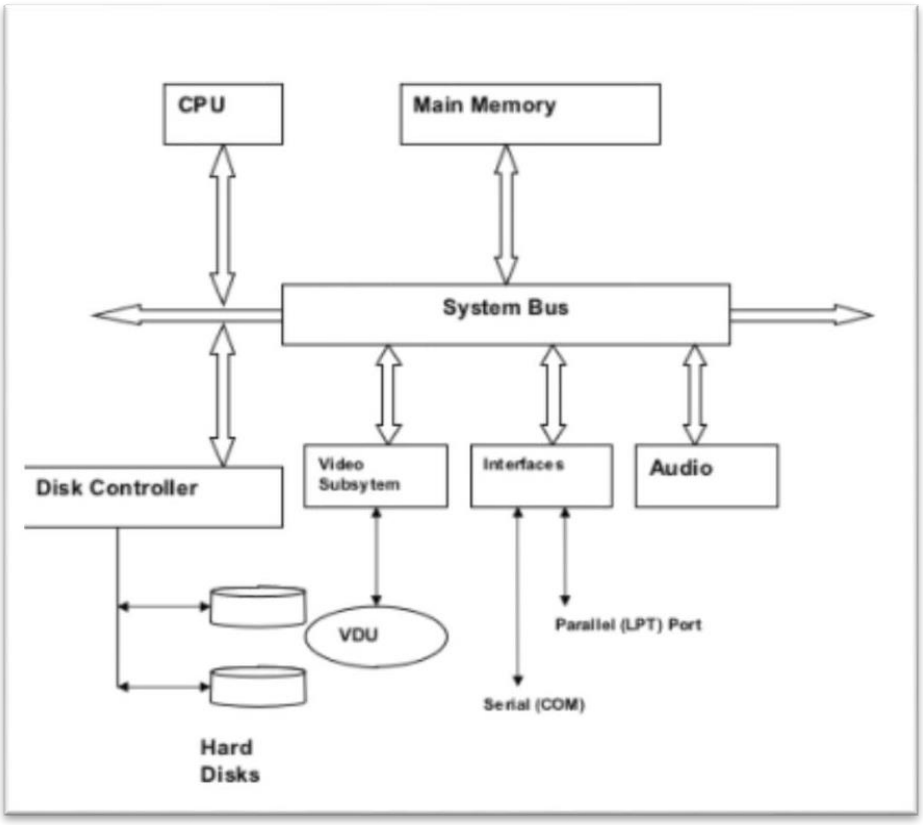


Рисунок 2.3 - Схема багаторівневої архітектури

2.2.4 Сервісно-орієнтована архітектура

Сервісно-орієнтована архітектура (SOA) - це архітектурний шаблон у розробці програмного забезпечення комп’ютера, при якому компоненти додатків надають послуги іншим компонентам через протокол зв'язку, як правило, через мережу.

Принципи орієнтації на послуги не залежать від будь-якого товару, постачальника чи технології.

SOA просто полегшує роботу програмних компонентів через різні мережі між собою.

Веб-служби, побудовані відповідно до архітектури SOA, як правило, роблять веб-сервіс більш незалежним. Самі веб-служби можуть обмінюватися даними між собою, і завдяки основним принципам, на яких вони створені, їм не потрібна будь-яка людська взаємодія, а також не потрібні будь-які модифікації коду. Це гарантує, що веб-служби в мережі можуть безперешкодно взаємодіяти між собою.

SOA базується на деяких ключових принципах, які згадані нижче:

Стандартизований контракт на надання послуг - послуги відповідають опису послуг. Послуга повинна мати якийсь опис, який описує, про що йдеться в службі. Це полегшує клієнтським програмам розуміння того, що робить служба.

Слабке зчеплення - Менша залежність один від одного. Це одна з основних характеристик веб-служб, яка просто стверджує, що між веб- службами та клієнтом, що викликає веб-службу, повинно бути якомога менше залежності. Отже, якщо функціональність служби змінюється у будь-який момент часу, це не повинно зламати клієнтську програму або зупинити її роботу.

Абстракція послуг - служби приховують логіку, яку вони інкапсулюють, від зовнішнього світу. Послуга не повинна розкривати, як вона виконує свою функціональність; він повинен просто повідомити програмі клієнта про те, що вона робить, а не про те, як вона це робить.

Повторне використання послуг - логіка поділяється на служби з метою максимізації повторного використання. У будь-якій розробницькій компанії повторне використання є важливою темою, оскільки очевидно, що не хочеться витрачати час і зусилля на створення одного і того ж коду знову і знову для багатьох додатків, які потребують їх. Отже, як тільки код веб-служби написаний, він повинен мати можливість працювати з різними типами програм.

Автономія служби - служби повинні контролювати логіку, яку вони інкапсулюють. Служба знає все про те, яку функціональність вона пропонує, а отже, вона також повинна мати повний контроль над кодом, який вона містить.

Стан послуги - в ідеалі, послуги повинні бути без стану. Це означає, що служби не повинні утримувати інформацію від одного стану до іншого. Це потрібно було б зробити з будь-якої програми клієнта. Прикладом може бути замовлення, розміщене на торговому сайті. Тепер ви можете мати веб-сервіс, який визначає ціну конкретного товару. Але якщо товари додаються в кошик для покупок, а веб-сторінка переходить до сторінки, де ви здійснюєте платіж, веб-служба не повинна нести відповідальність за ціну товару, який буде перенесено на сторінку платежу. Натомість це має робити веб-додаток.

Виявлення послуг - служби можна виявити (зазвичай у реєстрі служб). Ми вже бачили це в концепції UDDI, який створює реєстр, який може містити інформацію про веб-службу.

Складність послуг - послуги розбивають великі проблеми на невеликі. Ніколи не слід вбудовувати всю функціональність програми в одну службу, а натомість розбивати службу на модулі, кожен з яких має окрему функціональність бізнесу.

Сумісність послуг - послуги повинні використовувати стандарти, які дозволяють послугам користуватися різним абонентам. У веб-сервісах використовуються стандарти, як XML та зв'язок через HTTP, щоб переконатися, що вони відповідають цьому принципу.

Приклад SOA можемо розглянути нижче на рис. 2.4.

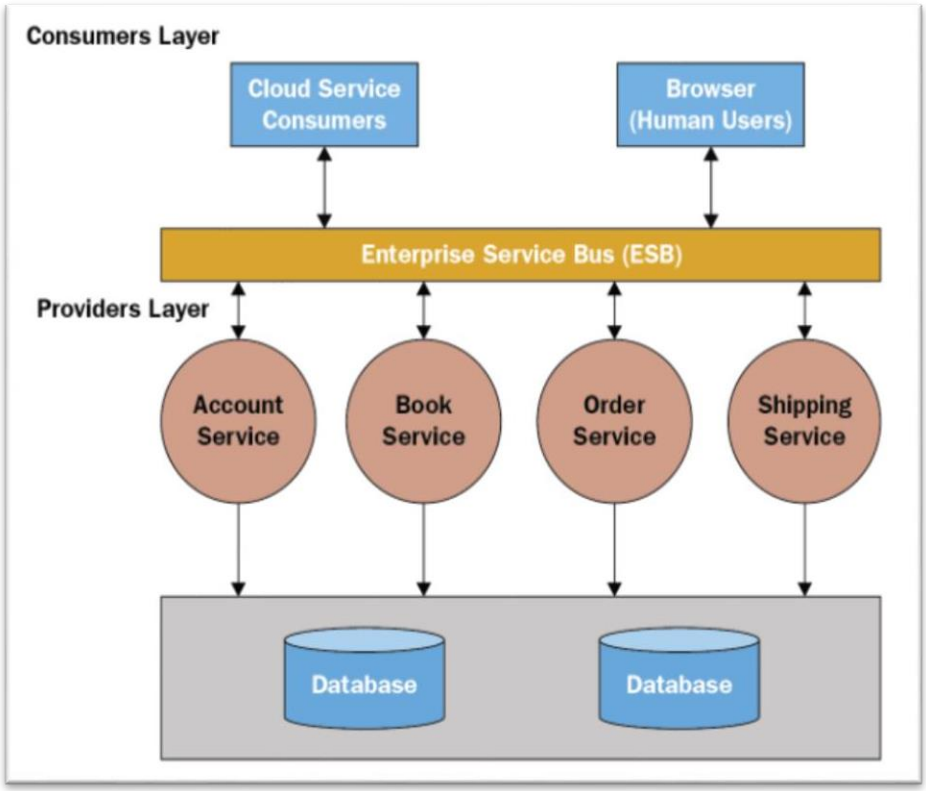


Рисунок 2.4 - Схема SOA

2.2.5 Архітектура мікропослуг

Мікросервіси (або архітектура мікропослуг) - це власний хмарний архітектурний підхід, при якому єдине додаток складається з безлічі вільно зв'язаних і незалежно розгортаються менших компонентів або служб. Ці послуги зазвичай можуть:

- Мати власний стек технологій, включаючи базу даних та модель управління даними.
- Спілкуватися між собою за допомогою комбінації REST API , трансляції подій та посередників повідомлень.
- Організовані відповідно до ділових можливостей, при цьому послуги, що розділяють лінії, часто називають обмеженим контекстом. Хоча велика частина дискусії про мікросервіси оберталася навколо архітектурних визначень та характеристик, їх значення можна зрозуміти частіше через досить прості ділові та організаційні переваги:

Код можна оновлювати простіше - нові функції або функції можна додавати, не використовуючи всієї програми.

Команди можуть використовувати різні стеки та різні мови програмування для різних компонентів. Компоненти можна масштабувати незалежно один від одного, зменшуючи витрати та витрати, пов’язані з необхідністю масштабувати цілі програми, оскільки одна функція може стикатися із занадто великим навантаженням.

Мікросервіси також можна зрозуміти тим, чим вони не є. Два найпоширеніших порівняння з архітектурою мікропослуг - це монолітна архітектура та архітектура, орієнтована на послуги (SOA) .

Різниця між мікропослугами та монолітною архітектурою полягає в тому, що мікропослуги складають єдину програму з безлічі менших, вільно пов’язаних служб, на відміну від монолітного підходу великої, тісно пов’язаної програми.

Приклад архітектури можемо побачити нижче на рис. 2.5.

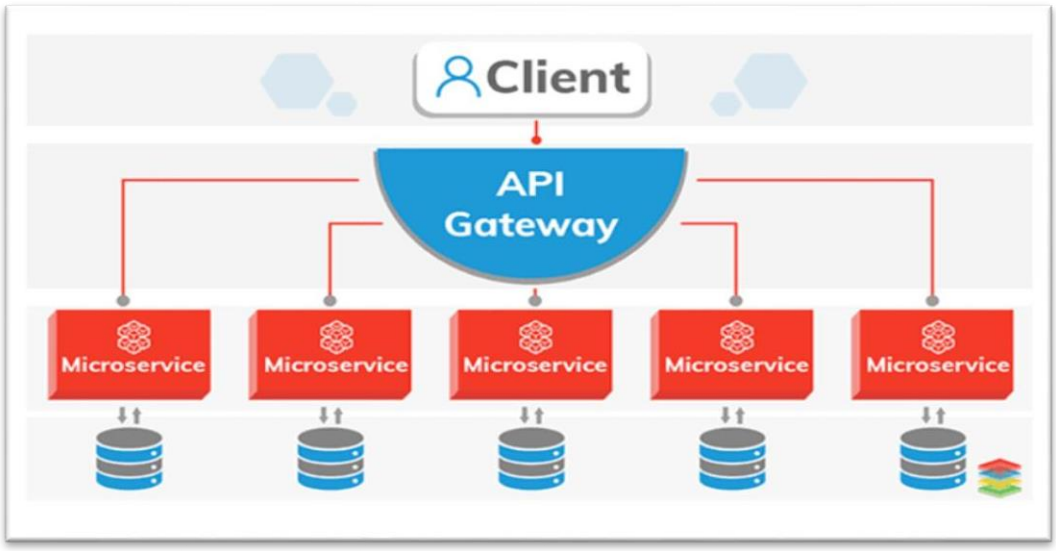


Рисунок 2.5 - Схема архітектури мікропослуг

2.3 Вибір мови програмування

Розглянемо найбільш популярні та ефективні мови програмування для створення веб-платформи.

2.3.1 HTML

Вперше розроблений Тімом Бернерсом-Лі в 1990 році, HTML - це скорочення від Hypertext Markup Language. HTML використовується для створення електронних документів (так званих сторінок), які відображаються у Всесвітній павутині. Кожна сторінка містить ряд з'єднань з іншими сторінками, які називаються гіперпосиланнями. Кожна веб-сторінка, яку ви бачите в Інтернеті, написана з використанням тієї чи іншої версії HTML-коду.

HyperText - це спосіб, за допомогою якого ви пересуваєтесь по Інтернету натискаючи на спеціальний текст, який називається гіперпосиланнями, що приводить вас до наступної сторінки. Той факт, що він гіпер, просто означає, що він не є лінійним - тобто ви можете зайти в будь-яке місце в Інтернеті, коли завгодно, натискаючи на посилання - немає жодного встановленого порядку робити щось.

Розмітка - це теги HTML, які роблять з текстом усередині них. Вони позначають це як певний тип тексту (наприклад, курсив).

HTML - це мова, оскільки вона має кодові слова та синтаксис, як і будь- яка інша мова.

HTML складається з ряду коротких кодів, набраних текстовим файлом автором сайту - це теги. Потім текст зберігається у форматі html і переглядається через браузер, наприклад Internet Explorer або Netscape Navigator. Цей браузер читає файл і переводить текст у видиму форму, сподіваючись, робить сторінку такою, як задумав автор. Написання власного HTML вимагає правильного використання тегів для створення вашого бачення.

Для створення HTML-сторінок можна використовувати що завгодно - від елементарного текстового редактора до потужного графічного редактора.

Теги - це те, що відокремлює звичайний текст від HTML-коду. Ви можете їх знати як слова між <angle-brackets>. Вони дозволяють всі цікаві речі, такі як зображення, таблиці та інше, просто повідомляючи браузеру, що відображати на сторінці. Різні теги будуть виконувати різні функції. Самі теги не відображаються, коли ви переглядаєте свою сторінку через браузер, але їх ефекти відображаються. Найпростіші теги роблять не що інше, як застосовувати форматування до якогось тексту, наприклад: Hello, how are you?.

У наведеному вище прикладі теги були обгорнуті навколо якогось тексту, і їх наслідком буде те, що вміщений текст буде виділений жирним шрифтом при перегляді через звичайний веб-браузер.

Вивчення прийомів та правильне використання ваших знань тегів надзвичайно покращить вашу роботу, а добре розуміння загального дизайну та аудиторії, яку ви намагаєтеся охопити, покращить шанси на успіх вашого веб-сайту. На щастя, ці речі можна дослідити і зрозуміти, якщо ви готові працювати над цим, щоб ви могли виводити кращі веб-сайти.

2.3.2 CSS

CSS є простою мовою дизайн покликаний спростити процес створення веб-сторінок презентабельно.

CSS обробляє зовнішній вигляд частини веб-сторінки. За допомогою CSS ви можете керувати кольором тексту, стилем шрифтів, інтервалом між абзацами, розміром та розміщенням стовпців, які фонові зображення або кольори використовуються, дизайном макета, варіаціями відображення для різних пристроїв та розмірами екрану а також безліч інших ефектів.

CSS легко вивчити і зрозуміти, але він забезпечує потужний контроль над поданням HTML-документа. Найчастіше CSS поєднується з мовами розмітки HTML або XHTML.

При використані CSS маємо такі переваги:

- CSS економить час - є можливість написати CSS один раз, а пізніше використати теж саме на інших HTML-сторінках. Ви можете визначити стиль для кожного елемента HTML і застосувати його до скільки завгодно веб- сторінок.
- Під час використання Css, сайт завантажується швидше, вам не потрібно кожного разу писати атрибути тегів HTML. Просто напишіть одне правило CSS тегу та застосуйте його до всіх випадків появи тегу. Тож менше коду означає швидший час завантаження.
- Простота обслуговування - щоб внести глобальні зміни, просто змініть стиль, і всі елементи на всіх веб-сторінках будуть оновлені автоматично.
- Покращені стилі HTML - CSS має набагато ширший набір атрибутів, ніж HTML, тому ви можете набагато краще виглядати на своїй HTML-сторінці в порівнянні з атрибутами HTML.
- Сумісність декількох пристроїв - таблиці стилів дозволяють оптимізувати вміст для більш ніж одного типу пристроїв. Використовуючи один і той же документ HTML, різні версії веб-сайту можуть бути представлені для портативних пристроїв, таких як КПК та мобільні телефони, або для друку.
- Глобальні веб-стандарти - зараз атрибути HTML застаріли, і рекомендується використовувати CSS. Тож непогано почати використовувати CSS на всіх HTML-сторінках, щоб зробити їх сумісними з майбутніми браузерами.

2.3.3 Javascript

JavaScript - це динамічна мова програмування, яка використовується для веб-розробки, веб-додатків, розробки ігор та багато іншого. Це дозволяє вам

реалізовувати динамічні функції на веб-сторінках, що неможливо виконати лише за допомогою HTML та CSS.

Багато браузерів використовують JavaScript як мову сценаріїв для здійснення динамічних дій в Інтернеті. Кожного разу, коли ви бачите спадне

меню, що відображається, додатковий вміст, доданий на сторінку, та динамічно змінювані кольори елементів на сторінці, щоб назвати декілька функцій, ви бачите ефекти JavaScript.

HTML визначає структуру вашого веб-документа та його вміст. CSS декларує різні стилі вмісту веб-документа.

HTML і CSS часто називають мовами розмітки, а не мовами програмування, оскільки вони в своїй основі забезпечують розмітки для документів із дуже незначним динамізмом.

JavaScript, з іншого боку, є динамічною мовою програмування, яка підтримує математичні обчислення, дозволяє динамічно додавати вміст HTML до DOM, створює динамічні декларації стилів, отримує вміст з іншого веб-сайту та багато іншого.

2.3.4 Bootstrap

Bootstrap - це найпопулярніший, безкоштовний фреймворк з відкритим кодом для створення адаптивного макета на веб-сторінках із значно меншими зусиллями. Він містить компоненти HTML, CSS та JS для створення форм, кнопок, навігації, випадаючого списку, модалів, макета та багатьох інших речей, список справді дуже довгий. Ви можете створити все це без особливих зусиль, для чого в іншому випадку знадобиться багато CSS, HTML та JS коду.

Веб-дизайнери та розробники люблять використовувати Bootstrap у своїх проектах. Вони використовують його для створення адаптивного веб- дизайну, який виглядає абсолютно точно на всіх розмірах екрана (смартфони, планшети, ноутбуки та ПК).

Якщо ви веб-розробник, вам слід вивчити переваги, які надає Bootstrap:

- Заощаджує ваш час - швидко створюйте функції, використовуючи заздалегідь визначені класи та шаблони дизайну, які надає bootstrap.
- Адаптивний дизайн - за допомогою Bootstrap вам не потрібно застосовувати медіа-запити у своєму файлі CSS. Він здійснює динамічне налаштування веб-сторінки на всі розміри екрану.
- Сумісний з усіма браузерами - вам не потрібно турбуватися про будь-який браузер, оскільки він сумісний з останніми версіями всіх браузерів - Google Chrome, Firefox, Opera, Safari та Edge.
- Легко і просто - це дуже легко і просто використовувати у веб-дизайні. Якщо у вас є базові знання HTML та CSS, вам слід продовжувати їх.
- Послідовність - це дає вам узгодженість між вашими проектами та іншими розробниками.
- Безкоштовні та з відкритим кодом - жодних обмежень немає. Bootstrap доступний на GitHub, де розробники можуть внести свої зусилля.

2.3.5 PHP

PHP - це мова сценаріїв, яка зазвичай використовується у веб-розробці на стороні сервера. Для того, щоб все це розібрати, дуже важливо спочатку зрозуміти, що таке мова сценаріїв. Мови сценаріїв (сімейство мов програмування, включаючи PHP, а також такі мови, як JavaScript і Ruby) - це підмножина мов кодування, що використовуються для автоматизації процесів, які в іншому випадку повинні були б виконуватися поетапно в коді сайту кожного разу, коли вони виникають.

Сюди входять такі речі, як діалогові вікна, що відкриваються на екрані у відповідь на дії користувача, чат-боти, що відповідають на визначену поведінку користувача відповідними повідомленнями, або анімація, яка відбувається, коли користувач прокручує повз певну точку сторінки - будь-які динамічні функції веб-сайту, які потрібно відбуваються на екрані без необхідності користувачеві

перезавантажувати сайт вручну. Мови сценаріїв, такі як PHP, відрізняються від мов розмітки, таких як HTML і CSS, в тому сенсі, що, хоча HTML і CSS визначають компонування та вигляд веб-сторінок, мови сценаріїв вказують статичній веб-сторінці (побудованій з HTML та CSS) "робити" конкретні дії. Якщо ви витратили якийсь час на читання про JavaScript, це може здатися звичним. Тож PHP - це ще один спосіб досягнення того, що можна зробити за допомогою JavaScript? Не зовсім.

Як вже згадувалося раніше, PHP зазвичай використовується як мова на стороні сервера (на відміну від мови, такої як JavaScript, яка зазвичай виконується на стороні клієнта). То що це означає? З точки зору програмування, на стороні клієнта мається на увазі діяльність веб-сайту, яка відбувається локально на комп'ютері користувача через веб-браузер користувача. Мови на стороні клієнта, такі як HTML, CSS та JavaScript, дають вказівки, що веб- браузери можуть аналізувати та перетворювати вміст на екрані вашого комп’ютера. Зверніть увагу, що в цьому списку є JavaScript (мова сценаріїв, така як PHP). Знову ж таки, процеси, сценарій яких виконується JavaScript, відбуваються на стороні клієнта - JS надає інструкції, які можуть бути зрозумілі та виконані у вашому веб-браузері. Клієнтська сторона - це сторона, яку ви бачите під час користування Інтернетом.

З іншого боку, діяльність на стороні сервера передбачає надсилання веб-браузером запитів на веб-сервер (програмне чи апаратне забезпечення, що зберігає сторінки веб-сайтів, зображення, медіа та інші ресурси), який потім відповідає на запит HTML-кодом, який може оброблятися та відображатись веб- браузером і перетворюватися на вміст на екрані користувача. Основна відмінність від активності на стороні клієнта полягає в тому, що цей процес передбачає спілкування з сервером і не завершується повністю в браузері клієнта. Іншими словами, мова сценаріїв на стороні клієнта, як JavaScript, може автоматизувати завдання, що включають вміст, який вже доступний користувачеві в його веб-браузері, але мова сценаріїв на стороні сервера, як PHP, використовується для запиту вмісту з сервера веб-сайту або бази даних та створення вміст, видимий та

доступний для користувача сайту. Наприклад, PHP-скрипт може зробити так, щоб ваші три останні публікації блогу автоматично з’являлися на першій сторінці вашого сайту. У цьому випадку самі повідомлення зберігаються на сервері сайту і викликаються, коли вони займають одне з трьох останніх опублікованих слотів. Це дозволяє уникнути як попередньої завантаження публікацій на вашому сайті, так і необхідності адміністратора сайту завантажувати та оновлювати публікації, коли публікуються нові статті.

Сценарії PHP також можуть включати умовні оператори (if / else / endif), які спрямовують ваш сайт на зміну відображення та додавання вмісту з веб- сервера за потреби. Сюди можуть входити такі дії, як диктування того, що якщо адміністратор сайту завантажує відеопосилання в поле «x», то сайт завантажує відео зі свого сервера і відображає його для користувача. Далі сценарій може стверджувати, що якщо адміністратор не завантажує посилання, то натомість на сторінці відображатиметься зображення за замовчуванням "y". Серверні дії PHP вводять абсолютно новий рівень динамічних можливостей веб-сайту (понад статичні функції, пропоновані HTML та CSS, і навіть динамічний вміст на стороні клієнта, що стає можливим завдяки JavaScript).

У цьому розділі було здійснено обстеження сучасних технологій та архітектурних методів програмного забезпечення, які використовують для реалізації нашої програми. Також була описана архітектура веб-платформи.

Для розробки веб-додатку для вивчення онлайн курсів, було вибрано технології, які призначенні для створення інтернет ресурсу та легко справляються з поставленими задачами. Для створення зовнішнього вигляду, було використано такі технології програмування як: JS, HTML, CSS, а для внутрішньої частини було використано: PHP.

Одним з найголовніших технологій було представлено та обрано базу даних MySql. Вона має у собі всі сучасні методи зберігання даних та їх обробки.

3 ЕТАПИ ТА МЕТОДИ РОЗРОБКИ ВЕБ-ПЛАТФОРМИ

3.1 Етапи розробки

Створення веб-сайту - це дуже складний і кропіткий процес, який займає багато часу. Для збільшення ефективності роботи, творець має використовувати сучасні етапи та методи розробки, які пришвидшують розробку веб-платформи.

Під час створення веб-платформи для вивчення онлайн курсів були використані такі етапи та методи, які наведені в наступних підрозділах.

3.1.1 Збір всієї необхідної інформації

Як і у більшості нестандартних проектів, спочатку робота почалась зі збору інформації. Хоча цей крок може бути на межі дріб’язковості, цей крок є найважливішим із усього процесу.

Якщо ми збираємо всю необхідну інформацію з самого початку, ми економимо величезну кількість часу в дорозі, особливо на початкових етапах етапу проектування.

В середньому було досліджено щонайменше десяток веб-сайтів, щоб вилучити з них найкращі ідеї для галузі, в якій буде будуватись сайт. Потім відбулось вдосконалення цих ідей та включення у план роботи.

3.1.2 Огляд вмісту

Ця фаза проходить паралельно зі створенням сайту через весь процес розробки. Причина в тому, що неминуче відбувається збір дедалі більше інформації, коли процес протікає на кожному кроці.

Починаючи наповнювати сайт цінним вмістом, починається все з головної сторінки (яку також називають домашньою) та основних внутрішніх сторінок.

Слід зазначити, що коли говориться про зміст, мається на увазі не лише текст. Зміст також включає візуальні ефекти, такі як фотографії, відео, таблиці та діаграми. Навіть аудіокліпи вважаються вмістом. По суті, це все, що завгодно, на що нам натрапляють відвідувачі.

3.1.3 Дизайн

Після створення мапи сайту, розробки каркасів та планування дорожньої карти настав час взяти руку за дизайн веб-сайту. Каркаси трансформуються у типографіку, кольорову графіку, анімацію, кнопки, меню та багато іншого.

Цільова аудиторія - один із ключових факторів, що враховується при розробці дизайну. Дизайн визначає, наскільки унікальним може бути веб-сайт, і це фактор для приємного користувацького досвіду. Щоб створити хороше враження для користувача, веб-дизайн повинен бути захоплюючим.

Повинна бути форма брендингу, яка відповідає меті сайту. Суміш кольорів, вибрана для веб-сайту, безумовно, керує взаємодією користувачів. Колір може викликати різні емоції, отже творчий спосіб його поєднання може бути дуже ефективним в дизайні.

3.1.4 Впровадження

Після схвалення дизайну, наступним моментом є розробка самого веб- сайту, що є головним завданням:

Розробка фронтенду - це розробка клієнтської частини веб-сайту для взаємодії з користувачами. Дизайн, спочатку зроблений на ранніх стадіях, перетворюється на спеціальну анімацію та ефекти. Функції згодом інтегруються на основі вибору технологій та інструментів.

Розробка бекенда - це зворотний бік розвитку інтерфейсу. Бекенд - це взаємодія на стороні користувача та на стороні сервера, пов’язуючи весь веб- сайт. Це більше схоже на машинне відділення. Код на сервері відповідає за сторону сервера, базу даних, інтеграцію бізнес-логіки тощо, залежно від призначення веб-сайту.

3.2 Методи розробки

Під час створення веб-платформи для вивчення онлайн курсів були використані такі методи:

- IntelliJ IDEA - це спеціальне середовище програмування або інтегроване середовище розробки (IDE), в основному призначене для Java. Це середовище використовується спеціально для розробки програм. Він розроблений компанією під назвою JetBrains, яка офіційно називалася IntelliJ. Він доступний у двох виданнях: Community Edition, яка ліцензована Apache 2.0, та комерційне видання, відоме як Ultimate Edition. Обидва вони можуть бути використані для створення програмного забезпечення, яке можна продати. Що робить IntelliJ IDEA настільки відмінним від аналогів, це простота використання, гнучкість та міцний дизайн. IntelliJ IDEA була розроблена компанією JetBrains, раніше відомою як IntelliJ. Вперше він був випущений в 2001 році, і він мав такі функції, як вдосконалена навігація кодом та можливість рефакторингу кодів, що зробило його дуже популярним. У 2010 році він навіть отримав відзнаку бути визнаним найкращим інструментом програмування, заснованим на Java, і виділив такі інструменти, як NetBeans, Eclipse та JDeveloper. Середовище розробки з відкритим кодом для Android, випущене Google в 2014 році, також базується на IntelliJ IDEA. IDE підтримує багато інших мов програмування, таких як Python, Lua та Scala. Найбільшою причиною, по якій його розглядають як один з найкращих інструментів програмування на основі Java, є його допоміжні функції, що робить його простим у використанні та робить створені ним програми дуже добре

розробленими. Він також має розширені функції перевірки помилок, які дозволяють швидше і простіше перевіряти помилки.

- Atom являє собою текстовий редактор, який є відкритим вихідним кодом. Це альтернатива розробника для використання більш простих текстових редакторів, таких як блокнот або блокнот плюс плюс. На відміну від notepad plus plus та Notepad Adam надає додаткові переваги для розробників, такі як підсвічування синтаксису для певних блоків коду.

Можливість переглядати всю структуру вашого проекту, щоб ви могли переходити між файлами набагато швидше, ніж якщо ви просто редагуєте в блокноті в notepad plus plus.

- MAMP - це одна з декількох популярних платформ . Він перетворює ваш комп'ютер на серверне середовище, яке може розміщувати веб-сайти під час роботи на них. MAMP використовує Apache, MySQL та PHP, що робить її дуже сумісною з WordPress. Доступна безкоштовна версія, або ви можете заплатити за власну версію, яка включає програми встановлення та інші функції, які допоможуть швидко налаштувати перший веб-сайт та покращити робочий процес. Як і всі сайти, розміщені локально, ваш веб-сайт для розробки та тестування MAMP не буде загальнодоступним. Це дозволяє вам вільно створювати або тестувати функції, не турбуючись про те, що це вплине на ваш фронтальний досвід користувача (UX). Це також запобігає відвідувачам натрапити на ваш напівзавершений сайт. Крім того, місцевий розвиток не вимагає з'єднання з Інтернетом, тому ви можете працювати з будь-якого місця. Місцеві сайти також, як правило, завантажуються швидше, що може дещо підвищити вашу продуктивність. Завершивши створення або внесення змін до свого сайту, ви можете перенести його на діючий сервер.

3.2.1 Клієнт-серверна модель

Клієнт-серверна модель або архітектура клієнт-сервер - це розподілена програма додатків, що розділяє завдання між серверами та клієнтами, які або проживають в одній системі, або обмінюються даними через комп’ютерну мережу або Інтернет. Клієнт покладається на відправлення запиту іншій програмі, щоб отримати доступ до послуги, що надається сервером. Сервер запускає одну або кілька програм, які діляться ресурсами та розподіляють роботу між клієнтами.

Взаємозв'язок клієнт-сервер взаємодіє за шаблоном обміну повідомленнями запит-відповідь і повинен дотримуватися загального протоколу зв'язку, який формально визначає правила, мову та шаблони діалогу, які слід використовувати.

Зв'язок клієнт-сервер зазвичай відповідає набору протоколів TCP / IP.

Протокол TCP підтримує з'єднання, поки клієнт і сервер не завершать обмін повідомленнями. Протокол TCP визначає найкращий спосіб розподілу даних додатків у пакети, які мережі можуть доставляти, передає пакети та отримує пакети від мережі, а також управляє контролем потоку та повторною передачею скинутих або спотворених пакетів. IP - це протокол без з’єднання, в якому кожен пакет, що подорожує через Інтернет, є незалежною одиницею даних, не пов’язаною з іншими одиницями даних.

Клієнтські запити організовуються та визначаються за пріоритетами в системі планування, яка допомагає серверам справлятися у випадку отримання запитів від багатьох різних клієнтів за короткий проміжок часу. Клієнт- серверний підхід дозволяє будь-якому комп'ютеру загального призначення розширити свої можливості, використовуючи спільні ресурси інших хостів. Популярні програми клієнт-сервер включають електронну пошту, всесвітню павутину та мережевий друк.

Існує безліч переваг моделі архітектури клієнтського сервера:

- Один сервер, що розміщує всі необхідні дані в одному місці, полегшує захист даних та управління авторизацією та автентифікацією користувачів.

- Такі ресурси, як сегменти мережі, сервери та комп'ютери, можна додавати до мережі клієнт-сервер без значних перебоїв.
- Доступ до даних може бути ефективним, не вимагаючи, щоб клієнти та сервер знаходились у безпосередній близькості.
- Усі вузли в системі клієнт-сервер незалежні, запитуючи дані лише від сервера, що полегшує легке оновлення, заміну та переміщення вузлів.
- Дані, які передаються через протоколи клієнт-сервер, є платформно-агностичними.

3.2.2 Веб-сервер

Для створення веб-платформи вивчення онлайн курсів був вибраний локальний сервер програми MAMP.

Веб-сервери - це програмне чи апаратне забезпечення (або обидва разом), яке зберігає та доставляє вміст до веб-браузера на базовому рівні.

Сервери спілкуються з браузерами за допомогою протоколу передачі гіпертексту (HTTP). Веб-сервери також можуть підтримувати SMTP (Простий протокол передачі пошти) і FTP (Протокол передачі файлів).

Веб-сервери також використовуються для розміщення веб-сайтів та даних для веб-додатків. Вони можуть розміщувати окремі веб-сайти та кілька веб-сайтів за допомогою віртуалізації.

Чому важливо зрозуміти відповідь на питання, як працює веб-сервер? Успіх веб-сайту залежить не тільки від його змісту та функціональних можливостей, а й від ефективності веб-сервера, який використовується для його роботи. Це вимагає розуміння можливостей та обмежень веб-сервера. Обговорюючи, як працює веб-сервер, недостатньо просто окреслити схему того, як низькорівневі мережеві пакети потрапляють і виходять із веб-сервера.

Багато років тому, коли веб-сервери були вперше прототипировані, вони обслуговували прості HTML-документи та зображення.

Робота відбувається у тому, що веб-сервер отримує запит на вміст веб-сторінки та відображає цей URL у локальний файл на хост-сервері. Потім сервер завантажує цей файл з диска і передає його через мережу до веб-браузера користувача. Весь цей обмін опосередковується браузером і сервером, що спілкуються між собою за допомогою HTTP. Цей робочий процес показано на рис. 3.1 нижче.

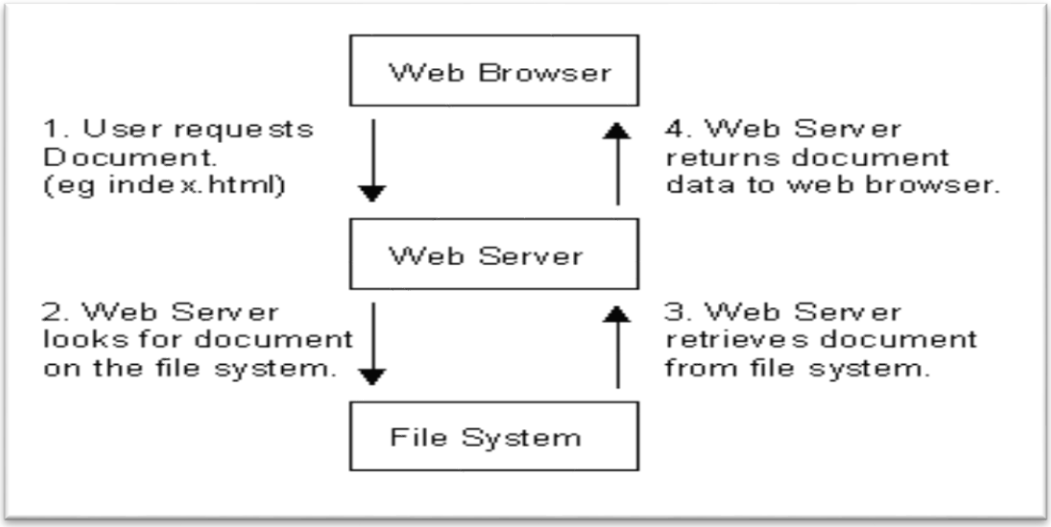


Рисунок 3.1 - Структурна схема робочого процесу між сервером та браузером

Оскільки це просте розташування, яке дозволяє подавати до веб-браузера статичний вміст, такий як мова розмітки HyperText (HTML) та файли зображень, було початковою концепцією того, що ми зараз називаємо World Wide Web. Краса його простоти полягає в тому, що це призвело до набагато більш складного обміну інформацією між браузерами та веб-серверами.

Можливо, найважливішим розширенням цього питання стала концепція динамічного вмісту (тобто веб-сторінки, створені у відповідь на введення користувача, прямо чи опосередковано). Найстарішим і найбільш часто використовуваним стандартом для цього є Common Gateway Interface (CGI) . Це досить безглузде ім'я, але воно в основному визначає, як веб-сервер повинен запускати програми локально і передавати свої результати через веб-сервер веб-браузеру користувача, який вимагає динамічний вміст.

Незважаючи на всі наміри та цілі, веб-браузер користувача ніколи не повинен знати, що вміст є динамічним, оскільки CGI в основному є протоколом розширення веб-сервера. На рис. 3.2 нижче показано, що відбувається, коли браузер запитує сторінку, яка динамічно генерується з програми CGI.

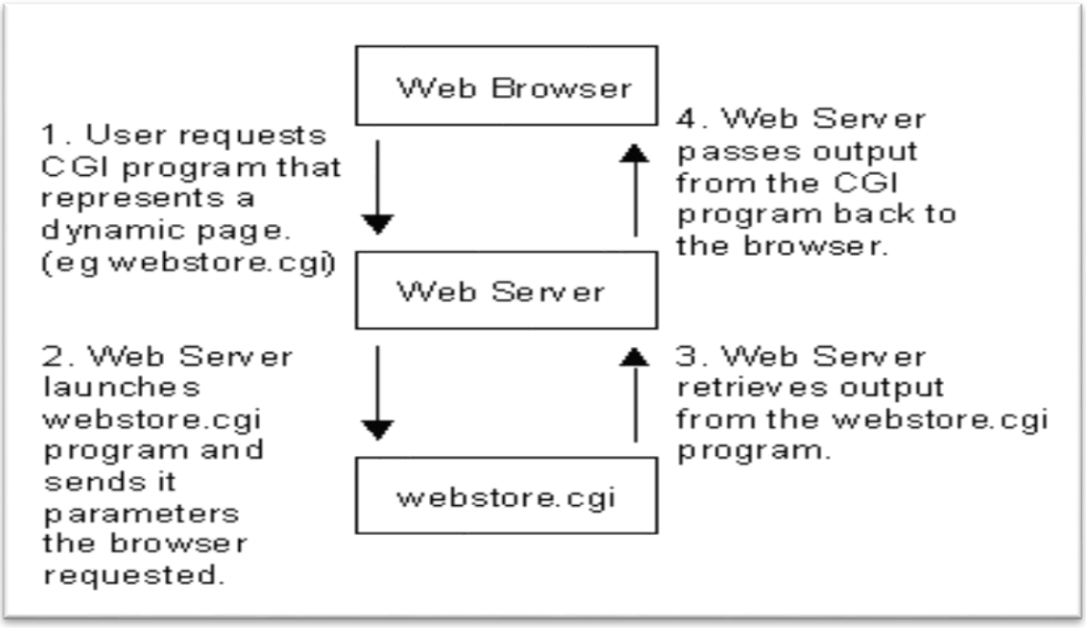


Рисунок 3.2 - Запит сторінки браузером

Другим важливим прогресом, і тим, що робить електронну комерцію можливим, було запровадження протоколу передачі HyperText Secure (HTTPS). Цей протокол дозволяє забезпечувати безпечний зв'язок між браузером та веб-сервером.

У двох словах, це означає, що користувачеві та серверу безпечно передавати конфіденційні дані один одному через те, що можна вважати небезпечною мережею. Інша справа, що трапляється, коли дані надходять у обидва кінці, і не слід ігнорувати .

3.2.3 База даних MySql

Для розробки веб-платформи було задіяно базу даних MySql.

MySQL - це система управління реляційними базами даних з відкритим кодом. Як і у інших реляційних базах даних, MySQL зберігається у таблицях, які відповідають за рядки та стовпці. Користувачі можуть визначати, обробляти, контролювати та запитувати дані за допомогою мови структурованих запитів, більш відомої як SQL. Ім'я MySQL - це поєднання “My”, імені дочки творця MySQL Майкла Віденіуса та “SQL”.

Гнучка та потужна програма, MySQL - найпопулярніша система баз даних з відкритим кодом у світі.

Як частина широко використовуваного стеку технологій LAMP (який складається з операційної системи на базі Linux, веб-сервера Apache, бази даних MySQL та PHP для обробки), він використовується для зберігання та отримання даних у найрізноманітніших популярних додатках, веб-сайти та послуги.

9

Рисунок 3.3 - Таблиця реєстрації користувачів

З даної таблиці ми бачимо, що є 4 поля:

- ‘id’ номер зареєстрованого користувача.
- ‘email’ пошта користувача, який реєструється.
- ‘name’ ім’я користувача.
- ‘pass’ пароль, який користувач вводить під час реєстрації та пізніше авторизації.

У цьому розділі було розглянуто головні етапи та методи для реалізації веб-платформи, які були детально описані. До того ж розглянуто базу даних MySql, яка застосована у розробці продукту. Програма використовує клієнт-серверну модель, що у декілька разів покращує створення і модернізацію веб- ресурсу.

Що стосується функціоналу веб-платформи, то вона має просту структуру, у якійє можливість реєстрації та доступ до данихкурсів. Описано втілення клієнт-серверної моделі у продукт, досліджено безліч переваг цієї архітектури та локальний веб-сервер на якому працює веб-платформа.

Також був проведений аналіз методів розробки за допомогою яких, було створено веб-продукт.

4 ТЕСТУВАННЯ СТВОРЕНОЇ ВЕБ-ПЛАТФОРМИ

У цьому розділі відбувається план тестування розробленої програми та ознайомлення з інтерфейсом даного ресурсу.

4.1 Ознайомлення з роботою даної веб-платформи

Створений онлайн ресурс є веб-платформою, у якій було розроблено комфортний та легкий зовнішній вигляд. Попередження у тому, що ознайомлення, тестування та план користування буде описано в одному розділі, оскільки всі ці питання взаємно підтримуються.

Для того, щоб відбувся старт даної веб-платформи, потрібно встановити програму MAMP та текстовий редактор Atom, приклад програм наведено нижче на рис. 4.1 та рис. 4.2.

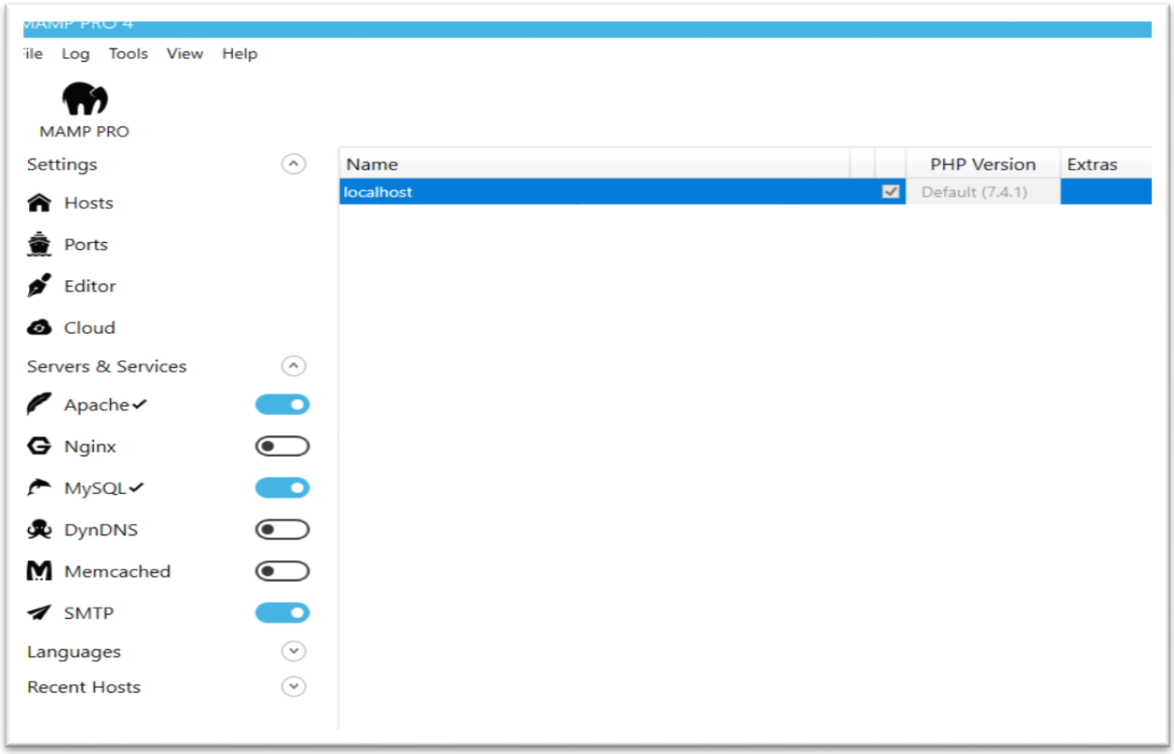


Рисунок 4.1 - Вигляд програми MAMP

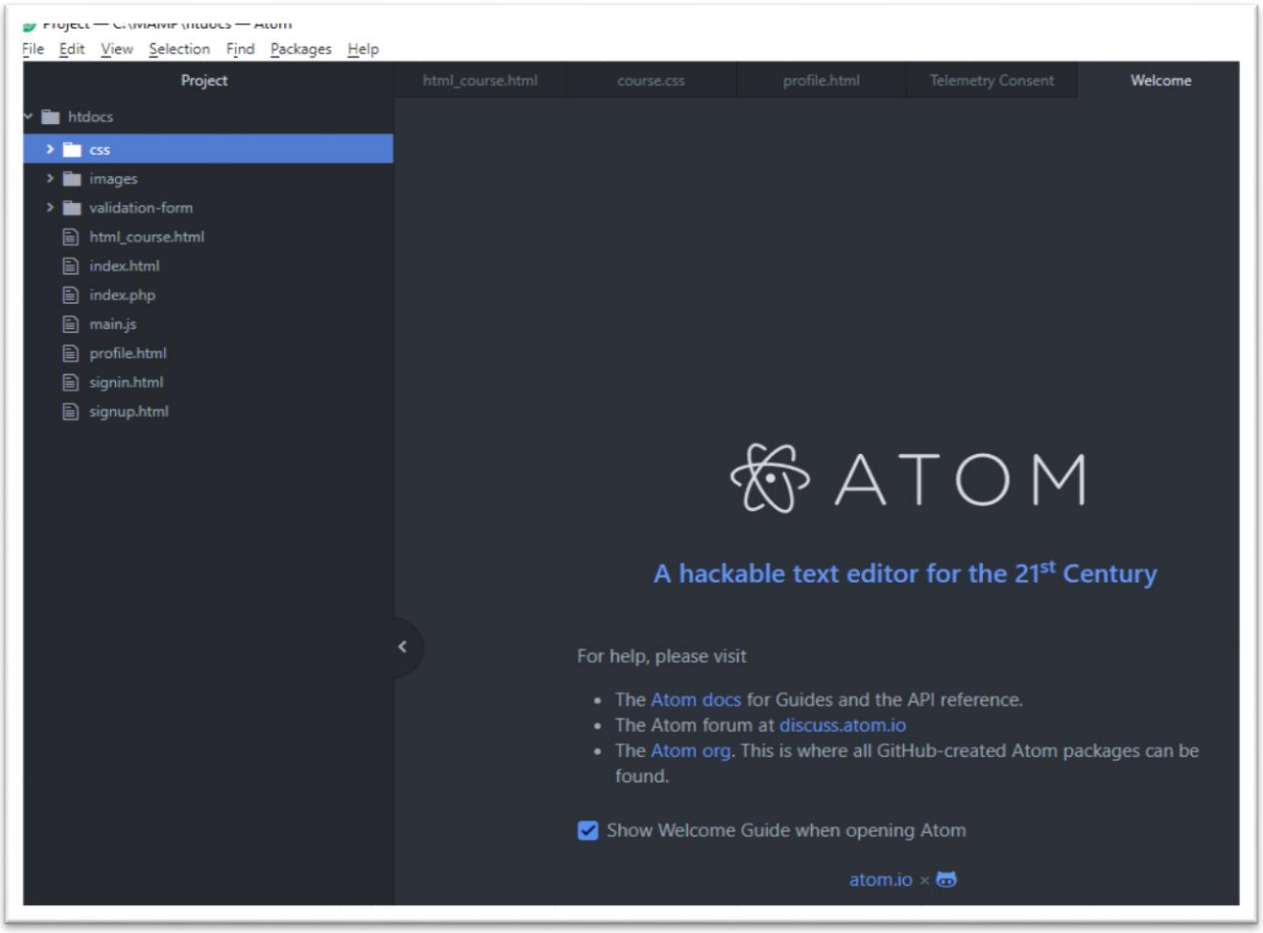


Рисунок 4.2 - Вигляд текстового редактора Atom

Щоб пов'язати ці програми, потрібно спочатку відкрити папку з програмою MAMP, далі знайти папку htdocs та перенести її в текстовий редактор, приклад можна побачити нижче на рис. 4.3.

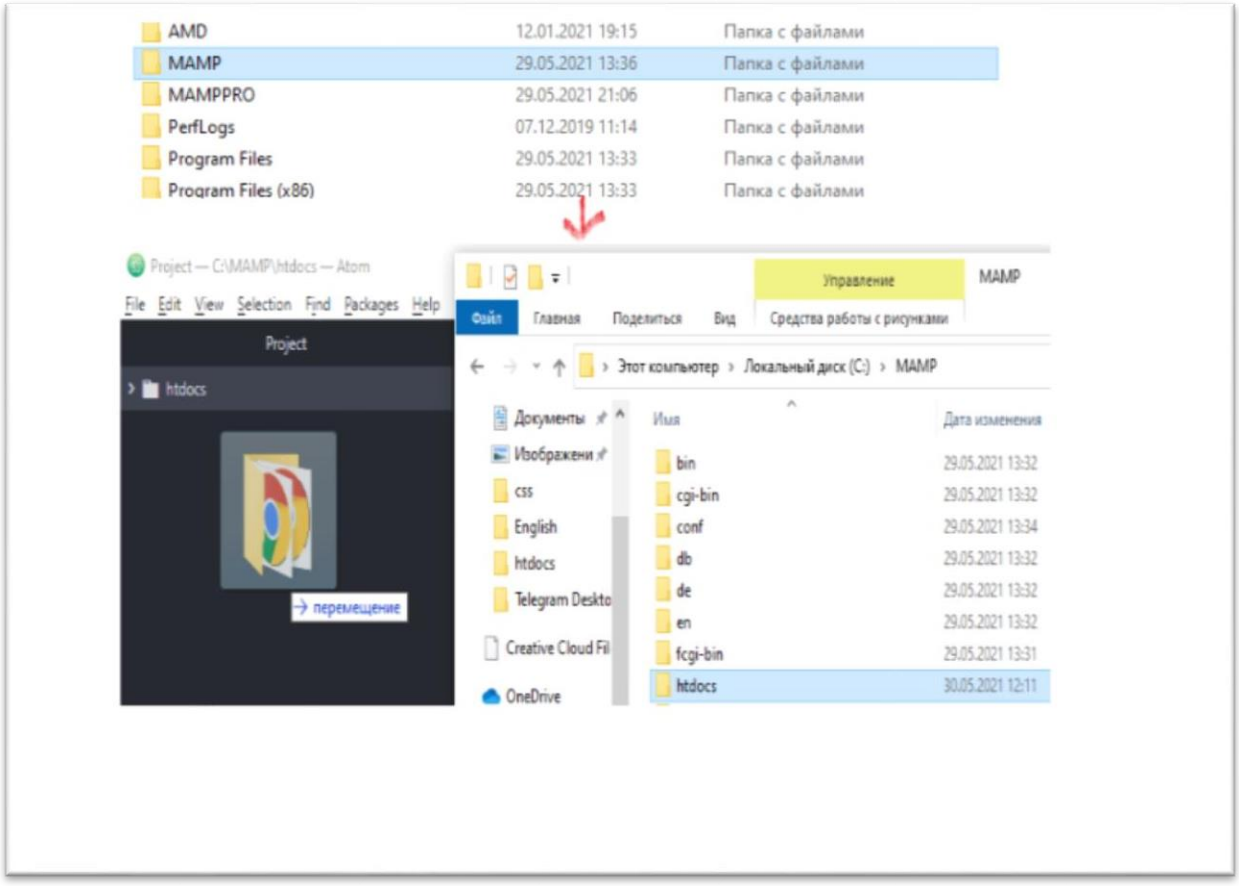


Рисунок 4.3 - Приклад початку роботи з програмами

Після вдалого запуску сервера, відкриваємо браузер та в пошуковій лінії вписуємо «localhost» та потрапляємо на головну сторінку представленої веб-платформи, яку бачимо на рис. 4.4.

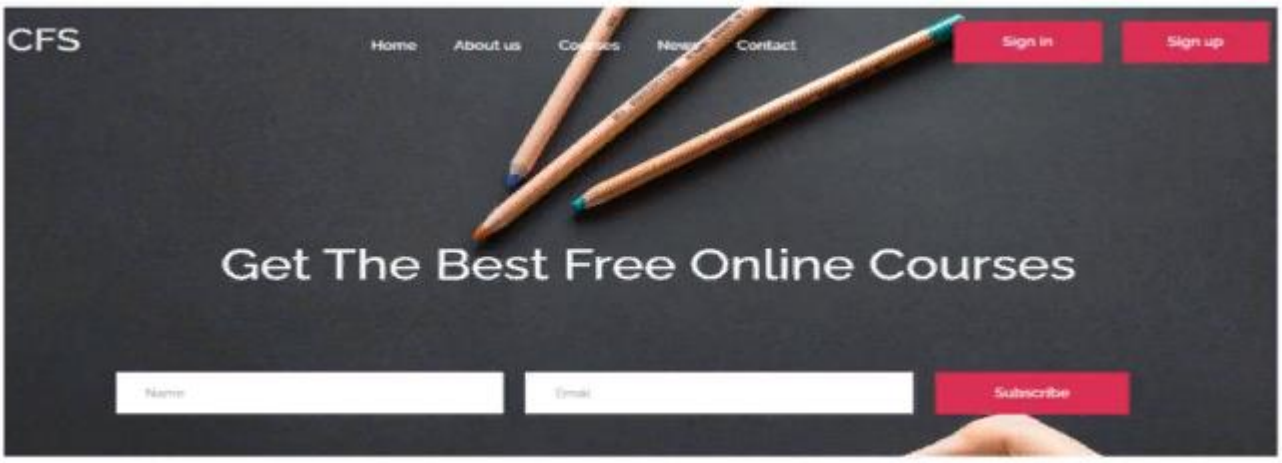


Рисунок 4.4 - Головна сторінка веб-платформи

Для того, щоб мати змогу зареєструватись новому користувачеві потрібно клацнути лівою клавішею мишкі на значок «Sign up», яку ми бачимо на рисунку вище у правому куточку веб-платформи.

Після описаних дій, потрапляємо на сторінку реєстрації користувача наведену нижче на рис. 4.5.

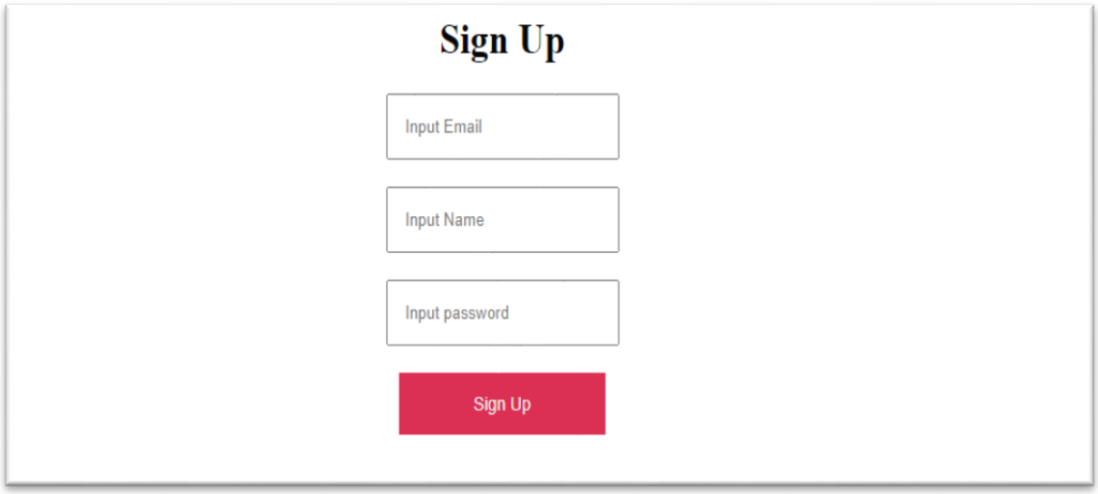


Рисунок 4.5 - Сторінка реєстрації користувача

Щоб успішно зареєструватись, необхідно заповнити всі поля, такі як:

- ‘Email’ пошта користувача.
- ‘Name’ ім’я користувача, який реєструється.
- ‘password’ пароль.

Якщо виникли якісь проблеми, при запису в поле вводу, буде перенесення на іншу сторінку, де можна побачити описану проблему, приклад можна побачити нижче на рис. 4.6. Тому потрібно повернутись, та редагувати запис.

Рисунок 4.6 - Невірний запис пошти

Після правильного заповнення всіх даних полів, потрібно натиснути кнопку «Sign up», звідки ми потрапимо на головну сторінку, приклад показано на рис. 4.7.

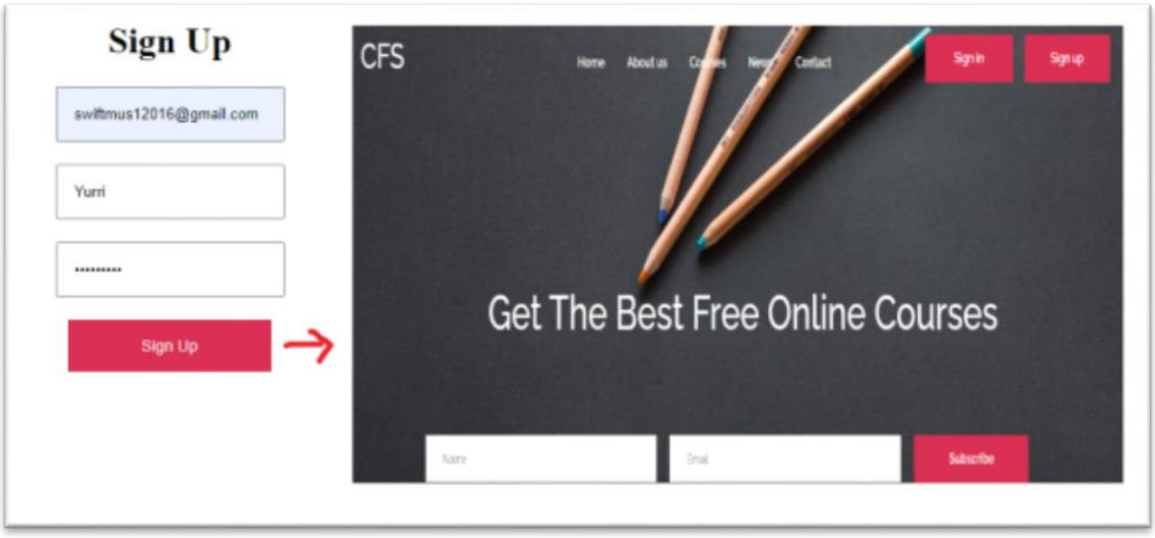


Рисунок 4.7 - Успішна реєстрація

Щоб увійти у профіль, потрібно натиснути «Sign in» та ввести дані, які вводили при реєстрації, приклад на рис. 4.8.

Рисунок 4.8 - Вхід у профіль

Після вводу даних та успішного входу, ми з’являємось на сторінці з доступними всім курсами, приклад наведений нижче на рис. 4.9.

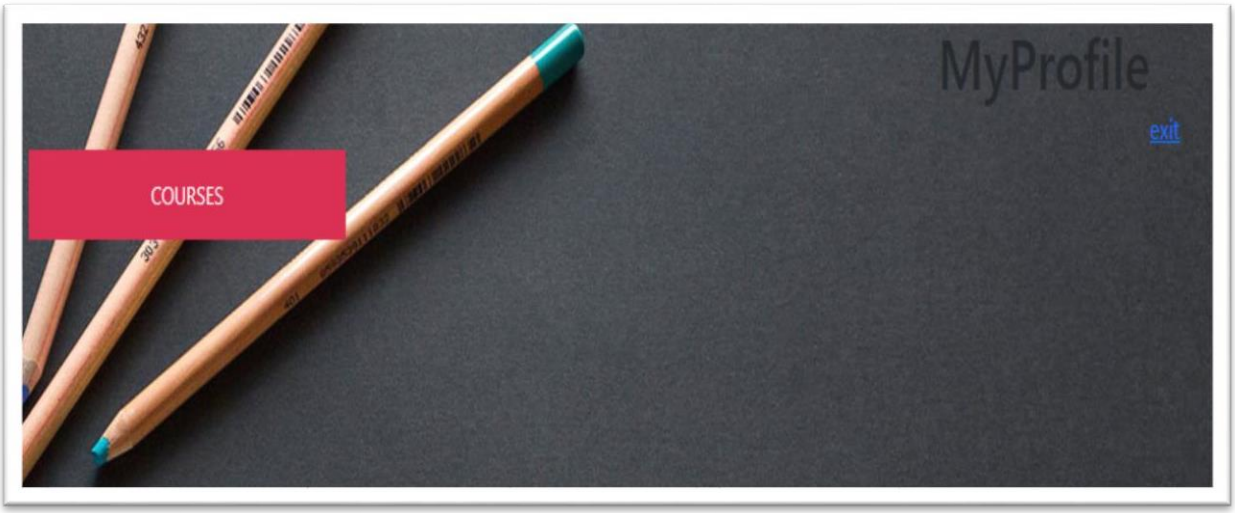


Рисунок 4.9 – Сторінка з доступними курсами

Щоб вибрати доступний курс для його проходження, необхідно навести мишкою на «Courses», де з’явиться випадаюче меню, яке можна побачити на рис. 4.10.

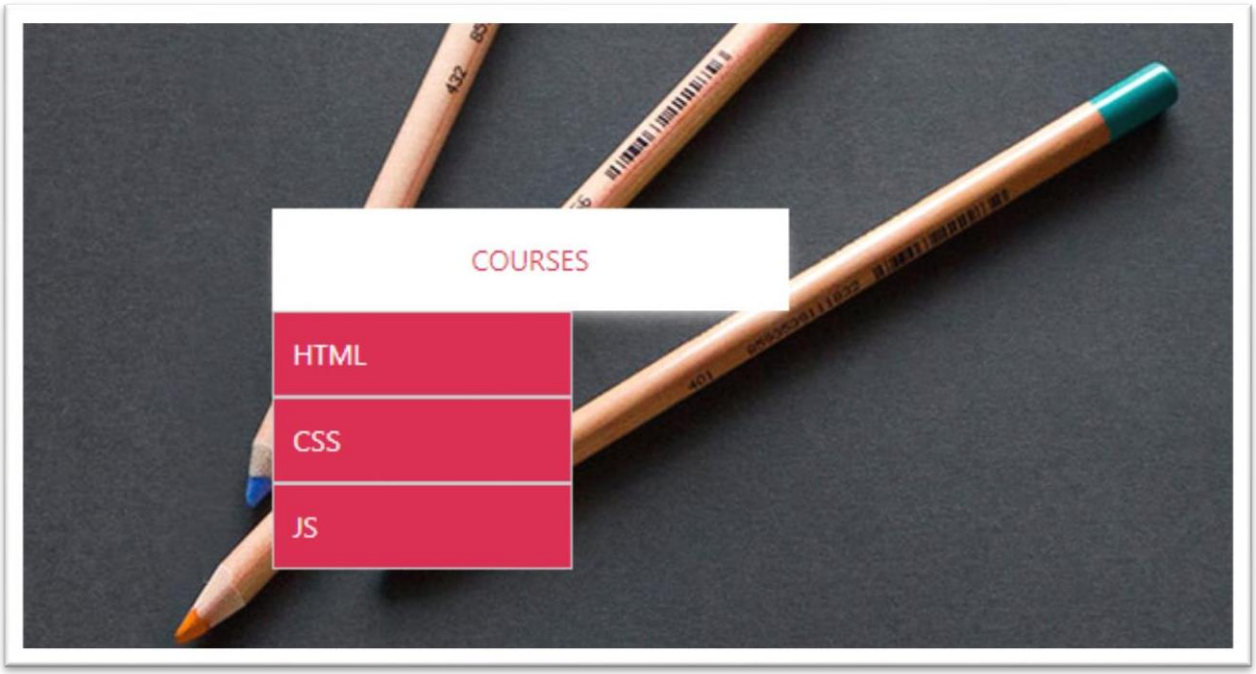


Рисунок 4.10 - Випадаюче меню доступних курсів

Далі вибираємо курс, у якому доступно перегляд відео, освітні матеріали та тести для засвоєння знань. Приклад на рис. 4.11

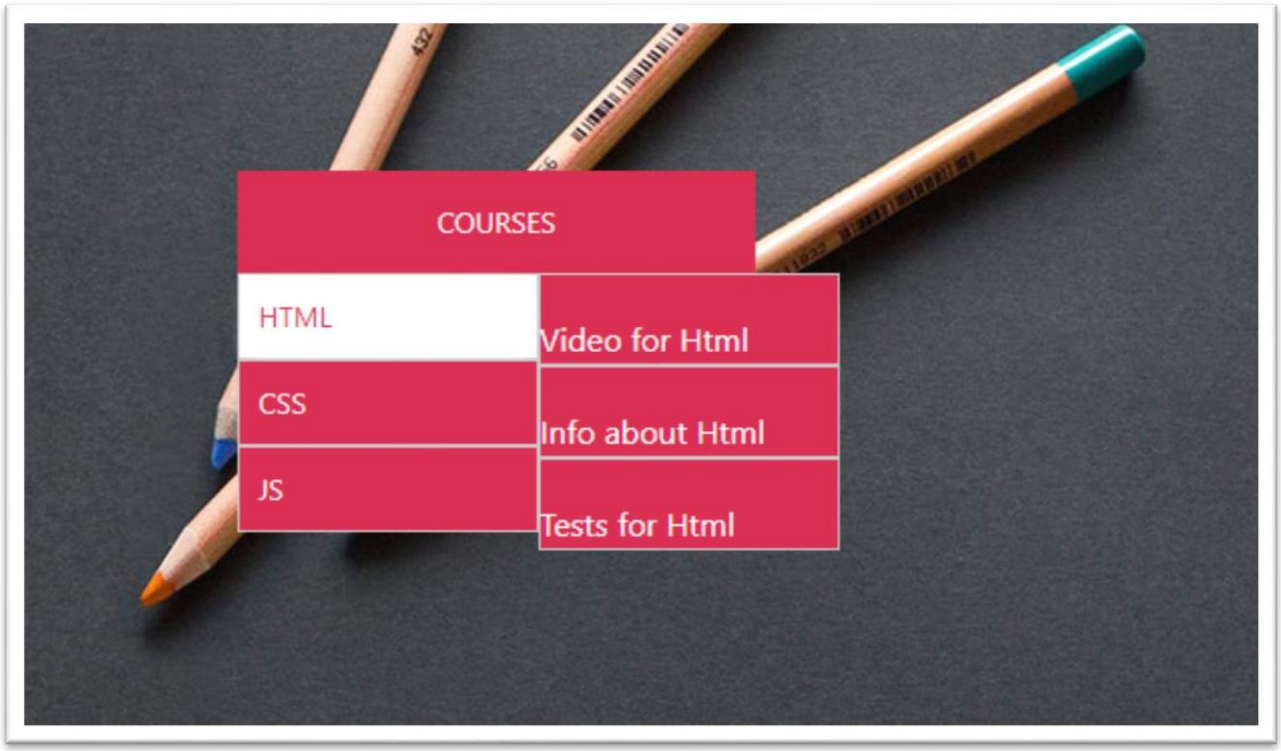


Рисунок 4.11 - Матеріали курсу

Для того, щоб вибрати відео потрібно натиснути на «Video for Html»,для перегляду. Приклад на рис. 4.12

Рисунок 4.12 - Відео урок по HTML

Щоб ознайомитись з матеріалами по курсу HTML, потрібно натиснути на «info about Html», приклад на рис. 4.13

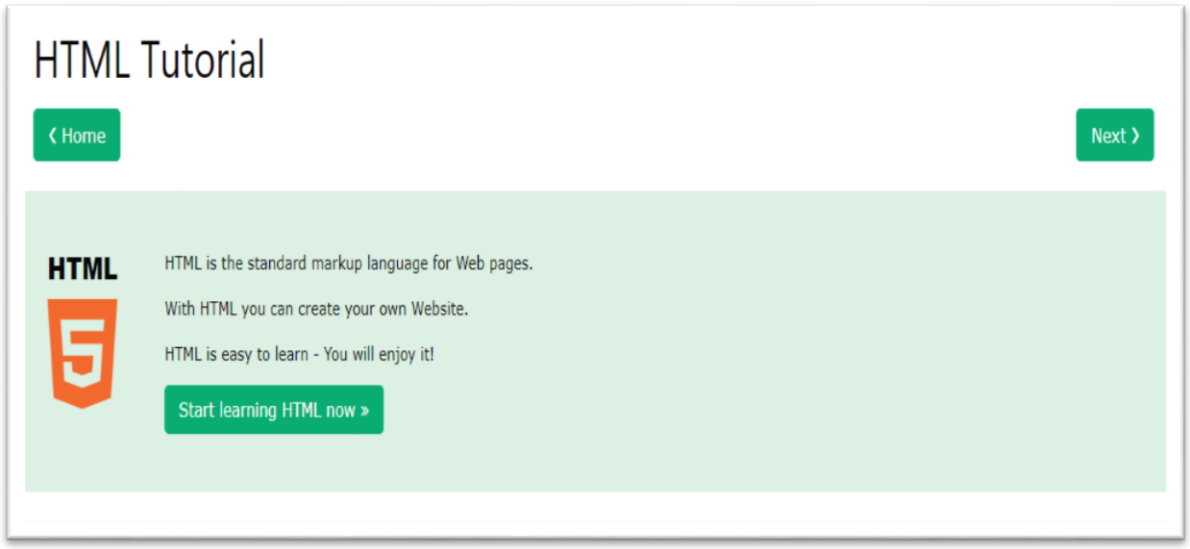


Рисунок 4.13 - Ознайомлення з курсом по HTML

Далі можна пройти тести, для цього потрібно натиснути на «Tests for Html», приклад на рис. 4.14

Рисунок 4.14 - Тести для засвоєння матеріалу по курсу HTML

4.2 План тестування створеної веб-платформи

Створено тест-кейс, у якому показано тестування функціональності створеної веб-платформи, зображення показано у Таблиці 1.

План тестування:

1. Запуск програми.
2. Перехід у браузері за посиланням «localhost».
3. Відображення головної сторінки.
4. Перехід на форму реєстрації.
5. Вхід у профіль.
6. Вибір доступного курсу.

Таблиця 1. Тест-кейс тестування

№	Описування	Сподіваний результат
1	Запуск програмного коду.	Успішний запуск та перехід до роботи у браузері.
2	Перехід у браузері за посиланням «localhost» огляд головної сторінки.	Успішний перехід на головну сторінку платформи.
3	Перехід на форму реєстрації.	Успішний перехід на форму реєстрації.
4	Заповнення всіх полів даними.	Ввід правильних даних.
5	Перехід на форму входу, там коректний ввід даних.	Успішний вхід у профіль.
6	Вибір доступних курсів.	Успішний перехід до матеріалів обраного курсу.

4.3 Погляди на рахунок покращення веб-платформи

Для ефективності подальшого навчання, необхідно покращити функціональність для веб-платформи. Потрібно збільшити кількість доступних курсів та матеріалів, а також організувати пошук по базі даних. Необхідно довести до кінця зовнішній вигляд онлайн ресурсу.

У даному розділі було показано створену веб-платформу для вивчення онлайн курсів, яка розроблялась продовж виконання магістерської роботи. Було розглянуто реалізацію та тестування даного програмного продукту. Детально на прикладах був досліджений функціонал веб-платформи.

Було проведено тестування, яке пройшло успішно. У таблиці описаний тест-кейс, який був запропонований для легкого та коректного тестування.

Також були написані погляди на рахунок покращення веб-платформи.

ВИСНОВКИ

Метою дипломного проекту було створення веб-платформи для вивчення онлайн курсів.

В першому розділі магістерської роботи було розглянуто теоретичні основи створення веб-сервісів, теорію розробки та інструменти веб-серверів.

У другому розділі було здійснено обстеження сучасних технологій та архітектурних методів програмного забезпечення, які використовують для реалізації представленої в роботі програми. Також була описана архітектура веб-платформи.

Для розробки веб-додатку для вивчення онлайн курсів, було вибрано технології, які призначенні для створення інтернет ресурсу та легко справляються з поставленими задачами. Для створення зовнішнього вигляду, було використано такі технології програмування як: JS, HTML, CSS, а для внутрішньої частини було використано: PHP.

Одним з найголовніших технологій було представлено та обрано базу даних MySql. Вона має у собі всі сучасні методи зберігання даних та їх обробки.

У третьому розділі було розглянуто головні етапи та методи для реалізації веб-платформи, які були детально описані. До того ж розглянуто базу даних MySql, яка застосована у розробці продукту. Програма використовує клієнт-серверну модель, що у декілька разів покращує створення і модернізацію веб- ресурсу.

Що стосується функціоналу веб-платформи, то вона має просту структуру, у якій є можливість реєстрації та доступ до даних курсів. Описано втілення клієнт-серверної моделі у продукт, досліджено безліч переваг цієї архітектури та локальний веб-сервер на якому працює веб-платформа.

Також був проведений аналіз методів розробки за допомогою яких, було створено веб-продукт.

В четвертому розділі було представлення функціональності та тестування програми, яка місить у собі інтерфейс, реєстрацію та авторизацію, а також вибір даного курсу, у якому можна переглядати відео, читати матеріали, та проходити тести

для засвоєння інформації.

У даному розділі було показано створену веб-платформу для вивчення онлайн курсів, яка розроблялась продовж виконання магістерської роботи. Було розглянуто реалізацію та тестування даного програмного продукту. Детально на прикладах був досліджений функціонал веб-платформи.

Було проведено тестування, яке пройшло успішно. У таблиці описаний тест-кейс, який був запропонований для легкого та коректного тестування.

Також були написані погляди на рахунок покращення веб-платформи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Хайці Лян, Вей Сун, Сінь Чжан, Чжунбо Цзян "Структура політики для спільного налаштування веб-сервісів" SOSE'06, 2016

2. Н. Дх'янеш, Г.К. Вініел, С.В. Рагхаван "DEVISE: Методологія побудови інфраструктури на основі веб-сервісів для спільних підприємств" WETICE'03, 2018

3. Ірфан Аван, Мухаммад Юнас "Диференційоване трактування спільних додатків на основі веб-сервісів" 9-та Міжнародна конференція з комп'ютерної кооперативної роботи в проектуванні

4. Джеймс Д. Гербслеб "Глобальна програмна інженерія: майбутнє соціотехнічної координації" FOSE'17, 2017

5. К. Сельчук, Вень-Сянь Лі, Томас Фан, Мінці Чжоу "Кордони в інформації та програмному забезпеченні як послуги" ICDE 2019

6. Джіоті Намджоші, Арчана Гупте, "Сервісно-орієнтована архітектура для хмарного програмного забезпечення для бронювання подорожей як послуга" Хмара 2021

7. Відьянанд Чоудхарі, "Програмне забезпечення як послуга: наслідки для інвестицій у розробку програмного забезпечення" HICSS'17, 2017

8. Сун Го, Фань Бай, Сяолінь Сюй, "Програмне забезпечення для моделювання як послуга та послуга- Орієнтований імітаційний експеримент» IRI 2021

9. Еспадас Хав'єр, Девід, Моліна Артуро "Розробка додатків на платформах програмного забезпечення як послуги" ICSEA 2022

10. Zhengxiong Hou та ін "ASAAS: Прикладне програмне забезпечення як послуга для високопродуктивних обчислень" HPCC 2016

11. General Architecture of Computer [Електронний ресурс] — Режим доступу: https://www.techantena.com

12. Service-oriented architecture (SOA) [Електронний ресурс] — Режим доступу: https://subscription.packtpub.com

13. Multilevel architecture [Електронний ресурс] — Режим доступу: https://pt.slideshare.net/attrimahesh/multilevel architecture structuredorg/7

14. Microservices Architecture and Design Patterns [Електронний ресурс] — Режим доступу: https://www.xenonstack.com/insights/microservices/

15. Monolithic architecture [Електронний ресурс] — Режим доступу: https://whatis.techtarget.com/

16. Architectural characteristics of web-based applications [Електронний ресурс] — Режим доступу: https www ibm com

17. What is CSS? [Електронний ресурс] — Режим доступу: https://www.tutorialspoint.com/css/what_is_css.htm

18. What is Bootstrap? [Електронний ресурс] — Режим доступу: https://www.yogihosting.com/what is bootstrap/

19. Everything You Need To Know About PHP [Електронний ресурс] — Режим доступу: https://skillcrush.com/blog/php/

20. Software Architecture [Електронний ресурс] — Режим доступу: https://www.castsoftware.com

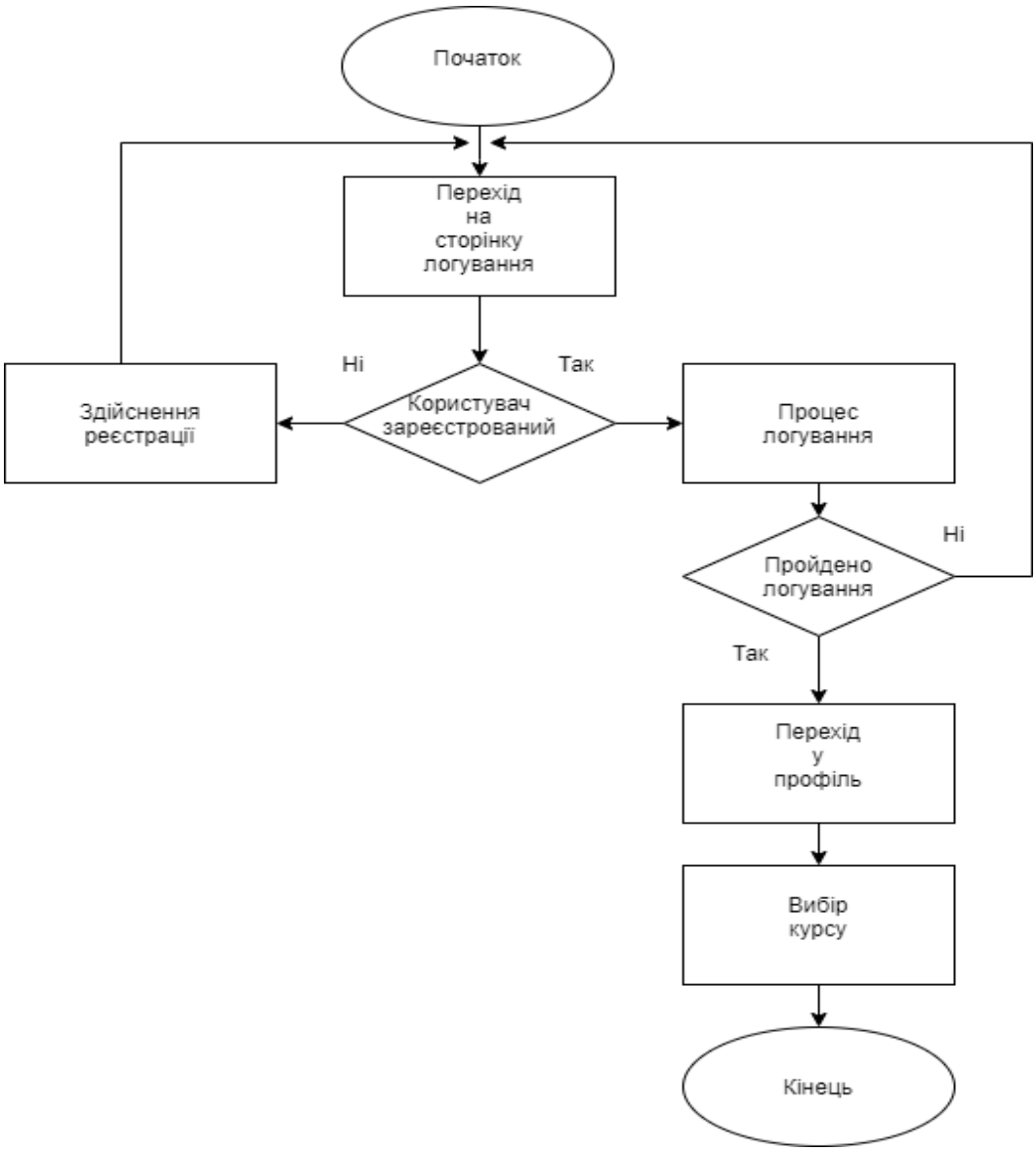
21. Intellij IDEA [Електронний ресурс] — Режим доступу: https://www.techopedia.com

22. What is Atom? [Електронний ресурс] — Режим доступу: https://medium.com

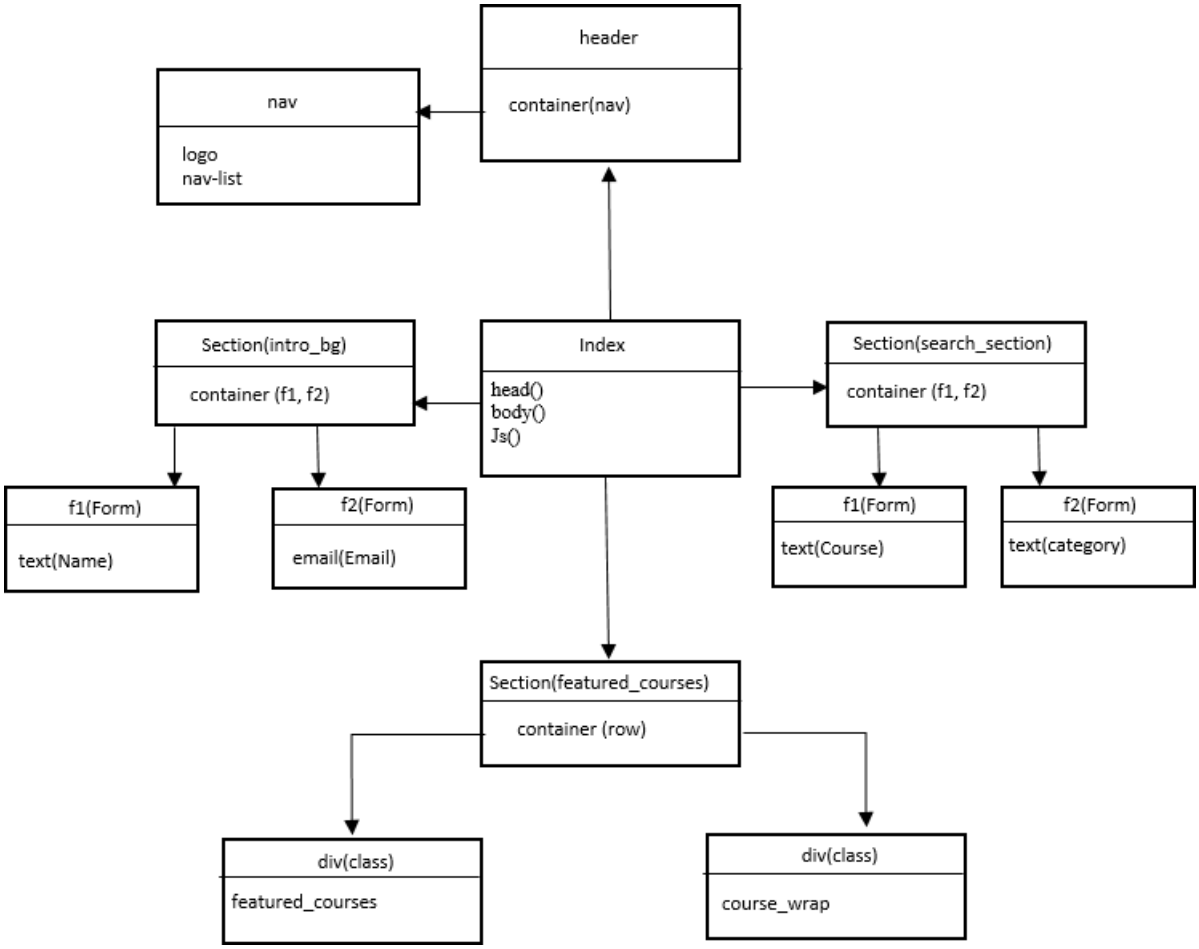
23. An Introduction to MAMP [Електронний ресурс] — Режим доступу: https://kinsta.com/knowledgebase/mamp not starting/

24. What is MySQL? [Електронний ресурс] — Режим доступу: https://www.digitalocean.com/community/tutorials /what is mysql

Додаток 1 - Алгоритм дій



Додаток 2 - Діаграма класів



Додаток 3 - Структурна схема веб застосунку