ЗМІСТ

ВСТУП

В наш час технології розвиваються неймовірно швидко, а з розвитком технологій росте і попит на кваліфікованих працівників та освідчених випускників, які зможуть виконувати важкі завдання та вирішувати поставлені задачі. Університети надають потрібні знання, але для студента цього може бути недостатньо, адже більшість матеріалу дається на самоопрацювання. Існує багато джерел інформації, з яких всі бажаючі можуть брати необхідні матеріали: книги, научні статті, для студентів – лекції в університеті. Але найбільш зручними на мою думку є платформи, на яких учень може не тільки вивчити новий матеріал, але й перевірити та засвоїти його.

Мета роботи. Проаналізувати дані для побудови серверної частини автоматизованої розподіленої системи навчання та тестування, враховуючи переваги та недоліки уже існуючих систем. Даний додаток буде надавати можливість всім бажаючим вивчати саме те, що їм потрібно. А тестові завдання матимуть спеціальний алгоритмом для кращого оцінювання користувачів.

Для досягнення поставленої мети були визначені наступні завдання:

- провести пошук та проаналізувати уже існуючі платформи та системи;
- Провести порівняльну характеристику даних систем, а також виділити їх основні переваги та недоліки;
- провести агрегацію переваг в створюваний проект;
- побудувати теоретичний базис для реалізації власного рішення.

Актуальність. В сучасних умовах надзвичайно швидкого розвитку інформаційних технологій актуальність створення систем навчання і тестування є надзвичайно високою. Їх вплив на сучасну освіту настільки

великий, що майже кожен зіштовхувався з необхідність пошуку інформації в мережі інтернет. Саме тому питання створення систем навчання та тестування повинно вирішуватися, виходячи з потреб користувачів, тобто людей, які мають бажання навчатися. За останні 5 років системи подібного роду дуже швидко набрали свою популярність, так, як це набагато легше та швидше, ніж шукати інформацію самому. Інформація в них зібрана та класифікована, а система тестування працює ефективніше, оскільки зменшує ймовірність похибки та являється більш гнучкою ніж звичайний принцип тестування. Даного роду система дозволить кожному користувачеві навчатись прямо вдома, не витрачаючи час та ресурси на пошук інформації.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА ІСНУЮЧИХ РІШЕНЬ.

АНОТАЦІЯ

В даному розділі було розглянуто перелік готових рішень різноманітних ресурсів, їх порівняння, а також аналіз їх переваг та недоліків. Проведена агрегація існуючих переваг та поставлена основна проблема з методами її вирішення. Був побудований теоретичний базис та розроблена загальна концепція для вирішення завдання побудови автоматизованої розподіленої системи навчання та тестування.

1.1 Аналіз та порівняння готових рішень.

На даний момент існує величезна кількість сервісів та веб-застосунків як дозволяють вам пройти певне опитування. Всі вони показують результат, деякі з них можуть показати наскільки цей результат кращий чи гірший ніж в інших користувачів і ще менше сервісів дозволяють поповнити свої знання по темі даного опитування. В даномі підрозділі будуть проаналізовані на мою думку найкращі сервіси даного формату. А також будуть виділені їх основні переваги та недоліки.

1.1.1 Платформа для підготовки до олімпіад з інформатики та програмування

Насамперед був розглянутий відомий російський сервіс з підготовки до інформатики та олімпіад з програмування[1]. На даному веб-сайті присутні велика кількість різноманітних статей та задач на велику кількість тем з програмування. Він містить величезну кількість ресурсів які зручно відсортовані за темами та категоріями. Також містить велику базу задач з минулих олімпіад. Вигляд веб-сервісу показано на рисунку 1.1. Структура веб-сервісу для кращого розуміння роботи показана на рисунку 1.2.

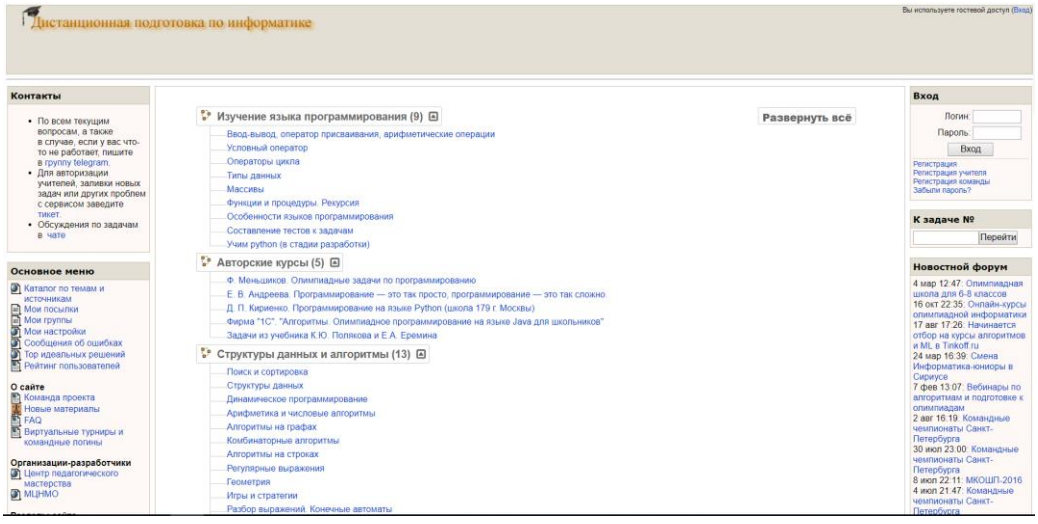

Рисунок 1.1 - Загальний вигляд веб-сервісу

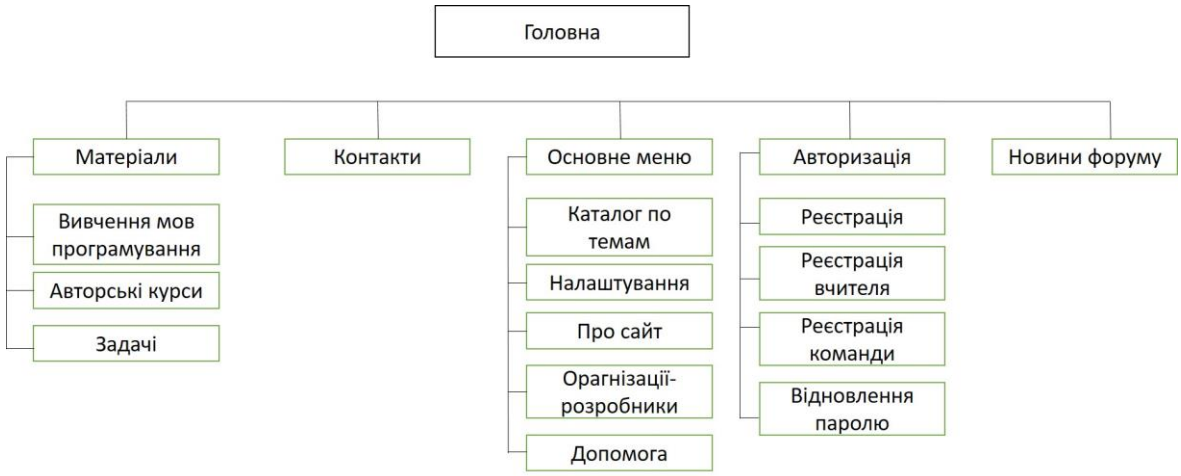

Рисунок 1.2 – Структурна схема веб-сервісу

До переваг можна віднести:

- можливість прочитати велику кількість текстів на різні теми для різних мов програмування, а також вирішити завдання пов’язані з ними;
- достатня кількість матеріалів для підготовки для певного завдання, для чого і призначений додаток, а також зручне сортування по темам;
- вчителі можуть публікувати свої статті та задачі, які учні можуть виконувати ;
- можливість обговорити дані задачі, але ця можливість не реалізована на самому сервісі, для неї потрібно встановити сторонній додаток для листування між користувачами. На мою думку дану функцію потрібно реалізувати на кожному розділі подібного роду сервісів з окремими гілками для кожного питання (аналог Stackoverflow архітектури), що вже можна віднести до недоліків.

Також до недоліків можна віднести:

- вузьку направленість, задачі, для оцінювання якості яких потрібен викладач, тобто людський ресурс, система не може бути автоматизована;

- неможливість порівняти результат з іншими учасниками;
- статті та тести можуть публікуватися тільки викладачами, для чого потрібна більш складна реєстрація, а також перевірка адміністрацією;
- дизайн не може задовільнити потреби користувачів. Цей, на перший погляд, недолік може відвернути від постійного користування сервісом багатьох можливих відвідувачів, оскільки на даний момент діджеталізація світу збільшила апетити сучасних користувачів до зручного UX та UI дизайну.

1.1.2 Платформа з підготовки та перевірки знань англійської мови *«examenglish.com»*

Далі була розглянута відома платформа для всіх бажаючих, на основі якої можна підготуватись до міжнародних екзаменів з англійської мови, таких, як IELTS, TOEFL, TOEIC, CAMBRIDGE та інших[2]. Даний сервіс працює тільки на англійській мові та не має можливості змінити мову, тому для доступу потрібно знання хоча б мінімуму англійської мови. На ньому можна визначити свій рівень володіння мовою та отримати рекомендації з подальшого вивчення та більш глибокого розуміння матеріалу. Також на даному сервісі доступний перелік усіх особливостей здачі наявних тестів, що дуже допомагає при їх здачі. На рисунку 1.3 показаний загальний вигляд веб-сервісу. А на рисунку 1.4 – структурна схема його роботи.

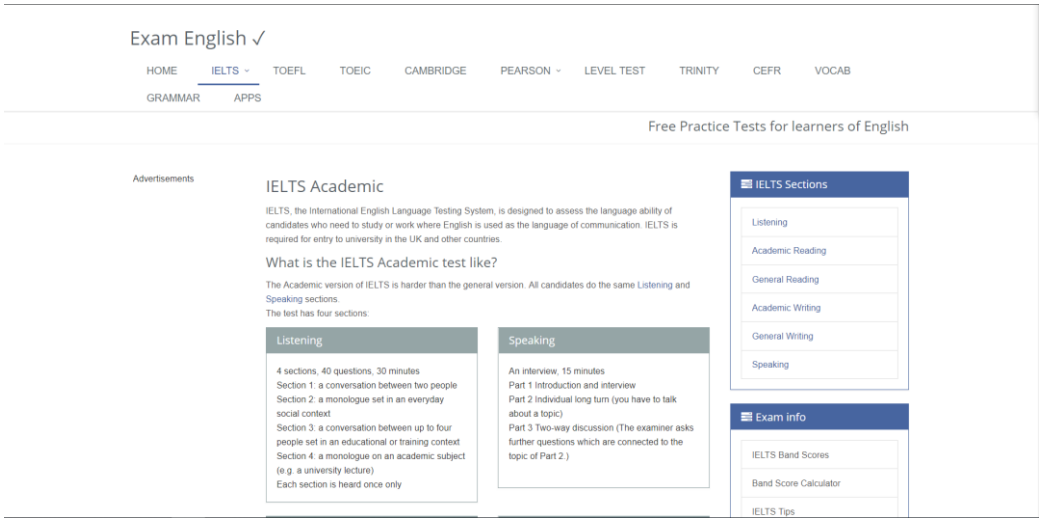

Рисунок 1.3 - Загальний вигляд веб-сервісу

Рисунок 1.4 - Структурна схема веб-сервісу

Перевагами даної платформи є:

- можливість оцінювання рівня знань матеріалу, що надає можливість новому користувачу не витрачати час на здобуття уже відомих йому навичок та проходження тестів, які йому не потрібні;
- при оцінюванні рівня знань не використовується людський фактор а вся система працює автоматизовано, що зменшує можливість похибки при підрахунку та перевірці результатів.

До недоліків можна віднести:

- недостатню кількість унікальних питань зі специфічних для англійської мови структур та правил;

- неможливість повністю оцінити рівень володіння розмовними навичками та їх покращення в подальшому.

1.1.3 Платформа з вивчення та тестування різноманітних навичок у сфері розробки програмного забезпечення, їх підтримки, захисту та засобів для створення візуального контенту *pluralsight.com*

Дана платформа добре підійде як для новачків у сфері *IT,* так і для людей, які мають певні навички[3]. Вона не є вузькоспеціалізованою, як згадані вище сервіси, що дозволяє оволодіти шикоким спектром знать та підібрати та вивчати саме ті технології, які вам потрібно і які актуальні на теперішній час, наприклад якщо користувачу потрібні навички в області веб-розробки, за допомогою зручних фільтрів та розподілу по зручно відсортованих темах він може перейти до відповідних розділів та обрати саме ту технологію, яка буде задовільняти його потреби. Якщо ж після деякого часу користувач зацікавиться технологіями просування свого продукту, сервіс надає можливість вивчати засоби, які надаюсь таку можливість. Також платформа дозволяє оцінити свій рівень знань відповідно до рівня *Junior, Middle, Senior,* і т.д. Це дозволяє розпочати вивчення матеріалу з потрібної точки та не витрачати часу на повторення уже вивченого. Також унікальна система тестування дозволяє побачити результати навчання у реальному часі відносно інших користувачів. На рисунку 1.5 зображено загальний вигляд веб-додатку. А на рисунку 1.6 – його структурна схема.

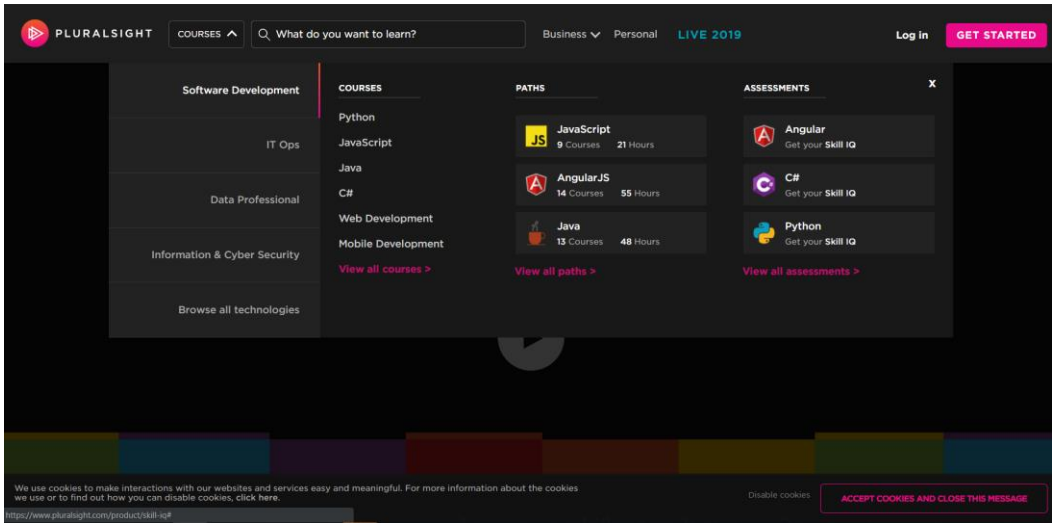

Рисунок 1.5 – Загальний вигляд веб-сервісу

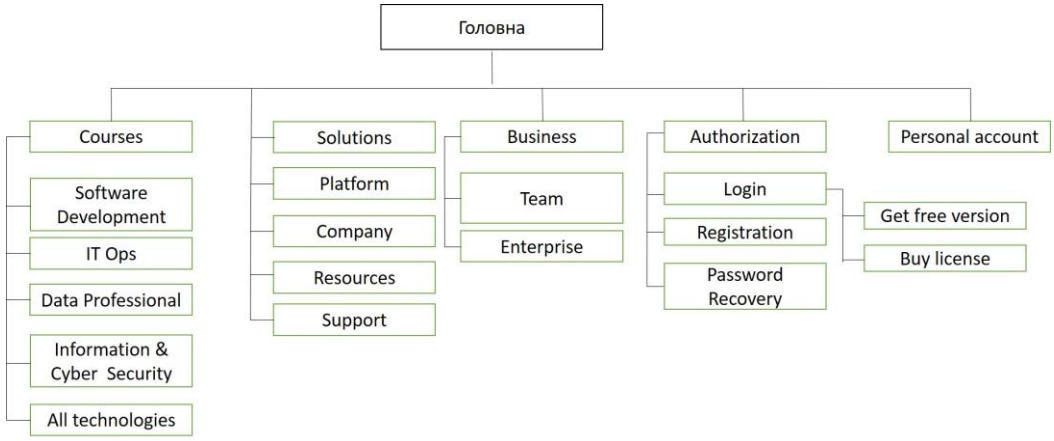

Рисунок 1.6 – Структурна схема веб-сервісу

З переваг цього сервісу можна виділити:

- велику кількість тем для вивчення матеріалу;
- унікальні тести, при проходженні яких закріплюються знання, здобуті при вивченні відповідних статей;
- за допомогою поля для пошуку можна відфільтрувати розділи, які вас цікавлять;
- можливість дуже зручно переходити по сторінках сайту;
- окрему увагу можна приділити UX/UI дизайну цього сервісу, який дозволяє швидко та без проблем знайти бажаний

матеріал і розпочати вивчення необхідних користувачу технологій;

- при проходженні тесту користувач має можливість побачити порівняння свого результату з результатами інших користувачів;
- можливість користування даним сервісом протягом 30-денного безкоштовного періоду, що дозволяє оцінити всі переваги та недоліки даної платформи, завчасно не витрачаючи коштів на придбання повної версії.

Але платну підписку скоріше можна віднести до недоліків, решта яких будуть розглянуті нижче:

- достатньо висока ціна у порівнянні з іншими платформами-конкурентами;
- для користування потрібне знання англійської мови технічного спрямування;
- неможливість оцінити повний функціонал веб-додатку без попередньої реєстрації;
- відсутність можливості побачити порівняння своїх результатів з членами локальної групи (країна, місто чи ВНЗ).

1.1.4 Веб-додаток для вивчення та поглиблення знань з програмування *metanit.com*

Даний ресурс присвячений різним мовам і технологіям програмування, комп'ютерам, мобільним платформам і IT-технологіям[4]. Тут публікуються різні керівництва і навчальні матеріали, статті та приклади, а також є можливість виконати практичні завдання на основі вивченого.

Пріоритетні напрямки навчання - мова *C#* і сімейство технологій *.NET* технології на базі Java, Python, робота з базами даних а також WEB-

технології, такі як HTML5, AJAX, jQuery, Node.js, Angular, React і ін. На рисунку 1.7 показано загальний вигляд веб-додатку. А на рисунку 1.8 – його структурна схема.

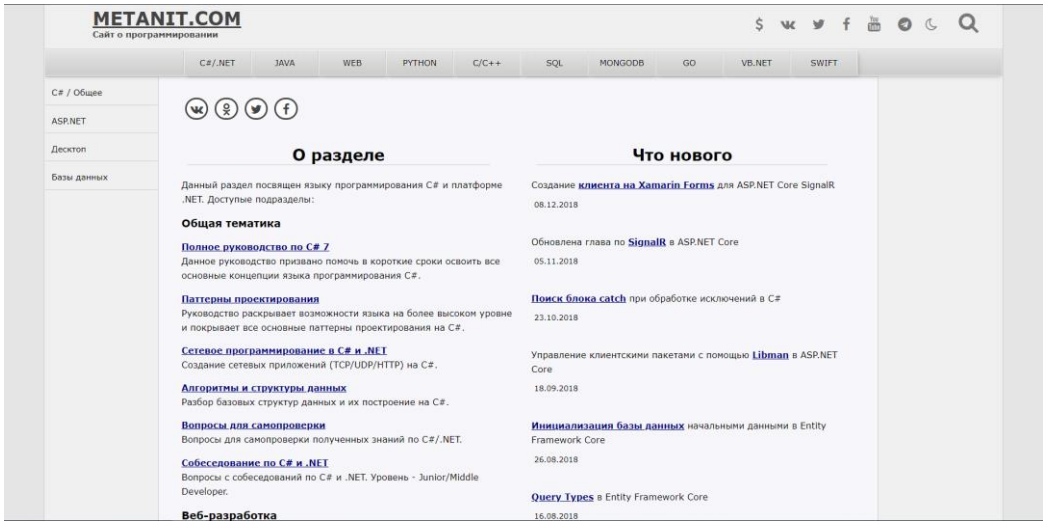

Рисунок 1.7 – загальний вигляд веб-сервісу

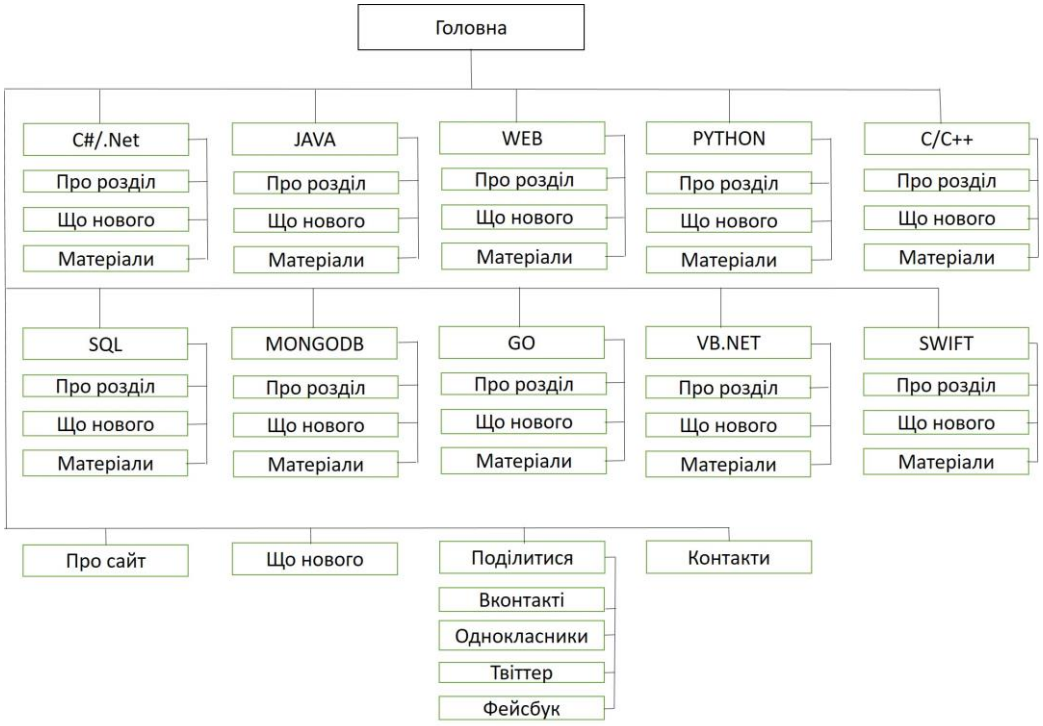

Рисунок 1.8 – Структурна схема веб-сервісу

Порівняно з вищерозглянутим сервісом *PLURALSIGHT.COM* даний ресурс є безкоштовним, хоча і не містить таку велику кількість тем для поглибленого вивчення.

Також він має такі переваги, як:

- користувачі мають змогу обговорити кожну статтю та задати питання напряму автору;
- реєстрація не є обов'язковою і тільки надає можливість спілкуватися, обговорювати та задавати питання іншим користувачам, що не заважає використовувати повний функціонал даного сервісу;
- зручна навігація між розділами;
- можливість використовувати без з'єднання з мережею, завантаживши *PDF*-версію .

Але дана платформа має й недоліки:

- інформація недостатньо швидко оновлюється й інколи є проблема з неактуальністю матеріалу;
- деякі теми не мають достатнього пояснення, що не сприяє поглибленому розумінню технологій;

Висновок:

В даному підрозділі було проаналізовано декілька існуючих рішень і виділено основні переваги і недоліки, які у подальшому можуть бути використані для вирішення поставленої задачі та уникнення власне самих недоліків. Також були проаналізовані структури роботи веб-сайтів та аналіз алгоритмів їх роботи і побудовані схеми загального вигляду. Вони також можуть бути використані для побудови та планування власного продукту.

1.2 Постановка задачі

Проведений в попередньому підрозділі аналіз існуючих рішень дозволяє виділити найбільш вагому переваги кожного з них та врахувати їх при побудові власного рішення. А також показує, що даного роду системи є розповсюджені і необхідні для людей, які займаються самонавчанням, тому власне розробка даного проекту є актуальною.

Такі платформи надають можливість вивчати необхідні матеріали, а також показати свій результат на основі пройденого тесту. В розроблюваному додатку результат може відображатись як абсолютний (відсоток правильних відповідей від кількості загальних питань) і відносний (на основі результатів інших користувачів). Також відносний результат може відображатись в рамках ВНЗ, міста чи навіть країни, або на основі результату всіх користувачів. При обрахуванні результату можна використати систему з графом, вітки якого матимуть різну вагу, що дозволяє вирішити проблему, що деякі питання можуть бути важчі ніж інші, тому потрібно надати певну вагу кожному питанню що зробити систему оцінювання більш чесною[5].

Аналіз переваг та недоліків дозволив виділити найбільш важливі та зробити висновки такого роду. Необхідно:

- Для повного функціонування додатку зберігати велику базу інформації, а також розділити її на теми та підтеми для зручного пошуку та доступу до них
- надати можливість користувачам публікувати власні теми та тести до них, що передбачає відкритість платформи
- надати можливість обговорення для статей
- Автоматизувати систему таким чином, щоб оцінювання проходило без участі людського ресурсу
- Створити дизайн який буде задовільняти сучасних користувачів та допомагати залучати нових

- Реалізувати зручний перехід по сайту
- Зробити систему безкоштовною, реалізувати підтримку платформи за рахунок користувачів

Щоб розробити даного роду додаток нам необхідно:

- розробити архітектуру проекту;
- розробити агрегатну сутність в СУБД для зберігання інформації;
- розробити веб-представлення ;
- реалізувати алгоритм для системи оцінювання на основі графа
- протестувати отриманий додаток.

ВИСНОВКИ ДО РОЗДІЛУ 1

В ході розробки даного розділу було проаналізовано дані для побудови серверної частини автоматизованої розподіленої системи навчання та тестування враховуючи переваги та недоліки уже існуючих систем. Було проведено пошук та аналіз уже існуючих платформ та систем, порівняльну характеристику даних систем, а також виділено їх основні переваги та недоліки. Провівши агрегацію переваг та врахувавши недоліки був побудований теоретичний базис для реалізації власного рішення. Також було виділено основні аспекти побудови власного рішення та складено план його розробки.

РОЗДІЛ 2

АНАЛІЗ ТА ВИБІР ТЕХНІЧНИХ РІШЕНЬ

У даному розділі буде проаналізовано основні технічні рішення та буде проведено їх порівняння та вибір найбільш підходящих. Спочатку буде розглянуто загальну технологію роботи розроблюваного веб-застосунку, далі основні технології, які будуть використані. Також буде проведено порівняння та вибір серверів та буде вибрано підходящий для даного проекту. Також однією з технологій, які необхідно вибрати є база даних та організація взаємодії веб-серверів.

2.1 Основні поняття

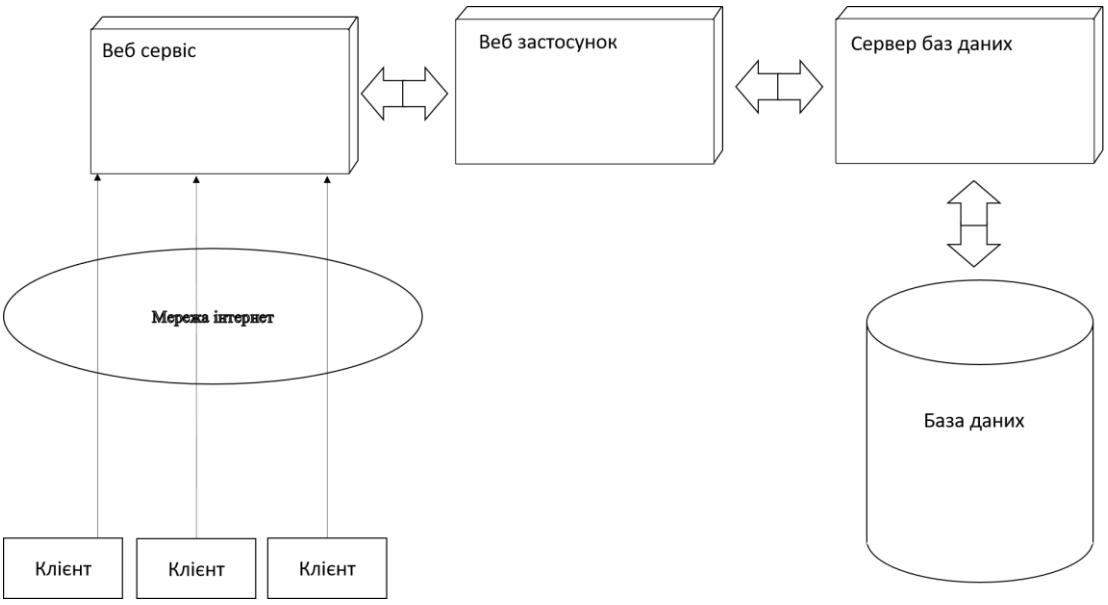

Рисунок 2.1 - Загальна технологія роботи веб орієнтованого застосування

На рисунку 2.1 відображено загальну технологію роботи веб орієнтованого застосування. Як видно з наведеного рисунку дана архітектура включає в себе три сервери і безпосередньо базу даних, к джерело інформації для поточної системи. Розглянемо кожен сервер та між серверну взаємодію детальніше. Оскільки мова йде про веб орієнтоване застосування формат комунікації клієнтів та веб серверу

завідома відомий і являє собою взаємодію по HTTP протоколу. Після відправки клієнтом запиту через мережу інтернет запит буде отримано веб сервером. Веб – сервер – частина архітектури, що відповідає за отримання та обробку запитів від клієнтів і їх подальшу маршрутизацію до серверу чи серверів застосувань, в рамках одно сервісної чи сервісно-орієнтованої архітектури, яка включає в себе оркестрацію набору серверів. Сервер веб застосувань отримує відповідний запит обробляє його та маршрутизує тіло запиту до відповідного застосунку. Застосунок звертається до джерела даних та взаємодіє через сервер застосувань з тим чи іншим сервером баз даних відповідно до підтримуваного ним протоколу. Сервер баз даних відповідно до запиту, маршрутизує його тіло до конкретної бази даних виконує його та повертає результат виконання серверу застосунків. Під компоненти дані на рисунку 2.1 на сервері можуть бути присутні один або кілька додатків. В останньому випадку буде присутня сервісно-орієнтована архітектура.

2.1.1 SOA - Сервіс-орієнтована архітектура

В даній роботі буде використано зовнішній сервіс як джерело даних, тому необхідно розглянути загальну сервісно орієнтовану архітектуру.

SOA - Сервіс-орієнтована архітектура - це набір архітектурних принципів, що не залежать від технологій і продуктів, як поліморфізм або інкапсуляція. Також її можна описати як модульний підхід до розробки програмного забезпечення, заснований на використанні розподілених, слабко пов'язаних замінних компонент, які оснащені стандартизованими інтерфейсами для взаємодії з протоколами.[6]

Основні принципи SOA:

- Стандартизований метод комунікації - сервіси підпорядковуються стандарту комунікації, який прийнятий колективно.
- Слабкий зв'язок – в разі виходу з роботи одного з сервісів, це не вплине на інші, так, як вони не залежать один від одного.
- Повторне використання сервісів – з ціллю повторного використання логіка розподілена по сервісах.
- Автономність - сервіс контролює логіку, яка інкапсульованв в ньому.

Більшість існуючих технологій та протоколів взаємодії сервісів побудовані на принципах SOA і не таким чином залежать від мови написання сервісу, надаючи користувачам можливість об'єднувати сервіси різних типів в розподілених системах. Даний підхід надає єдиний шаблон опису сервісів в якості універсальної сполучної ланки.

Структура SOA відображена на рисунку 2.2.

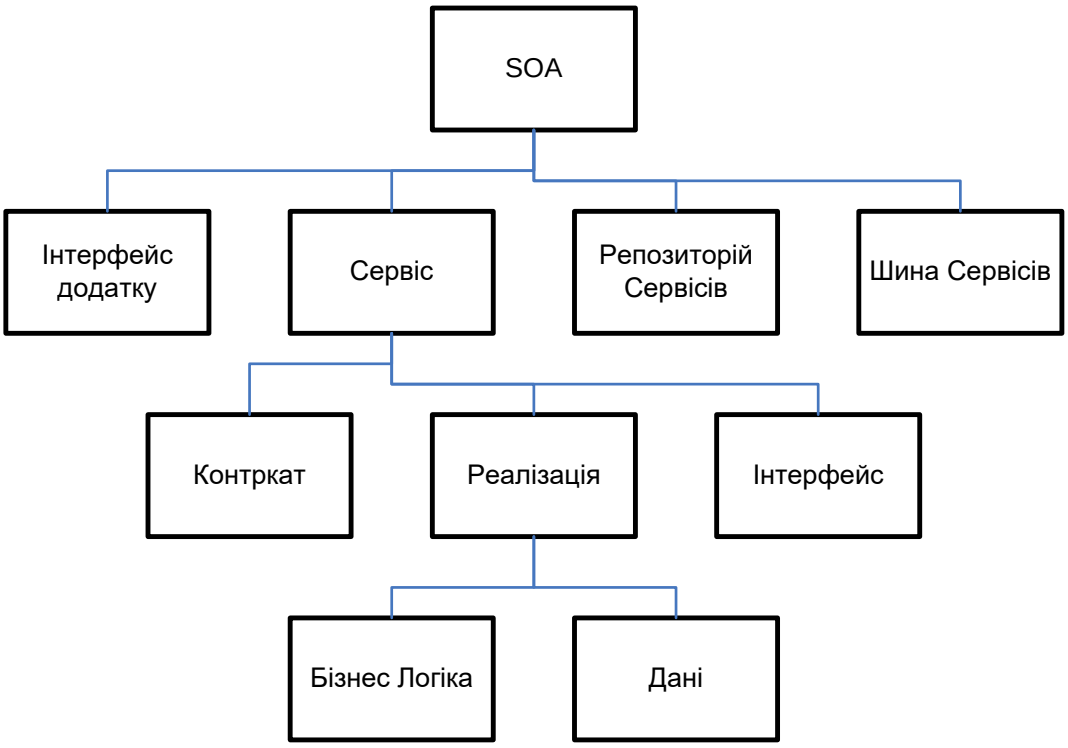

Рисунок 2.2 - струтктура SOA.

Веб-сервіси є концепцією створення таких додатків. Їх функції можна використати за допомогою стандартних протоколів мережі Інтернет, а концепція реалізується за допомогою ряду технологій, які стандартизовані World Wide Web Consortium (W3C).

Одним із варіантів реалізації компонентної архітектури є веб-сервіси.

А фундаментом для створення більшості технологій, пов'язаних з веб-сервісами є XML.

Згідно з визначенням W3C, веб-сервіси – це застосування, доступні за протоколами, які зазвичай використовуються для мережі Інтернет. Немає вимог, для використання якогось певного транспортного протоколу. Специфікація Simple Object Access Protocol(SOAP) використовується для віддаленої взаємодії з веб-сервісами. Вона забезпечує взаємодію розподілених систем, які не залежать від об1єктної моделі, ОС або засобу програмування. Дані передаються у вигляді XML документів спеціального формату. Існує декілька альтернативних рішень для взаємодії з даними веб-сервісами (CORBA, WCF та ін.). Така специфікація визначає, яким чином зв'язуються повідомлення SOAP і транспортний протокол.

Найбільш часто для передаці SOAP повідомлень вкористовується протокол HTTP. А в якості транспортного протоколу дуже поширеними є SMTP, FTP, TCP.

За визначенням W3C, "WSDL – це формат документу XML для опису мережних сервісів як набору певних операцій, що працюють за допомогою повідомлень, що містять документно-орієнтовану або процедурно-орієнтовану інформацію". Документ WSDL повністю описує взаємодію веб-сервісу з користувачами. WSDL файл – це документ у форматі XML, що описує методи, що надаються веб-сервісом, а також їх параметри, типи, назви та місцезнаходження Listener'a сервісу. По суті,

він надає можливість використовувати сервіси, створені на базі різних платформ[7].

WSDL містить всі деталі опису веб-сервісів, включаючи:

- URL веб-сервісу;
- Механізми комунікації, котрі розуміє веб-сервіс;
- Операції, які може виконувати веб-сервіс;
- Структура повідомлень веб-сервісу.

Приклад WSDL файлу:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="Guid"
targetNamespace="http://www. acbdf.com/Guid/"
xmlns:tns="http://www. acbdf.com/Guid/"
xmlns:soap="http://schemas. acbdf.com/wsdl/wsdl/"
xmlns:xsd="http://www.m2.org/2010/XMLScheme"
xmlns:soapenc="http://schemas.acbdf.com/ encoding/"
xmlnsгwsdl="http://schemas. acbdf.org/wsdl/"
xmlns="http://schemas. acbdf.org/wsdl/">
<portType name="GuidPortType">
      <operation name="getGuid" parameterOfOrder="prefix">
        <input message="tns:getGuidRequest"/>
        <output message="tns:getGuidResponse" />
      </operation>
  </portType>
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>
<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>
```

```
<portType name="glossaryTerms">
  <operation name="getTerm">
     <input message="getTermRequest"/>
     <output message="getTermResponse"/>
  </operation>
</portType>
          </output>
        </operation>
    </binding>
    <service name="GuidService"
        <port name="GuidPort" binding="tns:GuidBinding">
           <soap:address location="http://localhost/php/abcdf.php"/>
        </port>
    </service>
</definitions>
```

При створенні більшості платформ реалізується формування WSDL опису сервісу, таким чином надаючи розробнику можливість працювати з даним сервісом як із звичайним класом[8].

Веб-сервіси використовуються як ПЗ проміжного шару. Використовувати веб-сервіси при роботі з користувачем можуть як клієнтські програми, так і інші додатки (або веб-сервіси).

Розробники концепції веб-сервісів пропонують наступні сценарії застосування веб-сервісів:

1. Веб-сервіси при розробці логіки додатку (бізнес-логіки). Тобто, створюється новий додаток, логіка якого реалізується у веб-сервіси (див. Рисунок 2.3).

Рисунок 2.3 - сценарій застосування сервісу як реалізації бізнес-логіки додатку.

2. Веб-сервіси як засіб інтеграції. Тобто, веб-сервіс можна використовувати як спосіб доступу віддалених клієнтів до внутрішньої ІС, або для реалізації взаємодії внутрішніх компонентів (EJB, COM-компонента) з різними віддаленими клієнтами (див. рисунок).

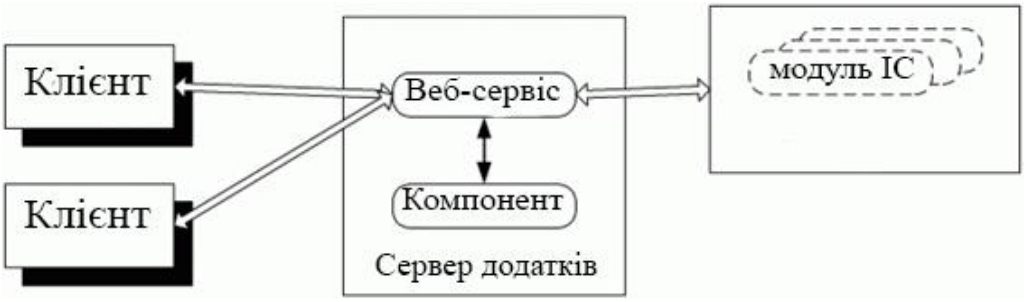

Рисунок 2.4 – сценарій застосування сервісу як засобу інтеграції

3. Використання веб-сервісу як основу при створенні проекту.

Платформа може використовувати веб-сервіси як вилучені компоненти, які можуть надати додаткову функціональність, залежно від того, чого саме потребує клієнт, різного роду обмежень в матеріалах і часі і складності завдання, що надає системі додаткової гнучкості (див. Рисунок 2.5).

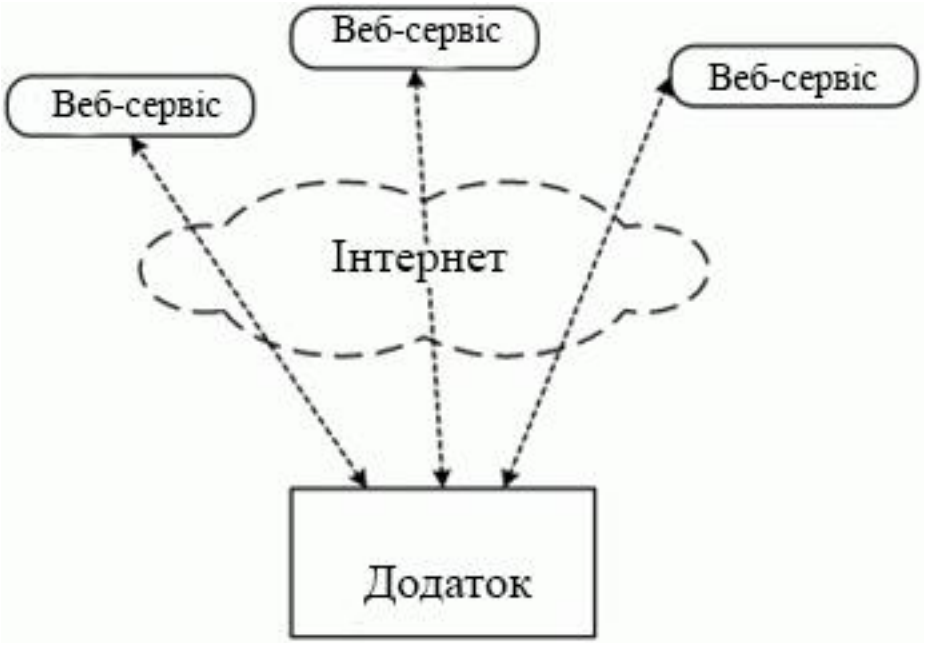

Рисунок 2.5 – застосування сервісу як віддаленого структурного блоку програми.

Крім цього, можливі й інші варіанти використання веб-сервісів. Наприклад, для побудови розподілених обчислювальних та інформаційних систем з різного роду структурами, але для використання такого роду веб-сервісу клієнт повинен мати певного робу інформацію, як адресу серверу, його ім'я, метод і пареметри.

У найпростішому випадку інформація може бути надана ще на етапі розробки, що показано на рисунку 2.6.

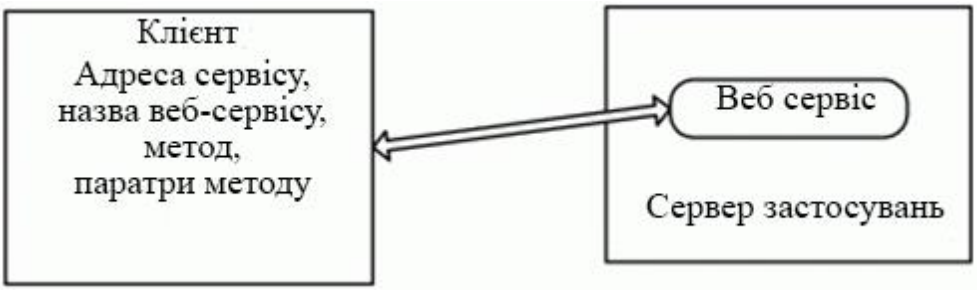

Рисунок 2.6 – інформація для застосування веб-сервісу, що є задана при розробці клієнта.

Документ WSDL повністю описує інтерфейс веб-сервісу із зовнішнім світом, він дає клієнтові можливість отримати необхідну інформацію для використання веб-сервісу (див. Рисунок 2.7).

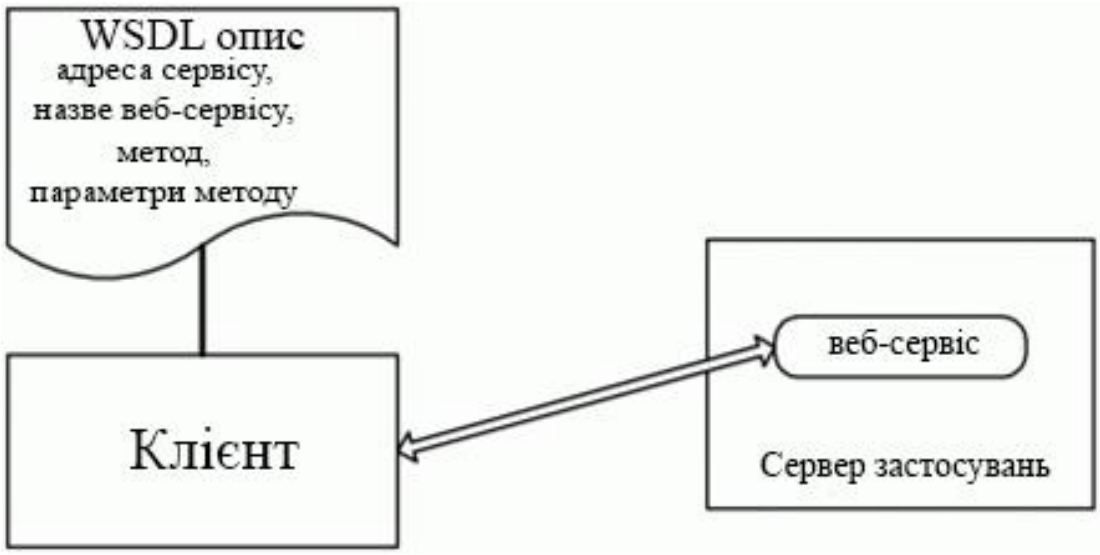

Рисунок 2.7 – опис інтерфейсу сервісу документом WSDL.

2.1.2 Хореографія та Оркестрація сервісів.

Основні технології як SOA надають засоби для опису, пошуку і виклику служб як самостійного об'єкту. Ця співпраця включає в себе послідовність дій і взаємозв'язків між ними, які створюють бізнес-процес. Існує два способи створення цього процесу: сервісна оркестровка і хореографія сервісу [9].

Організація оркестровки являє собою єдиний централізований виконуваний бізнес-процес (оркестр), який координує взаємодію між різними службами. Організатор відповідає за виклик і об'єднання сервісів.

Зв'язок між усіма що послугами описується однією кінцевою точкою (тобто складовою послугою). До складу оркестровки входить управління транзакціями між окремими службами. Оркестрація використовує централізований підхід до складу служби.

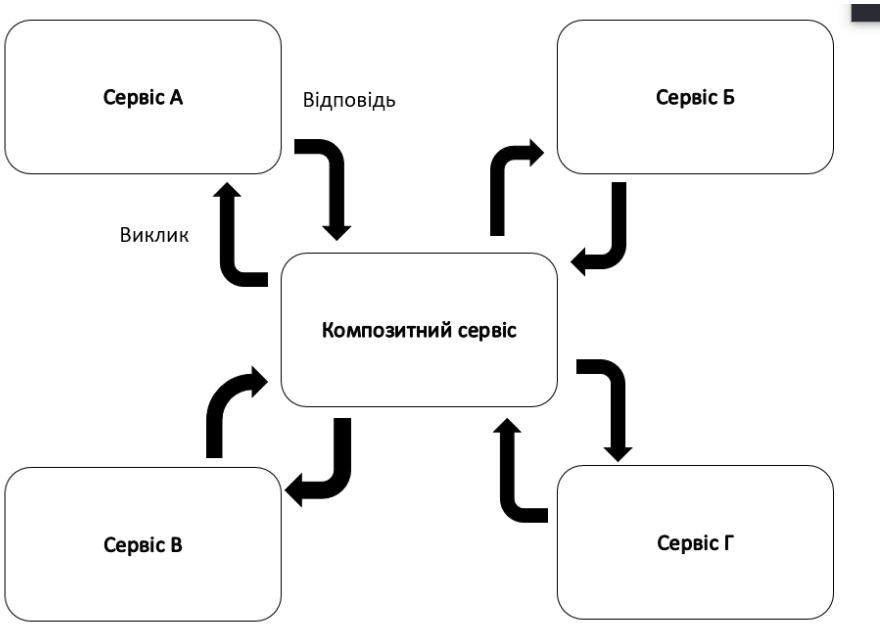

Рисунок 2.8 – схема принципу оркестрації

Хореографія послуг - це глобальний опис тих сервісів які беруть участь, який визначається обміном повідомленнями, правилами взаємодії і угодами між двома або більше кінцевими точками. У хореографії використовується децентралізований підхід до складу служби.

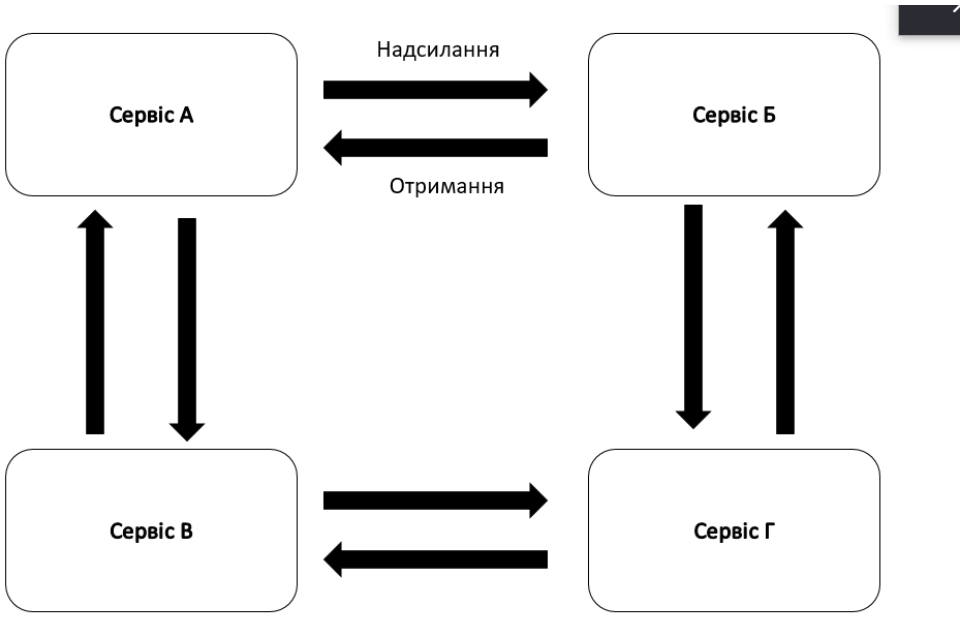

Рисунок 2.9 – схема принципу хореографії

У хореографії описуються взаємодії між декількома службами, де в якості оркестровки є контроль з одного боку. Це означає, що хореографія відрізняється від оркестровки щодо того, де повинна знаходитися логіка, яка контролює взаємодію між задіяними службами.

Тобто існує два основних підходи, які можуть допомогти розв'язати проблему координації роботи сервісів в розподілених системах:

Web Service Choreography - метод визначає протоколи взаємодії веб-сервісів між собою. Сервіси працюють разом для виконання одного глобального завдання і кожен виконує окремі його частини. Ефективна для невеликих завдань, так як зі збільшенням складності завдання та кількості сервісів, які будуть задіяні стає неефективною.

Service orchestration - метод який об'єднує сервіси за допомогою сервісу-окрестартора, в один багатофункціональний бізнес-сервіс. Подібний підхід найбільш цікавий в контексті поставленої задачі, оскільки надає можливості для впровадження агентів в систему як «сервісів-оркестраторів» без зміни її ієрархічної структури, таким чином комбінуючи переваги оркестрації з перевагами MAC.

2.2 Вибір технологій для проектування веб додатку

2.2.1 Опис архітектури додатку

Спочатку буде розглянуто принцип і сама побудова ієрархії роботи. Приблизна схема роботи будь-якого веб-додатку представлена на рисунку 2.10.

Рисунок 2.10 – схема роботи веб-додатку

На даному рисунку видно, що кожен веб-додаток відправляє http запит на веб-сервер для отримання певних даних. А програма, керуючись веб-сервером використовує певну модель щоб зберігати дані. На даний час найбільш популярними є такі бази даних, як SQL або NoSQL[10].

В найпростішому вигляді веб-додаток складається з трьох частин:

1. Частина, яка виконується веб-браузером. Дана частина реалізовується за допомогою будь-якої мови програмування. Єдина вимога – браузер повинен її підтримувати. Найбільш популярною мовою на даний час є JavaScript через те, що вона підтримується майже будь-яким браузером, а також має велику кількість бібліотек, що значно спрощує завдання.
2. Частина, що виконується сервером, якого контролює веб-сервер. Дана частина реалізовується за допомогою будь-якої мови програмування, однак знову умова – інтерпретацію даної мови знову ж повинен підтримувати веб-сервер.
3. База даних. Дана частина зберігає дані. Її реалізація може бути також дуже різноманітною. Є велика кількість типів баз даних, які будуть розглянуті нижче. Вибір бази даних ґрунтується на цілях і області вирішуваних завдань.

На етапі побудови потрібно якомога більше зменшити кількість блоків. Зазвичай додаток проектується на основі трьох структурних одиниць. Приклад зображено на рис. 2.11.

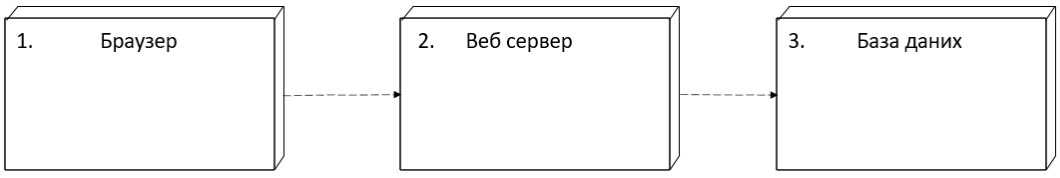

Рисунок 2.11 – схема побудови додатку

1. Модуль, яким управляє браузер.
2. Модуль, яким управляє веб-сервер.
3. База даних.

Даного роду структури створюють такі типи зв’язків.

1. Зв'язок між браузером і серверною частиною.
2. Зв'язок між серверною частиною і базою даних.

Щоб реалізувати максимальну незалежність між блоками, нам необхідно щоб кожен з них керував саме тим набором даних, який йому необхідний. Далі буде розглянуто детальніше:

Веб-браузер (браузер) - це програма, або ПЗ які дозволяють користувачеві знаходити веб-сторінки, отримувати до них доступ та відображати їх.

HTML - це стандартна мова розмітки документів. Вона є дуже зручною, так, як більшість браузерів підтримують її.

Веб-сервер - це ПЗ, яке може отримувати певні HTTP запити від клієнтів, обробляти їх і відповідати на них відповідно до стандартного протоколу.

База даних - це сукупність інформації, організованої таким чином, щоб її можна було легко отримати, керувати нею та оновлювати її.

Мінімізація залежностей

Для зменшення кількості залежностей між браузером і веб-сервером необхідно, щоб веб-сервер надавав інтерфейс для отримання необхідних даних для сторінки, таким чином мова HTML може бути задіяна лише в самому браузері,

Щоб зробити це нам необхідно:

- Визначити, що саме буде робити інтерфейс, його можливості.
- Визначити API серверної частини.

- Визначити, який саме протокол буде використовуватись для взаємодії між серверною і клієнтською частинами. При створенні протоколу необхідно врахувати на базі чого його створювати, щоб сучасні браузери були здатні його підтримувати.

2.2.2 Взаємодія веб-сервера та бази даних

Взаємодію бази даних і веб-сервера можна описати та створити двома шляхами. Логіка може знаходитись:

- В самій базі даних.
- В коді веб-сервера.

У випадку з базою даних, вона зберігає дані і надає інтерфейс доступу до них:

1. Вибірка даних - вирішується через представлення.
2. Модифікація даних - вирішується через процедури, які зберігаються.

Програма для веб-сервера є драйвером для доступу до логіки. Тобто вона просто пов'язує браузер з логікою, яка реалізована в базі даних.

У другому випадку база даних зберігає дані, і надає доступ до даних напряму. Логіка реалізована в коді веб-сервера. В цьому випадку база даних надає транзакції для проведення атомарних операцій.

Для мінімізації залежностей між веб-сервером і базою даних, необхідно, щоб бізнес-логіка була визначена тільки в одному місці. Тобто або в коді веб-сервера, або в базі даних. Це дуже важливо, тому що веде до зменшення витрат на обробку запитів на зміни.

2.2.2 Вибір веб-серверу

Вибір даної технології пов’язаний з сервером застосунків та веб-сервером. Сервер вибирається по факту того, який саме формат роботи нам необхідний, а також, як необхідно зберігати дані. Саме поняття веб-

серверу може відноситись як і в значенні апаратного забезпечення, так і програмного. Або навіть як до обох частин, які працюють сумісно.

В даному підрозділі веб-сервер розглядається як програмне забезпечення, яке включає в себе декілька компонентів, які в свою чергу управляють та контролюють доступ користувачів до файлів серверу. Самим базовим прикладом та сумісно мінімумом є HTTP-сервер. Це частина ПЗ, яка розпізнає веб-адреси та протокол доступу до веб-сторінок.

Простими словами, при запиті браузера до серверу, коли йому потрібен певний файл, він посилає запит на нього через HTTP протокол. Сервер приймає запит та знаходить необхідний файл та надсилає його назад за тим же протоколом. Якщо ж файл не знайдено, сервер повідомляє про помилку (помилка 404), що такого файлу не існує. Алгоритм роботи веб-сервера показано на рисунку 2.12

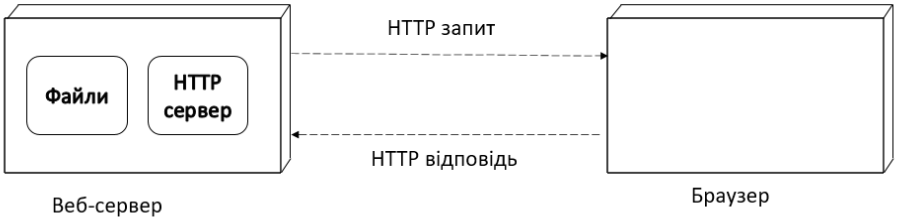

Рисунок 2.12 – алгоритм роботи веб-сервера

Сам сервер може бути статичний та динамічний.

Статичний складається комп'ютера (апаратного забезпечення) на якому знаходиться сервер HTTP, тобто програмне забезпечення.

Динамічний складається зі статичного веб-сервера та додаткогово ПЗ і є більш складним. Частіше за все роль ПЗ відіграє сервер додатку та баз даних. В ньому сервер додатків змінює вихідні дані перед надсиланням в браузер по HTTP. Тобто для отримання результуючої веб-сторінки сервер може заповнити існуючий HTML-шаблон даними з бази

даних, що дозволяє прискорити час опрацювання даних та супровід веб-додатків.

Файли веб-сайту, з якими працює сервер, тобто HTML-документи та пов’язані з ними ресурси зручніше зберігати на виділеному веб-сервері, який повинен завжди працювати та мати доступ до мережі інтернет, а також мати свою незмінну IP адресу. Тому необхідно також знайти хостинг-провайдера, який зможе задовільнити потреби додатку.

Розглянемо зв’язок по HTTP більш детально. Як було сказано вище, підтримку HTTP забезпечує сам сервер, а сам гіпертекстовий транспортний протокол вказує як передавати гіпертекст між двома комп’ютерами.

Сам протокол задає певні суворі правила взаємодії між клієнтом і сервером:

- клієнти можуть надсилати HTTP-запити тільки на сервери. А в свою чергу сервери здатні тільки відповідати на HTTP-запити клієнта;
- коли запитується файл по HTTP клієнт повинен сформувати файловий URL;
- веб-сервер мусить відповісти абсолютно на кожен запит, навіть якщо він має просто повідомити про помилку.

На власне веб-сервері HTTP сервер обробляє вхідні запити та відповідає на них:

- Коли отримується запит, спочатку перевіряється, чи існує файл за заданим URL;
- Якщо помилки немає, файл надсилається назад до браузера, якщо є, сервер додатку генерує необхідний ресурс;

- Якщо ні один з попередніх варіантів не можливий, генерується «помилка 404», яка повідомляє про те, що заданий файл не знайдено.

2.2.3 Огляд бази даних

База даних – це певна сукупність інформації, яка організована таким чином, щоб її можна було легко отримати, керувати нею та оновлювати її.

Дані організовані таким чином, що утворюються рядки, стовпці та таблиці. Вони індексуються для полегшення пошуку відповідної інформації. Інформація оновлюється, розширюються та видаляються після додавання нової інформації. Бази даних обробляють робочі навантаження, щоб створювати і оновлювати себе, запитуючи дані, які вони містять.

Комп'ютерні бази даних зазвичай містять узагальнення записів даних або файлів, таких як операції з продажу, каталоги продукції та запаси, а також профілі клієнтів[11].

Як правило, менеджер баз даних надає користувачам можливість керувати доступом до читання та запису, задавати генерацію звіту та аналізувати його роботу. Деякі бази даних пропонують ACID (атомність, узгодженість, ізоляцію та довговічність), щоб гарантувати, що дані є послідовними і що транзакції завершені.

Бази даних поширені у великих систем, але також присутні в менших розподілених робочих станціях і середніх системах, таких як AS/400 у IBM і персональних комп'ютерах.

З одного виду баз даних можна класифікувати за типом вмісту: бібліографічні, повного тексту, числові та типу зображення. В обчислювальних системах бази даних іноді класифікуються відповідно до їх організаційного підходу. Існує багато різних типів баз даних,

починаючи від найпоширенішого підходу, реляційної бази даних, розподіленої бази даних, хмарних баз даних або бази даних NoSQL.

2.2.3.1 Типи баз даних.

Існує багато різних типів баз даних. Вибір найкращої залежить від того, як саме мають зберігатись ваші дані:

- Реляційні бази даних. Елементи в реляційній базі даних організовані як набір таблиць з колонками і рядками. Технологія реляційних баз даних забезпечує найбільш ефективний і гнучкий спосіб доступу до структурованої інформації.
- Об'єктно-орієнтовані бази даних. Інформація в об'єктно-орієнтованій базі даних представлена у вигляді об'єктів, як у об'єктно-орієнтованому програмуванні.
- Розподілені бази даних. Розподілена база даних складається з двох або більше файлів, розташованих на різних сайтах. База даних може зберігатися на декількох комп'ютерах, розташованих в одному фізичному місці, або розкиданих по різних мережах.
- Сховища даних. Центральне сховище даних, сховище даних - це тип бази даних, спеціально розроблений для швидкого запиту та аналізу.
- Бази даних NoSQL. NoSQL, або нереляційна база даних, дозволяє зберігати і маніпулювати неструктурованими і напівструктурованими даними (на відміну від реляційної бази даних, яка визначає, як повинні бути складені всі дані в базі). Бази даних NoSQL стали популярними, оскільки веб-додатки стали більш поширеними і складними.
- Графові бази даних. Графові бази даних зберігають дані в термінах об'єктів і між сутностями.

- Бази даних OLTP. База даних OLTP - це швидка, аналітична база даних, призначена для великої кількості транзакцій, що виконуються кількома користувачами.

Це лише деякі з декількох десятків типів баз даних, що використовуються сьогодні. Інші, менш поширені бази даних пристосовані до дуже конкретних наукових, фінансових або інших функцій[12]. На додаток до різних типів баз даних, зміни в підходах до розробки технологій, такі як хмарні та автоматизовані. Деякі з останніх баз даних включають:

- Бази даних з відкритим вихідним кодом. Система бази даних з відкритим вихідним кодом є такою, вихідний код якої є відкритим кодом; такими базами даних можуть бути бази даних SQL або NoSQL.
- Хмарні бази даних. Хмарна база даних являє собою набір даних, структурованих або неструктурованих, які знаходяться на приватній, державній або гібридної платформі хмарних обчислень. Існують два типи хмарних моделей баз даних: традиційні та бази даних як послуги (DBaaS). За допомогою DBaaS адміністративні завдання та обслуговування виконуються постачальником послуг.
- База даних мультимоделей. Бази даних мультимоделей поєднують різні типи моделей баз даних в єдину, інтегровану базу. Це означає, що вони можуть містити різні типи даних.
- База даних документа / JSON. Бази даних документів призначені для зберігання, вилучення та керування документально-орієнтованою інформацією, це сучасний спосіб зберігання даних у форматі JSON, а не рядків і стовпців.
- Бази даних самостійного керування. Найновіші та найновіші типи баз даних, що самостійно керуються базами даних

(також відомими як автономні бази даних), засновані на хмарних і використовують машинне навчання для автоматизації налаштування бази даних, безпеки, резервного копіювання, оновлення та інших рутинних завдань керування, що традиційно виконуються адміністраторами баз даних.

2.2.3.2 Вибір підходящої бази даних

В зв'язку з тим, що дані в даному проекті не типізовані, нам потрібна така база даних, яка могла б зберігати їх і за допомогою якої було б зручно ними керувати. На мою думку при таких даних найкраще підходить така база даних, як NoSQL(Not Only SQL database), тобто нереляційна база даних.

NoSQL - це підхід до проектування баз даних, який може вміщувати широкий спектр моделей даних, включаючи ключові, документальні, колонкові та графічні формати. NoSQL є альтернативою традиційним реляційним базам даних, в яких дані розміщуються в таблицях, а схема даних ретельно розробляється до створення бази даних. Бази даних NoSQL особливо корисні для роботи з великими наборами розподілених даних.

Термін NoSQL може бути застосований до деяких баз даних, що передували реляційній системі управління базами даних. У цих додатках вимоги до продуктивності та масштабованості переважали над необхідністю безпосередньої, жорсткої консистенції даних, яку РСУБД надала транзакційним корпоративним додаткам.

Ранні бази даних NoSQL для веб- і хмарних додатків, як правило, зосередилися на дуже специфічних характеристиках управління даними. Можливість обробляти дуже великі обсяги даних і швидко розповсюджувати, ці дані через обчислювальні кластери були важливими ознаками у веб-дизайні та хмарних технологіях. Розробники, які впровадили хмарні та веб-системи, також намагалися створити гнучку

схему даних - або взагалі не мати схеми - для кращого включення швидких змін до програм, які постійно оновлювалися.

У базах даних NoSQL для доступу до даних і управління ними застосовуються різні моделі даних, в тому числі документна, графова, пошукова, з використанням пар «ключ-значення» і зберіганням даних в пам'яті. Бази даних таких типів оптимізовані для додатків, які працюють з великим обсягом даних, потребують низької затримки і гнучких моделях даних. Все це досягається шляхом пом'якшення жорстких вимог до несуперечності даних, характерних для інших типів БД.

У реляційній базі даних запис про книгу часто розділяється на кілька частин (або «нормалізується») і зберігається в окремих таблицях, зв’язок між якими визначається обмеженнями первинних і зовнішніх ключів. Реляційна модель створена таким чином, щоб забезпечити цілісність посилальних даних між таблицями в базі даних. Дані нормалізовані для зниження надмірності і в цілому оптимізовані для зберігання.

У базі даних NoSQL запис зазвичай зберігається як документ JSON. Для кожного елемента значення зберігаються в якості атрибутів в єдиному документі. У такій моделі дані оптимізовані для інтуїтивно зрозумілою розробки і горизонтальної масштабованості.

Переваги

- Бази даних NoSQL добре підходять для багатьох сучасних додатків, наприклад мобільних, ігрових, інтернет-додатків, коли потрібні гнучкі масштабовані бази даних з високою продуктивністю і широкими функціональними можливостями, здатні забезпечувати максимальну зручність використання.
- Гнучкість. Як правило, бази даних NoSQL пропонують гнучкі схеми, що дозволяє здійснювати розробку швидше і

забезпечує можливість поетапної реалізації. Завдяки використанню гнучких моделей даних БД NoSQL добре підходять для частково структурованих і неструктурованих даних.

- Масштабованість. Бази даних NoSQL розраховані на масштабування з використанням розподілених кластерів апаратного забезпечення, а не шляхом додавання дорогих надійних серверів. Деякі постачальники хмарних послуг проводять ці операції в фоновому режимі, забезпечуючи повністю керований сервіс.
- Висока продуктивність. Бази даних NoSQL оптимізовані для конкретних моделей даних (наприклад, документної, графічної або з використанням пар «ключ-значення») і шаблонів доступу, що дозволяє досягти більш високої продуктивності в порівнянні з реляційними базами даних.
- Широкі функціональні можливості. Бази даних NoSQL надають API і типи даних з широкою функціональністю, які спеціально розроблені для відповідних моделей даних.

2.2.4 Вибір платформи та мови програмування для реалізації веб-додатку

Від того, яка платформа буде обрана, залежить як і те, як буде реалізований сам додаток, так і можливість подальшого його розвитку. Також вибір сови програмування для реалізації веб-додатку є дуже важливим, тому потрібно почати з неї.

Мови програмування поділяються на 2 типи – типізовані та нетипізовані (ті, які не мають певного типу). Нетипізовані мови є простішими і не поділяються на інші підтипи. Типізовані ж в свою чергу поділяються на ще декілька категорій:

- Статична/динамічна типізація. При статичній типізації (приклад - C, Java, C#) кінцеві типи встановлюються на етапі компіляції, тобто компілятору до початку роботи програми відомо який тип мають змінні та методи та де вони знаходяться. При динамічній типізації (приклад - Python, JavaScript, Ruby) всі типи визначаються при виконанні програми.
- Сильна/слабка типізація. При сильній типізації (приклад - Java, Python, Haskell, Lisp) мова не дозволяє змішувати типи в виразах і не може виконати автоматичні неявні перетворення. Мови ж з слабкою типізацією (приклад - C, JavaScript, Visual Basic, PHP) можуть виконати перетворення неявно та автоматично, але при цьому може втратитись точність.
- Явна/неявна типізація. Явно типізовані мови (приклад - C++, D, C#) роблять необхідним вказувати типи функцій, змінних та їх аргументів явно. Неявно ж типізовані мови (приклад - PHP, Lua, JavaScript) віддають це завдання компілятору або інтерпретатору.

Також дані категорії можуть перемежовуватись, наприклад, мова програмування C має статичну, слабку типізацію.

Для розробки поставленого завдання необхідно вибрати саме нетипізовану мову (або з динамічною типізацією, так, як на етапі компіляції при використанні динамічної типізації, змінна може приймати будь-який тип і ще не є типізованою) тому, що при виборі бази даних, враховуючи тип самих зберігаємої інформації, була вибрана нетипізована база даних. Тому розглянемо дані типи.

Динамічна типізація в мовах програмування.

Динамічна типізація означає, що велика частина перевірок типів даних виконується на етапі виконання програми, хоча деякі перевірки (наприклад, перевірка синтаксичної коректності коду) виконуються на етапі компіляції[13]. Крім того, типи даних асоціюються з конкретними значеннями, а не зі змінними.

Динамічна типізація дозволяє створювати більш гнучке програмне забезпечення, хоча і ціною значної ймовірності помилок типізації. Модульне тестування набуває особливого значення при розробці програмного забезпечення на мовах програмування з динамічною типізацією, так як воно є єдиним способом знаходження помилок типізації, допущених в рідко використовуваних гілках логіки програми.

З відомих динамічно типізованих мов найбільш відомою є JavaScript, з допомогою якої в рамках платформи Node.js можна створювати серверну частину веб-додатку.

Розглянемо мову програмування JavaScript з допомогою якої буде побудоване дане завдання.

JavaScript – скриптова мова програмування, яка послужила основою для стандарту ECMAScript і стала його найпопулярнішим діалектом.

Найбільш поширеним використанням JavaScript є написання клієнтської сторони веб-додатку, що дозволяє створювати динамічні сторінки та користувацькі інтерфейси. Але також є можливим створення серверної частини, що робить дану мову дуже універсальною. Також великим плюсом є підтримка його майже усіма браузерами.

Мова програмування JavaScript є доволі компактною, але не менш гнучкою ніж аналоги, навіть більш. До неї вже створено дуже багато бібліотек та додатків, що додають велику кількість функціоналу, як наприклад:

- API, вбудовані в браузери, забезпечують різноманітні функціональні можливості, такі як динамічне створення

HTML і встановлення CSS стилів, захоплення і маніпуляція відео, робота з веб-камерою або генерацією 3D графіки і аудіо семплів. API - це готові набори блоків коду, які дозволяють розробнику реалізовувати програми, які в іншому випадку було б важко або неможливо реалізувати.

- Сторонні API дають змогу додати до можливостей своїх сайтів функціонал, створений іншими розробниками, як Twitter або Facebook.
- Також ви можете підібрати ваш HTML-код і бібліотеки, які дозволяють вам прискорити створення сайтів і додатків.

JavaScript є об'єктно-орієнтованою мовою, але вона має певні відмінності в порівнянні зі звичайними клас-орієнтованими мовами[14]. Крім того, дана мова має павні властивості функціональних мов, такі, як:

- функції як об'єкти першого класу;
- об'єкти як списки;
- анонімні функції;

Також, кадучи про мінуси, у мові відсутні певного роду можливості, як стандартна бібліотека, інтерфейс програмування додатків по роботі з файловою системою, управління потоками вводу-виводу, стандартні інтерфейси до веб-серверів і баз даних, а також система управління пакетами, що може відстежувати та встановлювати залежності.

Виходячи з цих даних можна зробити висновки що функціоналу даної мови програмування достатньо для виконання поставленого завдання. Для написання серверної частини додатку необхідно використовувати додаткову платформу, як Node.js.

Node або Node.js - програмна платформа, яка здійснює трансляцію JavaScript в машинний код, тобто перетворює JavaScript з вузьконаправленої мови в мову загального призначення. Node.js надає можливість JavaScript взаємодіяти з пристроями введення-виведення,

підключати зовнішні бібліотеки, написані на різних мовах для розширення функціоналу.

Node.js застосовується переважно на сервері, виконуючи роль веб-сервера. В її основі лежить подієво-орієнтоване і асинхронне (або реактивне) програмування з неблокуючим введенням/виведенням.

Однією з відмінних рис даної технології є її підхід до роботи з введенням-виведенням і відсутність багатопоточності. Щоб зрозуміти, що з себе представляє мова, наведено простий приклад - виконання HTTP-запиту:

- Зазвичай, при стандартному підході викликається функція/метод, яка здійснює запит, і виконання потоку блокується до тих пір, поки не прийде відповідь або повідомлення про помилку. Якщо потрібно виконувати якісь дії під час виконання запиту, то потрібно використовувати ще один потік.
- У Node ви викликаєте функцію/метод, і, крім параметрів запиту, передаєте їй ще один аргумент - функцію, яка буде викликана по завершенню запиту. Виклик не блокує виконання, і код залишиться активним.

Для роботи з платформою Node.js необхідно також вибрати фреймворк для роботи з серверною частиною. Він спрощує розробку додатку тому було розглягуто найбільш популярні фреймворки на даний час.

Ось деякі з найбільш популярних фреймворків JavaScript для JavaScript у 2019 році:

- Express.js все ще є найпопулярнішою структурою Node.js. Це швидка, невизначена і мінімалістська веб-структура. Вона розвивалася швидко, тому що вона була простою та зрозумілою. Вона, напевно, знаходиться значно ближче до

основних ідей Node.js щодо легкої системи з модульним підходом.

- Meteor, з іншого боку, використовує чистий JavaScript і Node.js всередині архітектури. Meteor - це сама екосистема, яка може бути корисною для створення більш складних серверних додатків. Однак, його використання може бути занадто важким, якщо необхідно зробити щось не вбудоване.
- Sails.js - це фреймворк MVC у реальному часі. Він був розроблений для емуляції шаблону MVC Ruby on Rails, але з підтримкою сучасних вимог додатків. Він робить це за допомогою API, керованих даних, з масштабованою, орієнтованою на обслуговування архітектурою.
- Koa.js був створений командою за Express. Компанія продається як "веб-фреймворк нового покоління для Node.js" - це менша, виразніша і надійніша основа для веб-додатків і API.
- Strapi - це інтерфейс API Node.js з можливостями CMS. Він є новим, але вже зарекомендував себе як один з найдосконаліших фреймворків керування вмістом Node.js.

З розглянутих вище фреймворків найбільш зручним у використанні є Express.js, так, як вона знаходиться значно ближче до основних ідей Node.js щодо легкої системи з модульним підходом.

ВИСНОВКИ ДО РОЗДІЛУ 2

В даному розділі було розглянуто основні технології для реалізації серверної частини веб-додатків. Спочатку було вибрано сервер, на основі якого буде побудований проект, і який буде обробляти всі вхідні запити та працювати з базою даних. Було розглянуто самі запити та їх типи, як вони працюють. Далі на основі вибору веб-серверу та поставленого завдання було вибрано базу даних, яка зможе зберігати нетипізовані дані та обробляти їх. Таким сховищем даних буде NoSQL database, тобто нереляційня база даних, яка і зможе працювати з даного роду даними. В зв'язку з пересіченням двох множин, на основі поставленого завдання, вибраних веб-серверу та сховища даних була вибрана мова програмування для реалізації додатку. JavaScript є доволі зручною мовою, яка може працювати з нетипізованими даними та нереляційною базою даних NoSQL. Для реалізації серверної частини було вибрано платформу Node.js та фреймворк до неї - Express.js

РОЗДІЛ 3

ОПИС РОБОТИ СЕРВЕРНОЇ ЧАСТИНИ ВЕБ-ДОДАТКУ

В даному розділі буде розглянуто використані для побудови проекту технології, використані при написанні коду бібліотеки та зв’язок з базою даних. Також буде показано приклад роботи веб-додатку та проведені до нього тести.

2.3 Використані бібліотеки

В рамках проекту було використано платформу Node.js за фреймворком Express.js.

Для побудови проекту були використані наступні бібліотеки:

- *bcryptjs*

Цей npm пакет використовує UNIX бібліотеку bcrypt. Це дозволяє хешірованного і шифрувати конфіденційні дані, такі як паролі користувачів, перед їх збереженням в базу даних.

- *git-tags*

Використовується для отримання версій додатку.

- *Gravatar*

Дана бібліотека допомагає отримати фото користувача, яке надсилається з теперішнього фото профілю за вказаною e-mail адресою.

Наступні бібліотеки будуть описані окремо, так, як вони потребують детального розгляду.

Бібліотека *validator*.

Дана бібліотека використовується для реалізації валідації користувача в системі.

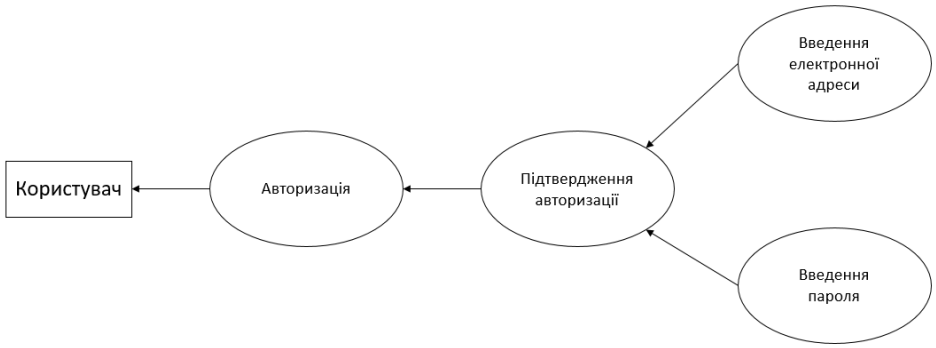

Рисунок 3.1 – Ієрархія прецеденту «Авторизація»

Бібліотека *mongoose.*

Дана бібліотека використовується для реалізації підключення бази даних до проекту.

В ході розробки проекту було вирішено використати хмарну базу даних MongoDB.

MongoDB - документоорієнтованих система управління базами даних (СУБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць[15]. Класифікована як NoSQL, використовує JSON-подібні документи і схему бази даних. Написана на мові C ++.

Отже, давайте розберемо, чому вам варто розглянути використання MongoDB в вашому проекті:

- Відсутність схеми.
- Легкість горизонтального масштабування.
- Багата функція аггрегации.
- Створена для денормалізації.
- Простий формат індексів.

У MongoDB є офіційні драйвери для основних мов програмування. Існує також велика кількість неофіційних або підтримуваних спільнотою драйверів для інших мов програмування і фреймворків.

Основним інтерфейсом до бази даних була командна оболонка «mongo». З версії MongoDB 3.2 в якості графічної оболонки

поставляється «MongoDB Compass». Існують продукти і сторонні проекти, які пропонують інструменти з GUI для адміністрування і перегляду даних.

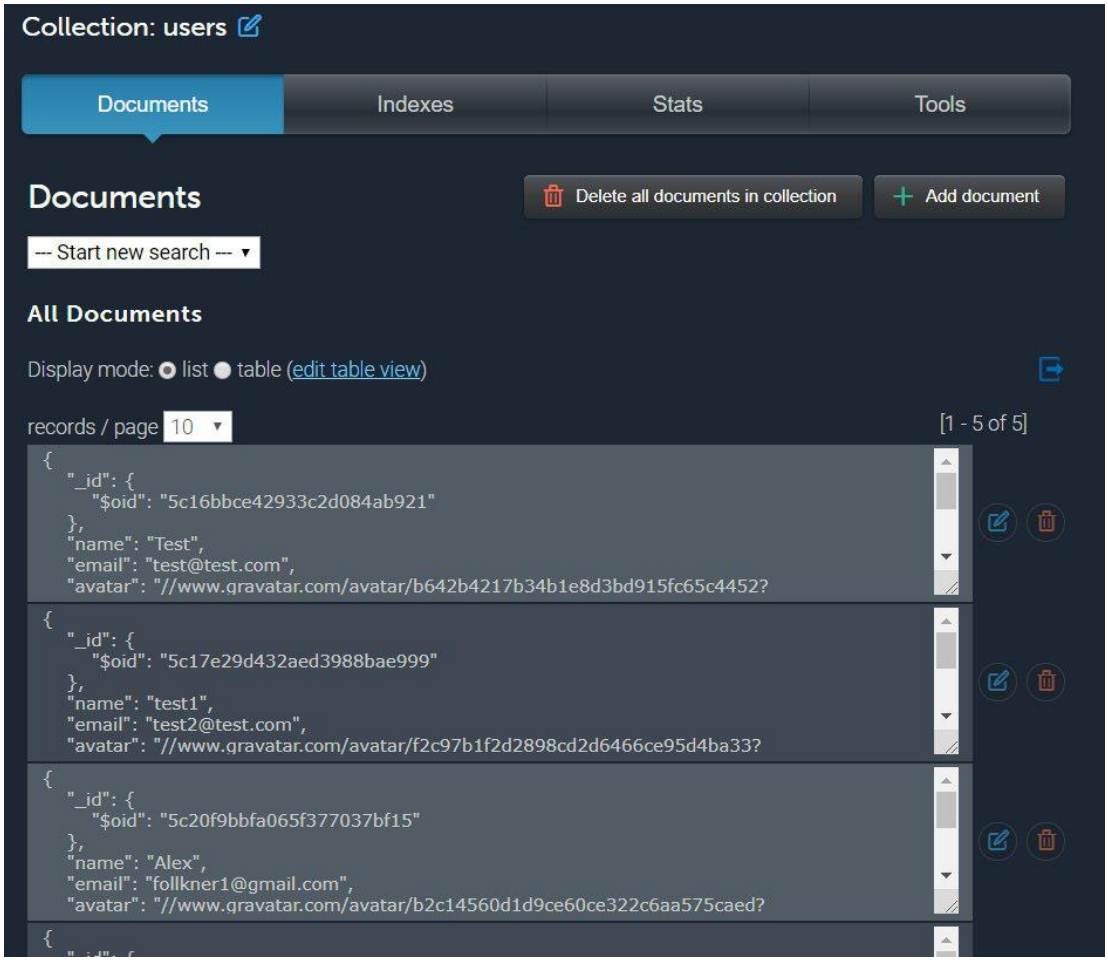

Рисунок 3.2 – приклад використання MongoDB

Захисту даних користувачів була приділена окрема увага, адже це є персональними даними. Для того, щоб захистити їх було використано наступні бібліотеки:

- *Passport*

Passport є проміжним програмним забезпеченням для express.js. Він підтримує різні типи входу, Basic, Token, Local (ім'я користувача, пароль), OAuth, OAuth2 і т.д. Ми можемо об'єднати їх, щоб користувачі могли аутентифицироваться, вступаючи в систему за допомогою Google, FB або будь-якого іншого сервісу з мінімальною кількістю коду. Ми також можемо використовувати це для об'єднання зовнішніх аут-сервісів, щоб

користувачі могли вибрати логін з однією з обраних стратегій, наприклад Google, Twitter.

- Passport-jwt

Стратегія Passport для аутентифікації за допомогою веб-маркера JSON. Цей модуль дозволяє перевірити автентичність кінцевих точок за допомогою маркера JSON. Він призначений для використання для захисту кінцевих точок RESTful без сесій.

- Jsonwebtoken

Використовується для роботи з токенами.

2.4 Приклад роботи серверної частини веб-додатку

Для того, щоб перевірити працездатність додатку, необхідно в термінал ввести команду *npm start.*

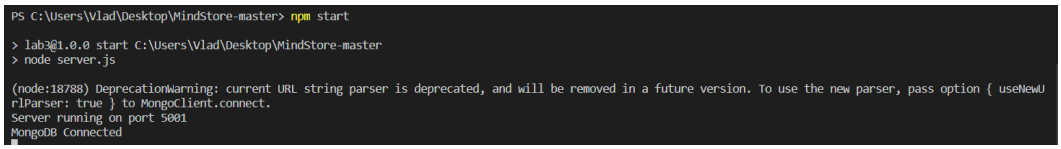

Рисунок 3.3 – приклад запущеного додатку

Як результат на консоль виводяться дані про порт та про те, що додаток підключений до бази даних.

2.5 Тестування веб-додатку

Також для того щоб показати працездатність додатку було проведено ряд тестувань. Для запуску тесту була використана команда npm run test –server.

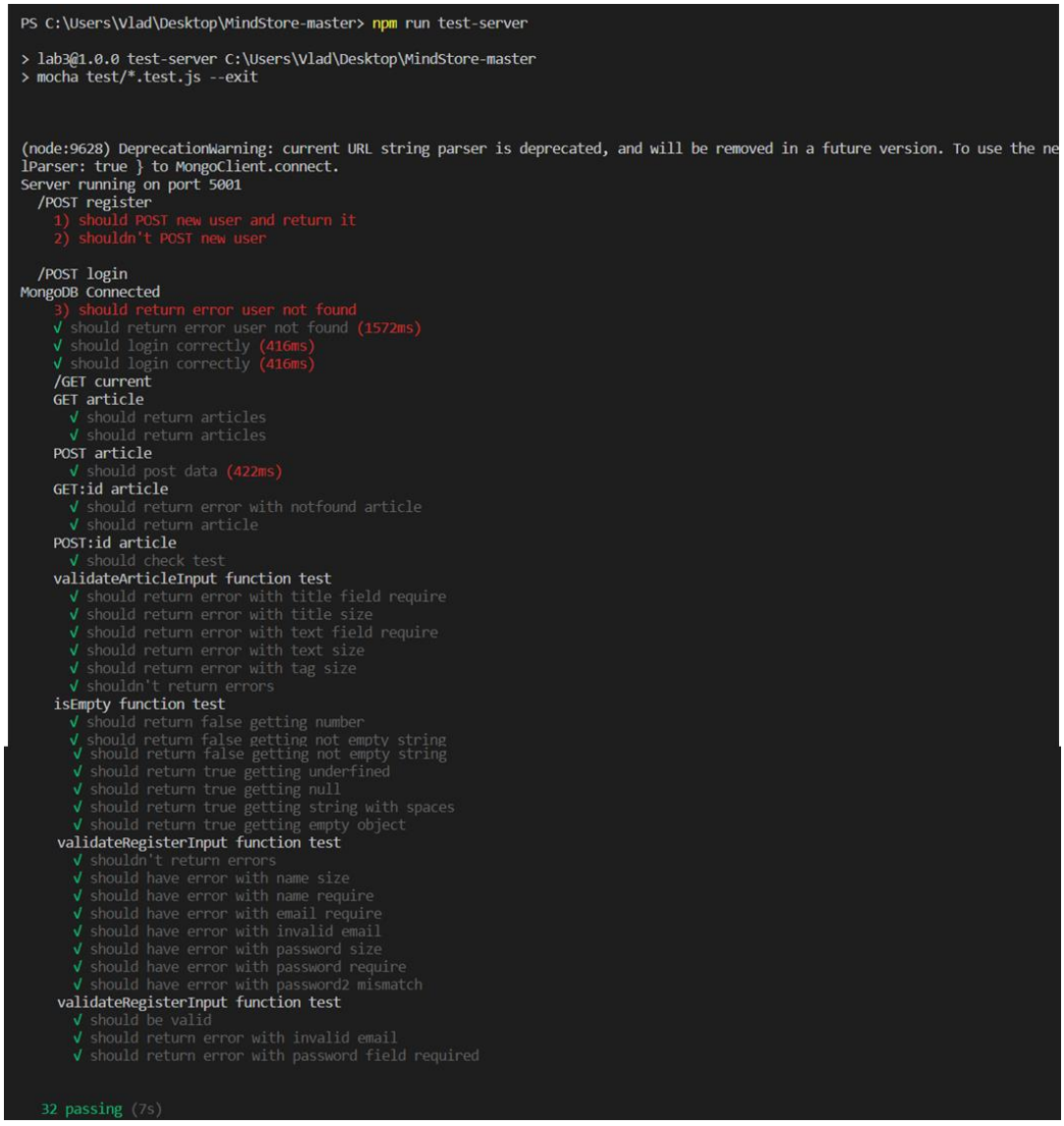

Рисунок 3.4 – тестування серверної частини веб-додатку

Проводиться тестування валідації, тобто перевірка запитів до API та перевірка даних логіну та пароля. А також тестувались запити до серверу.

При розробці було також задіяно технологію Travis CI, що надало можливість тестувати та своєчасно виявляти та виправляти помилки, які могли виникнути при розробці додатку.

Travis CI - розподілений веб-сервіс для побудови та тестування програмного забезпечення, що використовує GitHub для хостингу вихідного коду. Програмна складова сервісу також розташовується на

GitHub'i, проте розробники не рекомендують використовувати її в закритих проектах.

Веб-сервіс підтримує збірку проектів на багатьох мовах, включаючи JavaScript. Різні проекти з відкритим вихідним кодом використовують Travis CI для безперервної інтеграції коду.

Експоненціальна вибірка

Якщо в користувача зникне доступ до мережі інтернет, то веб-додаток буде намагатись перепідключитись до мережі раз в деякий час.

В процесі експоненційної вибірки отримання наступного часу повторення запиту відбувається за допомогою множення попереднього отриманого значення на два.

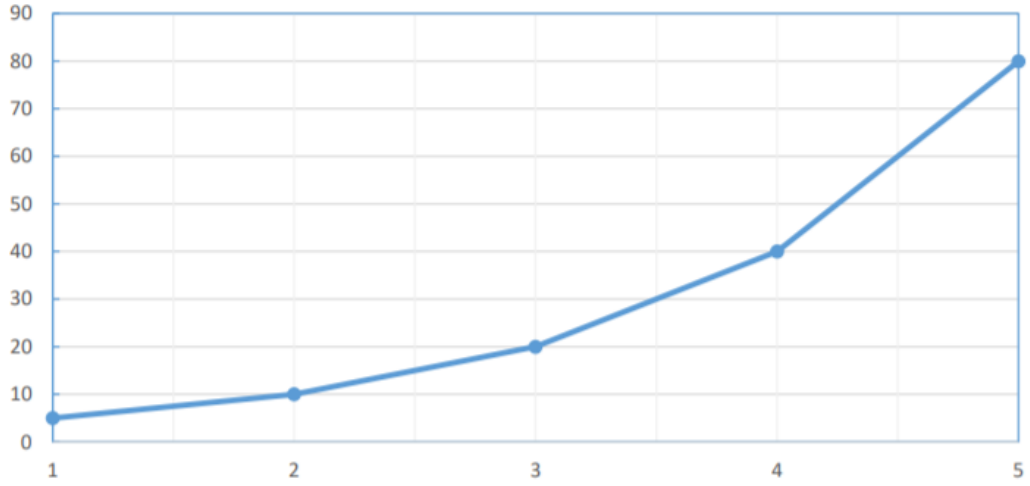

Рисунок 3.5 – графік часової залежності повторних запитів користувачів

ВИСНОВКИ ДО РОЗДІЛУ 3

В ході даного розділу було розроблено серверну частину веб-додатку, підключено її до хмарної бази даних та протестовано. Також було описано принцип роботи веб-додатку та реалізовано весь запланований функціонал.

ВИСНОВКИ

В ході даної дипломної роботи було проаналізовано існуючі рішення для того щоб виділити основні переваги і недоліки, які у подальшому були використані для вирішення поставленої задачі та уникнення власне самих недоліків. Виділені переваги були включені до складу додатку, а виявлені недоліки платформ уникнуті в ході розробки. Також були проаналізовані структури роботи веб-сайтів та аналіз алгоритмів їх роботи. Вони були використані для побудови та планування власного продукту.

Далі було розглянуто основні технології для реалізації серверної частини веб-додатку.

Спочатку було вибрано сервер, на основі якого був побудований проект, і який обробляє всі вхідні запити та працює з базою даних. Було розглянуто самі запити та їх типи, та як саме вони працюють.

Далі на основі вибору веб-серверу та поставленого завдання було вибрано базу даних, яка зможе зберігати нетипізовані дані та обробляти їх. Таким сховищем даних було вибрано NoSQL database, тобто нереляційня база даних, а саме – MongoDB, яка і зможе працювати з такого роду даними.

В зв'язку з пересіченням двох множин, на основі поставленого завдання, вибраних веб-серверу та сховища даних була вибрана мова програмування для реалізації додатку - JavaScript. Вона є доволі зручною мовою, яка може працювати з нетипізованими даними та нереляційною базою даних NoSQL.

Для реалізації серверної частини було вибрано платформу Node.js та фреймворк до неї - Express.js.

Далі було розроблено серверну частину веб-додатку, підключено її до хмарної бази даних та протестовано.

Результати тестів показали працездатність додатку. Також було описано принцип роботи веб-додатку та реалізовано весь запланований функціонал.

Список використаної літератури

1. Платформа для підготовки до олімпіад з інформатики та програмування – Режим доступу до ресурсу: https://informatics.mccme.ru/#/.
2. Платформа з підготовки та перевірки знань англійської мови – Режим доступу до ресурсу: https://www.examenglish.com/
3. Платформа з вивчення та тестування різноманітних навичок у сфері розробки програмного забезпечення, їх підтримки, захисту та засобів для створення візуального контенту – Режим доступу до ресурсу: https://www.pluralsight.com/.
4. Веб-додаток для вивчення та поглиблення знань з програмування – Режим доступу до ресурсу: https://metanit.com/.
5. Сучасні Web-технології– Режим доступу до ресурсу: https://www.lessons-tva.info/edu/contents-ttd.html.
6. IBM [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/developerworks/ru/library/ws-soa-term1/index.html.
7. Microsoft.com [Електронний ресурс] – Режим доступу до ресурсу:https://msdn.microsoft.com/en-us/library/bb833022.aspx.
8. IBM [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/developerworks/ru/library/j-jws20/index.html
9. OSP [Електронний ресурс] – Режим доступу до ресурсу: https://www.osp.ru/os/2004/11/184785/
10. Zametkinapolyah [Електронний ресурс] – Режим доступу до ресурсу: https://zametkinapolyah.ru/servera-i-protokoly/http-

server-ili-veb-server-naznachenie-funkcii-i-rol-servera-v-http.html.

11. Організація баз даних: практичний курс : Навч. посіб. для студ. / А. Ю. Берко, О. М. Верес; Нац. ун-т "Львів. політехніка". - Л., 2003. - 149 с. - Бібліогр.: 8 назв..
12. Когаловский М. Р. Енциклопедія технологій баз даних. — М.: Фінанси статистика, 2002. — 800 с. — ISBN 5-279-02276-4.
13. Luca Cardelli, Peter Wegner. On Understanding Types, Data Abstraction, and Polymorphism — ACM Computing Surveys, 1985. — С. 471—523. — ISSN 0360-0300.
14. Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/uk/.
15. Кайл Бенкер. MongoDB в дії = MongoDB in Action. — ДМК Пресс, 2014. — 394 с. — ISBN 978-5-97060-057-3.