Державний вищий навчальний заклад
“Прикарпатський національний університет імені Василя Стефаника”
Кафедра інформаційних технологій

УДК 004

ДИПЛОМНИЙ ПРОЕКТ

Тема: Розробка web-сервісу для перевірки і оцінювання знань

Спеціальність: 121 Інженерія програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДП.ПІ-05.ПЗ
(позначення)

Рецензент
професор Кузь М.В.
(посада) (підпис) (дата) (розшифровка підпису)

Студент
ІПЗ-41 Данилів І.С.
(шифр групи) (підпис) (дата) (розшифровка підпису)

Нормоконтролер
професор Кузь М.В.
(посада) (підпис) (дата) (розшифровка підпису)

Керівник дипломного проекту
ст. викладач Савка І.Я.
(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту

Завідувач кафедри

Козленко М.І.
(посада) (підпис) (дата) (розшифровка підпису)

2020
(рік)

Державний вищий навчальний заклад
«Прикарпатський національний університет імені Василя Стефаника»
Факультет математики та інформатики Кафедра інформаційних технологій
Спеціальність 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Завідувач кафедри Козленко М.І.
_____
„_____” _____ 20___р.

ЗАВДАННЯ
НА ВИКОНАННЯ ДИПЛОМНОГО ПРОЕКТУ

Студенту Даниліву Івану Степановичу
(прізвище, ім’я, по батькові студента)
1. Тема проекту Розробка web-сервісу для перевірки і оцінювання знань
затверджена розпорядженням по факультету математики та інформатики від „25” жовтня 2019 р.№7
2. Термін здачі студентом закінченого проекту 22 травня 2020 р. 3. Вихідні дані до дипломного проекту Стандарт кафедри Інформаційних Технологій ПНУ “Вимоги до змісту та оформлення”, технологія програмування серверної частини – Ruby on Rails, технології програмування клієнтської частини – HTML, CSS, Bootstrap 4, мова шаблонів – Slim, навчальний посібник – Бізнес-планування.
4. Зміст пояснювальної записки (перелік питань, що їх належить опрацювати)
1. Аналіз, дослідження особливостей та постановка задачі веб-сервісу для перевірки і оцінювання знань. 2. Методи та засоби реалізації веб-сервісу. 3. Опис програмної реалізації веб-сервісу для перевірки і оцінювання знань. 4. Бізнес-план веб-сервісу для перевірки і оцінювання знань
5. Перелік графічного матеріалу (з точним забезпеченням обов’язкових креслень)_____
_____
_____
_____
_____
_____
6. Дата видачі завдання 11.09.2019

Керівник _____ Савка І.Я.

Завдання прийняв до виконання _____ Данилів І.С.

КАЛЕНДАРНИЙ ПЛАН

Номер і назва етапів дипломного проекту	Термін виконання етапів проекту	Примітка
Обґрунтування актуальності, формулювання мети, завдання, предмету та об’єкту дослідження проекту	10.10.2019	виконав
Опрацювання джерел з теми проекту	30.10.2019	виконав
Підготовка І розділу проекту	15.12.2019	виконав
Підготовка ІІ розділу проекту	31.01.2020	виконав
Підготовка ІІІрозділу проекту	20.02.2020	виконав
Підготовка ІV розділу проекту	20.03.2020	виконав
Виправлення зауважень попередніх звітів. Підготовка вступу та висновків	20.04.2020	виконав
Оформлення проекту згідно вимог	10.05.2020	виконав

Студент _____ Данилів І.С.
(підпис)

Керівник проекту _____ Савка І.Я.
(підпис)

РЕФЕРАТ

Пояснювальна записка: 73 сторінки (без додатків), 29 рисунків, 13 таблиць, 25 джерел, 6 додатків на 11 сторінках.

Ключові слова: МОВА ПРОГРАМУВАННЯ, ФРЕЙМВОРК, АКАУНТ, RUBY, RUBY ON RAILS, САЙТ, СЕРВЕР, ВЕБ-СЕРВІС, ER-ДІАГРАМА.

Об`єктом дослідження є web-сервіс для перевірки і оцінювання знань.

Мета роботи: спроектувати та розробити клієнтську та серверну частину web-сервісу для перевірки і оцінювання знань.

Стислий опис тексту пояснювальної записки:

У даному дипломному проекті описано реалізацію веб-сайту написаного з використання мови Ruby і фреймворку Ruby on Rails. Для збереження інформації використовувалась база даних PostgreSQL. Для розробки зовнішнього вигляду використовувались Bootstrap та мова шаблонів Slim.

ABSTRACT

Explanatory note: 73 pages (without appendix), 28 figures, 13 tables, 25 references, 5 appendix on 11 pages.

Key words: PROGRAMING LANGUAGE, FRAMEWORK, ACCOUNT, RUBY, RUBY ON RAILS, SITE, SERVER, WEB- SERVICE, ER-DIAGRAM.

The object of study is the web service for knowledge checking and evaluation.

The purpose of the work is to design and develop the client's and server parts for knowledge checking and evaluation web service.

Brief description of the text of the explanatory note:

This course project describes the main stages of design and development of the website written by means of Ruby programming language and Ruby on Rails framework. The PostgreSQL database was used to store the information. Bootstrap and the Slim template language were used to develop the look.

ЗМІСТ

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ПЗ – програмне забезпечення.

СУБД – система управління базами даних.

API – Application Programming Interface.

CSS – Cascading Style Sheets.

HTML – Hypertext Markup Language.

HTTP – Hypertext Transfer Protocol.

MVC – Model-View-Controller.

ROR – Ruby on Rails

SOAP – Simple Object Access Protocol.

UDDI – Universal Description Discovery & Integration

URL – Uniform Resource Locator.

WSDL – Web Service Definition Language

XML – Extensible Markup Language.

ВСТУП

В наш час вже не можна уявити собі суспільство без використання веб-технологій. На сучасному етапі розвитку інформаційних технологій для отримання та передачі будь-яких видів інформації використовують комп'ютер та мобільні гаджети, які стали єдиними засобами, що надають широкі можливості керування інформацією. Все більше людей отримують доступ до мережі Інтернет, а багато установ використовують інформаційні мережеві системи для організації та управління внутрішніми процесами. Одним із пріоритетних напрямків процесу інформатизації сучасного суспільства є інформатизація освіти, яка являє собою систему методів, процесів і програмно-технічних засобів, інтегрованих з метою збирання, обробки, зберігання, розповсюдження та використання інформації в інтересах її споживачів.

Сьогодні все більше вчителі зустрічаються з проблемою, як зацікавити і здивувати учнів, так як вони – покоління інформаційного століття і мають необмежені можливості доступу до різної інформації. І тут на допомогу вчителям можуть прийти різні інтерактивні сервіси, за допомогою яких можна урізноманітнити всій урок, зробити його більш цікавим. А однією з основних причин посиленої уваги вчителів до впровадження веб-технологій є зручність та простота використання наявних веб-ресурсів.

Застосування інформаційних технологій в освіті сприяє підвищенню мотивації навчання учнів, економії навчального часу, а інтерактивність і наочність сприяє кращому уявленню, розумінню та засвоєнню навчального матеріалу, дозволяє значно полегшити організацію навчального процесу, перевірку та оцінювання знань учнів.

Останнім часом, дистанційне навчання стало трендом сучасної освіти в світі, особливо під час пандемії COVID-19. Вітчизняні педагоги активно приєднуються до новаторів освіти, впроваджуючи дистанційні технології навчання в процес навчання у закладах освіти.

Основними принципами дистанційного навчання є інтерактивна взаємодія у процесі роботи, надання учням можливості самостійного освоєння досліджуваного матеріалу, а також консультаційний супровід у процесі дослідницької діяльності.

Відкритим залишається питання про вибір оптимальних засобів організації дистанційного навчання. У бакалаврському проекті розробляється веб-сервіс для перевірки та оцінювання знань учнів. Розробка є актуальною, оскільки в даний час всі вчителі України зустрілись з проблемою впровадження дистанційного навчання для всіх учнів і пандемією COVID-19. Вимушене дистанційне навчання поставило вчителів перед непростими викликами: як організувати навчання дітей в умовах карантину, коли вчитель не може бути поруч, як зрозуміти, чи засвоїла дитина даний матеріал та як перевірити й оцінити знання учнів.

В даному проекті розглядаються особливості роботи з сервісом, зокрема створення дистанційних курсів, додання матеріалів до нього, здійснення зворотного зв'язку вчителя та учня. Постійний контроль знань учнів значно підвищує їх мотивацію до навчання, проте його проведення під час дистанційного навчання викликає багато труднощів у вчителів.

Тому, було запропоновано дослідити можливість створення веб-сервісу для перевірки та оцінювання знань учнів, що дозволить урізноманітнити дидактичні матеріали для мережевої навчальної діяльності з акцентом на інтерактивні форми, дуже швидко здійснювати контроль знань, максимально автоматизувати процеси тестування, а також використовувати різноманітні тести у дистанційному навчанні.

1 АНАЛІЗ, ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ ТА ПОСТАНОВКА ЗАДАЧІ ВЕБ-СЕРВІСУ ДЛЯ ПЕРЕВІРКИ І ОЦІНЮВАННЯ ЗНАНЬ

1.1 Задача розробки веб-сервісу для перевірки і оцінювання знань

У теперішньому світі за останні роки із розвитком новітніх технологій, поширенням та доступністю інтернет-зв'язку з'являється унікальна можливість для освіти. Інтернет – це необмежене джерело різної інформації, невичерпна скарбниця інтелектуальної активності сучасного школяра чи студента, який має можливість отримувати нові знання, удосконалювати уміння та навички. Учитель, за допомогою інтернету, може задавати учням завдання та перевіряти їх, оптимізувати систему перевірки, зокрема проводити тестування. Значний інтерес до розробки і використання освітніх веб-сервісів зумовлено значними очікуваннями підвищення ефективності навчання та оцінювання знань.

Основними завданнями проекту є:

- забезпечення вчителя персональним кабінетом, де можна створювати власний курс, завдання та тести до нього;
- забезпечення учня можливістю отримання завдань від вчителя та проходити тестування;
- швидке здійснення контролю знань;
- можливість дистанційного навчання;
- забезпечення користувачів зручним інтерфейсом.

Програмне забезпечення повинне бути гнучким у розробці та зруним у використанні, а архітектура веб-сервісу повинна складатися із трьох компонентів: БД, сервер та клієнт.

Користувачами сервісу мають бути:

- адміністратор;
- вчитель;
- учень.

Адміністратор має мати змогу перевіряти дані про користувачів, змінювати інформацію, при потребі видаляти користувачів та застарілу інформацію.

Вчителі мають мати змогу створювати власний курс, додавати завдання та тести до курсу. Також вони повинні мати можливість отримувати результати виконання завдання та тесту учнями.

Учні мають мати змогу отримувати завдання від учителя, проходити відповідний тест, відправляти розв'язки завдання і переглядати результати тестів.

Сервіс має надавати повноцінну інформацію про тести, створені вчителями для учнів. Він повинен бути зручним та швидкісним у використанні.

1.2 Існуючі програмні рішення для перевірки і оцінювання знань

З кожним роком все більше і більше набуває поширення різноманітних веб-сервісів, які вирішують дуже великий обсяг різнопланових завдань. Одним із таких завдань є розробка веб-сервісу для перевірки і оцінювання знань.

Під час пошуку інформації, дослідження особливостей вирішення ідентичних та аналогічних задач за обраним напрямком дипломного проекту, було виявлено, що на теперішній час такі сервіси є не досить простими у використанні, або не в повному обсязі реалізують поставлену задачу.

Одним із таких веб-сервісів є Google Classroom – безплатний веб-сервіс розроблений компанією Google для освітніх закладів із метою полегшення створення та поширення завдань дистанційним шляхом. Головна мета сервісу – пришвидшити процес розповсюдження інформації між вчителями та учнями [1].

Запросити учнів вступити до класу можна двома способами: за допомогою приватного коду або автоматично імпортувати зі шкільного домену. Кожний клас створює свій каталог, де учень може скинути роботу, яку вчитель матиме змогу перевірити. Вчителі можуть стежити за ходом роботи кожного учня, а після оцінювання роботи, повернути роботу разом із коментарями.

Також існує мобільний додаток Classroom, який доступний на iOS та Android. Логотип веб-сервісу Google Classroom зображений на рисунку 1.1.

Рисунок 1.1 – Логотип Google Clasroom

На Урок – це Український освітній портал для вчителів. Тут вчителі мають змогу ділитися інформацією, розробляти для учнів різноманітні тестування та завдання. Логотип освітнього порталу На Урок зображений на рисунку 1.2

Рисунок 1.2 – Логотип На Урок

Courséra – американська онлайн-платформа для навчання, заснована в 2012 році професорами Стенфорда Ендрю Нґ та Дафни Коллер, яка пропонує широкі відкриті онлайн-курси, спеціалізації та ступені [2].

Coursera пропонує користувачам безліч безкоштовних онлайн-курсів із різноманітних дисциплін, у випадку успішного закінчення цих курсів користувачеві видається відповідний сертифікат.

Також ця онлайн-платформа співпрацює із багатьма вищими навчальними закладами для викладання цих онлайн курсів. Зараз Coursera пропонує курси в різних сферах освіти.

Всі курси на онлайн-платформі доступні на безкоштовній основі, але якщо користувач хоче отримати сертифікат, то потрібно заплатити 30-40 доларів. Тривалість проходження навчання різна. Під час навчання студент повинен переглядати лекції, читати різноманітну інформацію, яка надається адміністратором курсу. Також студент повинен виконувати домашні завдання. Для перевірки знань студент проходить тестування і виконує різноманітні завдання. Із 2013 року деякі курси субтитруються українською мовою

Щоб уникнути різні типи шахрайства було розроблено “Кодекс Честі”, із ним користувач ознайомлюється при реєстрації.

Логотип онлайн-платформи Coursera зображений на рисунку 1.3

Рисунок 1.3 – Логотип Coursera

МійКлас – українська електронна інформаційно-освітня система. Функціонал системи дозволяє вчителям використовувати вже розроблені завдання шкільної програми з 1 по 11 класи або розробити власну навчальну програму для дистанційного навчання учнів [3].

МійКлас – це відкрита система, нею можна користуватись будь-коли, будь-де та на будь-якому пристрої, який має підключення до інтернету. На сайті є ІКТ-курс, що дає змогу вчителям самостійно і за короткі терміни впровадити дистанційне навчання.

Основні функції інформаційно-освітньої системи МійКлас:

- різні завдання із предметів інваріантної складової навчальних планів 1-11 класів;
- можливість вчителя впроваджувати дистанційне навчання;
- можливість дистанційного проведення домашніх, самостійних та контрольних робіт;
- отримання адміністрацією закладу освіти звіту про використання даної системи вчителями та учнями;
- контроль з боку батьків.

Дана система дає змогу впроваджувати ігрові технології у навчальний процес. Учні завжди змагаються за першість серед однокласників, що посилює їхню зацікавленість в навчанні та підвищує успішність на 20%.

Логотип інформаційно-освітньої системи МійКлас зображений на рисунку 1.4

Рисунок 1.4 – Логотип інформаційно-освітньої системи МійКлас

Основними недоліками цих сервісів є складний інтерфейс користувача та їх громіздкість. Також вони специфічні у плані налаштування.

Оскільки веб-сервіси для перевірки і оцінювання знань, а також дистанційного навчання є досить затребувані, то розробка зручного, зрозумілого, гнучкого та невимогливого такого сервісу є актуальним завданням.

1.3 Поняття веб-сервісу для перевірки і оцінювання знань у навчальному процесі

1.3.1 Поняття веб-сервісу

Веб-сервіс – це система, доступна в інтернет-просторі, що працює на основі спеціальної програми, ідентифікація якої виконується за допомогою URL-рядка. Основним завданням є взаємодія програмних систем на різних платформах, для чого використовуються відкриті протоколи [4]. Взаємодія клієнта з веб-сервісом зображена на рисунку 1.4

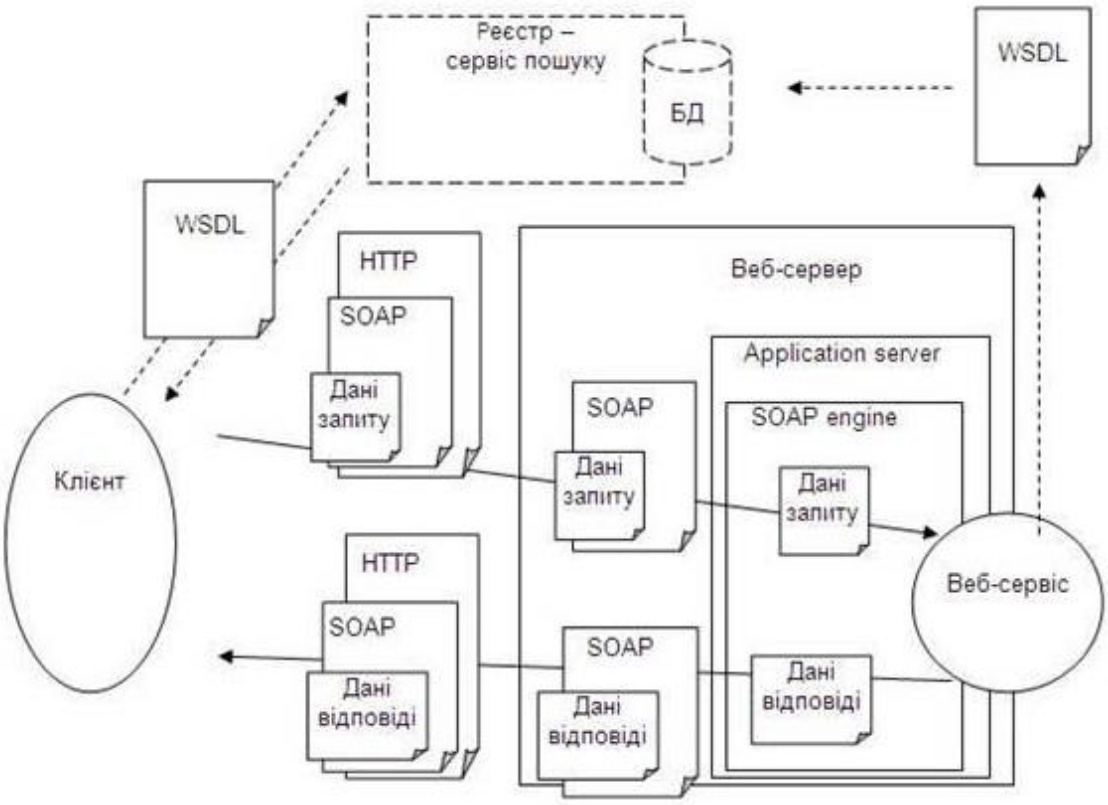

Рисунок 1.4 – Взаємодія клієнта з веб-сервісом

Стандарти, які використовуються веб-сервісами:

- XML – мова розмітки, яка визначає набір правил для кодування документів у форматі, який читається людиною і машиною. Цей формат досить гнучкий, щоб бути придатним для використовування в різних галузях. Іншими словами, даний стандарт визначає метамову, на базі якої шляхом впровадження обмежень на структуру і зміст документів визначаються специфічні, предметно-орієнтовані мови розмітки даних [5]. Структура XML-документа зображена на рисунку 1.5

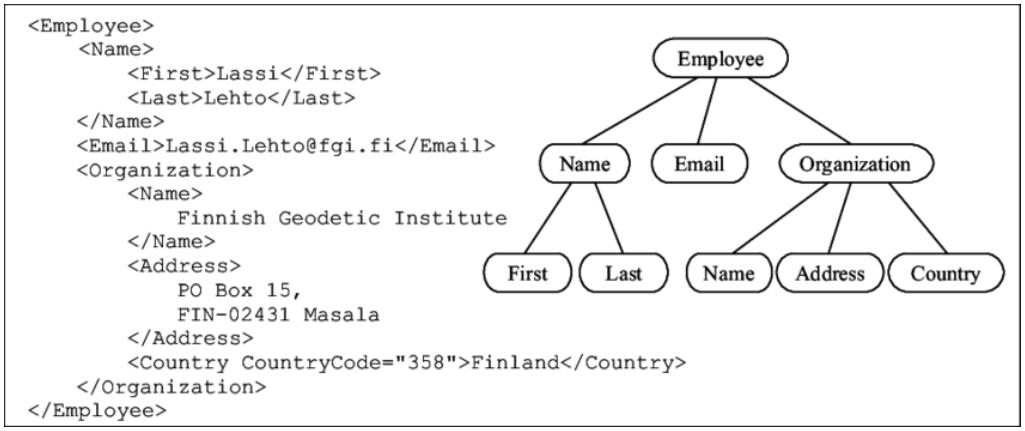

Рисунок 1.5 – Структура XML-документа

- SOAP – протокол обміну структурованими XML-повідомленнями у розподілених обчислювальних системах [6]. Повідомлення SOAP складається із: SOAP-конверта, SOAP-заголовка, тіла SOAP і даних про імена користувачів. Заголовок передає дані про запит, визначений у тілі SOAP. Тіло містить запит Web-сервісу чи відповідь на нього. Структура протоколу SOAP зображена на рисуну 1.6

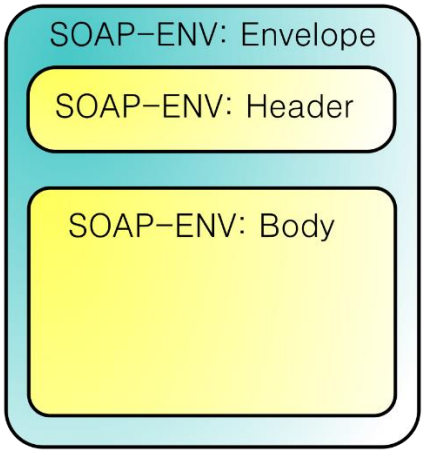

Рисунок 1.6 – Структура SOAP-протоколу

- WSDL – мова визначення інтерфейсу веб-сервісу заснована на XML, що описує функціональність веб-сервісу і спосіб доступу до нього. Файл WSDL використовується, щоб описати, що робить веб-сервіс, і надає клієнту всю інформацію, необхідну для підключення до веб-сервісу та використання всіх функцій, наданих веб-сервісом [7].

  Структура WSDL-файлу:

  1. Definition
  2. Target Namespace
  3. DataTypes
  4. Messages
  5. Porttype
  6. Bindings
  7. Service

- UDDI – платформово-незалежний механізм для розміщення, публікації, пошуку описів веб-сервісів. Він підтримує різноманітні види описів сервісів. UDDI насправді являє собою веб-сервіси, які дозволяють користувачам реєструвати інтерфейси на вузлі, переглядати, провіряти, зв’язуватись до зареєстрованих сервісів [8]. Основна архітектура UDDI-реєстру відносно запитів зображена на рисунку 1.7

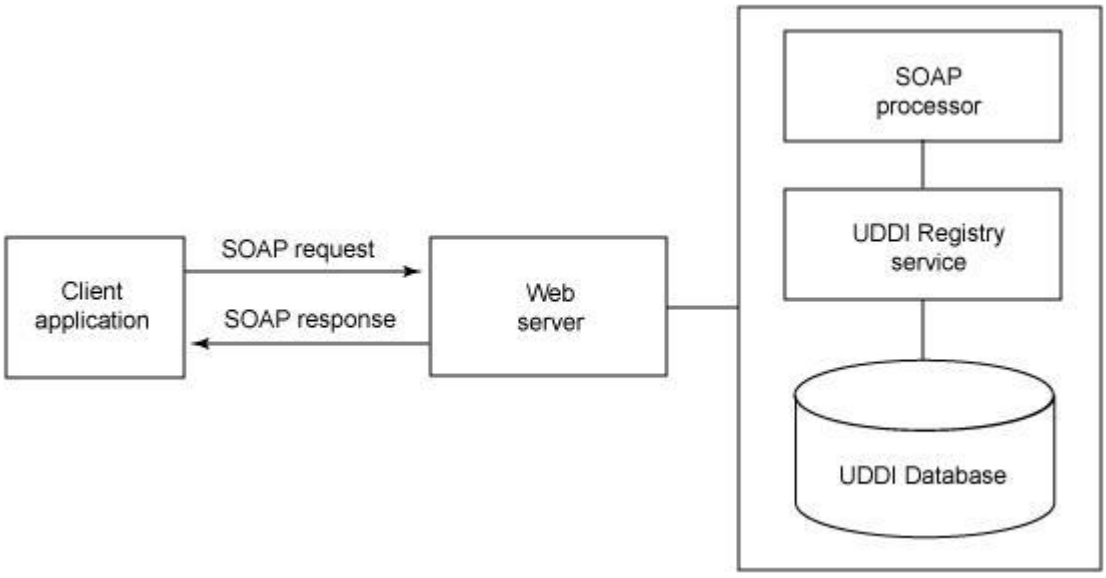

Рисунок 1.7 – Структура UDDI-реєстру

1.3.2 Веб-сервіси у навчальному процесі

Навчальний процес являє собою систему організаційних та дидактичних заходів, що складається із деяких елементів, має складові, відповідно її функціям, і тенденції до невпинного саморозвитку й самовдосконалення. Також ця система реалізовує сам зміст освіти на відповідному освітньому рівні відповідно до державних стандартів освіти

Навчальний процес розробляється відносно можливостей сучасних технологій і рівняються на формування грамотної, освіченої, розвиненої особи, яка здатна до постійного навчання. В України склалась така ситуація у освіті, що інформаційні технології стали складовими нового освітнього середовища. Використання новітніх веб-сервісів дає змогу суттєво модернізувати систему освіти [9].

Освітні веб-сервіси виконують дидактичні функції, які містять у собі:

- засоби збереження змісту навчання та його відтворення;
- можливість перегляду учнями навчального матеріалу;
- середовище, де учні можуть завантажувати виконані завдання;
- засоби організації тестування та керування навчальним процесом.

Використання новітніх веб-сервісів надає можливість суттєво покращити систему освіти, а значить, її подальша модернізація є незворотною та неминучою. Освітні веб-сервіси зараз є найкращим інструментом покращення професійної підготовки нових педагогів.

Все більше й більше набувають значення освітні веб-сервіси. Розробка таких сервісів дає змогу навчальним закладам проводити навчання дистанційно.

Функціональні призначення освітніх веб-сервісів поділяються на:

- програмні засоби;
- довідкові;
- педагогічні;
- навчально-методичні;
- наукові;

- навчальні;
- нормативні.

Також освітні веб-сервіси відкривають нові можливості до взаємодії вчителя і учня та дозволяють:

- інтерактивно доносити інформацію до учнів, незалежно від того, де вони перебувають;
- дистанційно задавати та перевіряти завдання;
- проводити тестування учнів;
- проводити інтернет конференції.

Освітні веб-сервіси є оптимальним способом навчання у непередбачуваних ситуаціях, таких як оголошення карантину або виникнення надзвичайних ситуацій, які не дозволяють стаціонарно навчатись у закладах загальної середньої освіти [10].

Використання інформаційних технологій у галузі освіти і у діяльності вчителів стало необхідним. Освітні веб-сервіси мають стати для педагогічної громади пріоритетним засобом та способом самоосвіти. Ці сервіси є надзвичайно ефективним елементом регіональної освіти. Одним із способів покращення якості освіти є покращення контролю знань. Здійснення контролю знань у навчальному процесі має на увазі об’єктивне оцінювання учнів вчителями.

Перевірка та оцінювання знань учнів є активним процесом. Вчитель не лише реєструє фактичні знання учнів, а й робить внесок у результат навчального процесу. Завдання вчителя – це знайти найефективніший спосіб перевірки й оцінювання знань, щоб об’єктивно оцінити знання учнів [11].

Одним з таких способів є тестування учнів за допомогою веб-сервісів. Веб-сервіс для перевірки та оцінювання знань – це сервіс, що призначений для об’єктивного оцінювання знань учнів та вимірювання їх знань. Переваги тестування на веб-сервісі:

- об’єктивність;
- універсальність;
- простота оцінювання;

- кількісний критерій оцінювання;
- чіткість завдань;
- можливість одночасно перевірити декілька учнів;
- реальним є самоконтроль;
- багатофункціональність.

Тестування знань є одним з елементів покращення якості освіти. Це призводить до таких висновків:

- різноманіття форм та методики контролю якості знань учнів є актуальним завдання, вирішення якого викликане потребами учительської практики та намаганнями інтеграції української освітньої системи з європейською;
- використання тестових завдань у комп'ютерній формі є одним із пріоритетних завдань в українській школі;
- впровадження у освітній процес національних навчальних розробок втілюється поступово завдяки співпраці вчителів;
- процес старіння освітньої інформації має потребу від вчителів постійного модернізування наукового матеріалу та тестових завдань;
- учнів закладів загальної середньої освіти потрібно підготувати до проходження тестування систематично.

Отже, проходження тестування за допомогою веб-сервісів – це шлях до покращення та модернізування навчального процесу.

Не менш важливим фактором модернізувати навчальний процес є можливість дистанційного навчання.

Дистанційне навчання – сукупність сучасних технологій, що забезпечують доставку інформації в інтерактивному режимі за допомогою використання ІКТ від тих, хто навчає (вчителів, викладачів, визначних постатей у певних галузях науки, політиків), до тих, хто навчається (учнів, студентів чи слухачів). Застосовується під час підготовки як у закладах загальної середньої освіти і ВНЗ, так і в бізнес-школах. Основними принципами дистанційного навчання є інтерактивна взаємодія у процесі роботи, надання студентам можливості самостійного освоєння досліджуваного матеріалу .

Також консультаційний супровід у процесі дослідницької діяльності. Дає змогу навчатися на відстані, за допомогою диспутів експертів із кількох країн, за відсутності викладача. Основну роль у здійсненні дистанційного навчання відіграють сучасні інформаційні технології [12].

Проте якісний показник освіти залежить від якості роботи вчителя. Отже, питання контролю та оцінки якості роботи вчителя є одним з важливих завдань в управлінні якістю освіти. І однією з важливих ознак кваліфікації вчителя є його володіння ІКТ. Кожен вчитель повинен на високому рівні володіти ІКТ, щоб мати змогу організувати дистанційне навчання. Виходячи із ситуації, яка склалась не тільки в Україні, але в усьому світі, можемо зробити висновок, що дистанційне навчання вийшло на одне з провідних місць в організації навчання здобувачів загальної середньої освіти.

Також важливим є вміння вчителя дистанційно оцінити знання і вміння учнів. Постійний контроль знань та вмінь учнів значно підвищує їх мотивацію до навчання та якість освітнього процесу. Проте, щоб провести тестування чи опитування, вчителі повинні виконати великий об’єм роботи для формування завдань та організації процесу. А особливо важко це зробити в умовах дистанційного навчання. Для вирішення цієї проблеми вчителі вдаються до використання веб-ресурсів для організації тестів і опитувань.

2 МЕТОДИ ТА ЗАСОБИ РЕАЛІЗАЦІЇ ВЕБ-СЕРВІСУ

2.1 Вибір архітектури веб-сервісу

Проаналізувавши поставлену задачу та способи її вирішення, я вирішив реалізувати поставлену задачу за допомогою веб-технологій. Головною перевагою веб-сервісу над іншими варіантами є його універсальність, можливість використання на різних пристроях без завантаження на операційну систему.

Для вирішення поставленого завдання було вирішено застосувати трирівневу архітектуру, що складається з трьох компонентів: база даних, сервер та клієнт. Схема даної триланкової архітектури зображена на рисунку 2.1.

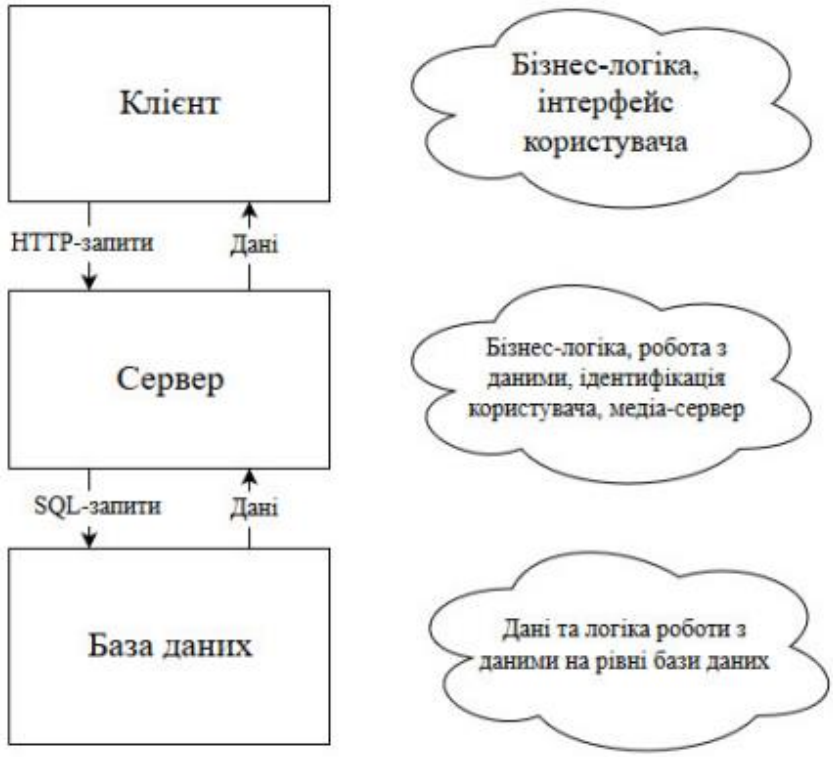

Рисунок 2.1 – Трирівнева архітектура веб-сервісу

Основним центром трирівневої архітектури є сервер. Він розміщується на другому рівні. У сервері розміщена основна бізнес-логіка, а також логіка доступу до БД. Завдяки серверу відбувається ідентифікація користувача, щоб надати йому індивідуальний доступу до веб-застосунку [13].

Сервер – це єдиний зв’язок між базою даних та користувачем, щоб унеможливити використання даних не за призначенням або їх пошкодження. Щоб користуватися веб-застосунком, потрібно авторизуватись у системі, ось чому дана логіка реалізовується на рівні сервера, оскільки на рівні користувача можлива втрата даних, підміна прав доступу чи інакші методи доступу до даних.

Клієнт – це інтерфейсний компонент, він являє собою перший рівень трирівневої архітектури. Перший рівень не має бути навантажений бізнес-логікою, він не повинен мати зв’язків з БД, також він має зберігати стан веб-застосунку. За допомогою інтерфейсу відбувається налаштування веб-сервісу і перегляд результату роботи. На користувацькому рівні може бути найпростіша бізнес-логіка: відбуватися попередня обробка даних, перевірка значень, алгоритми шифрування та деякі нескладні операції. А також на цьому рівні відбувається перша стадія автентифікації користувача, щоб обмежити неконтрольований доступ до веб-застосунку.

Не менш важливим завданням на рівні бази даних є забезпечення зберігання даних, яке виноситься на третій рівень трирівневої архітектури веб-сервісу. Переважно це реляційна чи об'єктно-орієнтована система управління базами даних. На цьому рівні також забезпечується цілісність даних шляхом зовнішніх зв’язків і ключів. На третьому рівні бази даних можна реалізувати деяку бізнес-логіку, для якої не потрібно використання інших джерел даних, крім самої бази даних та її таблиць.

2.2 Опис архітектури веб-сервісу

Для реалізації веб-сервісу використовувався API. Прикладний програмний інтерфейс – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення. Спрощено - це набір чітко визначених методів для взаємодії різних компонентів. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек [14].

За допомогою прикладного програмного інтерфейсу здійснюється взаємодія між користувачем та додатком. API-підхід – являє собою архітектурний підхід програмного інтерфейсу для різних програм, що обслуговують різні типи користувачів.

При використанні у контексті веб-розробок, прикладний програмний інтерфейс визначається як набір специфікацій, таких як повідомлення запитів протоколу передачі гіпертексту (HTTP), а також визначається структура відповідей повідомлень, як правило, в розширенні мові розмітки (XML) або у форматі нотації JavaScript. Незважаючи на те, що прикладний програмний інтерфейс історично став синонімом веб-служби, заснованої на простому об'єктному доступі (SOAP) та сервіс-орієнтованої архітектури для більш прямих передач репрезентативного стану (REST) та ресурсів орієнтованої архітектури ресурси (ROA). Прикладний програмний інтерфейс дозволяє поєднувати декілька API в нові додатки, відомі як гібридні. Також API дозволяє користувачам полегшувати обмін даними. Наприклад, API REST Twitter дозволяє розробникам отримувати доступ до основних даних Twitter, а API-пошуку надає способи взаємодії розробників із пошуковими запитами та даними про тенденції.

2.3 Опис архітектури клієнтського веб-сервісу

Для реалізації клієнтського веб-сервісу було використано шаблон MVC.

Статична сторінка на HTML не вміє реагувати на дії користувача. Для двосторонньої взаємодії потрібні динамічні веб-сторінки. MVC – ключ до розуміння розробки динамічних веб-застосунків, тому розробнику потрібно знати цю модель.

MVC розшифровується як модель-вид-контролер. Це спосіб організації коду, який передбачає виділення блоків, що відповідають за вирішення різних завдань. Один блок відповідає за дані додатки, інший відповідає за зовнішній вигляд, а третій контролює роботу додатка. Компоненти MVC:

- Модель – це компонент відповідає за дані, а також визначає структуру програми. Наприклад, якщо ви створюєте додаток, код компонента моделі визначатиме список завдань і окремі завдання.
- Вигляд – це компонент відповідає за взаємодію з користувачем. Тобто код цього компонента визначає зовнішній вигляд програми і способи його використання.
- Контролер – цей компонент відповідає за зв'язок між моделлю та виглядом. Код контролера визначає, як сайт реагує на дії користувача. По суті, це мозок MVC шаблону.

MVC шаблон (рис. 2.2) – підхід до проектування веб-застосунків, який передбачає виділення коду в блоки моделі, вигляду і контролера. Контролер обробляє вхідні запити. Модель дістає з бази даних інформацію, потрібну для виконання конкретних запитів. Подання визначає результат запиту, який отримує користувач.

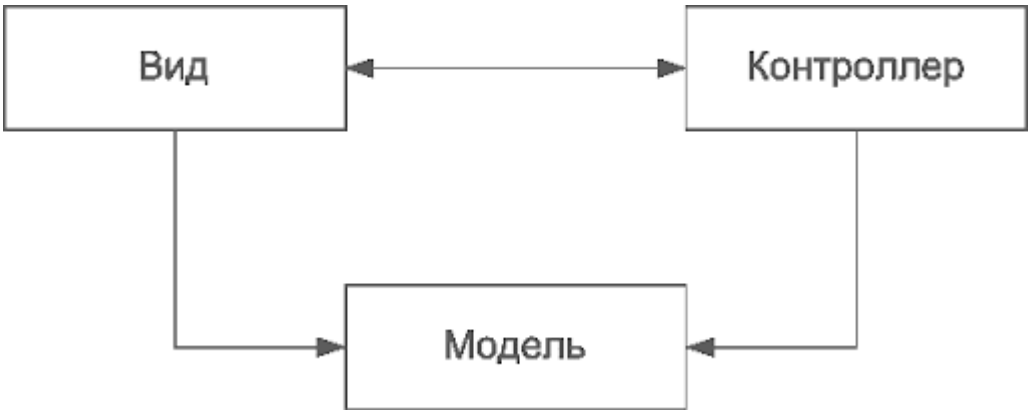

Рисунок 2.2 – Шаблон MVC

Також шаблон MVC не тільки не прив'язаний до якоїсь конкретної мови програмування, він також не прив'язаний і до використовуваної парадигмі програмування. Тобто, ви цілком можете проектувати свій додаток по MVC, при цьому не застосовуючи ООП.

Популярні мови програмування, такі як JavaScript, Python, Ruby, PHP, Java, C# і Swift, мають MVC шаблон, який використовується для розробки веб- або мобільних додатків.

2.4 Опис інструментів розробки

Цей веб-сервіс розроблений за принципами трирівневої архітектури побудови програм. Кожен рівень реалізовано із застосуванням різноманітних технологій.

Для клієнтського рівня я використав такі технології: мова програмування CoffeeScript, для покращення зовнішнього вигляду – Bootstrap, мова шаблонів Slim.

Для серверного рівня у проекті використано такі технології: мова програмування Ruby, фреймворк Ruby on Rails, веб-сервер Puma.

Для рівня бази даних було обрано PostgreSQL

2.4.1 Прості форми, мова шаблонів Slim та Bootstrap

Проста форма має на меті бути максимально гнучким, одночасно допомагаючи потужним компонентам створювати форми. Основна мета простої форми – не чіпати свій спосіб визначення макету, дозволяючи вам знайти кращий дизайн для своїх очей.

Щоб підключити прості форми до проекту, потрібно підключити додаткову бібліотеку за допомогою gem 'simple_form'.

Проста форма була розроблена так, щоб вона була налаштована так, як вам потрібно. В основному це стек компонентів, які викликаються для створення повного вводу HTML для вас, який за замовчуванням містить мітку, підказки, помилки та сам вхід.

Приклад простих форм:

```
<%= simple_form_for @user do |f| %>
  <%= f.input :username %>
  <%= f.input :password %>
  <%= f.button :submit %>
<% end %>
```

Slim – мова шаблонів, мета якої істотно зменшити синтаксис. Він розпочався як вправа, щоб побачити, скільки можна видалити зі стандартного HTML-шаблону. Оскільки все більше і більше людей викоримтовували Slim, функціональність зростала, а також зростала гнучкість синтаксису.

Основні переваги:

1. Елегантний синтаксис:
   - налаштування ярликів швидкого доступу;
   - короткий синтаксис без закриття тегів.
2. Безпека:
   - автоматичне вимкнення HTML за замовчуванням;
   - підтримка html_safe? Rails.
3. Розширюється за допомогою наступних плагінів;
4. Висока продуктивність:
   - порівняльна швидкість з ERB / Erubis;
   - потокова підтримка в Rails.
5. Підтримується всіма основними фреймворками (Rails, Sinatra);
6. Повна підтримка тегів та атрибутів Unicode;
7. Вбудовані двигуни такого типу, як Markdown та Textile.

Щоб підключити Slim потрібно додати до файлу Gemfile gem 'slim-rails' або виконати команду gem install slim.

Для покращення зовнішнього вигляду веб-сервісу було використано Bootstrap 4. Bootstrap – це інструментарій із відкритим кодом для розробки за допомогою HTML, CSS та JS.

Bootstrap, спочатку названий Twitter Blueprint, був розроблений Марком Отто та Джейкобом Торнтоном. Після декількох місяців розвитку невеликою групою, багато розробників почали робити свій внесок. Він був перейменований на Bootstrap і випущений [15].

Щоб підключити Bootstrap до Ruby om Rails потрібно встановити gem 'bootstrap'.

2.4.2 Мова програмування Ruby

Ruby – це динамічна мова програмування з відкритим кодом високого рівня та з акцентом на простоту та продуктивність. Вона має елегантний синтаксис, який є природним для читання та легким для запису. Це інтерпретована мова програмування, яка була розроблена для продуктивності програмістів, щоб зробити програмування цікавим [16].

Ця мова дотримується принципу “найменшої несподіванки”: застосунок має поводити себе так, як цього хоче програміст. Також Ruby успадкував ідеологію мови Perl у частині можливості програміста вирішити завдання декількома різними способами. Однією з основних цілей розробки було звільнення програмістів від рутинної роботи, яку комп’ютер може виконувати швидше і якісніше. Особлива увага, зокрема, приділялася буденним рутинним заняттям (обробка текстів, адміністрування), і для них мову налаштована особливо добре.

Ruby працює на багатьох платформах, включаючи Linux та безліч ароматів UNIX, MS-DOS, Windows 9x / 2000 / NT, BeOS та MacOS X. Також Ruby добре підходить для таких проблемних доменів:

- обробка тексту – класи файлів, рядків та Regexp допомагають швидко та чисто обробляти текстові дані.
- програмування CGI – у Ruby є все необхідне для програмування CGI, включаючи класи обробки тексту, бібліотеку CGI, інтерфейс бази даних і навіть eRuby та mod_ruby для Apache.
- мережеве програмування може бути цікавим за допомогою добре розроблених різних пакетів класів.
- програмування графічного інтерфейсу – доступні інтерфейси GUI-інструментів, такі як Ruby / Tk and Ruby / Gtk.
- програмування XML – функції обробки тексту та функціональний механізм регулярних виразів UTF-8 роблять програмування XML зручним у Ruby.

- прототипування – завдяки високій продуктивності, Ruby часто використовується для виготовлення прототипів. Прототипи іноді стають виробничими системами шляхом заміни вузьких місць на письмових розширеннях C.

Можливості мови програмування Ruby:

- має лаконічний і простий синтаксис;
- підтримується додавання конкретного екземпляру та методів у клас під час виконання програми;
- дозволяє перевизначати оператори, які насправді є методами;
- цілком об'єктно-орієнтована мова програмування. Все у Ruby є об'єктом, крім керуючих конструкцій. Наприклад, число 1 – екземпляр класу Fixnum;
- не підтримує множинне успадкування, але замість нього можна використовувати концепцію “домішок”, засновану на механізмі модулів;
- створювати розширення для Ruby на C дуже просто частково через збирача сміття, частково через нескладне і зручне API;
- можна використовувати блоки коду. Блоки можна використовувати у методах або перетворювати у замикання;
- не вимагає оголошення змінних, але для інтерпретатора бажано, щоб змінним присвоювалося пусте значення nil;
- можна динамічно завантажувати розширення, якщо це дозволяє операційна система;
- дозволяє обробляти виключення;
- містить автоматичний збирач сміття, працює для всіх об'єктів Ruby, в тому числі для зовнішніх бібліотек;
- підтримує замикання з повною прив'язкою до змінних;
- Цілі змінні в Ruby автоматично конвертуються між типами Fixnum (32-розрядні) і Bignum (більше 32 розрядів) в залежності від їхнього значення, що дозволяє обробляти цілочисельні математичні розрахунки із будь-якою точністю.

- у Ruby безпосередньо реалізовано багато шаблонів проектування, так, наприклад, singleton може бути реалізований додаванням необхідних методів до одного конкретного об'єкту;
- має незалежну від ОС підтримку не витискаючи багатопотоковість.

Мову програмування Ruby розробив Юкіхіро Мацумото, перша публічна версія була випущена у 1995 році. Після цього мова Ruby привернула до себе увагу програмістів. На даний час Ruby входить у десятку найпопулярніших мов програмування (рис. 2.3).

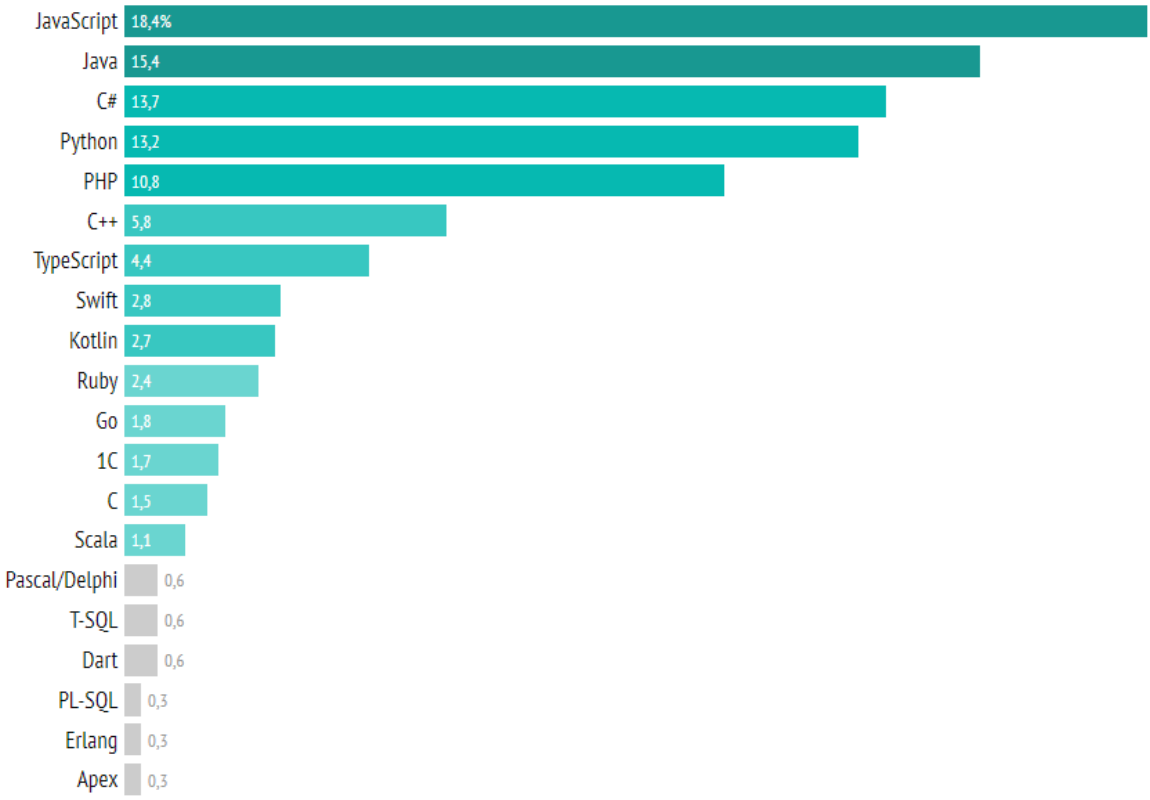

Рисунок 2.3 – Популярність мов програмування

Ruby дуже гнучка мова програмування. Вона дозволяє користувачам як завгодно міняти її частини. Ruby намагається ні в чому не обмежувати користувача. Оператори у Ruby – це синтаксичний цукор для методів. Їх, також, можна перевизначати. Блоки – дуже виразна та гнучка конструкція у Ruby. Програміст має змогу додати замикання до любого методу, описуючи дії цього метода. Також у Ruby не потрібно оголошувати змінні. Творці Ruby, випускаючи оновлення як мінімум двічі на рік, уберігають цю мову від смерті і стежать за її постійним розвитком [17].

На сьогоднішній час останньою стабільною версією є Ruby 2.7.1. Але різні проекти написані на різних версіях Ruby, щоб здійснити контроль версій Ruby використовують RVM.

RVM (Ruby Version Manager) – програмна платформа для unix-подібних операційних систем, призначена для керування кількома установками Ruby на одному пристрої.

Основні завдання:

1. Фізичне розділення версій ruby і наборів gemsets.
2. Можливість мати кілька версій ruby і перемикатися між ними.
3. Можливість для кожної версії ruby мати кілька gemsets-наборів і перемикатися між ними

Завдяки своєму інтуїтивному, простому і читабельному синтаксису, Ruby прекрасно підходить для стартапів і будь-яких компаній, які хотіли б якомога швидше постачати і розширювати свої продукти програмного забезпечення. Більш швидка розробка означає велику економію, а це важливо для стартапів з обмеженим бюджетом. Завдяки цьому більше грошей можна вкласти в розробку додаткового функціоналу, в маркетинг.

Великомасштабні проекти, в свою чергу, можуть використовувати Ruby on Rails як інструмент для прототипування. Оскільки створювати і розширювати програми на Ruby відносно легко і дешево, це також прекрасний варіант для створення внутрішніх інструментів, для яких продуктивність не є найвищим пріоритетом.

2.4.3 Фреймворк Ruby on Rails

Ruby on Rails – найпопулярніший фреймворк з відкритим кодом для веб-сервісів. Він створений за допомогою мови програмування Ruby.

Фреймворк – це сукупність коду, інструментів та утиліт, які дають вам конкретну структуру, з якою можна працювати, коли ви пишете програмне забезпечення.

Навідоміші веб-сайти розроблені на Ruby on Rails:

- GitHub
- Twitter
- Shopify
- Ask.fm
- Redmine
- Kickstarter
- Instacart
- ConvertKit
- Twitch
- Zendesk
- SoundCloud

Переваги використання RoR:

- це комплексне рішення, надає вам все необхідне для створення веб-програми
- Rails конвенції та конфігурація за замовчуванням економить вам багато роботи!
- Rails має чудову екосистему, тому ви можете знайти всі інструменти та підтримку, які вам можуть знадобитися
- Це активно розвивається, тому ви регулярно отримуєте виправлення та нові функції!

Ruby on Rails – це впевнена структура. Однією з таких думок є те, що конвенція має бути важливішою за конфігурацію. Це означає, що вам доведеться приймати менше рішень, оскільки творці Rails вже прийняли їх за вас. Оскільки вам доведеться приймати менше рішень, ви будете більш продуктивними і швидше все буде зроблено, але якщо ви хочете змінити деякі з них, ви можете.

Rails приймає запити, спрямовує їх на відповідні дії, які потім взаємодіють із базою даних (через ActiveRecord) для виконання запиту [18]. Потім повертає результати (HTML або JSON) назад користувачеві (рис. 2.4). Rails використовує архітектуру MVC.

Рисунок 2.4 – робота Rails у веб-застосунку

Дії організовані в контролери, контролери приймають рішення про те, як обробити запит, і вони запитують у базі даних будь-які потрібні їй дані. Потім контролер надає подання. Перегляд – це дизайн та зміст сторінки. Це кінцевий продукт, який буде повернутий користувачеві. Схема роботи Rails застосунку зображена на рисунку 2.5

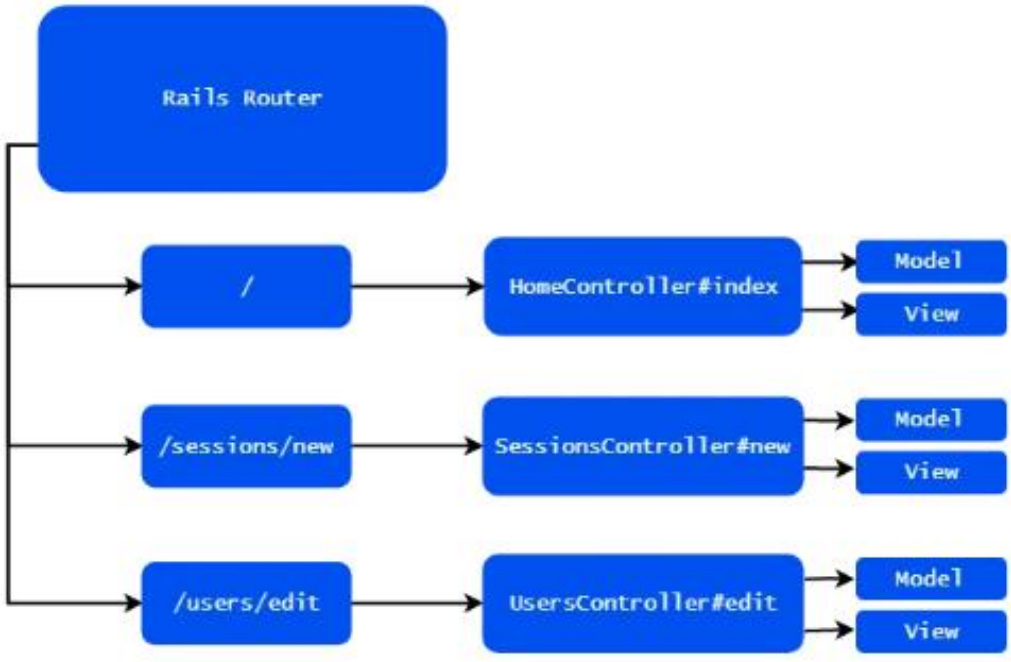

Рисунок 2.5 - Схема роботи Rails застосунку

Ruby on Rails розобив Девід Гайнемаєр Генссон і був випущений у липні 2004 року. Ruby on Rails є відкритим програмним забезпеченням і розповсюджується за ліцензією MIT.

Під час розробки додатків на RoR, використання gem не рідкість. Gem – це готовий набір класів, що полегшує роботу програміста.

Таким чином програміст зосереджується на основних тонкощах своєї роботи. Як результат – збільшується швидкість розробки. Gem – це архів, усередині якого, крім самої бібліотеки знаходиться файл специфікації, який крім усього іншого включає в себе інформацію:

- версію бібліотеки;
- залежності від інших гемів;
- опис гема;

Геми створюються незалежними програмістами, при розробки типових задач. Встановлюються в систему однойменної утилітою gem.

Основні геми, які я використав у проекті:

- gem 'pg' – інтерфейс Ruby до бази даних PostgreSQL;
- gem 'sass-rails' – забезпечує інтеграцію проектів Ruby on Rails з мовою таблиці стилів Sass;
- gem 'turbolinks'– робить швидше переміщення веб-додатків;
- gem 'jquery-ui-rails' – підключає пакет jQuery (JavaScripts, таблиці стилів та зображення) для Ruby on Rails;
- gem 'bootstrap' – підключає Bootstrap 4;
- gem 'slim-rails' – пропонує генератори Slim для Rails;
- gem 'simple_form' – підключає Simple form для Ruby on Rails;
- gem 'cocoon' – полегшує обробку вкладених форм;
- gem 'tinymce-rails' – інтегрує редактор TinyMCE;
- gem 'breadcrumbs_on_rails' – це простий плагін Ruby on Rails для створення та керування навігацією паніруванння для проекту Rails;
- gem 'momentjs-rails' – підключає Moment.js (легка бібліотека дат JavaScript для розбору, маніпулювання та форматування дат);
- gem 'draper' – додає об'єктно-орієнтований шар логіки презентації у додаток Rails;
- gem 'ransack' – дозволяє створити як прості, так і розширені форми пошуку для вашої програми Ruby on Rails;
- gem 'devise' – це гнучке рішення для аутентифікації користувача;

- gem 'authority' – допомагає вам авторизуватися у вашому додатку Ruby;
- gem 'carrierwave' – забезпечує простий та надзвичайно гнучкий спосіб завантаження файлів;
- gem 'acts_as_list' – надає можливості для сортування та упорядкування декількох об’єктів у список;

Оскільки геми різних версій можуть конфліктувати, під кожен проект прийнято створювати окремі гемсети, в яких міститься набір певних гемів, з певними версіями, заточеними під потрібну версію Ruby і RoR.

2.4.4 Поняття Active Record

Active record (AR) – шаблон проектування додатків. AR є популярним способом доступу до даних реляційних баз даних в об'єктно-орієнтованому програмуванні.

Active record стає основною частиною більшості програм програмування. Це робить більш простими завдання CRUD, можливість їх швидшої розробки. Наприклад, замість того, щоб писати багато SQL для вставки, оновлення та видалення багатьох загальних і простих об'єктів даних, AR дозволяє просто призначити значення об'єкту даних та виконати команду, наприклад, $ object-> save(), SQL складається і виконується для вас.

Більшість фреймворків також реалізують відносини даних у відповідних моделях Active Record, що може значно спростити доступ до даних, пов’язаних із вашим об'єктом.

Переваги Active record:

- викликати методи об’єкта набагато простіше, ніж писати власні оператори SQL. Це набагато полегшує написання запитів даних із бази даних, особливо це стосується приєднання таблиць;
- більшість поширених запитів можна вирішувати автоматично за допомогою Active Record. Це дозволяє Active Record писати найкращий запит для цих загальних операцій;

- Active Record – це абстракція, яка сидить над шаром SQL. Це означає, що якщо ви переходили на базовий механізм бази даних, вам не доведеться міняти свій код.

Шаблон Active record включає в себе один рядок бази даних в об'єкті. Це дозволяє дуже легко створювати, оновлювати та видаляти дані з бази даних. Ми також можемо використовувати клас Active Record для запиту даних із бази даних. Можливість запиту, створення, оновлення та видалення з одного об’єкта – одна з причин, чому Active Record робить роботу з базою даних у веб-додатках такою простою!

2.4.5 Веб-сервер Puma

Puma – порівняно новий веб-сервер, який може робити і багатопроцесність, і багатопоточність. Від повільних клієнтів він не страждає, оскільки може обслуговувати кількох одночасно. Heroku рекомендує на своїх серверах користуватися саме ним.

Puma – це високопродуктивний веб-сервер для додатків Ruby. Він заснований на веб-сервері Mongrel, який свого часу додав кілька революційних функцій і багато в чому вплинув на розробку додатків Ruby. Розробник Puma трансформував операційну структуру Mongrel, перейшовши на Rack (і, таким чином, усуваючи деякі проблеми з продуктивністю), і розробив додаток для підтримки паралелізму.

Особливості веб-серверу Puma:

- Puma не вимагає багато місця і використовує мало ресурсів;
- веб-сервер Puma надає кілька режимів роботи: він дозволяє задати мінімальну та максимальну кількість потоків, а також підтримує кластерний режим, в якому ви можете використовувати розгалужені процеси для одночасної обробки запитів;
- він заснований на Mongrel і багато в чому успадкував його код;
- веб-сервер Puma розроблений для Rubinius, але також може працювати і з JRuby;

- він надає простий, але досить великий і гнучкий набір конфігурацій, що дозволяє легко підготувати веб-сервер як до розробки, так і до оточення;
- “з коробки” Puma не підтримує розміщення і виробництва декількох додатків, але це можна зробити за допомогою спеціального інструменту Jungle.

На відміну від інших веб-серверів Ruby, Puma був побудований для швидкості та паралелізму. Puma – це невелика бібліотека, яка забезпечує дуже швидкий і одночасний сервер HTTP 1.1 для веб-додатків Ruby. Він призначений лише для роботи додатків Rack.

Що робить Puma настільки швидким, це ретельне використання розширення Ragel для забезпечення швидкого, точного розбору протоколів HTTP 1.1. Графік порівняння швидкості веб-серверу Puma з іншими веб-серверами зображений на рисунку 2.6.

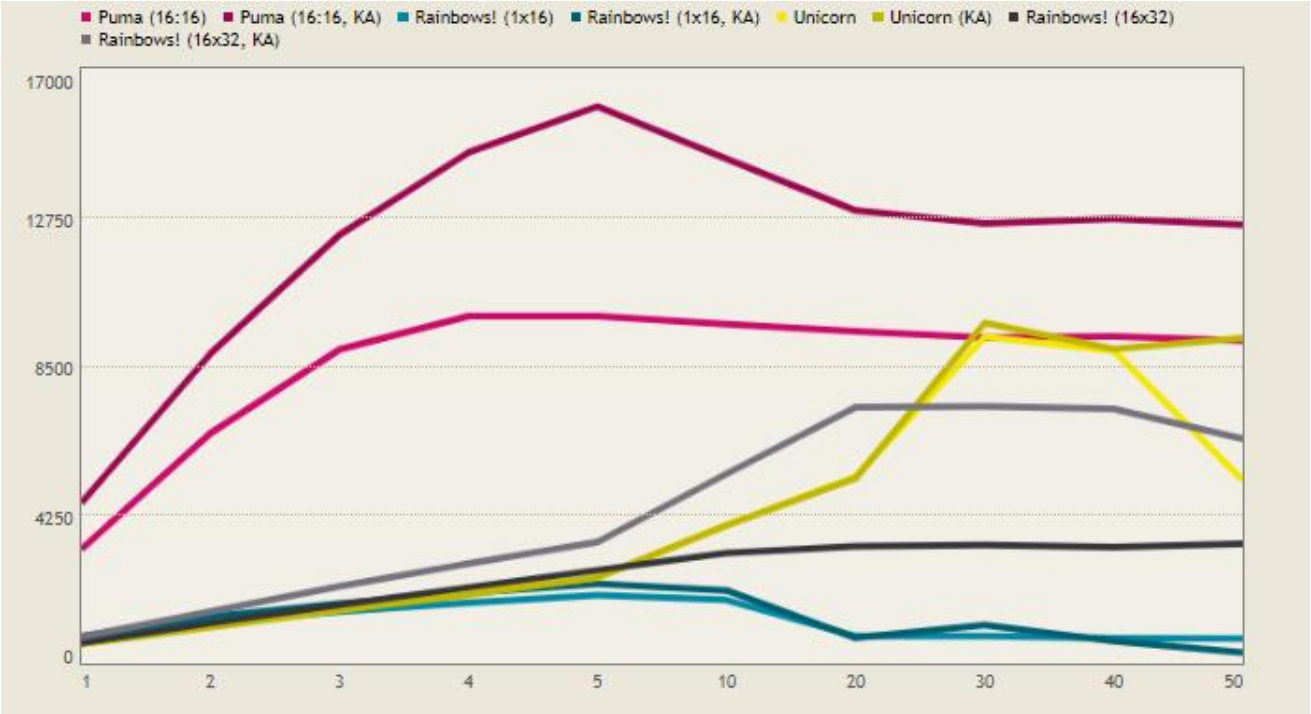

Рисунок 2.6 – Порівняння швидкості веб-серверів

Puma був створений Еваном Феніком наприкінці 2011 року як похідне від Mongrel. Тепер більшість оригінальних кодів Mongrel було переписано за винятком аналізатора. Історія git проекту надає часовий графік розвитку проекту.

2.4.6 База даних PostgreSQL

Для правильної роботи веб-сайту треба не лише файли з кодом, но й бази даних. Для того, щоб користувач міг взаємодіяти з базами даних користуються системи управління базами даних.

У базу даних можна записати багато різноманітної інформації. Надзвичайно важливу роль відіграє зв'язок даних у базі: зміна будь-якого рядка може призвести до значних змін інших рядків. Тому працювати з інформацією простіше та швидше, ніж коли б ці зміни стосувалися тільки одного місця в базі даних. Проте це не значить, що БД обов'язково повинне бути у кожному сайті. Наприклад, у сайті-візитці не потрібно розміщувати ніякі дані на, тому не треба бази даних.

Система управління базами даних – сукупність програмних і лінгвістичних засобів загального або спеціального призначення, що забезпечують управління створенням і використанням баз даних.

Простіше “база даних” – це ті самі дані, що представлені у вигляді сукупності файлів на дисках, з якими якраз працює «система управління базами даних» (СУБД) – програмний продукт, що має засоби для створення, наповнення, модифікації і пошуку по базах даних.

Весь сенс використання бази даних в тому, що коли даних стає більше, ніж кілька книг ці дані раптово стає неймовірно складно структурувати, а спроби отримати повний список призводять середньостатистичний комп'ютер в стан синього екрану від навантаження, то в справу вступають СУБД, які беруть на себе цю нелегку і досить оплачуване з боку великих підприємств тягар.

Переваги СУБД:

- продуктивність: СУБД має забезпечувати певний рівень продуктивності для вирішення відповідних завдань;
- надійність: не повинно бути ніякої втрати інформації, а збої системи мінімізовані.

Загальна схема роботи з базою даних зображена на рисунку 2.7

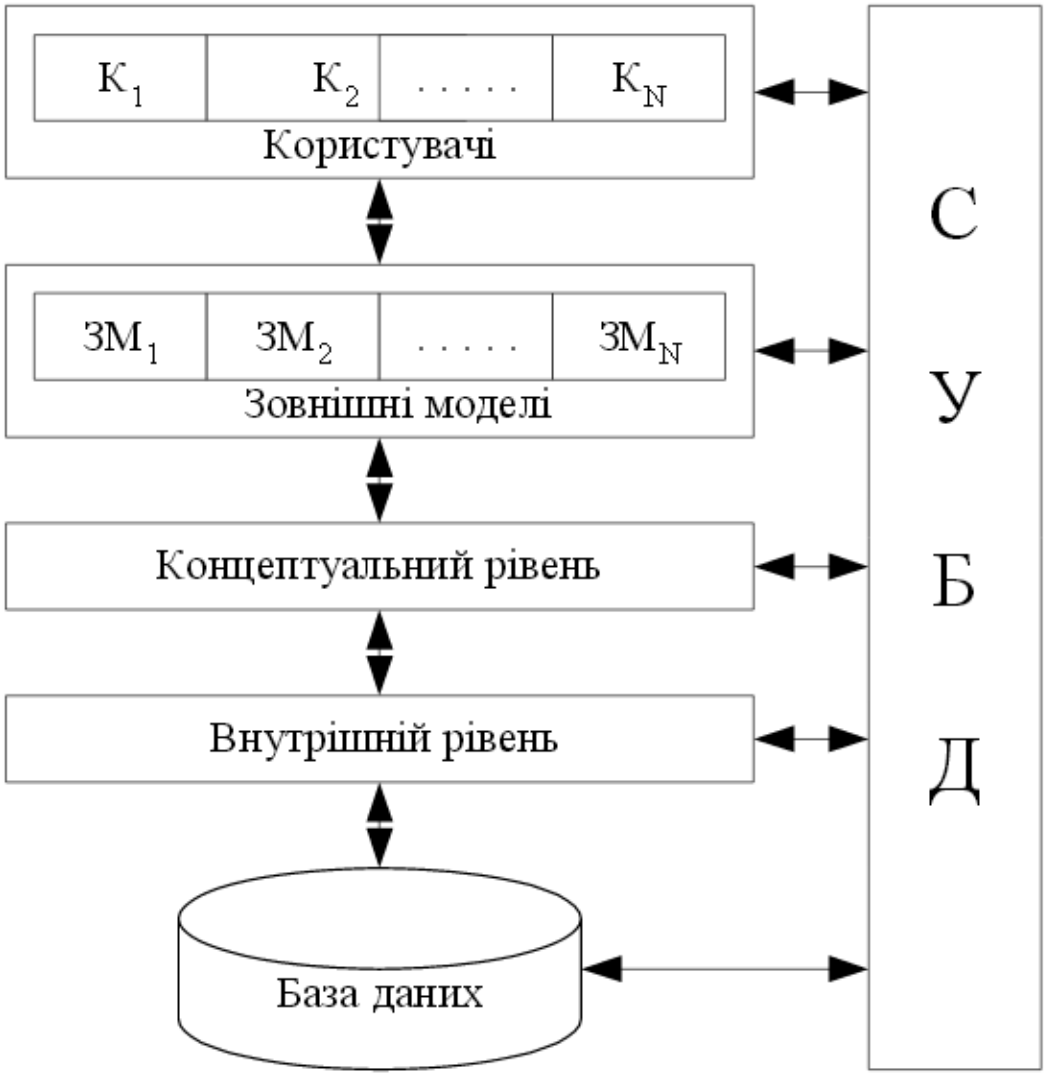

Рисунок 2.7 – Загальна схема роботи з базою даних

За замовчуванням в якості бази даних Rails пропонує використовувати SQLite 3, автономну базу даних, яка непогано підходить для отримання першого досвіду в розробці додатків. Однією з особливостей SQLite є неможливість одночасно виконувати більше однієї операції запису, тому чим швидше ви відмовитеся від її використання, тим краще. При розробці додатків часто практикується підхід, при якому development середовище має мінімум відмінностей від production середовища. Це дозволяє значно зменшити ймовірність появи проблем, які можуть виникнути через використання різних інструментів при розробці і при роботі додатка в продакшені. Цей підхід є однією з причин для того, щоб задуматися про використання PostgreSQL в розробці.

PostgreSQL – це потужна об'єктно-реляційна база даних із відкритим кодом, яка використовує мову SQL у поєднанні з багатьма функціями, які безпечно зберігають та масштабують найскладніші навантаження даних [19].

PostgreSQL заслужив потужну репутацію за свою перевірену архітектуру, надійність, цілісність даних, надійний набір функцій, розширюваність та відданість спільноті з відкритим кодом, що стоїть за цим програмним забезпеченням, щоб послідовно пропонувати ефективні та інноваційні рішення. PostgreSQL працює на всіх основних операційних системах, сумісний з ACID з 2001 року і має потужні додатки, такі як популярний розширювач геопросторових баз даних PostGIS. Не дивно, що PostgreSQL став реляційною базою даних з відкритим кодом для багатьох людей і організацій.

У PostgreSQL існує багато функцій, спрямованих на допомогу розробникам при розробці програм, захисті цілісності інформації та побудови середовищ. Також ці функції допомагають керувати вашими даними незалежно від того об’єму даних. У PostgreSQL можна визначати власні типи даних, створювати власні функції, навіть писати код з різних мов програмування!

Різні функції, які присутні у PostgreSQL:

1. Типи даних:
    - структури: дата / час, масив, діапазон, UUID;
    - примітиви: Integer, Numeric, String, Boolean;
    - геометрія: точка, лінія, коло, багатокутник;
    - документ: JSON / JSONB, XML.
2. Цілісність даних
    - унікальний, не нуль;
    - зовнішні ключі;
    - первинні ключі;
    - обмеження виключення.
3. Паралельність, продуктивність
    - Індексація;
    - розширена індексація;

- операції, вкладені транзакції;
- паралелізація запитів читання та побудова індексів;
- розбиття таблиці;
- всі рівні ізоляції транзакцій, визначені в стандарті SQL.

4. Надійність та відновлення
   - WAL;
   - реплікація: асинхронна, синхронна, логічна;
   - точне відновлення.
5. Безпека
   - аутентифікація;
   - надійна система контролю доступу;
   - захист інформації.
6. Інтернаціоналізація, пошук тексту
   - підтримка міжнародних наборів символів, наприклад через співпраці ICU;
   - співвідношення, нечутливі до регістру та нечутливості до акцентів;
   - повнотекстовий пошук.

Деякі обмеження PostgreSQL описані у таблиці 2.1

Таблиця 2.1 – деякі обмеження PostgreSQL

Назва	Значення
Максимальний розмір БД	Unlimited
Максимальний розмір таблиці	32 TB
Максимальна довжина запиту	400Gb
Максимальна довжина атрибута	1 GB
Кількість запитів у таблиці	Unlimited
Кількість атрибутів у таблиці	250-1600
Кількість індексів у таблиці	Unlimited

Отже, PostgreSQL – це вільно поширювана об'єктно-реляційна система управління базами даних, найбільш розвинена з відкритих СУБД.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ВЕБ-СЕРВІСУ ДЛЯ ПЕРЕВІРКИ І ОЦІНЮВАННЯ ЗНАНЬ

3.1 Середовище розробки веб-сервісу

Visual Studio Code – редактор коду для Linux, OS X і Windows. Він поєднує простоту редактора коду з тим, що потрібно розробникам для редагування та налагодження. Visual Studio Code забезпечує комплексну підтримку редагування коду, навігації та розуміння, а також легке налагодження, багату модель розширення та легку інтеграцію з існуючими інструментами.

Основною перевагою Visual Studio Code є те, що до нього можна підключити розширюваний плагінами дебагер, практично для будь-якої мови програмування.

Незалежно від того, яке завдання вам потрібно виконати – реалізувати довгостроковий проект, розробити гілку компонента з невеликим терміном існування або просто швидко переглянути запит – Visual Studio Code допоможе підвищити продуктивність, надавши повністю налаштоване середовище розробки за лічені хвилини.

Також у Visual Studio Code є браузерні редактор з підтримкою репозиторіїв Git та розширень, а також вбудований інтерфейс командного рядка для редагування, запуску та налагодження додатків на будь-якому пристрої.

3.2 Опис функціональності веб-сервісу

Веб-сервіс для перевірки та оцінювання має трьох користувачів системи. Даний веб-сервіс містить всі необхідні функції для того, щоб вчитель міг організувати дистанційне навчання і об’єктивно оцінювати та перевіряти знання учнів.

Функції та дії користувачів у веб-сервісі зображено на рисунку 3.1

Рисунок 3.1 – Діаграма прецедентів веб-сервісу

За діаграмою існують такі користувачі: учень, вчитель та адміністратор. Вчитель створює курс для свого предмету. До цього курсу він додає урок, а до уроку він додає завдання та тести за бажанням. Також педагог має змогу давати тести по цілому курсу. Учень може проходити тести та виконувати завдання, які задав учитель. Адміністратор має можливість видаляти користувачів веб-сервісу та застарілу інформацію. Всі користувачі веб-сервісу можуть переглядати інформацію, яка знаходиться на сайті.

Веб-сервіс для перевірки та оцінювання знань складається із декількох модулів. Ці модулі розбиті на різні функціональні підблоки. Основним модулем буде особистий кабінет користувача. Така структура, дозволяє легко та інтуїтивно користуватися веб-сервісом.

3.3 Опис бази даних веб-сервісу

Першим етапом розробки веб-сервісу є створення сховища, для зберігання інформації – розробка бази даних.

У проекті було використано об’єктно-реляційну систему керування базами даних – PostgreSQL.

База даних складається з наступних таблиць: “admins”, “users”, “subjects”, “lessons”, “quizzes” , “questions”, “answers”, “tasks”, “task_answers”, “rating”.

Щоб зберігати дані кожного користувача було розроблено модель “users”. Одним із завдань, яке стояло під час створення моделі “users” – це які дані необхідно зберігати. Найперше, це ідентифікатор користувача. Він має цілочисельний тип даних, допомагає швидко звертатись до користувача системи і також є унікальним значенням. Наступне, що потрібно зберігати – це особисті дані користувачів: ім’я, прізвище, електронну адресу та пароль. Одним із найважливіших полів у моделі “users” є поле “is_teacher”. Воно містить булеве значення (true або false), по замовчуванню це – “false”. Якщо при реєстрації на сайт користувач реєструється як вчитель, то це значення змінюється на “true”. Таким чином я розділяю користувача на учень і вчитель.

Структура моделі “users” представлена наступим чином (таблиця 3.1):

Таблиця 3.1 – Структура моделі “users”

Ім’я поля	Тип даних
id	integer
first_name	string
last_name	string
email	string
password	string
is_teacher	boolean

Оскільки головним завданням адміністратора сайту є видалення або редагування застарілої інформації, то адміністратори винесені у окрему модель “admins”.

У цій моделі зберігаються всі дані про адміністраторів веб-сервісу: ім’я, прізвище, електронна адреса та пароль.

Структура моделі “admins” представлена наступим чином (таблиця 3.2):

Таблиця 3.2 – Структура моделі “admins”

Ім’я поля	Тип даних
id	integer
first_name	string
last_name	string
email	string

Однією з найпростіших сутностей у даному проекті є сутність “subjects”. Ця таблиця зберігає у собі тільки ідентифікатор та назву предмету, який викладає педагог. Ця модель потрібна для того, щоб вчитель міг створити предмет, який викладає та додати до нього уроки.

Структура сутності “subjects” представлена таблицею 3.3:

Таблиця 3.3 – Структура моделі “subjects”

Ім’я поля	Тип даних
id	integer
name	string

Сутність “lessons” містить у собі поля ідентифікатора, імені та відео. Поле “video” потрібне для того щоб можна було загрузити відео з будь-якого відео-хостингу. Також у цій сутності присутнє поле “description”, де описується тема заняття. Структура моделі “lessons” представлена наступним чином (таблиця 3.4):

Таблиця 3.4 – Структура моделі “lessons”

Ім’я поля	Тип даних
id	integer
name	string
description	text
video	text

Наступна модель яка присутня у проекті – це модель “quizzes”.

Ця модель містить тільки ідентифікатор та ім’я тесту. Проте без цієї сутності не можливе проведення тестування.

Структура моделі “quizzes” представлена наступим чином (таблиця 3.5):

Таблиця 3.5 – Структура моделі “quizzes”

Ім’я поля	Тип даних
id	integer
name	string

Оскільки модель “quizzes” створена, наступним кроком створення тесту є створення моделі, де б зберігалися питання до тесту. Такою моделлю є модель “questions”. Вона містить у собі питання тесту.

Структура сутності “questions” представлена наступим чином (таблиця 3.6):

Таблиця 3.6 – Структура моделі “questions”

Ім’я поля	Тип даних
Id	integer
question	text

Для того, щоб можна було б створити повноцінний тест, потрібно створити модель, де б зберігалися відповіді на питання. Такою моделлю є модель “answers”. У цій моделі містяться ідентифікатор, відповіді і також є поле “is_true”. Це булеве поле, яке по замовчуванню має значення false. Воно потрібно для визначення правильної відповіді. Педагог створюючи тест вибирає яка відповідь є правильною і вона змінює значення поля “is_true” із false на true. Таким чином вибирається правильна відповідь із вибірки відповідей.

Структура сутності “answers” представлена наступим чином (таблиця 3.7):

Таблиця 3.7 – Структура моделі “answers”

Ім’я поля	Тип даних
Id	integer
answer	string
is_true	boolean

Щоб вчитель зміг учням не тільки задавати тести, а й різноманітні завдання, було створено сутність “tasks”.

Структура моделі “tasks” представлена наступим чином (таблиця 3.8):

Таблиця 3.8 – Структура моделі “tasks”

Ім’я поля	Тип даних
Id	integer
title	string
description	text

Для того, щоб учні змогли завантажити вирішення завдання для викладача необхідно щоб ці дані кудись зберігались. Для цього існує модель “task_answers”. У цій сутності зберігаються тільки відповіді учні.

Структура моделі “task_answers” представлена наступим чином (таблиця 3.9):

Таблиця 3.9 – Структура моделі “task_answers”

Ім’я поля	Тип даних
Id	integer
task_answer	text

Для того, щоб можна було зберігати оцінки за тестування була розроблена сутність “ratings”

Структура моделі “ratings” представлена наступим чином (таблиця 3.10):

Таблиця 3.10 – Структура моделі “ratings”

Ім’я поля	Тип даних
Id	integer
rating	float

Для взаємодії між таблицями, їх потрібно з’єднати зв’язками.

Асоціації у Ruby on Rails:

- один до одного (belongs_to) – коли один екземпляр моделі "належить" одному екземпляру іншої моделі;
- один до одного (has_one) – цей зв'язок показує, що кожен екземпляр моделі містить або володіє одним екземпляром іншої моделі;
- багато до багатьох (has_many :through) – цей зв'язок вказує, що оголошена модель може відповідати нулю або більше екземплярів іншої моделі через третю модель;
- один до одного (has_one :through) – цей зв'язок показує, що оголошена модель може бути пов'язана з одним екземпляром іншої моделі через третю модель;
- багато до багатьох (has_and_belongs_to_many) – створює пряме з'єднання з іншою моделлю, без проміжної моделі;
- один до багатьох (has_many) – цей зв'язок вказує на те, що кожен екземпляр моделі має нуль або більше примірників іншої моделі.

Для повноцінного зображення бази даних із асоціаціями використовують ER-діаграму

ER-діаграма – це діаграма взаємозв'язків, яка відображає відносини наборів об'єктів, що зберігаються в базі даних. ER-діаграми найчастіше використовуються для проектування або налагодження реляційних баз даних у галузі програмної інженерії, бізнес-інформаційних систем, освіти та досліджень. Ця діаграма використовує визначений набір символів, таких як прямокутники, ромби, овали та з'єднувальні лінії для зображення взаємозв’язку сутностей, відносин та їх атрибутів. Вони відображають граматичну структуру із сутностями та зв’язками.

Діаграми ER пов'язані з діаграмами структури даних (DSD), які зосереджуються на зв'язках елементів всередині об'єктів, а не на відносинах між самими сутностями. ER-діаграми також часто використовуються спільно з діаграмами потоку даних (DFD), які відображають потік інформації для процесів або систем.

Використання ER-діаграм:

- дизайн бази даних – використовуються для моделювання та проектування реляційних баз даних, з точки зору логіки та бізнес-правил та з точки зору конкретної технології, яка має бути впроваджена. часто є початковим кроком у визначенні вимог до проекту інформаційних систем
- виправлення неполадок у базі даних – використовуються для аналізу існуючих баз даних для пошуку та вирішення проблем у логіці чи розгортанні
- освіта – бази даних є сьогоднішнім методом зберігання реляційної інформації для освітніх цілей та подальшого пошуку, ER-діаграми можуть бути корисними при плануванні цих структур даних;
- дослідження – оскільки стільки досліджень зосереджено на структурованих даних, ER-діаграми можуть відігравати ключову роль у створенні корисних баз даних для аналізу даних;
- інформаційні системи для бізнесу – використовуються для проектування або аналізу реляційних баз даних, що використовуються в бізнес-процесах. Будь-який бізнес-процес, який використовує дані, що включають сутності, дії та взаємодію, потенційно може отримати користь від реляційної бази даних;
- реінжиніринг бізнес-процесів – допомагають аналізу баз даних, що використовуються для перепроектування бізнес-процесів та моделювання нових налаштувань бази даних.

Для того, щоб продемонструвати залежність таблиць та асоціації я розробив ER-діаграму бази даних веб-сервісу для оцінювання і перевірки знань (рис.3.2).

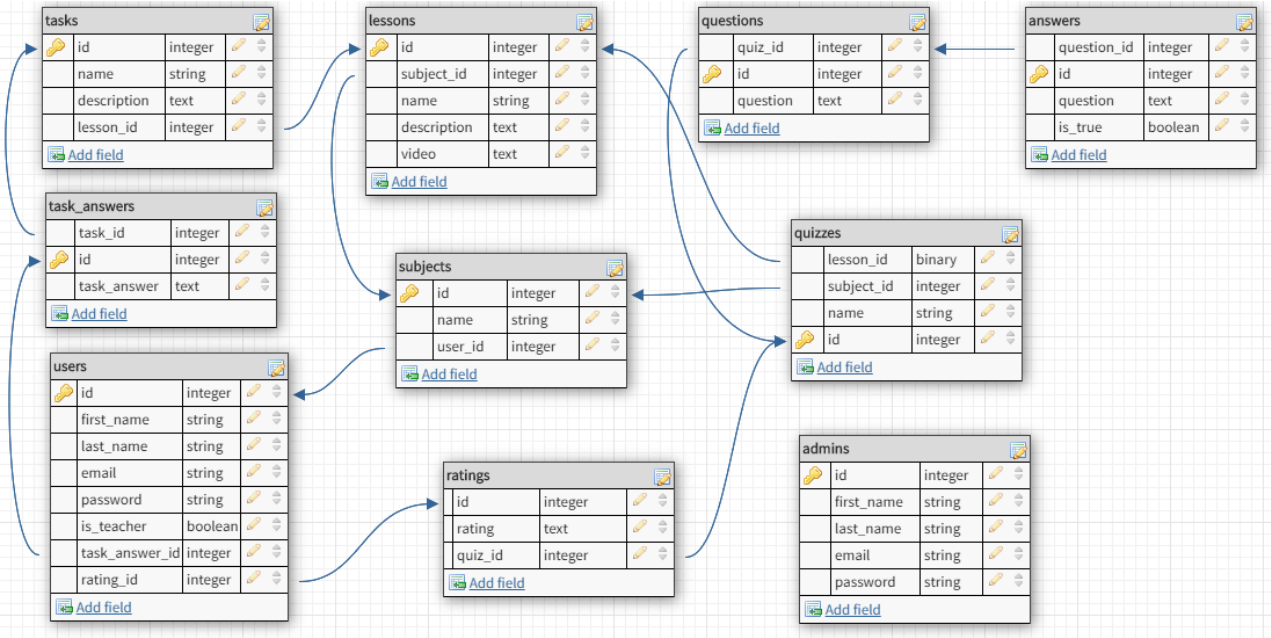

Рисунок 3.2 – ER-діаграма веб-сервісу для перевірки та оцінювання знань

Не менш важливим етапом створення бази даних є валідація даних. Валідації використовуються, щоб бути впевненими, що тільки валідні дані зберігаються в вашу базу даних. Наприклад, для вашого застосування може бути важливо, що кожен користувач надав валідну електронну адресу. Валідації на рівні моделі – найкращий спосіб переконатися, що в базу даних будуть збережені тільки валідні дані. Валідації у проекті прописані у моделях, наприклад модель user.rb (додаток А). Вони не залежать від бази даних, не можуть бути обійдені кінцевими користувачами і зручні в тестуванні і обслуговуванні. Rails являє простоту в обслуговуванні, вбудовані хелпери для загальних потреб, а також дозволяє створювати свої власні методи валідації.

Є кілька способів валідації даних, перш ніж вони будуть збережені в вашу базу даних, включаючи обмеження вбудовані в базу даних, валідації клієнтської частини і валідації на рівні контролера.

Плюси і мінуси:

- обмеження бази даних або збережені процедури роблять механізми валідації залежними від бази даних, що робить тестування і підтримку більш важкими;
- валідації клієнтської частини можуть бути дуже корисні, але в цілому ненадійні, якщо використовуються їх поодинці;

- валідації на рівні контролера часто призводить до громіздкості і труднощів тестування і підтримки;

Загальні опції валідацій:

- :allow_nil – пропускає валідацію, коли значення, яке провіряється дорівнює nil;
- :allow_blank – пропускає валідацію, якщо значення атрибута blank?, наприклад, nil або порожній рядок;
- :message – дозволяє визначити повідомлення, яке буде додано в колекцію errors, коли валідація провалюється;
- :on – дозволяє визначити, коли має відбутися валідація.

3.4 Розробка веб-сервісу для перевірки і оцінювання знань

3.4.1 Розробка автентифікації користувачів

Автентифікацію користувачів у проекті було реалізовано за допомогою гему “devise”. За допомогою devise створити користувача, який може входити та виходити з вашої програми, настільки просто, тому що devise піклується про всі контролери, необхідні для створення користувачів (users_controller) та для сеансів користувача (users_sesions_controller). Цей гем заснований на warden і обробляє автентифікацію за допомогою bcrypt.

Дійсно немає причин не використовувати цей гем для автентифікації під час роботи з RoR. Реалізація автентифікації вручну є безладною і вимагає декількох контролерів та декількох годин налаштування.

Переваги гему devise:

- розроблений на Rack;
- використовує MVC-модель;
- дозволяє мати кілька користувачів одночасно;
- базується на концепції модульності: використовує тільки те, що потрібно для проекту.

Під час встановлення гему devise як залежності будуть встановлені додатково наступні геми:

- warden – сполучне ПЗ, яке дає змогу автентифікації для Rack-додатків;
- orm_adapter – надає тільки одну точку входу для того, щоб використовувати основні функції Ruby ORM;
- bcrypt-ruby – надає просту обгортку для роботи з паролями. За основу взято криптографічну хеш-функцію bcrypt ();
- thread_safe – надає потоково-безпечні утиліти та колекції для RoR;
- railties – внутрішні компоненти Rails: завантажувачі додатку, rake-завдання, плагіни та генератори.

Devise використовує десять модулів для налаштування автентифікації користувачів. Ви можете знайти ці модулі всередині вашої моделі користувача.

Модулі для налаштування автентифікації:

- Database Authenticable – хешує та зберігає пароль у базі даних для підтвердження автентичності користувача під час входу. Автентифікацію можна зробити як через POST-запити, так і базову автентифікацію HTTP;
- Recoverable – додає посилання "Забув пароль", що дозволяє користувачеві скинути свій пароль за допомогою електронної пошти;
- Validatable – забезпечцє перевірку для адреси електронної пошти та пароля;
- Rememberable – запам’ятовує користувача за допомогою файлів cookie, додає прапорець “запам’ятай мене”;
- Registerable - створюється процес реєстрації, забезпечує користувачів можливістю редагувати та видаляти свій обліковий запис;
- Trackable – відстежує IP-адреси користувачів, кількість входів, останній вхід та часові позначки;
- Omniauthable – додає підтримку постачальника Omniauth, що дозволяє входити через сторонніх постачальників, таких як Facebook, Twitter тощо;
- Lockable – блокує обліковий запис через певний час або певну кількість спроб входу;

- Confirmable – блокує доступ до облікового запису користувача, якщо користувач не підтвердив його електронною поштою;
- Timeoutable – закінчує сеанс користувача через певний проміжок часу.

Шість із цих модулів включені за замовчуванням: Database Authenticable, Recoverable, Validatable, Rememberable, Registerable, Trackable. Інші ж потрібно налаштовувати.

Devise також пропонує кілька дуже корисних допоміжних функцій:

- before_action :authenticate_user! – обмежує доступ до будь-яких дій, якщо користувач не ввійшов у систему;
- skip_before_action :authenticate_user! – скасовує обмеження до сторінки веб-сервісу;
- current_user_ – повертає примірник користувача, який виконує дію;
- user_signed_in? – перевіряє, чи користувач увійшов до виконання певної дії.

Devise використовується як для автентифікації користувачів, так і для автентифікації адміністраторів. Щоб описати шляхи для користувача, то потрібно у файлі “routes.rb” прописати devise_for :users, а для адміністратора – devise_for :admins.

3.4.2 Розробка функції тестування

Основний контролер, який відповідає за функціонування системи тестування – quizzes_controller.rb (додаток Б). У цьому контролері організовані 4 базові функції (CRUD) для тестування.

Створення тесту відбувається за допомогою вкладених форм. Вкладені форми – це форми, які обробляють вкладені моделі та атрибути в одній формі. ля того щоб використовувати вкладені форми у застосунку я підключив гем “cocoon”. Для видалення вкладених моделей Rails використовує віртуальний атрибут під назвою _destroy. Коли встановлено _destroy, вкладена модель буде видалена.

Якщо запис зберігається, Rails здійснює пошук поля id, щоб знищити реальний запис, тому якщо ідентифікатор не вказаний, він буде обробляти поточний набір параметрів, як параметри для нового запису.

Приклад вкладених моделей зображено на рисунку 3.3

```
def quiz_params
  params.require(:quiz).permit(:name, questions_attributes: [:id, :question, :_destroy,
                               answers_attributes: [:id, :answer, :is_true, :_destroy]])
end
```

Рисунок 3.3 – Вкладені моделі

Cocoon визначає дві помічні функції:

- link_to_add_association – додає посилання на вашу розмітку, яка при натисканні на неї динамічно додає нову часткову форму для даної асоціації
- link_to_remove_association – додає посилання на вашу розмітку, яка при натисканні на неї динамічно видаляє навколишню часткову форму

Для створення тесту у веб-сервіс вносяться вхідні дані, які обробляються. Після створення тесту учні мають змогу пройти його. Після того, як учні пройдуть тест веб-сервіс виконує підрахунки та виводить результат. Оцінка за пройдений тест відображається у відсотках. Максимальна оцінка – 100%. Цей результат можуть переглядати користувачі сервісу.

Формула підрахунку оцінки учня:

$$A = 100 * \frac{k}{n},$$

де A – оцінка учня, k – кількість правильних відповідей у тесті, а n – загальна кількість питань у тесті.

Перевірка чи відповідь є правильною здійснюється за допомогою оператора умови: if is_true == true.

На рисунку 3.4 зображена схема структури тестування.

.

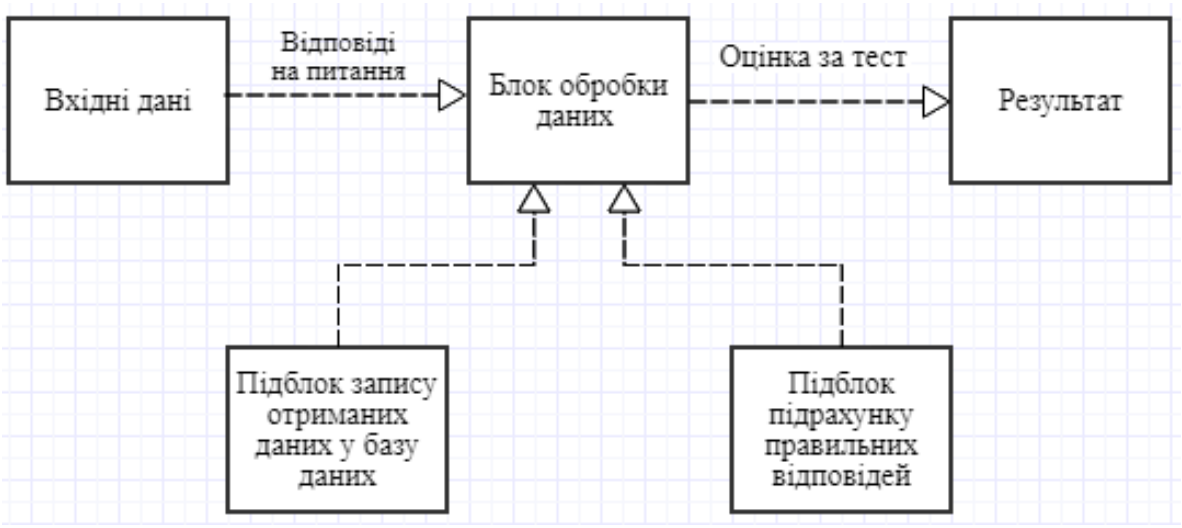

Рисунок 3.4 Схеми структури тестування

Ця схема наглядно описує як працює тестування у веб-сервісі для перевірки та оцінювання знань. Система тестування складається з кількох модулів, кожен з яких розбивається на різну кількість функціональних підблоків. Така структура, дозволить легко та інтуїтивно користуватися системою за рахунок того, що модулі розташовуються у порядку, який зазвичай використовують для вирішення схожих задач

3.4.3 Розробка можливості задання завдання

Для того, щоб створити завдання для учнів вчителеві прийдеться створити свій предмет, урок, а вже тоді створювати завдання.

Створення уроку і завдань для уроку реалізовано за допомогою вкладених ресурсів. Вкладені ресурси – це ресурси, які які логічно підпорядковані іншим ресурсам. Щоб організувати вкладені ресурси потрібно прописати вкладені маршрути. Також можна вкладати ресурси у вкладені ресурси Приклад вкладених маршрутів:

```
resources : subjects do
  resources : lessons do
    resources :tasks
  end
end
```

Після того, як ми вказали, що наш вкладений ресурс має контролер з неймспейса, все готово. Коли ми проводимо перевірку rake routes в терміналі, ми побачимо, що наш новий контролер знаходиться в просторі імен і що ми готові рухатися далі.

Глибоко вкладені ресурси швидко стають громіздкими. Один із способів уникнути глибокої вкладеності полягає в тому, щоб генерувати екшени колекції в області видимості батька, отримуючи уявлення про ієрархію, але не вкладати екшени елементів. Іншими словами, створювати маршрути з мінімальною кількістю інформації для однозначної ідентифікації ресурсу.

Для того щоб вчитель міг задати завдання учням було створено модель у базі даних, де ці завдання зберігаються. Також для зручного опису завдання було використано редактор TinyMCE (рис. 3.5). Основна логіка створення завдання розміщена к контролері task_controller.rb (додаток В)

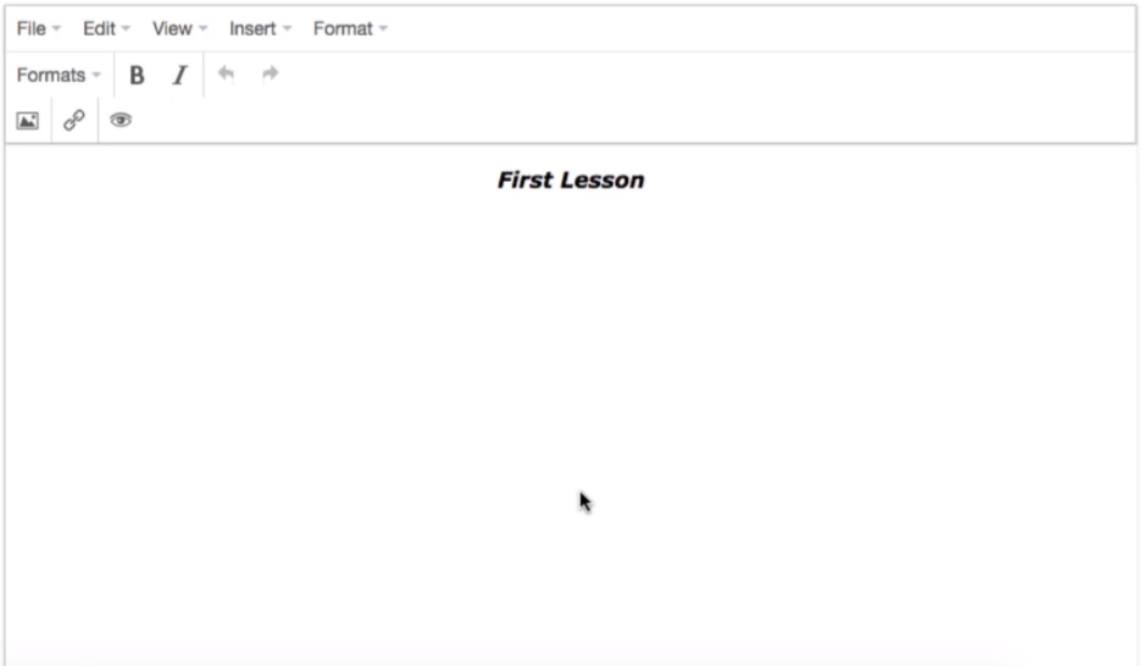

Рисунок 3.5 – Редактор TinyMCE

TinyMCE – це багатий текстовий редактор у хмарі. Він незалежний від платформи веб-редактор Javascript HTML WYSIWYG, випущений Moxiecode Systems AB як Open Source під LGPL. Цей редактор має можливість конвертувати поля HTML TEXTAREA або інші елементи HTML в екземпляри редактора.

TinyMCE дуже легко інтегрувати в інші системи CMS. Простіше кажучи, це як Word, який можна реалізувати у вашій програмі.

Додати редактор TinyMCE до проекту можна трьома способами:

- за допомогою Tiny Cloud
- з використанням TinyMCE self-hosted
- використання gem 'tinymce-rails'

Я підключив цей редактор за допомогою гему 'tinymce-rails'. Цей гем підтримує Сем Поленз для інтеграції TinyMCE в конвеєр активів Ruby on Rails. Ця процедура створює базовий додаток Ruby on Rails, що містить редактор TinyMCE на основі нашого базового прикладу. За допомогою цього редактора педагогу зручніше задавати завдання учням.

Для того щоб учні змогли дати відповідь вчителеві також використовується редактор TinyMCE.

3.5 Методика роботи користувача

3.5.1 Реєстрація, авторизація та редагування користувача

Для взаємодії з веб-сервісом користувачеві потрібно зареєструватися на сайті за допомогою форми “Sign up” (рис. 3.6).

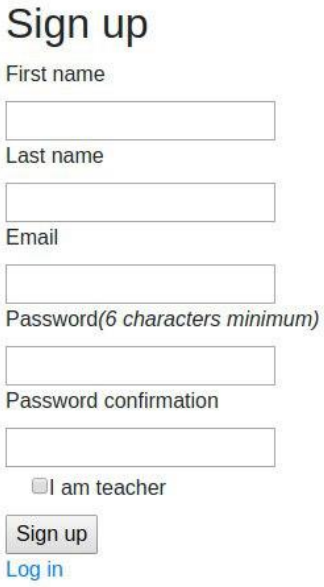

Рисунок 3.6 – Форма реєстрації на веб-сервіс

Якщо реєструється вчитель, він повинен при реєстрації вказати це. Форма реєстрації містить ім’я, прізвище, електронну адресу та пароль. Ця форма прописана у файлі new.html.slim (додаток Г). Для успішної реєстрації пароль потрібно ще раз ввести для його перевірки.

Для створення форм реєстрації було використано прості форми та технологію bootstrap.

Якщо ж користувач вже зареєстрований на сайт, то для взаємодії з веб-сервісом, потрібно авторизуватись (рис. 3.7).

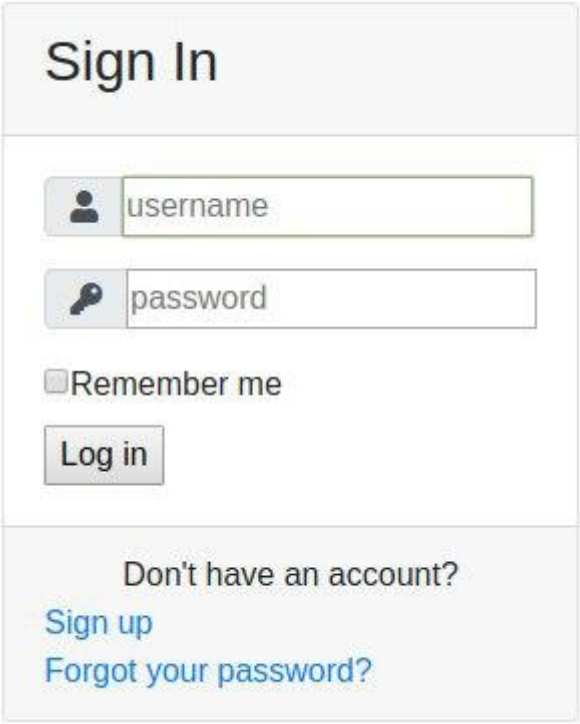

Рисунок 3.7 – Форма авторизації на веб-сервіс

Для того, щоб змінити дані користувача потрібно натиснути кнопку “Edit user”. Після чого появиться форма зміни даних користувача (рис.3.8). Можна змінювати імя, прізвище, елктронну адресу та пароль. Проте не можна змінювати свою роль: учень, вчитель чи адміністратор. Змінивши всі потрібні дані треба натиснути на кнопку “Update”.

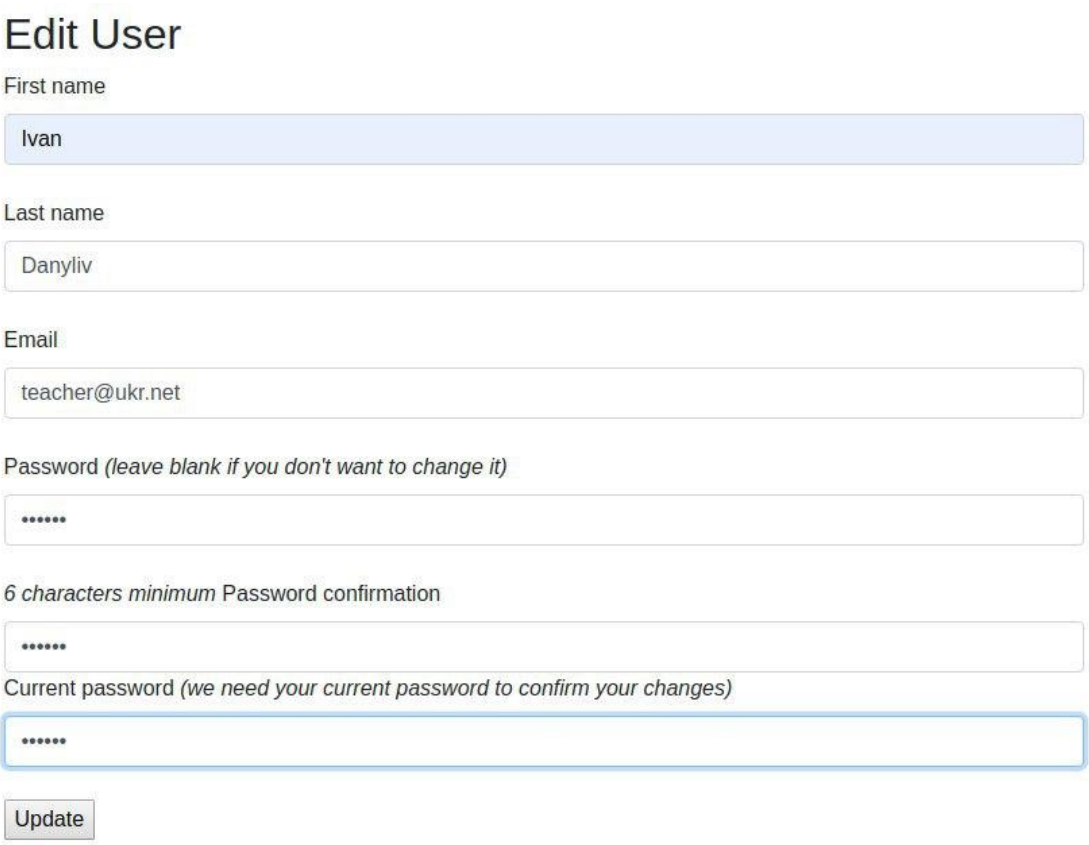

Рисунок 3.8 – Форма редагування користувача веб-сервісу

3.5.2 Можливості адміністратора веб-сервісу

Адміністратор веб-сервісу може видаляти та редагувати користувачів та застарілу інформацію. Для видалення або редагування користувачів потрібно перейти в пункт “Users” (рис. 3.9)

Рисунок 3.9 – Пункт меню “Users”

Так само як і користувачів адміністратор може видалити чи змінити будь-яку інформацію .

3.5.3 Створення тестів та завдань

Для створення тестів потрібно спочатку потрібно створити предмет, який учить вчитель, наступний крок – це створення уроку. Для створення уроку потрібно натиснути на кнопку “New Lesson” післячого появиться форма створення уроку (3.10).

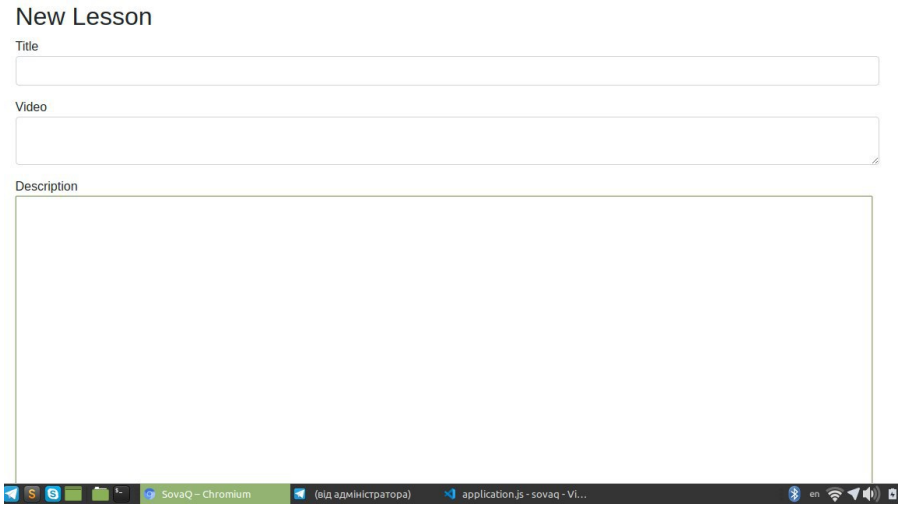

Рисунок 3.10 – Форма створення уроку

Під час створення чи редагування уроку можна додати до нього відео із будь-якого відео-хостингу, скопіювавши код для вставки і вставивши у відповідне поле. Після створення уроку можна створити тест. Для створення тесту потрібно натиснути на кнопку “New Quiz”. Натиснувши на цю кнопку появиться форма для створення тесту (рис. 3.11). Форма створення тесту представлена у додатку Д.

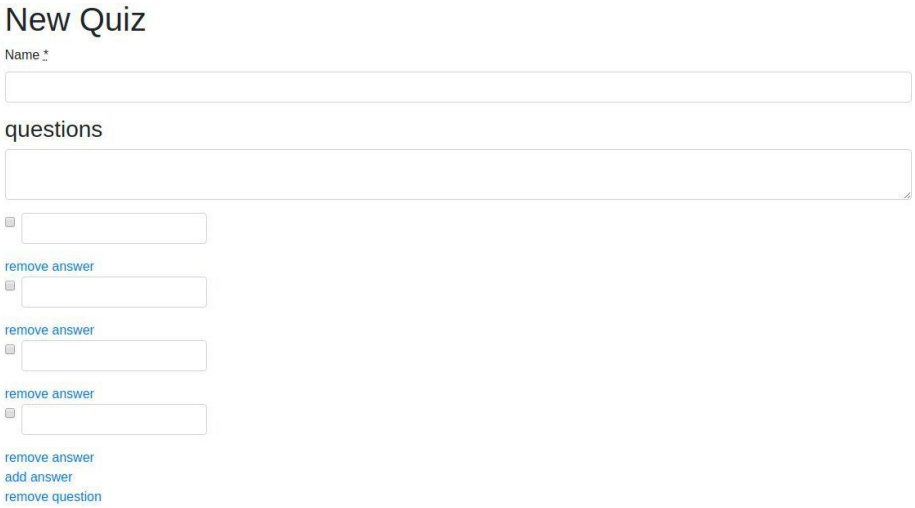

Рисунок 3.11 – Форма створення тесту

Щоб додати питання до тесту треба натиснути на кнопку “add question”, а щоб додати відповідь до питання потрібно натиснути на кнопку “add answer”. Щоб вказати яка з відповідей є правильною потрібно позначити її “прапорцем”. Щоб видалити питання чи відповідь треба натиснути на відповідні кнопки. Якщо видалити питання автоматично видаляється і відповіді до цього питання.

Щоб створити завдання для учнів, потрібно натиснути на кнопку “New Task”. Після цього появиться форма створення завдання. Завдання створюється за допомогою редактора TinyMCE (рис. 3.5).

3.5.4 Проходження тестування

Учень має змогу пройти тест. Для цього потрібно зайти на сторінку із всіма тестами натиснувши на кнопку “Quizzes”. Перейшовши а на цю сторінку можна вибрати потрібний тест із запропонованих (рис. 3.12)

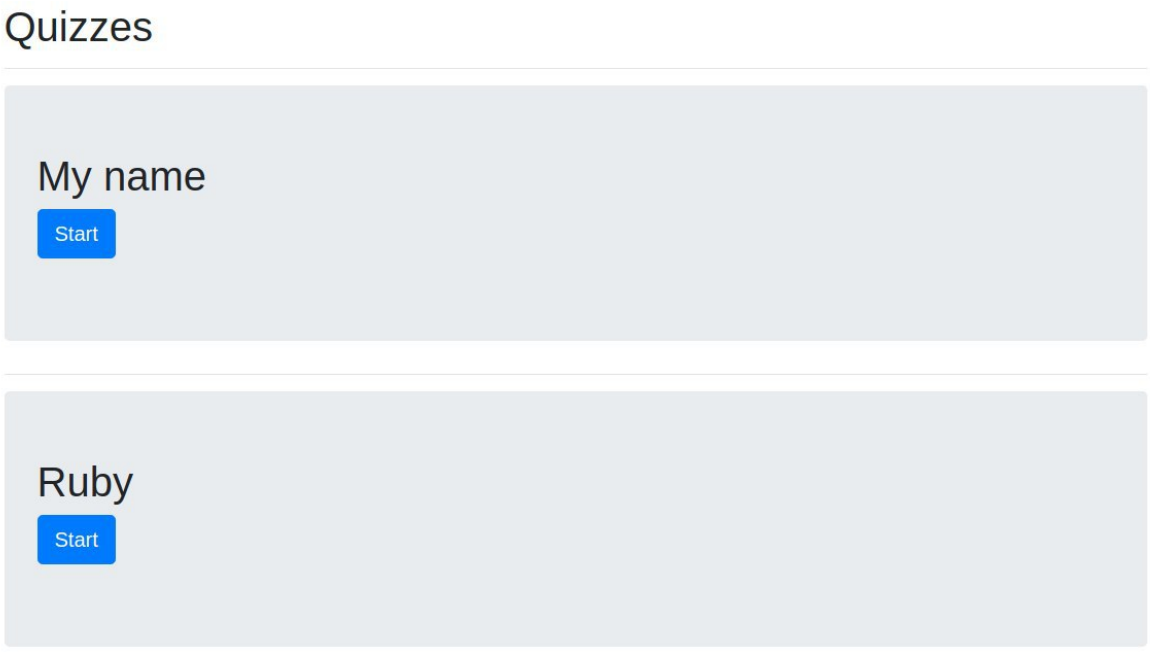

Рисунок 3.12 – Сторінка “Quizzes”

Натиснувши на кнопку “Start” учень починає проходити тест, відповідаючи на поставлені запитання (рис. 3.13).

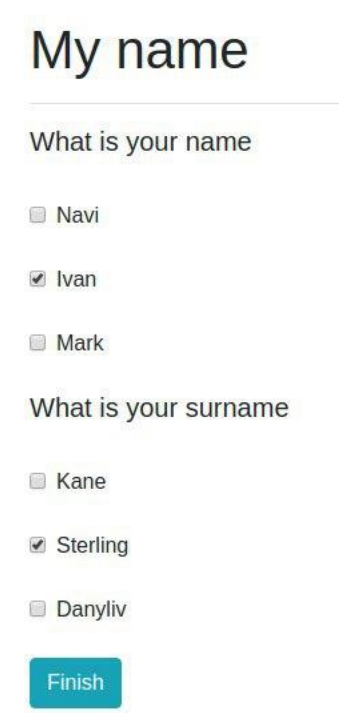

Рисунок 3.13 – Проходження тесту

Після того як учень дав відповіді на всі запитання потрібно натиснути на кнопку “Finish” щоб дізнатися свій результат (рис. 3.14)

Рисунок 3.14 – Результат проходження тесту для учня

Пройшовши тестування учень отримує оцінку за свої знання. Результати за тестування також може переглядати учитель (рис. 3.15). Таким чином можна здійснювати дистанційне навчання.

Рисунок 3.15 – Перегляд результатів тесту учителем

4 БІЗНЕС-ПЛАН ВЕБ-СЕРВІСУ ДЛЯ ПЕРЕВІРКИ ТА ОЦІНЮВАННЯ ЗНАНЬ

4.1 Резюме

Розроблений веб-сайт для перевірки та оцінювання знань стане в пригоді вчителю при проведенні перевірки та оцінювання знань учнів. Вчитель зможе розробити власний тест, створити експрес-тест для перевірки знань на уроці або у якості домашнього завдання. Учні виконують роботу у зручний час, використовуючи будь-який пристрій. Вчитель миттєво отримує інформацію про виконання завдань та зможе відстежити прогрес успішності кожного учня. Це особливо стане в пригоді вчителям під час проведення дистанційного навчання, особливо під час вимушеного дистанційного навчання для всіх учасників навчального процесу. Також веб-застосунок буде інформувати учнів про їхні результати оцінювання. І кожен учень зразу бачить свій результат і свій рейтинг відносно інших [20].

А також даний проект забезпечує користувача персональним кабінетом, в якому можна створювати тести і опитування, дуже швидко здійснювати контроль знань, надає можливості максимально автоматизувати процеси тестування та опитування, використовувати тести і опитування у дистанційному навчанні. Користувач буде забезпечений зручним графічним інтерфейсом, який може працювати як на комп’ютері, так і на мобільному телефоні [21].

4.2 Прогнозування витрат на розробку веб-сервісу

4.2.1 Розрахунок витрат на розробку даного проекту

Кошторис на розробку даного веб-сервісу буде включати такі витрати:

- на основну заробітну плату;
- на додаткову заробітну плату;

- амортизація техніки, що використовувались в процесі розробки;
- на силову електроенергію;
- нарахування на заробітну плату;
- інші витрати [22].

Основна заробітна плата ($З_о$) розраховується за формулою [23]:

$$З_о = \frac{М}{Т_р} \cdot t,$$

де $М$ – місячний посадовий оклад конкретного розробника, грн.;

$Т_р$ – кількість робочих днів у місяці;

t – число днів роботи розробника.

Над розробкою даного веб сервісу працюватимуть двоє людей: розробник та керівник. Місячний оклад розробника складає близько 13500 грн., керівника – близько 18000 грн. Робота з розробки даного проекту в середньому триватиме два місяці. Керівник надаватиме консультації два рази в тиждень.

Отже, основна заробітна плата розробника ($З_{оР}$) та керівника ($З_{оК}$) складатиме:

$$З_{оР} = \frac{13500}{21} \cdot 42 = 27000\,грн;$$

$$З_{оК} = \frac{18000}{21} \cdot 18 = 15428{,}57\,грн.$$

Всього витрат на основну заробітну плату ($З_{осн.}$) - 4242857 *грн.*

Додаткова заробітна плата розраховується у розмірі 10-12% від основної заробітної плати (формула 2):

$$З_Д = \frac{З_{осн}}{100\%} \cdot 10...12\%.$$

$$З_Д = \frac{4242{,}57}{100\%} \cdot 10\% = 424{,}286\ (грн).$$

Нарахування на заробітну плату розробників, які братимуть участь у розробці даного проекту розраховуються за формулою [23]:

$$Н_{зп} = (З_{осн} + З_Д) \cdot \frac{\beta}{100}$$

де $З_о$ – основна заробітна плата розробників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, %;

$З_Д$ – додаткова заробітна плата всіх розробників та робітників, грн.

Ставка єдиного внеску на загальнообов'язкове державне соціальне страхування складає 22%.

$$H_{зп} = (4242,57 + 4242,86) \cdot \frac{22}{100} = 10267,71\ грн.$$

Амортизаційні відрахування по кожному обладнанню розраховуються прямолінійним методом [24].

$$A = \frac{ОФ_п \cdot H_а}{100} \cdot \frac{T}{12},$$

де $ОФ_п$ – балансова (первісна) вартість об'єкта основних фондів, грн;

$H_а$ – річна норма амортизаційних відрахувань, %;

T – термін використання обладнання, приміщень, місяці.

Під час створення веб сервісу буде використовуватись ноутбук з операційною системою Linux. Отже. Амортизаційні відрахування:

$$A = \frac{20000 \cdot 15}{100} \cdot \frac{2}{12} = 500\ грн$$

Витрати на силову електроенергію розраховуються за формулою [25]

$$В_Е = В \cdot П \cdot Ф \cdot К_П,$$

де $В$ – вартість однієї кВт*год енергії, грн. (В = 1,68 грн./кВт*год);

$П$ – установлена потужність обладнання, кВт (0,09 кВт);

$Ф$ – фактична кількість годин робот обладнання, год. (274 год);

$К_П$ – коефіцієнт використання потужності (0,9).

$$В_Е = 1,68 \cdot 0.09 \cdot 274 \cdot 0,9 = 37,29\ грн.$$

Інші витрати: $8000\ грн$. Загальні витрати представлено в таблиці 4.1.

Таблиця 4.1 – Загальні витрати на розробку програмного продукту

Витрати	Сума, грн
на основну заробітну плату	42428,57
на додаткову заробітну плату	4242,86

Кінець таблиці 4.1 – Загальні витрати на розробку програмного продукту

амортизація техніки, що використовувались в процесі розробки	500
нарахування на заробітну плату	10267,71
на силову електроенергію	37,29
інші витрати	8000
Всього	65476,43

4.2.2 Розрахунок собівартості однієї копії програмного продукту

Даний програмний продукт суттєво відрізняється від продуктів звичайного матеріального виробництва. Ця різниця проявляється у формуванні витрат на виробництво та збут програмних продуктів. Програмний продукт буде реалізовуватись через мережу інтернет, тому не потрібно враховувати вартість матеріального носія з програмним продуктом. Але потрібно включити витрати на інтелектуальну власність, яка обчислюється за формулою [25]

$$I_в = k \cdot I_p,$$

де k – коефіцієнт, який враховує відповідні нарахування на заробітну плату, $(k = 1{,}22;\ 22\%\ ЄВС)$;

I_p – кошти, які будуть отримані від продажу однієї копії програмного продукту.

Кошти, які будуть отримані від продажу одного продукту, становлять 300 грн. Інші витрати становлять 20% від ціни на продукт.

Результати калькуляції собівартості програмного продукту занесено в таблицю 4.2

Таблиця 4.2 Собівартість програмного продукту

Стаття калькуляції	Витрати, грн.
Інтелектуальну власність	$1{,}22 \cdot 300 = 366$
Інші витрати	$0{,}2 \cdot 300 = 60$
Всього	426

4.2.3 Розрахунок ціни реалізації

Нижню межу ціни реалізації $(Ц_{Н.М.})$ розраховують за формулою [25]:

$$Ц_{Н.М.} = S_B \left(1 + \frac{P}{100}\right)\left(1 + \frac{\alpha_{ПДВ}}{100}\right),$$

де S_B – собівартість продукту, грн.;

P – норматив рентабельності, % ($P = 30\%...60\%$);

$\alpha_{ПДВ}$ – ставка податку на додану вартість, % ($\alpha_{ПДВ} = 20\%$).

Тому нижня межа ціни реалізації даного продукту становить:

$$Ц_{Н.М.} = 426 \cdot \left(1 + \frac{50}{100}\right)\left(1 + \frac{20}{100}\right) = 766{,}8\,(грн.)$$

Верхню межу ціни реалізації $(Ц_{В.М.})$ розраховують за формулою:

$$Ц_{В.М.} = Ц_{Н.М.} \cdot K_{В.Я.},$$

де $K_{В.Я.}$ – відносний коефіцієнт якості ($K_{В.Я.} = 1{,}36$).

Тому верхня межа ціни реалізації даного продукту:

$$Ц_{В.М.} = 766{,}8 \cdot 1{,}36 = 1042{,}85\,(грн.)$$

Тоді ціну реалізації ($Ц_Р$) даного програмного продукту можна прийняти у розмірі 900 грн.

Аналітично критичний обсяг виробництва продукту можна визначити за формулою [20]:

$$Q_K = \frac{ПВ}{Ц_{Н.М.} - ЗМ},$$

де $ПВ$ – постійні витрати, в нашому випадку витрати на розробку;

$ЗМ$ – змінні витрати або виробничі витрати, в нашому випадку собівартість програмного продукту.

$$Q_K = \frac{6547{,}43}{766{,}8 - 426} = 193$$

Щоб працювати без збитковості потрібно реалізувати 193 копії програмного продукту.

4.2.4 Розрахунок чистого прибутку від реалізації програмного продукту

Для розрахунку чистого прибутку ($П$) скористаємося формулою [25]:

$$П = \left(Ц_р - \frac{(Ц_р - MP)\cdot f}{100} - S_в - \frac{q \cdot S_в}{100} \right) \cdot \left(1 - \frac{h}{100} \right) \cdot N \ (грн.),$$

де $Ц_р$ – ціна реалізації програмного продукту;

MP – витрати на матеріальні та інші ресурси, які були придбані для виготовлення готової продукції (розраховують як 40-60% від ціни реалізації);

f – зустрічна ставка податку на додану вартість ($f = 16{,}67\%$);

q – норматив, який визначає величину адміністративних витрат , витрат на збут та інші витрати (рекомендується приймати $q = 5\% - 10\%$);

$S_в$ – собівартість продукту;

h – ставка податку на прибуток ($h = 18\%$);

N – кількість продукту, яка планується реалізувати за рік.

$$П = \left(900 - \frac{(900 - 0{,}4 \cdot 900)\cdot 16{,}67}{100} - 426 - \frac{5 \cdot 426}{100} \right) \cdot \left(1 - \frac{18}{100} \right) \cdot 300 = 89219{,}28 \ (грн.)$$

Отже, чистий прибуток становить 89219,28 грн.

4.2.5 Розрахунок терміну окупності виробника

Термін окупності виробника розраховується за формулою:

$$T_О = \frac{B_З}{П},$$

де $B_З$ – загальні витрати на розробку продукту;

$П$ – чистий прибуток від реалізації продукту.

$$T_О = \frac{65476{,}43}{89219{,}28} = 0{,}73.$$

Розробка вважається ефективною, якщо $1 < T_О < 3$.

В нашому випадку термін окупності складає 0,73 року. Отже, розробка програмного продукту є економічно вигідною.

ВИСНОВОК

У дипломному проекті розроблено web-сервіс для перевірки і оцінювання знань. Цей продукт вирішує проблему тестування учнів, а також здійснення дистанційного навчання.

В даному проекті проведено аналіз предметної області та існуючих варіантів розв’язування поставленої задачі. Також під час аналізу існуючого програмного забезпечення було досліджено аналогічні системи для вирішення даної задачі. Обґрунтовано вибір розробки веб-сервісу, який побудований за трирівневою архітектурою. Це дає можливість зробити сервіс гнучким у розробці та зручним у використані.

Під час перевірки функції тестування була підтверджена коректність результатів. Перевірено можливість здійснення дистанційного навчання. Отже, веб-сервіс відповідає вимогам.

Було розроблено бізнес план проекту, де обраховані витрати на розробку веб-сервісу та ціну його реалізації. Розраховано ефективність вкладених інвестицій та час, за який вони окупляться. Після всіх розрахунків було встановлено, що даний веб-сервіс є конкурентоспроможним на ІТ-ринку.

Під час розробки веб-сервісу та оформлення дипломного проекту були вдосконалені навички програмування і закріплені теоретичні знання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

REFERENCES

1. Веб-сервіс Google Classroom:
   https://uk.wikipedia.org/wiki/Google_Classroom
   (дата звернення: 17.12.2019).
2. Онлайн-платформа для навчання Coursera:
   https://uk.wikipedia.org/wiki/Coursera
   (дата звернення: 18.12.2019).
3. Інформаційно-освітня система МійКлас:
   https://uk.wikipedia.org/wiki/%D0%9C%D1%96%D0%B9%D0%BA%D0%BB%D0%B0%D1%81
   (дата звернення: 20.12.2019).
4. Поняття веб-сервіс: https://promo.ingate.ru/seo-wikipedia/web-service/
   (дата звернення: 25.12.2019).
5. Розши́рювана мо́ва розмі́тки XML: https://uk.wikipedia.org/wiki/XML
   (дата звернення: 26.12.2019).
6. Протокол обміну структурованими повідомленнями:
   https://uk.wikipedia.org/wiki/SOAP
   (дата звернення: 26.12.2019).
7. Мова опису вебсервісів: https://uk.wikipedia.org/wiki/WSDL
   (дата звернення: 27.12.2019).
8. Платформово-незалежний інструмент UDDI:
   https://uk.wikipedia.org/wiki/UDDI
   (дата звернення: 28.12.2019).
9. Веб-сервіси, як складова навчального процесу:
   http://timso.koippo.kr.ua/hmura11/osvitni-veb-resursy-yak-skladova-navcha/
   (дата звернення: 29.12.2019).

10. П. Федорук і М. Дутчак, Побудова бази знань адаптивних систем дистанційного навчання на основі фреймової та продукційної моделей представлення знань, Управляючі системи і машини (УСіМ), №5, с.35-42, 2012.
11. М. Дутчак, Моделювання процесу автоматизованої побудови індивідуалізованого навчання в інтелектуальних освітніх онлайн системах, Прикладні науково-технічні дослідження: матеріали IV міжнар. наук.-практ. конф., Івано-Франківськ, 2020, т.1., с. 47-51.
12. Дистанційне навчання: https://uk.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D1%82%D0%B0%D0%BD%D1%86%D1%96%D0%B9%D0%BD%D0%B5_%D0%BD%D0%B0%D0%B2%D1%87%D0%B0%D0%BD%D0%BD%D1%8F (дата звернення: 10.01.2020).
13. Клієнт-серверна архітектура: https://medium.com/@IvanZmerzlyi/%D0%BA%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0-%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0-%D1%82%D0%B0-%D1%80%D0%BE%D0%BB%D1%96-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D1%96%D0%B2-9893d8048229 (дата звернення: 20.01.2020).
14. Прикладний програмний інтерфейс: https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81 (дата звернення: 03.02.2020).

15. Сильвио Морето Bootstrap в примерах (пер. с англ. Рагимов Р. Н.). Москва : ДМК Пресс, 2017. 314 с.
16. Мова програмування Ruby: https://uk.wikipedia.org/wiki/Ruby (дата звернення: 10.02.2020).
17. David A., BlackJoseph Leo III, The Well-Grounded Rubyist Third Edition. USA: Manning Publications Co., 2019, 584 с.
18. Брюс А., Ruby on Rails. Быстрая веб-разработка, БХВ-Петербург, 2018, 477 с.
19. База даних PostgreSQL: https://www.postgresql.org/ (дата звернення: 15.02.2020).
20. Козловський В.О., Лесько О.Й. Бізнес-планування: навчальний посібник, УНІВЕРСУМ-Вінниця 2008.
21. Кошторис витрат виробництва: https://pidruchniki.com/1373112064744/ekonomika/koshtoris_vitrat_virobnitstva_ponyattya_sklad_metodika_skladannya (дата звернення: 16.02.2020).
22. Кошторис витрат і калькуляція собівартості продукції: https://pidruchniki.com/82242/ekonomika/koshtoris_vitrat_kalkulyatsiya_sobivartosti_produktsiyi (дата звернення: 16.02.2020).
23. Нарахування заробітної плати: https://byhgalter.com/rozraxunok-naraxuvan-ta-utriman-iz-zarobitnoї-plati-2018-prikladi/ (дата звернення: 18.02.2020).
24. Амортизаційні відрахування по обладнанню: https://web.posibnyky.vntu.edu.ua/fmib/17nebava_ekonomika_organizaciya_virobnichoyi_diyalnosti/55.htm (дата звернення: 18.02.2020).
25. Калькуляція собівартості одиниці матеріального носія з програмним продуктом: https://studfile.net/preview/3904173/page:8/ (дата звернення: 18.02.2020).

Додаток А

Модель user.rb

```
class User < ApplicationRecord
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable, :trackable and
:omniauthable

  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :validatable,
         :omniauthable, omniauth_providers:[:google_oauth2]

  has_many :subjects, dependent: :destroy
      has_many :quizzes, dependent: :destroy

  validates :email, format: { with: URI::MailTo::EMAIL_REGEXP }
  validates :password, presence: true,
                       confirmation: true,
                       length: { within: 6..40 },
                       on: :create

  validates :password, confirmation: true,
                       length: { within: 6..40 },
                       allow_blank: true,
                       on: :update

  validates :first_name, presence: true, length: { within: 3..25 }
  validates :last_name, presence: true, length: { within: 3..50 }
end
```

Додаток Б

Контролер quizzes_controller.rb

```
class QuizzesController < ApplicationController
  before_action :authenticate_user!
  before_action :find_quiz, only: %i[show edit update destroy]

  def index
         respond_to do |format|
      format.html { current_user.quizzes }
      format.json { render json: current_user.quizzes, include:
[:question, include: [answer]]}
    end
  end

  def new
    @quiz = Quiz.new
  end

  def show
respond_to do |format|
      format.html { @quiz }
      format.json { render json: @quiz, include: 'question',
'answer' }
    end
  end

  def edit
  end

 def update
    respond_to do |format|
      if @quiz.update(quiz_params)
```

Продовження додатку Б

```
        format.html { redirect_to @quiz, notice: 'Quiz was
successfully updated.' }
        format.json { render :show, status: :ok, location: @quiz }
      else
        format.html { render :edit }
      end
    end
  end

  def destroy
    @quiz.destroy
    respond_to do |format|
      format.html { redirect_to quizzes_path, notice: 'Quiz was
successfully destroyed.' }
      format.json { head :no_content }
    end
  end

def create
    @quiz = Quiz.new(quiz_params)
    @quiz.user_id = current_user.id

    respond_to do |format|
      if @quiz.save
        format.html { redirect_to @quiz, notice: 'Quiz was
successfully created.' }
        format.json { render :show, status: :created, location:
@quiz }
      else
        format.html { render :new }
      end
    end
  end
```

Продовження додатку Б

```
  private

  def quiz_params
    params.require(:quiz).permit(:name, questions_attributes:
[:id, :question, :_destroy, answers_attributes: [:id, :answer,
:is_true, :_destroy]])
  end

  def find_quiz
    @quiz = Quiz.find params[:id]
  end
end
```

Додаток В

Контролер tasks_controller.rb

```
class TasksController < ApplicationController
  before_action :authenticate_user!
  before_action :find_task, only: [:show, :edit, :update, :destroy]

  def index
    @task = current_user.task
  end

  def new
    @task = Task.new
  end

  def show
  end

  def edit
  end

  def update
    if @task.update task_params
      redirect_to task_path
    else
      render 'edit'
    end
  end
 def destroy
    @task.destroy
    redirect_to tasks_path
  end
```

Продовження додатку В

```
def create
      @task = Task.new task_params
    @task.user_id  =  current_user.id
    if @task.save
      redirect_to @task
    else
      render 'new'
    end
  end

  private

  def task_params
    params.require(:task).permit(:title, :description)
  end

  def find_task
    @task = Task.find params[:id]
  end
end
```

Додаток Г

Сторінка входу new.html.slim

```
head
  title Login Page
  /! Fontawesome CDN
  link crossorigin="anonymous"
href="https://use.fontawesome.com/releases/v5.3.1/css/all.css
" integrity="sha384-
mzrmE5qonljUremFsqc01SB46JvROS7bZs3IO2EmfFsd15uHvIt+Y8vEf7N7f
WAU" rel="stylesheet" /
body
  = form_for(resource, as: resource_name, url:
session_path(resource_name)) do |f|
    .container
      .d-flex.justify-content-center.h-100
        .card
          .card-header
            h3 Sign In
          .card-body
            form
              .input-group.form-group
                .input-group-prepend
                  span.input-group-text
                    i.fas.fa-user
                .field
                  = f.email_field :email, placeholder:
'username', autofocus: true
              .input-group.form-group
                .input-group-prepend
span.input-group-text
                    i.fas.fa-key
                .field
```

Продовження додатку Г

```
                    = f.password_field :password,
placeholder: 'password', autocomplete: "current-password"
              - if devise_mapping.rememberable?
                .field
                  = f.check_box :remember_me
                  = f.label :remember_me
              .actions
                = f.submit "Log in"
          .card-footer
            .d-flex.justify-content-center.links
              | Don't have an account?
            = render "devise/shared/links"
```

Додаток Д

Форма створення тесту

Д1 – new.html.slim

```
%h1 New Quiz
= render "form"
```

Д2 – _form.html.slim

```
= simple_form_for @quiz do |f|
  = f.input :name
  h3 questions
  #Questions
    = f.simple_fields_for :questions do |q|
      = render 'question_fields', f: q
      = q.simple_fields_for :answers do |a|
        = render 'answer_fields', f: a
    .links
      = link_to_add_association 'add question', f, :questions
  = f.submit
```

Д3 – _question_fields.html.slim

```
.nested-fields
  = f.input :question, label: false
  .links
    = link_to_add_association 'add answer', f, :answers
  = link_to_remove_association "remove question", f
```

Продовження додатку Д

Д4 – _answer_fields.html.slim

```
.nested-fields
  .btn-group
    = f.input :is_true, label: false, as: :boolean, checked_value:
true, unchecked_value: false, class: 'mt-2'
    = f.input :answer, label: false

= link_to_remove_association "remove answer", f
```

Додаток Е

Сирцевий код програмного забезпечення

Код програмного забезпечення, яке було розроблене в ході написання дипломного проекту, доступний за посиланням:
https://github.com/IvanDanyliv/sovaq.