Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

УДК 004.942

На правах рукопису

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: “ Розробка веб-застосування для автоматичного тестування по мові SQL”

Спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

БР.ІПЗ - ___.___.____.____ ІПЗ

Студент

ІПЗ-44 МС_____/Поліна ОВСЯННІКОВА/

Науковий керівник

к.ф.-м.н. _____ /Сергій ПОЛЯКОВ/

Консультант
з питань нормоконтролю

фахівець_____/Тамара ЧАПОВСЬКА/

Допускається до захисту

Завідувач кафедри

д.т.н., професор _____/ Олексій БИЧКОВ /

Київ – 2021

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

Освітньо-кваліфікаційний рівень - бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖЕНО
Зав. кафедри програмних систем і технологій
_____ (Олексій БИЧКОВ)

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ

Овсяннікової Поліни Олександрівни

(прізвище, ім'я, по батькові)

1. **Тема роботи: «**Розробка веб-застосування для автоматичного тестування по мові SQL**»**

Керівник проекту: к.ф.-м.н. Поляков Сергій Анатолійович

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

Затверджені наказом вищого навчального закладу від «___» _____2020р. №_____

2. **Строк здачі студентом закінченої роботи** «___» _____2021р.

3. **Вихідні дані до дипломної роботи:** монографії, підручники, навчальні посібники, статті та тези конференцій вітчизняних і зарубіжних авторів, Інтернет-ресурси з питань розробки веб-застосунків.

4. **Зміст пояснювальної записки:**

1) Аналітична частина:

проаналізувати існуючі програмні рішення та засоби тестування по мові SQL, обґрунтувати доцільність обраної теми дослідження, порівняти досягнення іноземних та українських фахівців вже існуючими рішеннями на зарубіжному та українському ринках.

2) Практична частина:

спроектувати архітектуру програмного забезпечення для тестування знань по мові SQL, розробити програмне забезпечення, розробити інтерфейс користувача.

5. **Перелік графічного матеріалу (із зазначенням обов'язкових креслень)**:

1) Аналітична частина: 8 рисунків.
2) Практична частина: 25 рисунків, 12 листів додатку програмного коду.

6. **Консультанти розділів проекту:**

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1 і 2 розділи аналітична частина	Сергій ПОЛЯКОВ	18.01.2021	18.01.2021
3 розділ практична частина	Сергій ПОЛЯКОВ	18.01.2021	18.01.2021

7. **Дата видачі завдання** «___» _____ 2021 р.

Керівник _____ (Сергій ПОЛЯКОВ)

Завдання прийняв до виконання _____ (Поліна ОВСЯННІКОВА)

Календарний план

№ з/п	Назва етапів виконання етапів бакалаврської роботи	Термін виконання етапів роботи	Відмітка про виконання
1	Уточнення постановки задачі	29.11.2020-05.12.2020	Виконано
2	Аналіз літератури	06.12.2020-04.01.2021	Виконано
3	Аналіз існуючих методів, концепцій та алгоритмів вирішення завдання	09.01.2021-16.01.2021	Виконано
4	Побудова алгоритмічної моделі основних процесів	28.01.2021-14.02.2021	Виконано
5	Опис розробленого алгоритму	15.02.2021-20.03.2021	Виконано
6	Розроблення програмного забезпечення	21.03.2021-30.04.2021	Виконано
7	Тестування розробленого програмного забезпечення	02.05.2021-15.05.2021	Виконано
8	Оформлення і друк пояснювальної записки	16.05.2021-25.05.2021	Виконано
9	Оформлення презентації	26.05.2021-03.06.2021	Виконано
10	Отримання рецензії		
11	Затвердження пояснювальної записки роботи завідувачем кафедри		
12	Захист дипломної роботи		

Студент – бакалавр _____ (Поліна ОВСЯННІКОВА)

Керівник роботи _____ (Сергій ПОЛЯКОВ)

АНОТАЦІЯ

Випускна кваліфікаційна бакалаврська робота: 60 с., 25 рис., 14 джерел, 3 додатки.

Тема: «Розробка веб-застосування для автоматичного тестування по мові SQL»

Об’єкт дослідження: процес тестування знань по мові SQL.

Предмет дослідження: програмні засоби для тестування знання SQL.

Мета роботи: покращення якості тестування знань по мові SQL, підвищення рівня якості знань студентів та працівників.

При розробці веб-застосунку для тестування по мові SQL було вирішено такі задачі:

1. Дослідження процесів розробки веб-застосунків.
2. Аналіз розробки веб-застосунків на основі підходу AJAX.
3. Аналіз архітектурних рішень даного способу розробки ПЗ;
4. Реалізація веб-застосунку за допомогою фреймворку Node.js.

Результати дослідження: створений веб-застосунок може бути використаний для перевірки знань студентів по мові SQL або кандидатів при наймі на роботу.

Висновок: розроблений веб-застосунок надає можливість дистанційного оцінювання знань по мові SQL за рахунок використання комплексу сучасних інструментів та технологій.

Ключові слова: веб-застосування, тестування, SQL, JavaScript, Node.js, AJAX, Express.js, SQLite.

АННОТАЦИЯ

Выпускная квалификационная бакалаврская работа: 60 с., 25 рис., 14 источников, 3 приложений.

Тема: «Разработка веб-приложения для автоматического тестирования по языку SQL»

Объект исследования: процесс тестирования знаний по языку SQL.

Предмет исследования: программные средства для тестирования знания SQL.

Цель работы: повысить качество тестирования знаний по языку SQL, увеличение уровня качества знаний студентов и работников.

При разработке веб-приложения для тестирования по языку SQL были решены следующие задачи:

1. Исследование процесса разработки веб-приложений;
2. Анализ разработки веб-приложений на основе подхода AJAX;
3. Анализ архитектурных решений данного способа разработки;
4. Реализация веб-приложения с помощью фреймворка Node.js.

Результаты исследования: созданное веб-приложение может быть использовано для проверки знаний студентов по языку SQL или кандидатов при найме на работу.

Вывод: Разработано приложение предоставляет возможность дистанционного оценивания знаний по языку SQL за счет использования комплекса современных инструментов и технологий.

Ключевые слова: веб-приложение, тестирование, SQL, JavaScript, Node.js, AJAX, Express.js, SQLite.

ABSTRACT

Final qualification bachelor's work: 60 p., 25 pictures, 14 sources, 3 annexes.

Topic: «Development of a web application for automatic testing of SQL language»

Object of study: the testing process of SQL knowledge.

Subject of study: software for testing.

Purpose of work: improving the quality of the testing process, preventing subjective assessment through the use of the developed web application.

When developing web applications for testing students, the following tasks were solved:

1. Researching of the web application development process;
2. Analysis of the development of web applications based on AJAX principle;
3. Analysis the architectural solutions of this development method;
4. Implementing the web application using the Node.js framework.

Research results: created a web application can be used for students testing in the department and faculty and hiring candidates.

Conclusion: The developed application provides the ability to remotely assess knowledge of the SQL language through the use of a set of modern tools and technologies.

Keywords: web application, testing, SQL, JavaScript, Node.js, AJAX, Express.js, SQLite.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

БД – база даних.

ОС – операційна система.

Фреймворк – це програмне забезпечення, що надає загальну функціональність програми.

MVC (Model, View, Controller) - архітектурне типове рішення для організації програми.

Клієнт - браузер, що функціонує як інструмент для передавання інформації на сервер та отримання відповіді.

UX (User experience) - призначене для користувача відчуття, враження, сприйняття роботи додатку.

JSON (JavaScript Object Notation) - текстовий структурований формат даних, що складаються з пар ключ-значення.

API (Application Programming Interface) - стандартизований інтерфейс для взаємодії клієнтів з сервером по стандартному протоколу, що надаються веб-застосувань.

DOM (Document Object Model) - об'єктна модель документа, програмний інтерфейс, що дозволяє програмам отримувати доступи до HTML, XHTML і XML документів, а також до структури і оформлення таких типів документів.

ВСТУП

На сьогоднішній день вміння формулювати SQL-запити є фундаментальним вмінням, яке є одною із вимог до будь-яких розробок програмного забезпечення. Оволодіння цим навиком – складний процес, що вимагає значних зусиль студента та багато годин практики.

Враховуючи стрімкий розвиток технологій у сучасному світі, використання новітніх програмних розробок в різних сферах життя людини має високий пріоритет. Зокрема, для закладів освіти та для особистого розуміння рівня знання з певного навика використання методу тестування є найефективнішим. Система тестування покликана підвищити якість та ефективність оцінювання знань.

Тести особливо популярні за кордоном, вони служать для відбору найбільш гідних кандидатів. Багато роботодавців вважають цю систему дуже зручною, так як вона дозволяє незначними зусиллями вирішити проблему комплектації кадрів.

Відповідно до формальної структури можна виділити елементарні тести, в яких може бути тільки один результат, також є складні тести, які складаються в свою чергу з підтестів, згідно будь-якого з яких повинна бути надана оцінка за виконання.

Переваги тестування:

- Можливість проводити групові вимірювання знань за певною темою або ж комплексом тем.
- Економічність у створенні.
- Всі тестовані участники знаходяться в однакових умовах.
- Об'єктивність оцінки знання.
- Простий спосіб перевірки підпису.

Недоліки даного виду тестування:

- Скрізь різний рівень знань виконавця.
- Можливість вгадування відповідей.
- Дані, одержувані в результаті тестування, хоча і включають в себе інформацію про прогалини в знаннях в деяких темах, але не дають інформацію про події, які послужили причиною цих прогалин.

- Розробка якісного тесту – є тривалим, трудомістким і дорогим процесом. Типові набори тестів для більшої частини дисциплін ще не були створені, але створені як правило мають невисоку якість.
- Тест не дає можливості перевірити та оцінити рівень знань, який є пов’язаним з творчістю та створенням нового.

Метою моєї дипломної підвищити якість оцінювання рівня знань по мові SQL, а також виключення суб’єктивності при оцінюванні шляхом розробки програмного застосунку.

Новизна. Система тестування, яка, на відмінну від існуючих, надає можливість дистанційного інтерактивного оцінювання знань по мові SQL за рахунок використання стеку сучасних технологій.

Практичне значення дипломної роботи полягає в тому, що створене програмне забезпечення може бути використано для тестування рівня знань по мові SQL студентів або визначення рівня знань кандидатів при наймі на роботу.

Дипломна робота складається з: вступу, трьох розділів, які включають 11 підрозділів, висновків, переліку джерел посилань із 14 найменувань. У листі дипломної роботи міститься 25 рисунків. Загальний обсяг роботи 60 листів.

РОЗДІЛ 1
ТЕОРИТИЧНІ ОСНОВИ РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ

1.1. Поняття та переваги веб-застосунку

Абсолютно всі користувачі персональних комп'ютерів знають що таке застосунок ОС Windows. Це програма, що встановлюється на ПК та функціонує в ОС Windows. Текстові, а також графічні редактори, медіаплеєри тощо. Далі проаналізовано, що ж являє собою саме веб-застосунок в сьогоденному житті.

Веб-застосунки – це програми, які написані скриптовими мовами (Perl, PHP тощо.) або, які є написаними високорівневими мовами та відкомпільовані під належну ОС (C, C++ тощо.), які працюють на серверній стороні та є призначеними для розробки інтерфейсу між користувачами та сайтом.

Таким чином, веб-застосунок – це програма, що функціонує в браузері, так само як Microsoft Word функціонує в операційній системі Windows. За цієї причини, для доступу до програми необхідні інтернет-браузер та вихід в мережу Інтернет. Збереження а також оброблення інформації при такого роду організації обчислень проходить на віддаленому сервері, а інтернет-браузер (клієнт) призначений для користування інтерфейсом (рис. 1.1).

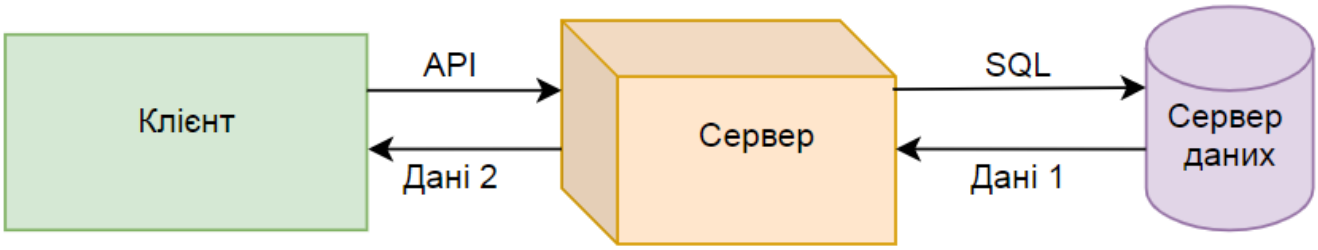

Рис. 1.1 Схема роботи веб-застосунку

Клієнтами є типові підключені до Інтернету пристрої користувача (наприклад, ваш комп'ютер, підключений до Wi-Fi, або телефон, підключений до мобільної мережі) та програмне забезпечення для доступу до Інтернету, доступне на цих пристроях (наприклад, веб-браузер Firefox або Chrome).

Сервери - це комп'ютери, на яких зберігаються веб-сторінки, сайти чи програми. Коли клієнтський пристрій хоче отримати доступ до веб-сторінки, а копія веб-

сторінки завантажується з сервера на клієнтську машину для відображення у веб-браузері користувача.

Текстовий пакет Microsoft Office функціонуватиме тільки на Windows. Але для веб-застосунку немає різниці яка операційна система встановлена на персональному комп’ютері, так як для застосунку операційною системою та його інтерфейсом є браузер.

Завдяки такій універсальності веб-застосунку, користувач може без якихось проблем працювати з улюбленою програмою на будь-якому із його девайсів.

У прикладному зрівнянні – веб-застосунки мають досить велику перевагу: вже було створено величезну кількість програмного забезпечення та сервісів, з чиєю допомогою будь-хто, навіть не являючись програмістом і просунутим користувачем, має змогу створити корисні для себе програми для розваг та взагалі для своєї зручності, при цьому це є безкоштовним.

Головні переваги веб-застосунків:

- додаток абсолютно не залежить від того, яка операційна система встановлена на комп'ютері користувача, тобто воно, по суті, є крос-платформних. Однак є тут і один момент, який може створити деякі труднощі при роботі з веб-додатками - це різні реалізації деяких специфікацій в браузерах, а також можливість налаштовувати різні параметри на зразок відображення шрифтів, що може привести до некоректної роботи деяких додатків. Але варто зазначити, що даний мінус хоча і має місце, але скільки-небудь істотного впливу на роботу більшості веб-додатків він не робить;
- сам факт існування веб-додатків повністю змінює спосіб поширення продукту. Тут творці відходять від традиційних способів поширення програмних продуктів шляхом продажу копій і установки їх на кожен комп'ютер користувачів. Тепер все набагато простіше: єдина версія додатка розташована на сервері, а всі користувачі мають доступ до неї, вірніше, до її призначеного для користувача інтерфейсу з будь-якого місця в світі. З будь-якого, де є Інтернет. При цьому користувачеві навіть не потрібно встановлювати нову версію програми - відразу після своєї появи вона доступна всім, причому багато

хто може і не помітити яких-небудь змін, тим більше якщо ці зміни не стосуються зовнішнього вигляду інтерфейсу. У всьому цьому явно видно і позитивний момент для розробників - їм не потрібно піклуватися про сумісність версій своїх додатків, оскільки всі користувачі одноразово отримують доступ і працюють з самою останньою версією програми;

– для користувача немає необхідності встановлювати і налаштовувати програмне забезпечення - все вже встановлено на серверах і налаштоване розробниками. Все, що потрібно від користувача, це змінити на свій смак кілька ключових параметрів. Це дуже приємно для користувачів, оскільки більшість не любить возитися з настройками і вважає за краще програмні продукти, повністю готові до використання відразу після їх інсталяції, хоча є й такі, які вважають за краще повністю налаштувати програму під свій смак і потреби. Однак в разі веб-додатків ми позбавлені навіть процесу інсталяції;
– для роботи з додатком від користувача, за великим рахунком, нічого і не потрібно. Хіба що комп'ютер і встановлений браузер. Але тут проблем немає - Інтернет-браузер вже є в будь-якій операційній системі, і для доступу до необхідного веб-додатком досить просто завантажити його URL в браузер. Використання веб-додатків багато в чому знімає обмеження, що накладаються на апаратну частину комп'ютера. Тобто певні системні вимоги до ПК все ж є, але їх рівень автоматично досягнутий комп'ютером, раз на ньому вже запущені ОС і браузер;
– з огляду на те, що основна частина веб-додатки сконцентрована на сервері в одному місці, куди простіше займатися його налаштуванням, не потрібно утримувати величезні команди фахівців технічної підтримки, що займаються консультаціями користувачів і налаштуванням програми на комп'ютерах в усьому світі. Це набагато менш затратно в фінансовому плані і куди більш ефективно. При цьому користувачу невидима архітектура додатки, в будь-який момент можна додати будь-яку кількість серверів, на яких встановлена основна складова програми, додати обчислювальні потужності, і користувач цього навіть не помітить.

В приклад можна привести електронну скриньку Gmail, він являється повноцінним поштовим клієнтом, який має функції всіх інших поштових клієнтів і має безліч інших функцій. Bloglines – веб-додаток для зображення актуальних новин, яке безпосередньо конкурує із звичайними аналогічними застосуваннями, конкурує та лідує. Ці веб-застосування працюють на сервері, а їх інтерфейс користувача (UI) відображається у вигляді веб-сторінок в браузері.

Хмарні технології дають змогу користувачам користуватись програмами без встановлення їх на персональний комп'ютер, а також користувач має змогу отримати прямий доступ до своїх власних файлів з будь-якого комп'ютеру дистанційно, який дозволяє вийти в мережу Інтернет. Хмарні технології допомагають користувачам ефективно обробляти, оновлювати та працювати з інформацією.

Найпростішими прикладами цих сучасних технологій можна представити сервіси електронної пошти, такі як Gmail та інші. Користувачу потрібно всього на всього доступ до мережі Інтернет, що дасть йому змогу легко відправити пошту, але при цьому нічого не встановлювати додатково на свій комп'ютер жодної програми окрім браузера.

Хмарні технології мають найрізноманітніші характеристики, основними з яких є:

- Спільна інфраструктура. Дана інфраструктура використовує віртуалізовану модель програмного забезпечення, що забезпечує спільний доступ до фізичних послуг, можливостей зберігання та мережевих можливостей. Хмарна інфраструктура, незалежно від моделі розгортання, прагне максимально використати доступну інфраструктуру для ряду користувачів.
- Динамічне забезпечення, що дозволяє надавати послуги на основі поточних вимог до попиту. Це робиться автоматично за допомогою програмного забезпечення автоматизація, що дозволяє розширювати та скорочувати можливості обслуговування, за потреби. Це динамічне масштабування потрібно робити, зберігаючи високий рівень надійності та безпеки.
- Мережний доступ. Потрібен доступ через Інтернет із широкого кола пристроїв, таких як ПК, ноутбуки та мобільні пристрої, використання API на основі

стандартів (наприклад, на основі HTTP). Розгортання служб у хмарі включає все, починаючи від використання бізнесу додатки до останньої програми на новітніх смартфонах.

– Кероване вимірювання використовує вимірювання для управління та оптимізації послуги, а також для надання інформації про звіти та виставлення рахунків. Таким чином, споживачам виставляється рахунок за послуги відповідно до того, скільки вони фактично використали протягом розрахункового періоду.

Основні моделі хмарних послуг для хостингу через мережу Інтернет такі, як:

– Програмне забезпечення як послуга (SaaS) - споживачі купують можливість доступу та використання програми чи послуги, розміщеної в хмарі.
– Платформа як послуга (PaaS). Споживачі купують доступ до платформ, що дозволяє їм розгортати власне програмне забезпечення та програми в хмарі. Операційні системи та доступ до мережі не керуються споживачем, і можуть існувати обмеження щодо того, які програми можуть бути розгорнуті. Прикладами можуть бути Amazon Web Services (AWS), Rackspace та Microsoft Azure.
– Інфраструктура як послуга (IaaS) - споживачі контролюють та управляють системами з точки зору операційних систем, програм, сховища та мережеве підключення, але самі не контролюють хмарну інфраструктуру.
– На відміну від PaaS та SaaS (навіть новіші обчислювальні моделі, такі як контейнери та без серверів), IaaS забезпечує найнижчий рівень управління ресурсами в хмарі.
– На основі цього можна підсумувати, що веб-застосунки є актуальними в нам час, це сучасні технології, які й досі набувають стрімкого розвитку і є незамінним інструментом для економії ресурсів та часу для створення універсальних програмних застосунків, які будуть надавати можливість для використання в будь-якій точці планети та на будь-якому комп’ютері, незважаючи на ОС та технічні характеристики девайсу.

1.2. Принцип роботи веб-застосунків

Веб-програми різного розміру та рівня складності відповідають одному архітектурному принципу, але деталі можуть відрізнятися.

Архітектура веб-додатків - це механізм, який визначає спосіб взаємодії компонентів програми між собою. Або, іншими словами, спосіб підключення клієнта та сервера встановлюється архітектурою веб-додатків.

Більшість веб-додатків розробляються шляхом розділення його основних функцій на рівні або підрівні. Це дозволяє легко замінити та оновити кожен шар програми самостійно. Цей архітектурний шаблон називається багаторівневою або ж трирівневою архітектурою (рис.1.2).

Рис. 1.2 – Структурні рівні програмного забезпечення

Базовий рівень є найнижчим рівнем ПЗ. Відповідає за взаємодію з базовими апаратними засобами. Основне ПЗ входить до апаратного забезпечення та зберігається у спеціальних мікросхемах пам’яті (ПЗП), утворюючи базову систему вводу та виводу BIOS. Програми та їх внутрішні дані записуються у ПЗП на етапі розробки і не можуть бути змінені під час виробництва.

Системний рівень допомагає користувачеві, апаратному та прикладному програмному забезпеченню взаємодіяти та функціонувати разом. Ці типи комп’ютерного програмного забезпечення дозволяють працювати в середовищі чи

платформі для роботи іншого програмного забезпечення та програм. Ось чому системне програмне забезпечення є важливим для управління цілою комп’ютерною системою.

Програми службового рівня взаємодіють з базовими і з системними програмами. Метою службових програм є автоматизація роботи перевірки та налаштування комп’ютерної системи, а також вдосконалення функцій системних програм. Деякі утиліти (програми обслуговування) негайно додаються до операційної системи, доповнюючи ядро, але більшість – це програми та розширюють функції операційної системи. Тобто, у розробленій програмі є дві сфери: інтеграція з ОС та автономна робота.

Веб-сервер є сполучною ланкою між веб-застосунком і клієнтом. Він приймає запити від клієнта у вигляді набору даних по протоколу HTTP і повертає у відповідь документи на мові HTML і файли, які потрібні додатково (картинки, листи стилів, інші об'єкти). При цьому документи та інші файли можуть бути як статичними (перебувати в файлової системі сервера), так і динамічними, тобто сформованими програмною частиною. Найбільш поширеним веб-сервером є Apache , його використовують для організації веб-вузлів більш ніж в 70% випадків (а для української частини інтернету цей показник досягає 90%).

Крім того, існують розробки спеціальних веб-серверів для різних додатків, що входять у великі комплекси, наприклад при використанні веб-додатків на технологіях компанії Microsoft використовують Internet Information Server (IIS). Від вибору сервера залежить набір можливостей з мов програмування, що застосовуються для розробки інформаційної системи.

Роль веб-сервера двозначна: з одного боку - створити комунікацію клієнта з серверної частиною, з іншого - відгородити його від виконання програмного коду і доступу до віддалених ресурсів. Щоб реалізувати описаний принцип, створено велике безліч рішень, але в процесі практичного використання виникають деякі питання, в тому числі одним з головних вважається питання безпеки. Також важливими параметрами є надійність роботи і захищеність від збоїв.

У веб-середовищі можуть виконуватися додатки, написані на різних мовах програмування. Їх можна розділити на дві групи: компільовані (виконувані як звичайні програми на комп'ютері) і інтерпретовані (що вимагають інтерпретатора для виконання). При розробці веб-додатків використовуються обидва цих типу, але особливої популярності заслужили саме інтерпретовані мови: Perl, PHP, Ruby, Python. Інтерпретовані програми повільніше виконуються, але процес їх розробки значно простіше і швидше.

Статичний веб-сайт у відсутності своєчасних та постійних оновлень контента має ризик провалу абсолютно всім конкурентним проектам. У цьому випадку інформацію, яка була підготовленою контент-менеджером з метою просунення та публікації, необхідно прийняти з керівництвом підприємства або власником ресурсу, а також направити розробнику програмного забезпечення або веб-майстру з метою внесення цих даних на веб-ресурс. Подібним принципом зміна колишньої або додавання нової інформації коштує дорожче, так як задіяно більше людей, і таким чином процес оновлення набагато ускладняється та є затратним.

Сервер баз даних виконує накопичення структурованої інформації, а також її видачу за запитами веб-додаткам. При цьому між програмною частиною системи і сервером баз даних можуть використовуватися різні інтерфейси (як стандартні так і спеціальні інтерфейси). Не всі мови програмування веб-додатків мають надійні і функціональні інтерфейси до серверів баз даних різних розробників. Від сервера баз даних залежать продуктивність програми і функціональність з точки зору даних. Наприклад, розвинені СУБД дозволяють використовувати розподілені сховища даних і мають потужні можливості щодо захисту від збоїв і відновлення після них. На рис. 1.3 представлена як приклад СУБД MySQL, що має серйозну репутацію в світі веб-розробок.

Вміст динамічних сайтів знаходиться не у варіанті статичних HTML-сторінок, а розташовується в базі даних і відображається безпосередньо відповідно до запиту користувача. Є досить велика кількість концепцій програмування ПЗ також широко відомих та загальноприйнятих мов програмування, наприклад: Perl, ASP, PHP і т.д. За їх допомогою можна сформувати базу з метою гнучкого веб-сайту будь-якої

складності, проте це далеко не всім під силу, і поріг входу в цю область є дуже високим. У процес формування подібного веб-сайту додається нова особа – розробник ПЗ на одну з мов програмування, у результаті чого розробка програмного забезпечення збільшується в термінах і ускладняється.

Застосовуючи 1-го розробника програмного забезпечення нереально створити функціонал, а також створити прийнятний дизайн майбутнього веб-сайту – це допустимо тільки лише за присутності дизайнера. При присутності такого підходу інформативний зміст ресурсу стане ізольованим від графічного, шляхом формування особливого стандарту веб-сайту, з урахуванням необхідного дизайну. Ще одним плюсом динамічних веб-застосунків стає порівняно просте управління ресурсом, за допомогою панелі адміністратора, і також можливо за доступною ціною розвитку проекту.

Подібним чином, в разі якщо веб-сайт складається з безлічі сторінок або є досить часта процедура його оновлення, то в такому випадку перевага динамічної структури безсумнівна. Розробникам ПЗ не буде потрібно цілком змінювати всі без винятку сторінки при незначній модифікації дизайну або при появі додактових розділів сайту. Відділення графічного дизайну та інформаційного вмісту і є важливою характерною особливістю, а також більш важливою перевагою динамічних веб-сайтів.

Центром клієнтської частини веб-додатки є браузер. Будь-який комп'ютер, що підключений до мережі інтернету, містить програму-браузер, через яку можна почати роботу з веб-застосунками.

Щодо використання пам'яті, то користувачеві не потрібно завантажувати всю програму на свій комп'ютер для роботи з нею. Навіть, весь інтерфейс не потрібно завантажувати. Досить завантажити лише ту його частину, яка необхідна для виконання конкретного поточного завдання. Як результат, веб-застосунки мають невеликі розміри, швидко завантажуються та швидко реагують на дії користувача. Навіть найскладніший застосунок завантажується всього за кілька секунд, а то й менше, і лише якщо канал зв’язку занадто повільний

Оскільки на комп'ютері користувача не встановлено програми, то користувач може працювати з веб-додатком із будь-якого місця.

Веб-застосунки мають вагому перевагу перед програмними аналогами і головним важелем є мобільність та простота створення та використання. Дана технологія ще й досі набирає популярність серед користувачів та розробників.

1.3. Порівняння та аналіз існуючих програмних рішень

TestDome надає послуги компаніям для тестування кандидатів при наймі на роботу задля перевірки їх знань по мові SQL. Ресурс містить перевірки не тільки по мові SQL, а й по різним мовам програмування.

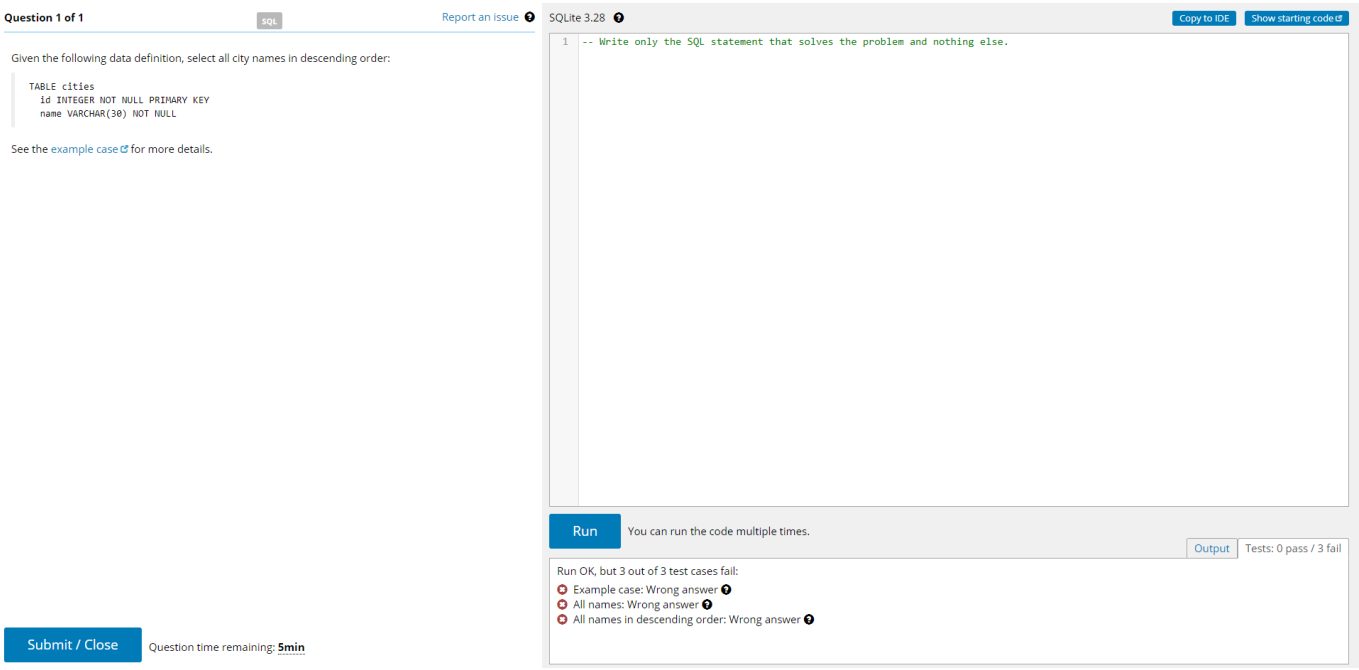

Рис. 1.3 Інтерфейс тестування з SQL в TestDome

Ресурс переважно платний, безкоштовно користувач може пройти лише два простих теста.

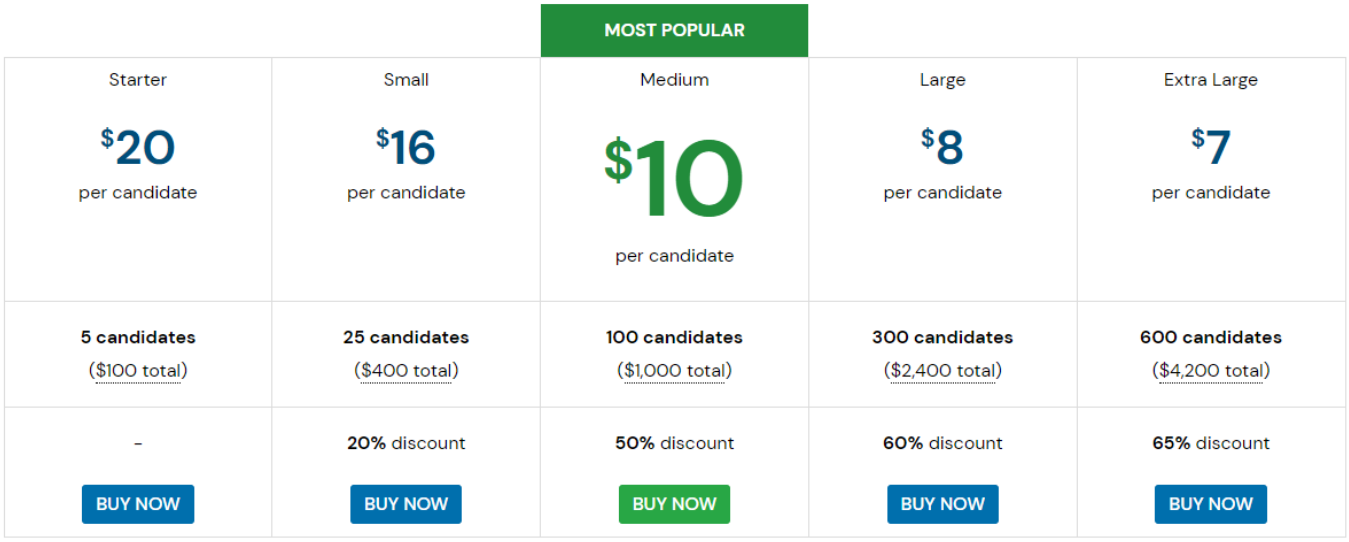

Рис. 1.4 Ціни на тестування по мові SQL на TestDome

Веб-застосунок SQLSchools орієнтований більше на вивчення мови SQL. Тестування знань по мові SQL на цьому ресурсі виконується за допомогою простих перевірок вписування відсутніх елементів запита.

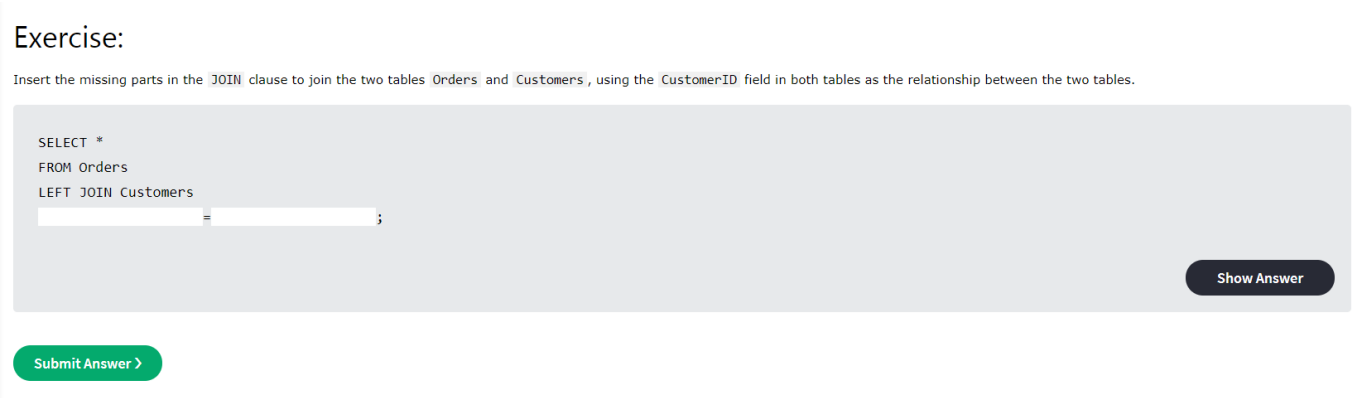

Рис. 1.5 Приклад тесту в застосунку SQL Schools

Недоліком системи є те, що дані тести не можливо кастомізувати та отримати розгорнутий звіт про сумарну кількість балів за пройдені тести. Також, даний метод є неоднозначним, оскільки рівень складності дуже малий та існує ймовірність того, що людина вгадає правильну відповідь.

Imocha - застосунок безкоштовній версії містить тести, що базуються на теорії SQL без можливості написання самого запиту.

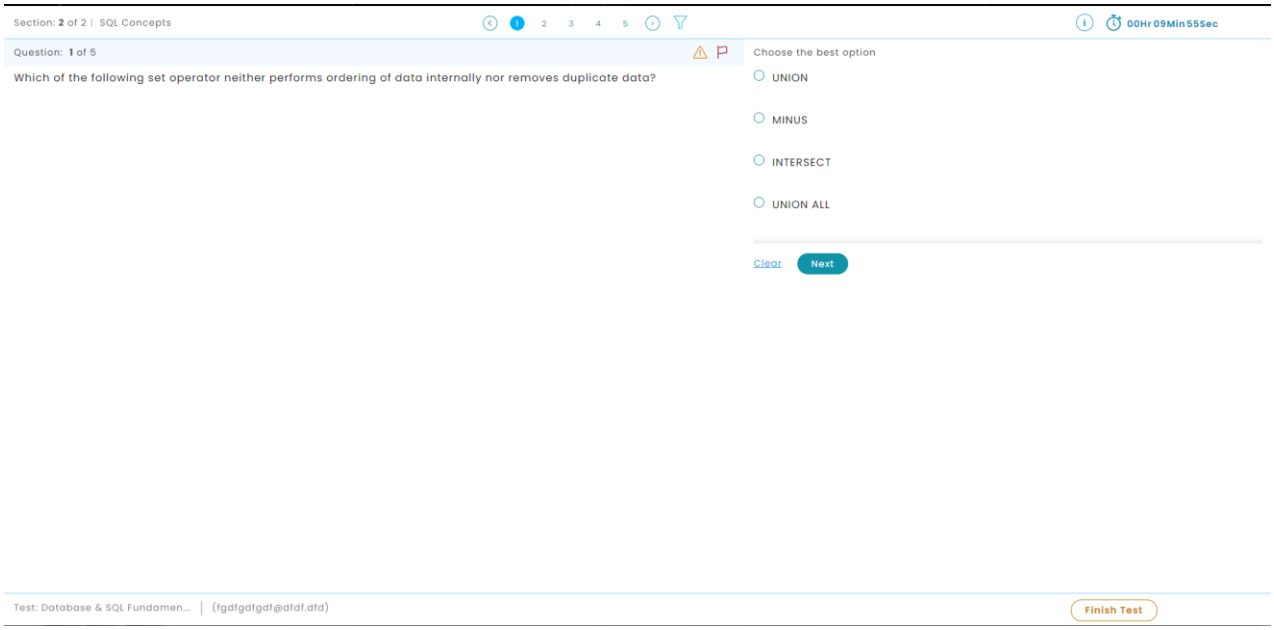

Рис. 1.6 Інтерфейс тесту від веб-застосування Imocha

Для повноцінної роботи із застосунком потрібно купувати підписку, ціна якої стартує від 150 доларів за місяць.

Висновки до розділу

Веб-застосунки є технологіями сучасності та майбутнього. Вони змінюють курс програмного забезпечення, яке використовують користувачі. Саме дані застосунки вже посідають значне місце в життях більшості людей. Явні переваги веб-застосувань перед своїми конкурентами робіть їх ще більш популярними та затребуваними на українському та закордонних ринках.

Веб-застосунком вважається конкретний динамічний сайт, що містить в собі інтерактивні модулі для взаємодії з користувачем. При цьому, не виключається можливість додавання інформативного контенту до даного застосунку.

РОЗДІЛ 2
ОГЛЯД ОБРАНОГО ОПТИМАЛЬНОГО ПІДХОДУ ДЛЯ РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ

У цьому розділі оглянуто та описано деякі системи та платформи, які використовувались для створення та коректної роботи з веб-застосунками, їх можливості та властивості, а також проведений аналіз. Зокрема, в даному розділі описано такі технології, як AJAX, яка є невід'ємним компонентом у створенні веб-застосунків, а також інші платформи, такі, як Google Cloud Platform та Microsoft Azure.

2.1. Підхід AJAX для створення UI веб-застосунків

AJAX (Asynchronous JavaScript and XML) - інструмент для побудови веб-додатків, які обмінюються даними з сервером у фоновому режимі. При цьому користувач отримує додаток з динамічним оновленням контенту, без перезавантаження всієї сторінки. Основним елементом AJAX є мова програмування JavaScript. На ньому реалізовано можливість завантаження контенту без перезавантаження сторінки.

В стандартному веб-застосування обробкою всієї інформації та даних займається сервер, а браузер відповідає тільки за взаємодію з користувачем за допомогою графічного інтерфейсу, передачу запитів і висновок надійшов через HTML, а в Ajax-застосуванні між користувачем і сервером з'являється ще один прошарок - рушій Ajax. Він позначує, які запити можна обробити одразу та по які необхідно звертатися на сервер.

Поведінка сервера додатку теж змінилась. Раніше на кожен запит сервер видавав нову сторінку із даними, а тепер сервер відсилає лише ті дані, які потрібні клієнтові у браузері, а HTML з них прямо в браузері формує рушій Ajax.

Асинхронність визначається в тому, що подекуди не кожен натиск користувача доходить до сервера, при цьому зворотне теж справедливо - далеко не кожна реакція

сервера обумовлена запитом користувача. Більшу частину запитів до сервера генерує рушій Ajax, при тому його можна написати так, що він буде завантажувати дані та інформацію, передбачаючи дії юзера.

Основне призначення AJAX полягає в наданні веб-додатку можливості ефективної обробки взаємодії між користувачем і веб-сторінкою, при цьому значно знижуються вимоги до оновлення або повної перезавантаження сторінки при кожній взаємодії з користувачем. Такий підхід надає широкі можливості при використанні браузера (аналогічні можливостям настільного додатки або веб-додатки на основі модуля).

Обробка взаємодії AJAX здійснюється асинхронно у фоновому режимі. Завдяки цьому користувач може продовжувати роботу зі сторінкою. Взаємодія AJAX ініціюється за допомогою коду JavaScript. Після виконання взаємодії AJAX код JavaScript оновлює вихідний текст HTML для сторінки. Зміни вносяться негайно без необхідності оновлення сторінки. Взаємодії AJAX можуть використовуватися для виконання таких завдань, як перевірка правильності формату вводяться записів на основі серверної логіки (безпосередньо під час їх введення користувачем), витяг докладних даних з сервера, динамічне оновлення даних на сторінці і передача елементів форм сторінки.

Покроковий вигляд роботи веб-застосунку за принципом AJAX:

- подія відбувається на веб-сторінці (сторінка завантажується, натискається кнопка);
- об'єкт XMLHttpRequest створюється JavaScript;
- об’єкт XMLHttpRequest надсилає запит веб-серверу;
- сервер обробляє запит;
- сервер надсилає відповідь назад на веб-сторінку;
- відповідь читається JavaScript;
- правильна дія (наприклад, оновлення сторінки) виконується JavaScript.

Remote Scripting (міжсайтовий скриптинг) - це тип вразливості інтерактивних інформаційних систем у веб-технологіях. XSS з’являється, якщо на сторінки, які

згенерувались за допомогою серверу, з якоїсь причини потрапляють скрипти користувача.

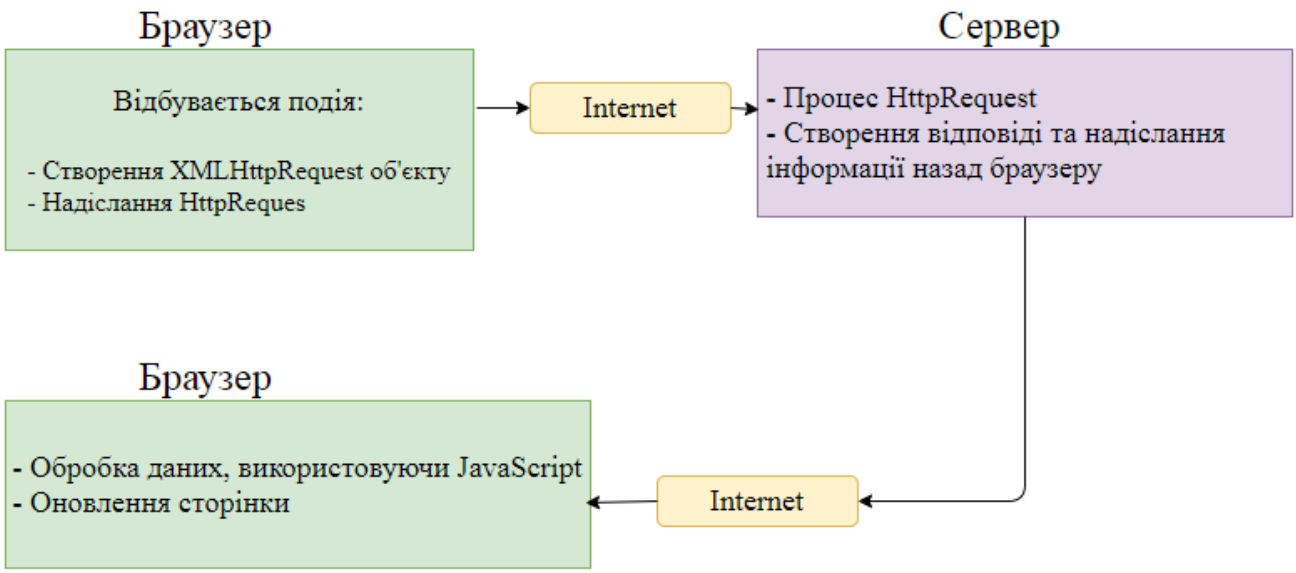

Рис. 2.1 Схема принципу роботи застосунку за технологією AJAX

AHAH (Asynchronous HTML and HTTP) – це схожий на AJAX підхід динамічного оновлення сторінок, з застосуванням JavaScript. Головною його різницею від AJAX є те, що відповіді сервера повинні бути на HTML. Перевага підходу полягає в порівняно більшій сумісності та функціональності.

Реалізується у форматі простих фреймів, які автоматом міняють свій розмір під розмір вмісту, або у варіанті прихованих фреймів, що здійснюють тільки функції завантаження даних.

Asynchronous XHTML and HTTP, або - AXAH – це по суті те саме, що і AHAH. Відмінність полягає тільки в тому, що в AHAH сервер повертає клієнту HTML, а в AXAH вже XHTML.

2.2. Розробка веб-застосунку за допомогою Node.js

Мова програмування JavaScript розроблена компанією Netscape для розробки інтерактивних HTML-документів. Це об'єктно-орієнтована мова розробки програмного забезпечення та вбудованих додатків, що виконуються як на стороні клієнта-браузера, так і на стороні сервера в клієнт-серверній архітектурі. Синтаксис

мови JS дуже схожий та нагадує синтаксис Java – через це його називають Java-подібним.

Основні області застосування мови JavaScript:

- Динамічне створення документа з даними за допомогою сценарію виконання;
- Швидка перевірка достовірності та коректності заповнених користувачем полів форм в мові HTML до передачі цих даних на сервер;
- Створення динамічних сторінок на мові HTML спільно з каскадними таблицями і об'єктним типом документів;
- Взаємодія з користувачем програми при вирішенні поточних локальних завдань, що вирішуються за допомогою додатку JavaScript, вбудованому в HTML-сторінку.

Nodejs (або просто Node) - це серверна платформа для роботи з JavaScript через двигун V8. JavaScript виконує дію на стороні клієнта, а Node - на сервері застосунку. За допомогою серверної платформи Node можна писати поовноцінні програми та веб-додатки. Node успішно функціонує із зовнішніми бібліотеками, викликає команди з коду на JavaScript і викконує роль веб-сервера.

Переваги Node над іншими серверними технологіями (Java, PHP, C #) вкрай прості: з Node простіше масштабувати. При одночасному підключенні до сервера тисяч користувачів Node працює асинхронно, тобто ставить пріоритети і розподіляє ресурси грамотніше. Java ж, наприклад, виділяє на кожне підключення окремий потік.

Прийнята в Node модель принципово відрізняється від поширених наразі платформ для побудови серверів застосунків, в яких масштабованість досягається за рахунок багатопоточності. Можна затвердити, що завдяки подієво орієнтованій архітектурі фреймворку знижується споживання пам'яті, підвищується пропускна здатність програми і спрощується модель програмування. Зараз платформа Node швидко розвивається та поширюється, і багато хто вважає її якісною альтернативою традиційному підходу до розробки веб-застосувань - на базі Apache, PHP, Python і та інших.

В основі планформи Node лежить автономна віртуальна машина Java Script з розширеннями, що роблять даний фреймворк придатним для програмування

загального призначення з упором на розробку серверів для веб-додатків. Платформу Node не є доцільно порівнювати ні з мовами програмування, які зазвичай використовуються для створення веб-застосувань (PHP , Python , Ruby , Java), ні з контейнерами, що реалізують протокол HTTP (Apache , Tomcat , Glassfish так інші) . На сьогодення багато хто вважає, що потенційно вона може замінити традиційні інструменти для розробки веб-додатків.

Nodejs Express являє собою веб-фреймворк, написаний на JavaScript і працює всередині середовища виконання node.js. Цей модуль висвітлює деякі ключові переваги цього фреймворка, установку середовища розробки і виконання основних завдань веб-розробки і розгортання.

Деякі переваги даного фреймворка:

- дозволяє налаштувати посередників для відповіді на запити HTTP;
- визначає таблицю маршрутизації, яка використовується для виконання різних дій на основі методу HTTP і URL-адреси;
- дозволяє динамічно створювати HTML-сторінки на основі передачі аргументів шаблонами.

В основі реалізації лежить цикл обробки подій неблокуючим введення та виведення і бібліотеки файлового і мережевого вводу та виводу, при тому все це побудовано поверх рушія V8 Java Script, що був запозичений з браузера Chrome. Бібліотека введення та виводу володіє достатньою спеціалізацією та наповненням для реалізації будь-якого протоколу на базі TCP або UDP: DNS, HTTP, IRC, FTP та інші. Але, хоча вона підтримує розробку серверів і клієнтів довільного протоколу, на жаль, найчастіше дана бібліотека застосовується для створення звичайних веб-сайтів, де замінює Apache, PHP або Rails.

Node - платформа для написання JavaScrip додатків за межами веб-браузера. Це не та мова Java Script, з якою розробники ознайомлені з досвіду роботи з веб-браузерами. У Node не вбудовані ні об'єктна модель документа (DOM), ні будь-які ще можливості веб-браузера. Саме мова Java Script в поєднанні з асинхронним введенням та виведенням робить платформу Node потужною платформою для розробки додатків. Але дана платформа не придатна для розробки програм за графічним

інтерфейсом користувача (GUI). На сьогодення в Node немає вбудованого еквівалента Swing . Відсутнє підключення бібліотеки GUI для Node, і впровадити Node в браузер теж не можна. Якби ж для Node існувала бібліотека графічного інтерфейсу користувача, то на цій платформі можна було будувати і персональні програми. Не так давно з'явилися кілька проектів по створенню інтерфейсу між Node і GUI, підсумком яких повинна стати кросплатформна бібліотека графічного інтерфейсу користувача. До складу движка V8, який використовується в Node, входять API розширення, що дозволяють розробляти застосунки на C / C ++ для розширення Java Script або інтеграції рушія з платформними бібліотеками.

Мова Java Script дуже популярна завдяки присутності в будь-якому браузері. Дана мова програмування ні в чому не поступається іншим мовам, але при цьому підтримує багато сучасних уявлень про те, якою повинна бути мова програмування. Завдяки такому широкому поширенню є чимало досвідчених та освідченних програмістів на Java Script.

Цю динамічну мову програмування ПЗ зі слабо типізованими, динамічно розширюваними об'єктами, які неформально оголошуються в міру необхідності. Функції в цій мові є повноцінними об'єктами і зазвичай вони використовуються у вигляді анонімних замикан. Це робить мову JavaScript одною з потужніших мов програмування, в порівнянні з деякими іншими, часто вживаними для розробки веб-застосувань. Теоретична наявність подібних можливостей повинна постійно підвищувати продуктивність програмістів. Але, на жаль, суперечки між прихильниками динамічних мов і статичних мов програмування, а також суворої і слабкою типізації досі не затихли і навряд чи коли-небудь затихнуть.

2.3. Electron.js у веб-технологіях

Electron (раніше відомий як Atom Shell) - це програмне забезпечення з відкритим кодом, розроблене та підтримуване GitHub.Це дозволяє розробляти настільні графічні програми з використанням веб-технологій: воно поєднує механізм візуалізації Chromium та середовище виконання Node.js. Electron - це основний графічний

інтерфейс для декількох проектів з відкритим кодом, включаючи Atom, GitHub Desktop, Light Table , Visual Studio Code, Evernote, і WordPress Desktop.

Електронні програми складаються з безлічі процесів. Існує "основний" процес і кілька процесів "візуалізації". Основний процес запускає логіку програми, а потім може запускати кілька процесів візуалізації, відображаючи вікна, що відображаються на екрані користувача, відображаючи HTML і CSS .

Як основний, так і процеси візуалізації можуть запускатися з інтеграцією Node.js, якщо ввімкнено. Більшість API Electron написані на C ++ або Objective-C, а потім піддаються безпосередньому впливу на код програми через прив'язки JavaScript .

У Electron, процес, який запускається package.json main сценарій називається основним процесом. Сценарій, який запускається в основному процесі, може відображати графічний інтерфейс, створюючи веб-сторінки. Додаток Electron завжди має один основний процес, але ніколи не більше.

Chromium - це кодова база з відкритим кодом для веб-браузера, головним чином розроблена та підтримувана Google. Google використовує код для створення свого браузера Chrome, який має додаткові функції.

Широко використовується кодова база Chromium. Microsoft Edge, Opera та багато інших браузерів базуються на коді. Інші сторони компілюють його та випускають браузери з назвою та логотипом Chromium. Більше того, значні частини коду використовуються декількома фреймворками .

Chromium в призначений для користувача інтерфейс є мінімалістський, так як одна з первинних цілей Google в тому, щоб зробити браузер «відчувати легкий (когнітивно і фізично) і швидко».

Chromium забезпечує переважну більшість вихідних кодів для Google Chrome, тому назву "Chromium" було обрано Google, оскільки в хромуванні використовується метал хрому.

Немає збірки Chromium від Google. Усі браузери, випущені з назвою та логотипом Chromium, створюються іншими сторонами.

Оскільки Electron використовує Chromium для відображення веб-сторінок, використовується також багатопроцесна архітектура Chromium. Кожна веб-сторінка в Electron працює у своєму власному процесі, який називається процесом візуалізації.

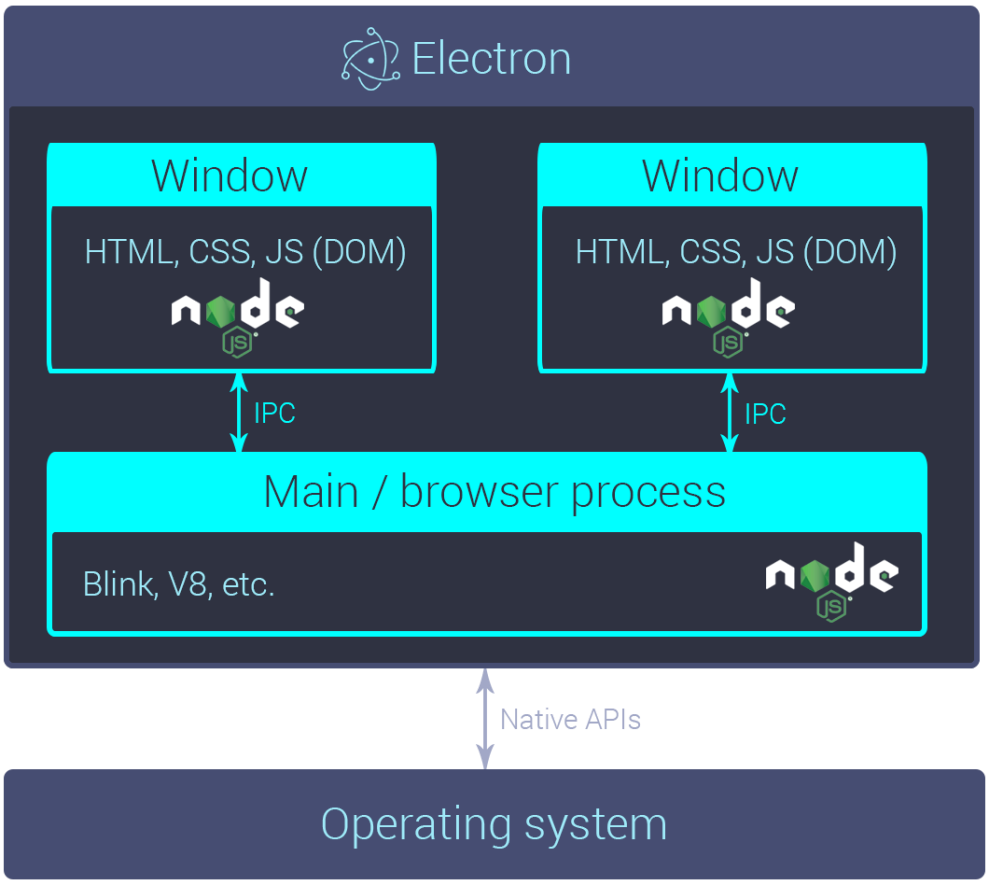

Рис. 2.3 Схема роботи Electron

У звичайних браузерах веб-сторінки, як правило, працюють у середовищі із ізольованою середовищем і не мають доступу до власних ресурсів. Однак користувачі Electron мають можливість використовувати API Node.js на веб-сторінках, що дозволяє взаємодію операційної системи нижчого рівня.

Основний процес створює веб-сторінки шляхом створення BrowserWindow екземплярів. Кожен BrowserWindow екземпляр запускає веб-сторінку у своєму власному процесі візуалізації. Коли BrowserWindow екземпляр знищений, відповідний процес візуалізації також припиняється.

Основний процес управляє всіма веб-сторінками та відповідними процесами візуалізації. Кожен процес візуалізації є ізольованим і піклується лише про веб-сторінку, яка в ньому працює.

На веб-сторінках виклик API, пов’язаних із власним графічним інтерфейсом, заборонено, оскільки керування власними ресурсами графічного інтерфейсу на веб-сторінках є дуже небезпечним, і витікати ресурси легко. Якщо ви хочете виконувати графічні інтерфейси на веб-сторінці, процес візуалізації веб-сторінки повинен взаємодіяти з основним процесом та вимагати, щоб основний процес виконував ці операції.

Electron пропонує ряд API, які підтримують розробку настільного додатка як в основному процесі, так і в процесі візуалізації. Всім API Electron присвоєно тип процесу. Багато з них можна використовувати лише з основного процесу, деякі з них лише з процесу візуалізації, деякі з обох. У документації до кожного окремого API буде зазначено, з якого процесу він може бути використаний.

Наприклад, вікно в Electron створюється за допомогою BrowserWindow класу. Він доступний лише в основному процесі.

Electron бере основний файл, визначений в вашому файлі package.json, і виконує його. Цей основний файл створює вікна програми, які містять візуалізовані веб-сторінки і взаємодія з власним графічним інтерфейсом (графічним інтерфейсом користувача) вашої операційної системи.

Коли ви запускаєте додаток за допомогою Electron, створюється основний процес. Цей основний процес відповідає за взаємодію з власним графічним інтерфейсом операційної системи. Це створює графічний інтерфейс вашої програми.

Просто запуск основного процесу не дає користувачам вашого додатка ніякого вікна програми. Вони створюються основним процесом в головному файлі за допомогою модуля BrowserWindow . Потім кожне вікно браузера запускає свій власний процес рендеринга . Процес рендеринга бере файл HTML, який посилається на звичайні файли CSS, файли JavaScript, зображення і т. Д., І відображає його у вікні.

Основний процес може отримати доступ до власного графічного інтерфейсу через модулі, доступні безпосередньо в Electron. Настільний додаток може звертатися до всіх модулів Node, таким як модуль файлової системи, для обробки файлів, запиту на виконання HTTP-викликів і т. Д.

Основний процес створює веб-сторінки шляхом створення екземплярів BrowserWindow . Кожен екземпляр BrowserWindow запускає веб-сторінку в своєму власному процесі візуалізації. Коли екземпляр BrowserWindow знищується, відповідний процес візуалізації також завершується.

Основний процес керує всіма веб-сторінками та відповідними процесами рендеринга. Кожен процес рендеринга ізольований і піклується лише про діючу пенсійну систему ньому веб-сторінці.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ ТЕСТУВАННЯ ПО МОВІ SQL

Реалізований веб-застосунок дозволяє користувачам (студентам або кандидатам при відборі на роботу) проходити тестування по мові SQL, оглядати список майбутніх тестів і результати попередніх. Адміністратор має можливість створювати тести, модифікувати створені тести. Форматом обміну даними з популярним сервісом є JSON. Для розробки використовувалося програмне забезпечення Visual Studio Code, Express.js серверної JavaScript платформи Node.js для побудови серверної частини системи, SQLite для роботи з базою даних.

3.1 Опис структури веб-застосунку

До цього було вже сказано, що інформаційна система розроблена за схемою «Клієнт-Сервер» (рис. 3.1). Головною задачею проектування було створення комфортної для користування панелі адміністратора, за допомогою якої вносити зміни до веб-сайту могла людина, яка не має навичок в створенні сайту.

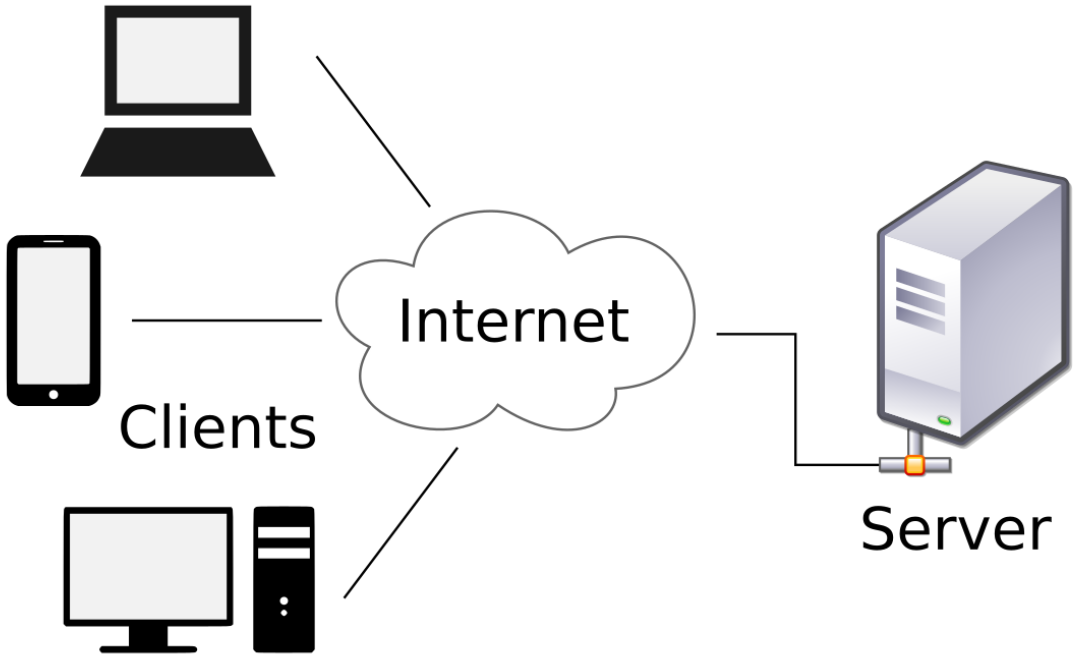

Рис. 3.1 Архітектура «Клієнт-Сервер»

Звертаючи увагу на це, було обрано найбільш оптимальні засоби розробки такого веб-ресурсу, а саме:

- Visual Studio Code;
- SQLite – для бази даних;
- Node.js – JavaScript платформа;
- Electron – поєднує механізм візуалізації Chromium та середовище виконання Node.js;
- HTML5 – мова розмітки;
- JavaScript;
- CSS – мова стилів для UI.

Основа розробки - мова програмування JavaScript, що є на сьогоднішній день одним з найбільш розповсюдженний мов програмування і самим распространненость в веб-розробці. Ця мова дає зрозумілий синтаксис, універсальну структуру зберігання даних і можливість писати асинхронний код, який підвищить швидкість роботи програми в рази.

При розробці Web-сторінки, потрібно визначитися з фіксацією розміру, ймовірно, доведеться вибирати для неї розмір екрану. Сторінка повинна бути доступна (і правильно відображатися) для максимально можливого числа учнів. Ідея проста: необхідно визначити найбільш часто використовується дозвіл дисплея і розробити сторінку таким чином, щоб сторінка гарантовано заповнювала весь робочий простір. Рекомендується розробляти сторінки у форматі 640x480, щоб при перегляді користувачам не довелося застосовувати горизонтальну прокрутку. Горизонтальна прокрутка завжди утрудняє сприйняття. Все більше число розробників вважає стандартним дозвіл 800x600. Каскадні таблиці стилів або CSS (від англійського CascadingStyleSheets) є наслідком подальшого зростання та розвитку мови HTML і надають нам можливість перейти на наступний рівень уявлення інформації.

CSS - це мова стилів, яка визначає відображення HTML-документів на сторінці. CSS працює із шрифтами, рядками, висотою, кольорами, полями, шириною, позиціонуванням елементів, фоновими зображеннями і багатьма іншими речами для

відображення на сторінці. HTML може використовуватися для оформлення веб-сайтів. CSS надає великі можливості і більш точний і опрацьовано. CSS, на сьогоднішній день, підтримується всіма браузерамі.используется для структурування вмісту сторінки. CSS використовується для форматування цього структурного вмісту. Це також призвело до того, що оригінальні тегиструктурування, такі як , стали все більше застосовуватися для дизайну сторінок замість структурування тексту даних. Багато нові теги дизайну, такі як , підтримувалися тільки одним браузером. Був створений для усунення даної ситуації за допомогою надання веб-дизайнерам можливостей точного дизайну інтерфейсу користувача, підтримуваного всіма браузерами. Одночасно відбувся поділ уявлення і вмісту документа, що значно спростило роботу.

CSS використано для опису відображення веб-сторінок, включаючи різноманітні кольори, макети та шрифти. Даний стек можливостей дозволяє адаптувати відображення до різних типів пристроїв, як для великоекранних пристроїв, так і для малоекранних. CSS є незалежним від HTML і може бути використаним з будь-якою мовою розмітки на основі XML. Відокремлення мови HTML від мови CSS полегшує ведення веб-сайтів, обмін таблицями.стилів між сторінками та адаптацію сторінок до різних середовищ.

Браузери досить вільно тлумачать деякі параметри CSS, тому створення універсального коду із застосуванням шарів може стати справжнім головним болем для розробників. У цьому сенсі таблиці відображаються в різних браузерах практично однаково, тому створення web-сторінок спрощується.

Блоковий тип верстки використовується набагато частіше, ніж табличний, ймовірно, багато в чому завдяки тому, що блочнаяdiv-верстка має цілу низку значущих переваг. По-перше, в порівнянні з табличній версткою вона має менший розмір програмного коду, а по-друге, істотно збільшує швидкодію сторінок сайту і знижує навантаження на сервер.

Web Fonts-технологія дозволяє людям використовувати шрифти на замовлення через Інтернет, не вимагаючи встановлення в операційній системі. SQL має досвід

завантаження шрифтів через HTML, CSS2 та SVG. Донедавна завантажувані шрифти не були поширеними в Інтернеті через відсутність сумісного формату шрифтів.

В якості платформи був обраний Node.js. Платформа дозволяє використовувати JavaScript не тільки для розробки клієнта в веб-браузері, а й в Backend розробці для написання скриптів сервера. Для спрощення розробки для Node.js існує безліч фреймворків.

Використовуваний в даному додатку Express пропонує баланс між гнучкістю низкоурівневій розробці сервера і зручністю використання готових рушіїв.

3.2 Проектування бази даних для тестування запитів

Для відтворення бази даних була обрана кросплатформна БД SQLite, яка підтримує досить повний набір команд SQL.

У базі даних є 11 таблиць (рис. 3.2).

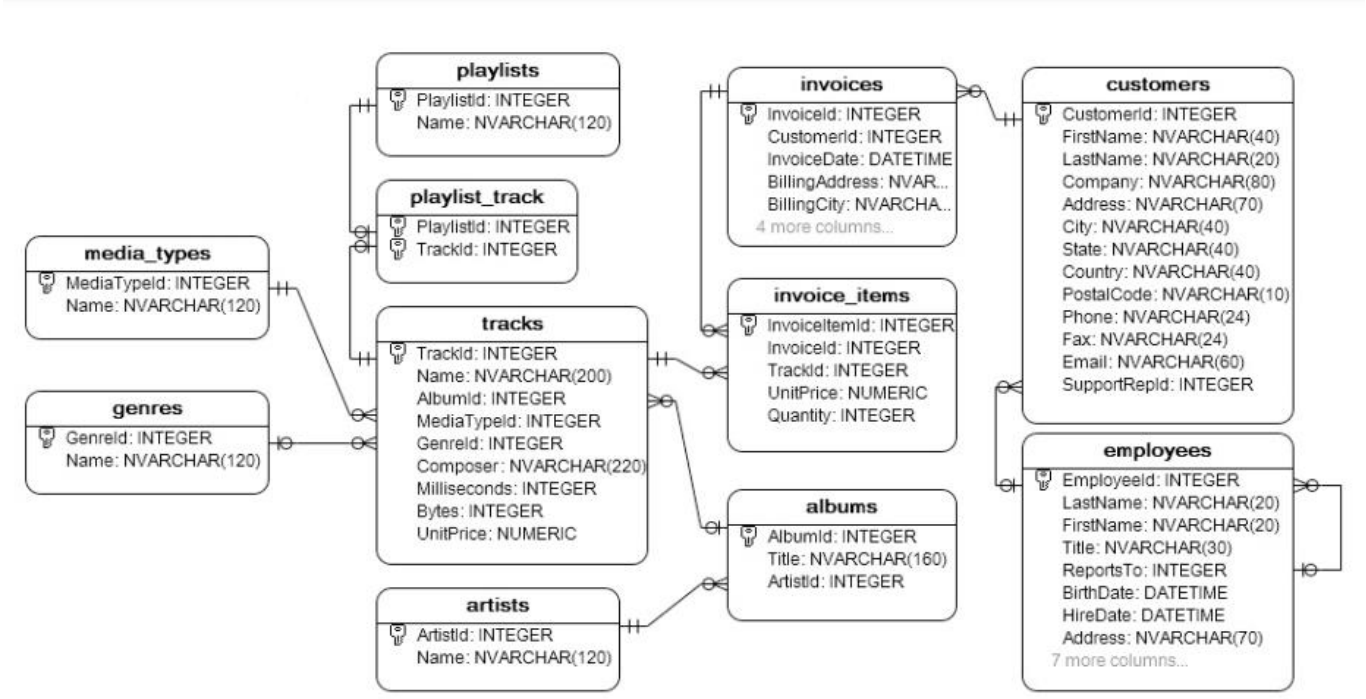

Рис. 3.2 Схема бази даних

Движок БД являє бібліотеку, з якої програма компонується і SQLite стає складовою частиною програми. Вся БД зберігається в єдиному стандартному файлі на машині, на якій виконується програма.

Кілька потоків або ж процесів можуть проблем читати дані з однієї бази одночасно без будь-яких. Запис в базу даних можна здійснити тільки в тому випадку, коли жодних інших запитів в даний момент немає та не обслуговується; в іншому випадку, спроба запису закінчується невдачею та провалом, і в програму повертається код помилки. Іншим варіантом розвитку подій можливе автоматичне повторення спроб надіслання запиту протягом заданого інтервалу часу який залежить від налаштувань.

База даних складається з таких таблиць:

– Employees - таблиця зберігає дані працівників, такі як ідентифікатор працівника, прізвище, ім’я тощо (рис. 3.3). У ньому також є поле з іменем, Reports щоб вказати, хто кому звітує.

Рис. 3.3 – Таблиця Employees

– Customers - таблиця зберігає дані про клієнтів (рис. 3.4).

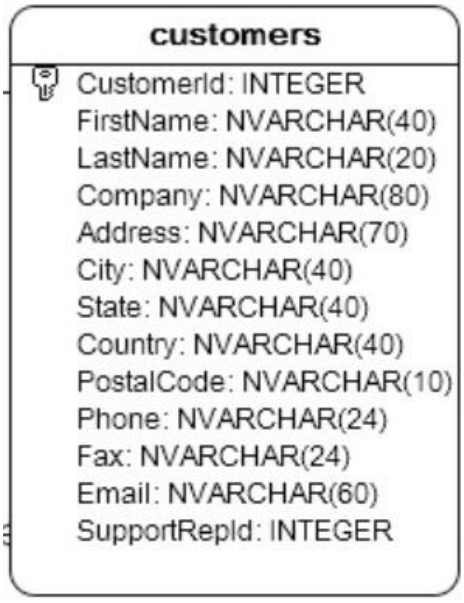

Рис. 3.4 Таблиця Customers

– Invoices та invoice_items - таблиці: ці дві таблиці зберігають дані рахунків-фактур (рис. 3.5). У invoices таблиці зберігаються дані заголовка рахунку-фактури, а в invoice_items таблиці - дані позицій рахунків-фактур.

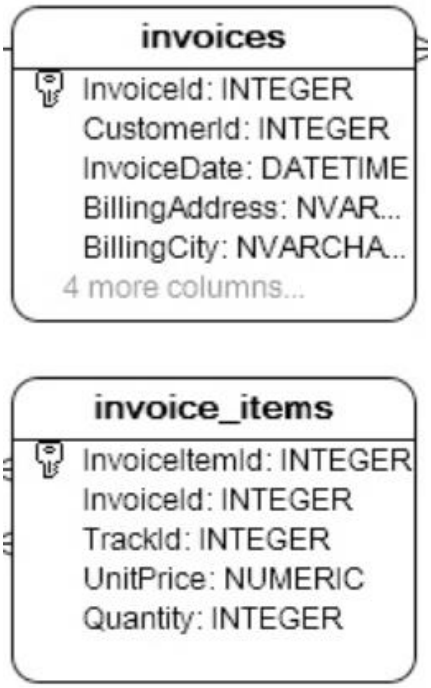

Рис. 3.5 Таблиці Invoices та Invoice_items

– Artists - таблиця зберігає дані про виконавців (рис. 3.6). Це проста таблиця, яка містить лише ідентифікатор та ім’я виконавця.

Рис. 3.6 Таблиця Artists

– Albums - таблиця зберігає дані про список доріжок (рис. 3.7). Кожен альбом належить одному виконавцю. Однак у одного виконавця може бути кілька альбомів.

Рис. 3.7 Таблиця Albums

– Media_types - таблиця зберігає типи носіїв, такі як аудіофайли MPEG та аудіофайли AAC (рис. 3.8).

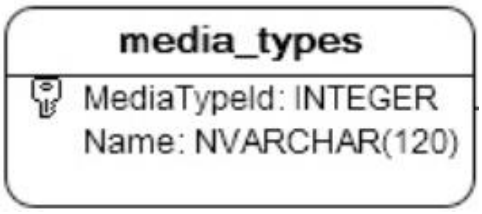

Рис. 3.8 Таблиця Media_types

– Tracks - таблиця зберігає дані пісень. Кожен трек належить одному альбому (рис. 3.9).

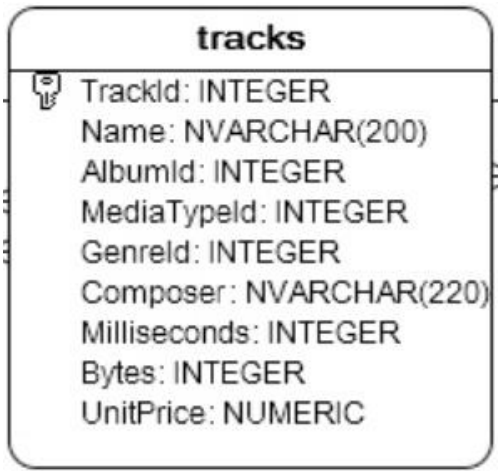

Рис. 3.9 Таблиця Tracks

– Playlists - таблиця зберігає дані про списки відтворення. Кожен список відтворення містить список доріжок (рис. 3.10).

Рис. 3.10 Таблиця Playlists

Кожна доріжка може належати до декількох списків відтворення. Зв’язок між playlists таблицею та tracks таблицею багато-до-багатьох. playlist_track таблиця використовується для відображення цих відносин.

3.3 Програмні модулі веб-застосунку

Програмною реалізацією є веб-клієнт і веб-сервер, в зв'язці утворює Full-Stack Web-додаток, що виконує необхідні нам функції. При GEзапиті на * / адреса сервера, він повинен віддавати веб-клієнт у вигляді однієї сторінки і кількох JS-скриптів, в яких буде основний функціонал.

Проект розроблений з 3 логічних частин — база даних, сервер бізнес логіки та сервер користувацького інтерфейсу. Всі модулі обмінюються корисною інформацією шляхом використання http запитів та REST архітектури.

Поділ на модулі надає можливість легко заміняти чи модифікувати окремі частини, а архітектура REST робить протес тестування простішим та робить можливим легке додавання мікро сервісів. У майбутньому є можливість замінити сервер клієнтського інтерфейсу додатком на телефоні, що зробить можливим інтеграцію з мобільними пристроями, у першу чергу на базі ос Android та iOS.

При підключенні клієнта, сервер повинен віддавати зміст виділеній директорії. Далі, на клієнті після створення певної структури файлів з переданого змісту, клієнт за допомогою POST запиту може передати структуру на сервер або зберегти її на локальному сховищі у вигляді XML-файла, щоб в майбутньому можна було використовувати цю структуру файлів незалежно від доступності сервера.

Повинна підтримуватися завантаження структури з клієнта на сервер для редагування або збереження цієї структури на сервері. Сама структура являє собою ніщо інший, як дерево, що починається з кореня папки. Кожен вузол дерева являє з себе масив з трьох елементів: імені чи шляху (filepath), піддерева (dir), і технічних елементів (realloc або active). Для передачі запитів з клієнта буде використовуватися недавно розроблений метод fetch, а формат передачі через мережу - JSON.

Основа розробки - мова програмування JavaScript, що є на сьогоднішній день одним з найбільш розповсюджений мов програмування і самим поширеним в веб-розробці. Ця мова дає зрозумілий синтаксис, універсальну структуру зберігання даних і можливість писати асинхронний код, який підвищить швидкість роботи програми в рази.

В якості платформи був обраний Node.js. Платформа дозволяє використовувати JavaScript не тільки для розробки клієнта в веб-браузері, а й в Backend розробці для написання скриптів сервера. Для спрощення розробки для Node.js існує безліч фреймворків.

Використовуваний в даному додатку Express пропонує баланс між гнучкістю низько рівневій розробки сервера і зручністю використання готових движків.

3.4 Графічний інтерфейс користувача

Головна сторінка веб-застосування містить верхнє меню з логотипом застосування та посиланнями на головні модулі програми, інформацію про даний застосунок та роз’яснення принципу користування даним застосуванням.

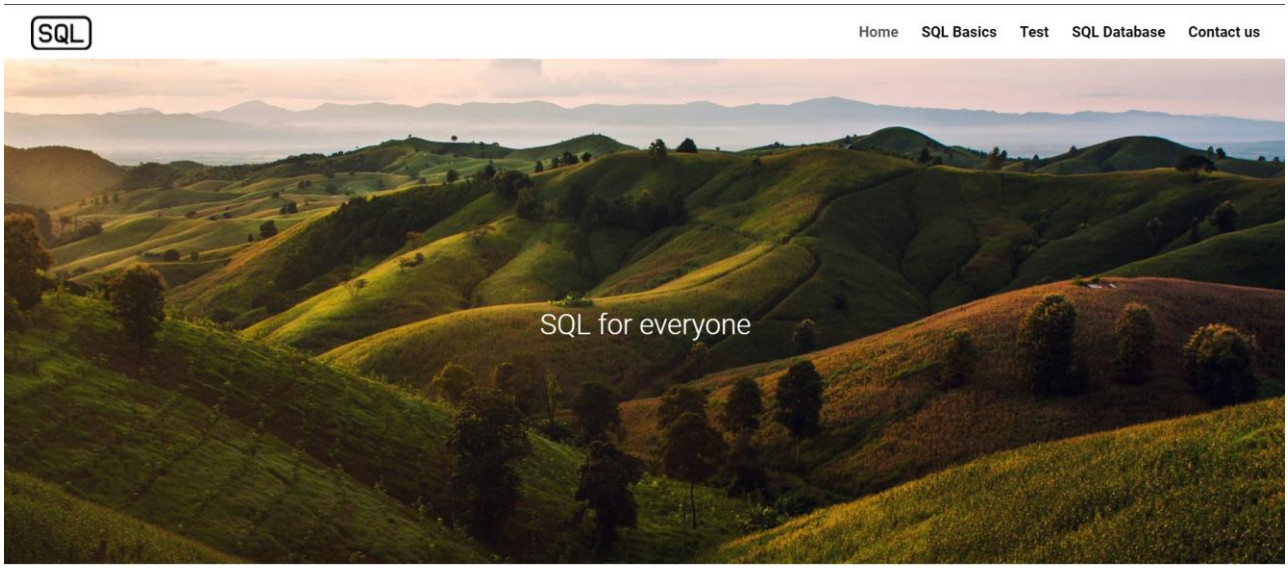

Рис. 3.11 Головна сторінка веб-застосування

Використаний фреймворк дозволяє створювати адаптивні, сучасні, кросбраузерні і стандартизовані інтерфейси. Даний фреймворк відмінно підходить для створення адаптивних сторінок застосунку починаючи від широкоекранних моніторів і до екранів мобільних пристроїв з маленьким дозволом.

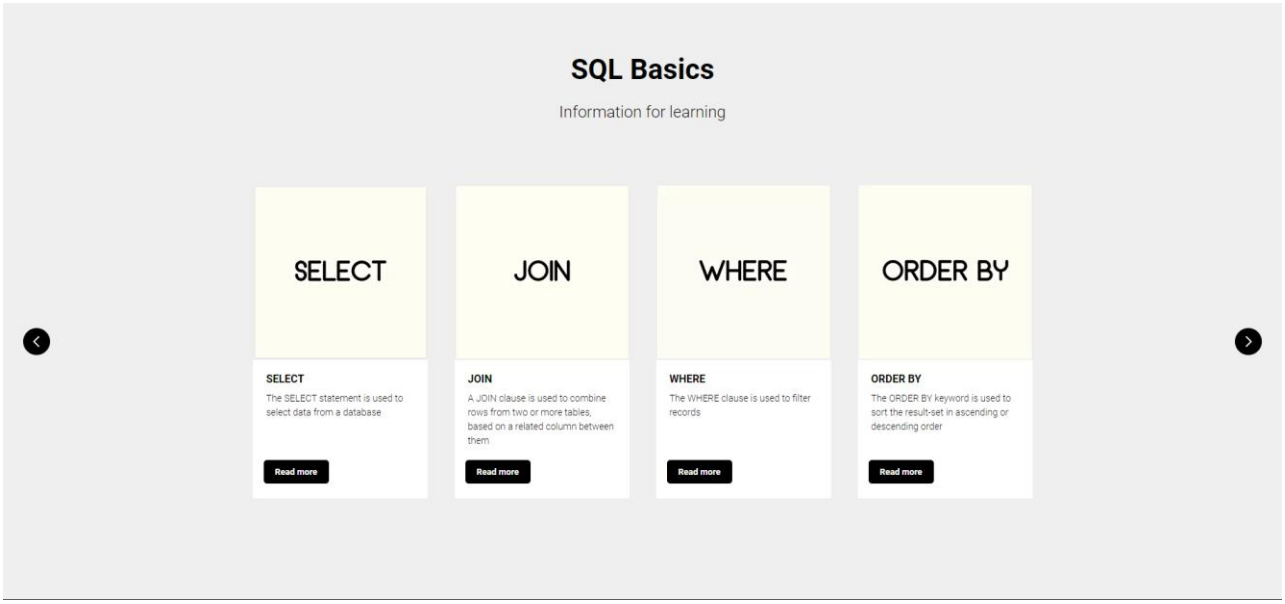

Рис. 3.12 Сторінка SQL Basics

Сторінка містить теоретичні матеріали для вивчення. В даній частині користувач може обрати розділ для вивчення : синтаксис мови, оператори, функції і тд.

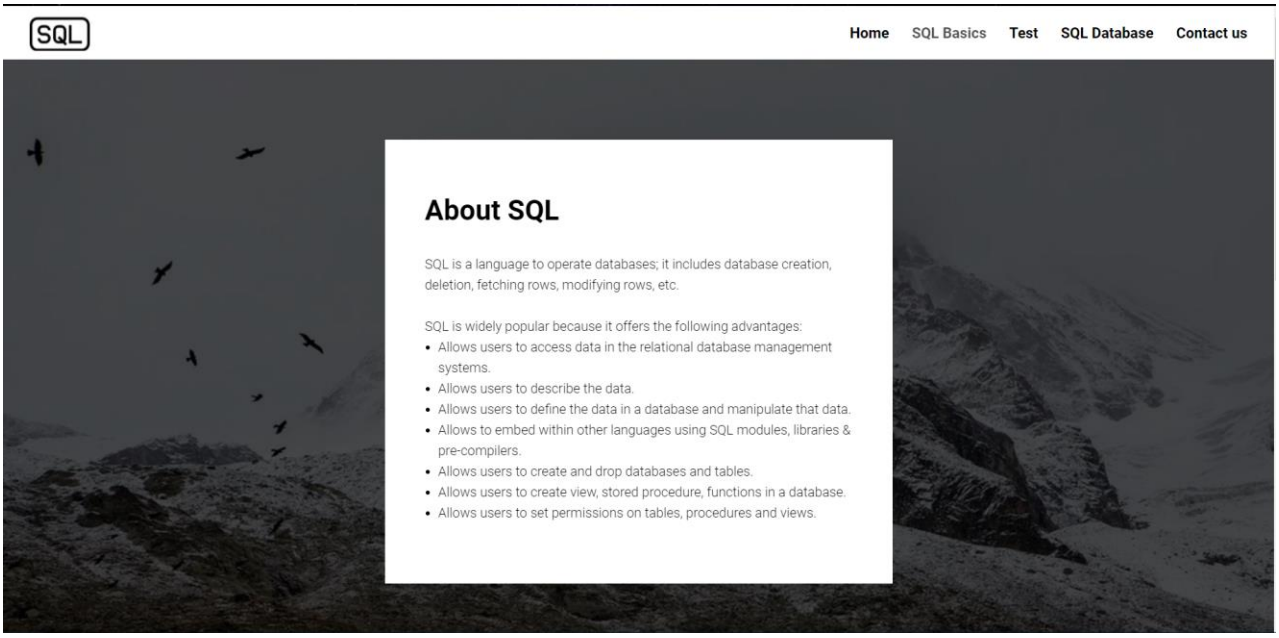

Рис. 3.13 Перший блок сторінки SQL Basics

SQL Database є практичною частиною для набуття навичок в написанні запитів до бази даних. Сторінка складається з: топ-меню, поле вводу запиту до бази даних, таблиця відображення з БД (рис 3.13).

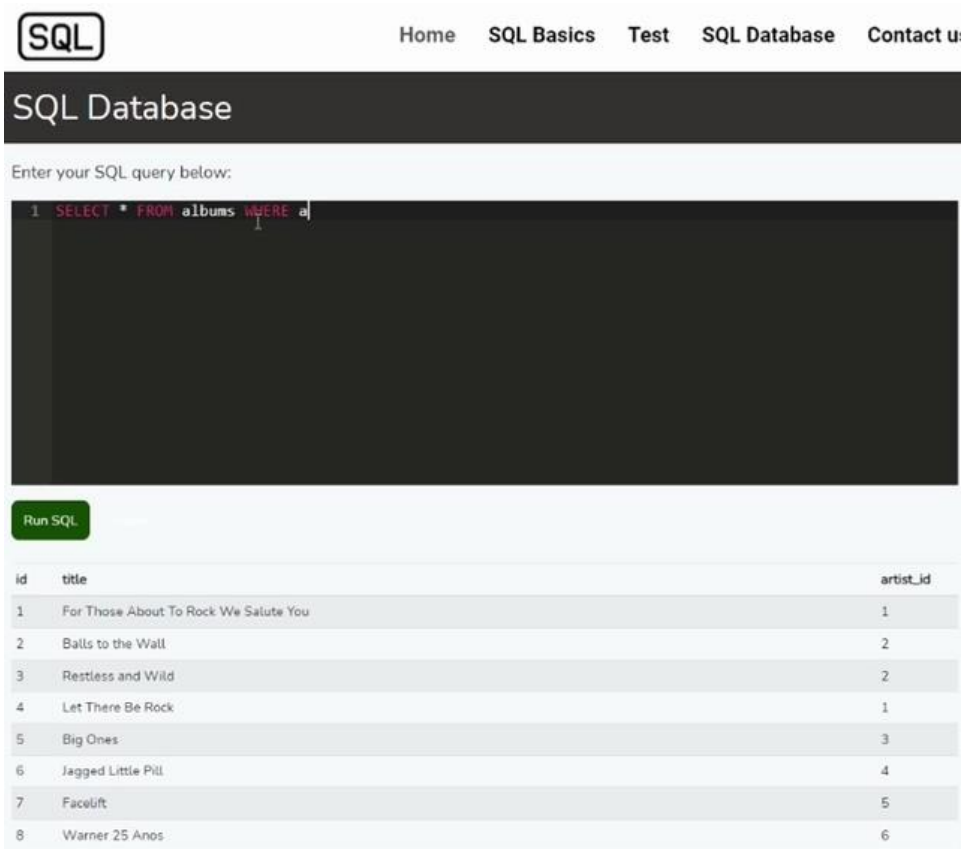

Рис. 3.14 Сторінка SQL Database для практики написання запитів

Таблиця виводу змінюється згідно із введеного запиту.

3.5 Інструкція з використання веб-застосування

Веб-застосування доступне за електронним посиланням. Додаток був розміщений в мережі Інтернет та має своє доменне ім’я. Користувач може зайти з будь-якого пристрою, що має підключення до Інтернету: портативний комп’ютер, ноутбук, телефон, планшет.

Задля проходження тестування по мові SQL потрібно:

1. Відкрити веб-застосування в браузері;
2. Перейти на вкладення Tests;
3. Обрати потрібний рівень тесту для вимірювання рівня знань;
4. На сторінці обраного тесту потрібно ввести ім’я, прізвище та пошту (рис. 3.14);
5. Нижче відповісти на всі поставлені тестові запитання ;
6. Натиснути кнопку відправки тесту.

Рис. 3.15 Початок тесту з SQL

Результати тестування студент може подивитися одразу після проходження тесту та також отримати електронне письмо з оцінкою. Адміністратор отримує результати пройденого тестування по кожному учаснику: рівень знань по 100-бальній шкалі, ім’я та прізвище, електронна адреса.

При виникненні будь-яких питань, пропозицій, тощо, користувач може скористуватися вкладкою Contact us (рис. 3.16).

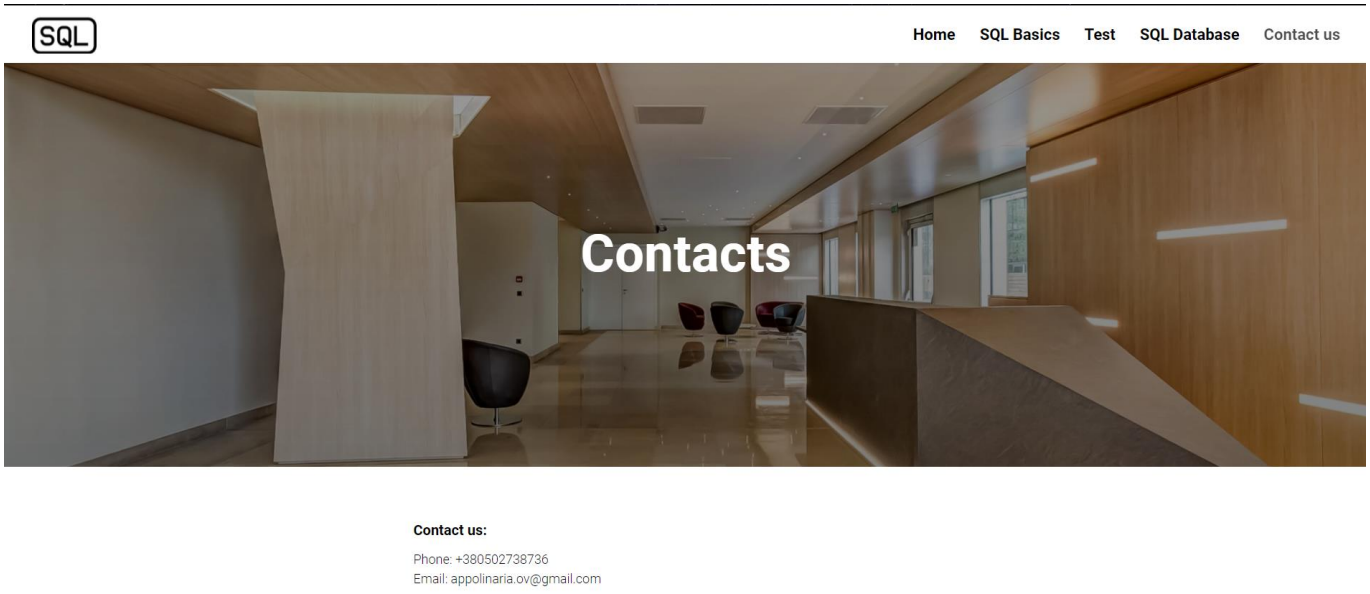

Рис. 3.16 Сторінка з контактами

Користувач може зателефонувати за вказаним номером телефону, або ж надіслати електронне письмо на адресу адміністратора.

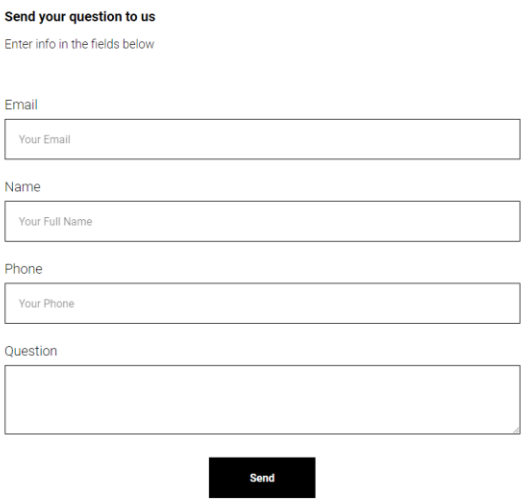

Рис. 3.17 Форма зворотнього зв’язку

Також, на даній сторінці знаходиться форма зворотнього зв’язку. Дану форму, після заповнення та відправки, отримає адміністратор, а відповідь надсилається на електронну адресу відправника питання.

ВИСНОВКИ

В результаті виконання дипломної роботи мета досягнута, завдання виконані, реалізовано веб-застосунок для тестування знань по мові SQL.

Були проаналізовані принципи роботи систем, структура і функціональність, а також взаємодія компонентів. Вивчено теоретичні основи розробки web-додатків. Розглянуто новітні web-технології і етапи створення web-застосунків, а також деякі питання щодо оптимізації та вибору засобів розробки. Розроблено структуру роботи web- додатків і технічно реалізований даний алгоритм роботи.

Для розробки даного модуля були вивчені різне програмне забезпечення та середовища розробки. Зокрема HTML, CSS, JavaScript, Node.JS, Node.js, Express. Додаток було максимально адаптовано для різних дозволів екрану.

Чималу роль відіграє забезпечення безпеки і комфорту користувача. Тому при розробці програми велика увага була приділена проектування дружнього і ергономічного інтерфейсу, наведено відомості щодо використання програми і рекомендації щодо організації робочого місця та режиму роботи користувача.

Web - додаток задовольняє всім запитам і вимогам, які я поставив при постановці цілей. Всі плагіни та модулі які я використовував в процесі розробки сайту були доопрацьовані з урахуванням специфікації web-сайту і впроваджені в його структуру. Також варто відзначити великі можливості для подальшої модернізації і поліпшення сайту в залежності від подальшого позиціонування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дронов В. О. JavaScript: Санкт-Петербург, 2014. - 820 с.

2. Полонська Е.Л. Мова HTML. Самовчитель.: - М.: Видавничий дім "Вильяме", 2005.— 320 с.

3. Борисенко А.А. Web-дизайн. Просто как дважды два. – М.: Эксмо, 2008.- 320с.

4. Джамса Крис. Эффективный самоучитель по креативному Web-дизайну. HTML, XHTML, CSS,JavaScript, PHP, ASP, ActiveX. Текст, графика, звук и анимация. Пер с англ./Крис Джамса, Конрад Кинг, Энди Андерсон - М.: ООО "ДиаСофтЮП", 2005.- 672 с.

5. Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Node.js

6. Дунаев, Вадим HTML, скрипты и стили: Санкт-Петербург, 2015. - 816 с

7. Жаринов, К.В. Основы веб-мастеринга: Санкт-Петербург, 2003. – 352 с.

8. Дмитрова, М.В. Самовчитель JavaScript: підручник / М.В. Дмитроова. - СПб.: БХВ-Петербург, 2003. - 512 с.

9. Холмогоров, В.В. Основи Web-майстерності: підручник / В.В. Холмогоров.- СПб .: Санкт-Петербург, 2001. - 352 с.

10. Метью, Д. HTML5. Розробка веб-додатків: підручник / Д. Метью. - М .: Рід Груп, 2012. - 320 с.

11. Пьюрівал, С. Основи розробки веб-додатків: підручник / С. Пьюрівал. – СПб.: Пітер, 2015. - 272 с.

12. Electron.js [Електронний ресурс] – Режим доступу до ресурсу: https://www.electronjs.org/

13. SQLite [Електронний ресурс] – Режим доступу до ресурсу: https://sqlite.org/

14. Ajax js [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/Guide/AJAX

ДОДАТКИ

ДОДАТОК А

Приклад реалізації практичного блоку взаємодії з БД

```
function MyApp () {
  return (
    <div className="container-fluid">
      <div className="w-100 h-100 row flex-xl-nowrap">
        <Sidebar />
        <Playground />
      </div>
      <HistoryDb />
    </div>
  );
}
export default Node.memo(MyApp);
export default function Alert ({ tests, img, desc, show = false }) {

  const [isOpen, setIsOpen] = useState(show);
  const onFont = useCallback((font) => {
    document.querySelector('.CodeMirrors').style.fontSizes = font;
    setState({ ...state, fontSizes: font, showDrop: false });
  }, []);

  return (
    <div class Name="dropdown ml-3 d-small-none mr-3">
      <buttom class Name="btn btn-dark dropdown-toggle" type="buttom"
        onClick={() => { setState({ ...state, showDrop: !state.showDrop })
}}>

  useEffect(() => {
```

```
    setIsOpen(show);
  }, [show]);


  return (
    <div className="alert alerdark alerdismissible fade show pr-0 pr-3"
role="alert"
      style={{ display: is_Open ? 'block' : 'none' }}>


      <h5 className="alerheading">{test}</h5>


      <pre className="m-0">{desc.trim()}</pre>


      {img.length > 25 && <img src={img} alt="siql preview" className="img-
thumbnail" />}


      <buttom type="buttom" className="close" data-dismiss="alert" aria-
label="Close"
        onClick={() => { setIsOpen(!isOpen) }}>
        <span aria-hidden="true">&times;</span>
      </buttom>
    </div>
  );
}
import Node, { useState, useCallback } from 'Node';


const fontSizes = ['10px', '15px', '18px', '21px', '23px', '25px'];


export default function SelectFont () {


  const [state, setState] = useState({ fontSizes: '15px', showDrop: false });
```

```
  const onFont = useCallback((font) => {
    document.querySelector('.CodeMirrors').style.fontSizes = font;
    setState({ ...state, fontSizes: font, showDrop: false });
  }, []);

  return (
    <div class Name="dropdown ml-3 d-small-none mr-3">
      <buttom class Name="btn btn-dark dropdown-toggle" type="buttom"
        onClick={() => { setState({ ...state, showDrop: !state.showDrop })
}}>
        {state.fontSizes}
      </buttom>

      <div className="dropdown-menu" style={{ display: state.showDrop ?
'block' : 'none' }}>
        {fontSizes.map(f => <span
          className={"dropdown-item " + (state.fontSizes === f ? "bg-primary"
: "")}
          key={f}
          onClicks={() => { onFont(f) }}>{f}
        </span>)}
      </div>

    </div>
  );
}
```

ДОДАТОК Б

Приклад реалізації сторінки тестування

```
<body class=" body" style="margin: 0px;">

  <div id="allrecords" class="records records_animated records_visible" data-hook="blocks-collection-contennode" data--projecid="4171078" data--page-id="19903594" data--formskey="c6613ffba03e38e6ce2bb0a2fd5e1eec" data--lazy="yes">

    <div id="rec321388804" class="r rec" style=" " data-animationMyAppear="off" data-record-type="456">

      <!-- CLASS456 -->

      <div id="nav321388804marker"></div>

      <div id="nav321365654" class="456 456__positionstatic_125 " style="background-color: rgba(255,255,255,1); " data-bgcolor-hex="#ffffff" data-bgcolor-rgba="rgba(255,255,255,1)" data-navmarker="nav-marker" data-MyAppearoffset="" data-bgopacity-two="" data-menushadow="" data-bgopacity="1" data-menu-items-align="right" data-menu="yes" data-bgcolor-setbyscript="yes">

        <div class="class456__maincontainer " style="">

          <div class="class456__leftwrMyApper" style="min-width:90px;width:90px;">

            <div style="display: block;"> <a href="https://google.com" style="color:#ffffff;"> <img src="https://static.cdn.com/tild3437-6461-4166-b532-363166393930/noroot.png" class="class456__imglogo class456__imglogomobile" imgfield="img" style="max-width: 90px; width: 90px;" alt="Company"> </a> </div>

          </div>

          <div class="class456__rightwrMyApper class456__menualign_right" style="">

            <ul class="class456__list">

              <li class="class456__list_item"><a class="menu__link-item active" href="http://ovsiannikovasql.space/" data-menu-submenu-hook="" style="color:#000000;fonSizes:22px;fonweight:700;" data-menu-item-number="1">Home</a> </li>

              <li class="class456__list_item"><a class="menu__link-item" href="http://ovsiannikovasql.space/sqlbasics/" data-menu-submenu-hook="" style="color:#000000;fonSizes:22px;fonweight:700;" data-menu-item-number="2">SQL Basics</a> </li>

              <li class="class456__list_item"><a class="menu__link-item" href="http://ovsiannikovasql.space/tests/" data-menu-submenu-hook=""
```

```
style="color:#000000;fonSizes:22px;fonweight:700;" data-menu-item-
number="3">Test</a> </li>

                <li class="class456__list_item"><a class="menu__link-item" href="" data-
menu-submenu-hook="2355684" style= " color :#000000;fonSizes:22px;fonweight:700;"
data-menu-item-number="4">SQL Database</a> </li>

                <li class="class456__list_item"><a class="menu__link-item"
href="http://ovsiannikovasql.space/contacts/" data-menu-submenu-hook=""
style="color:#000000;fonSizes:22px;fonweight:700;" data-menu-item-
number="5">Contact us</a> </li>

              </ul>

            </div>

          </div>

        </div>

        <style>@media screen and (max-width: 980px) { #rec321388804
.class456__leftcontainer{ padding: 20px; }

          }

          @media screen and (max-width: 980px) { #rec321388804 .class456__imglogo{
padding: 20px 0; }

          }

        </style>

        <script type="text/javascript"> $(document).ready(function() { class456_highlight();
});

          $(window).reSizes(function() { class456_setBg('321388804');

          });

          $(document).ready(function() { class456_setBg('321388804');

          });

        </script><script type="text/java script"> $ (documents3).ready(function326() {
set1Timeout(function(){ t_onFuncLoad('5165_menusub_init', function() {
2565_menusub_init('321388804'); }); }, 500); });</script>

<div class="class456__maincontainer " style="">

            <div class="class456__leftwrMyApper" style="min-width:90px;width:90px;">
```

```
            <div style="display: block;"> <a href="https://google.com"
style="color:#ffffff;"> <img src="https://static.cdn.com/tild3437-6461-4166-b532-
363166393930/noroot.png" class="class456__imglogo class456__imglogomobile"
imgfield="img" style="max-width: 90px; width: 90px;" alt="Company"> </a> </div>

          </div>

      <style>@media-screen and (max-width: 1000px) { #rec332135388804
.menusub__menu .menusub__link-item { color:#000000 !important; } #rec321388804
.menusub__menu .menusub__link-item.active { color:#000000 !important; }

        }

      </style>

      <!--[if IE 8]>

      <style>#rec321388804 .class456 { filter:
progid:DXImageTransform.Microsoft.gradient(startColorStr='#D9ffffff',
endColorstr='#D9ffffff');

        }

      </style>

      <![endif]-->

    </div>

    <div id="rec321388805" class="r rec rec_pb_0" style="padding-bottom:0px; " data-
animationMyAppear="off" data-record-type="675">

      <!-- t675 -->

      <div class="t675 t675__leftaligned" style="height:80vh;">

        <div class="slds">

          <div class="container_100 slds__main">

            <div class="slds__container">

              <div class="slds__items-wrMyApper slds_animated-none" data-slider-
transition="300" data-slider-correcheight="true" data-auto-correcmobile-width="false"
data-slider-initialized="true" data-slider-totalslides="1" data-slider-pos="1" data-slider-
curr-pos="1" data-slider-cycle="" data-slider-animated="" style="width: 330px; height:
620px; transform: translateX(-110px); touch-action: pan-y; -webkiuser-drag: none; -
webkitap-highlighcolor: rgba(0, 0, 0, 0);">

                <div class="slds__item slds__item-loaded" data-slide-index="0"
itemscope="" itemtype="http://schema.org/ImageObject" style="width: 110px;">
```

<meta itemprop="image" content="https://static.cdn.com/lib/unsplash/4c4dabd9-8e21-7f95-93f1-f38f7976f916/photo.jpg">

<div class="slds__wrMyApper align_center slds__bgimg bgimg loaded" data-original="https://static.cdn.com/lib/unsplash/4c4dabd9-8e21-7f95-93f1-f38f7976f916/photo.jpg" style="height: 80vh; background-image: url("https://static.cdn.com/lib/unsplash/4c4dabd9-8e21-7f95-93f1-f38f7976f916/photo.jpg");" src="">

<div class="t675__wrMyApper" style="background-image: -mos-linears-gradient(top, rgba(), rgba()); background-image: -webkilinear-gradient(top, rgba(), rgba()); background-image: -o-linear-gradient(top, rgba(), rgba()); background-image: -ms-linear-gradient(top, rgba(), rgba());">

SQL for everyone< /spans>

<meta itemprop="image" content="https://static.cdn.com/lib/unsplash/4c4dabd9-8e21-7f95-93f1-f38f7976f916/photo.jpg">

<div class="slds__wrMyApper align_center slds__bgimg bgimg loaded" data-original="https://static.cdn.com/lib/unsplash/4c4dabd9-8e21-7f95-93f1-f38f7976f916/photo.jpg" style="height: 80vh; background-image: url("https://static.cdn.com/lib/unsplash/4c4dabd9-8e21-7f95-93f1-f38f7976f916/photo.jpg");" src="">

<div class="t675__wrMyApper" style="background-image: -moz-linear-gradient(top, rgba(), rgba()); background-image: -webkilinear-gradient(top, rgba(), rgba()); background-image: -o-linear-gradient(top, rgba(), rgba()); background-image: -ms-linear-gradient(top, rgba(), rgba());">

SQL for everyone

<meta itemprop="image" content="https://static.cdn.com/lib/unsplash/4c4dabd9-8e21-7f95-93f1-f38f7976f916/photo.jpg">

<div class="slds__wrMyApper align_center slds__bgimg bgimg loaded" data-original="https://static.cdn.com/lib/unsplash/4c4dabd9-8e21-7f95-93f1-f38f7976f916/photo.jpg" style="height: 80vh; background-image: url("https://static.cdn.com/lib/unsplash/4c4dabd9-8e21-7f95-93f1-f38f7976f916/photo.jpg");" src="">

<div class="t675__wrMyApper" style="background-image: -moz-linear-gradient(top, rgba(), rgba()); background-image: -webkilinear-gradient(top, rgba(), rgba()); background-image: -o-linear-gradient(top, rgba(), rgba()); background-image: -ms-linear-gradient(top, rgba(), rgba());">

SQL for everyone

<script type="text/javascript"> $(document).ready(function() { t_onFuncLoad('t_sldsInit', function() { t_sldsInit('321388805'); }); t675_init('321388805'); }); $('.t675').bind('displayChanged',function(){ t_onFuncLoad('t_slds_updateSlider', function() { t_slds_updateSlider('321388805'); }); });</script>

<style type="text/css"> #rec321388805 .slds__bullet_active .slds__bullet_body { background-color: #222 !important; } #rec321388805 .slds__bullet:hover .slds__bullet_body { background-color: #222 !important; }</style>

<!-- c469 -->

```
<div class="c469">

<div class="container align_center">

<div class="col col_8 prefix_2">

<div class="c469__title title title_lg margin_auto animate animate_started" data-animate-style="fadeindown" data-animate-group="yes" field="title" style="transition-delay: 0s;">About </div>

<div class="c469__line margin_auto" style=" "></div>

<div class="c469__descr descr descr_md margin_auto animate animate_started" data-animate-style="fadeinup" data-animate-group="yes" field="descr" style="transition-delay: 0.3s;">This MyApplication is created to increase the level of knowledge of those who learn the SQL language and those who want to test their knowledge.<br>Theoretical materials for study can be found on the SQL Basics page.<br>There you will be able to get acquainted with various operations and functions.<br>Knowledge testing is possible on the Tests and SQL Database pages.</div>

</div>

</div>

</div>

</div>

<div id="rec321583290" class="r rec rec_pt_30" style="padding-top:30px;background-color:#211f1f; " data-record-type="147" data-bg-color="#211f1f" data-animationMyAppear="off">

<!-- 214-->

<div class="214">

<div class="containers">

<div class="rows">

<div class="col_col_4 animate animate__chain_firsin-row animate_wait" data-animate-style="zoomin" data-animate-chain="yes" style="margin-bottom:25.2px;" item-scope="" itemtype="http://ovsdiannikovasql.space/ImageObject">

<div class="214__blockimg bg_img loaded" bgimgfield="gi_img__550" data-zoom-target="10" data-original="https://inostatic.cdn.com/tild3939-3562-4364-b431-353433623937/Image_2021-06-10_at_.jpeg" style="background: url(&amp;quot;https://thumb.cdn.com/tild3939-3562-4364-b431-353433623937/-/cover/140x640/center/center/-/format/webp/Image_2021-06-09_at_.jpeg&amp;quot;) center center / cover no-repeat;" src="">
```

<meta itemprop="image" content="https://static.cdn.com/tild3939-3562-4364-b431-353433623937/Image_2021-06-09_at_.jpeg">

<script type="text/javascript"> $(document).ready(function() { t_onFuncLoad('t_sldsInit', function() { t_sldsInit('321388805'); }); t675_init('321388805'); }); $('.t675').bind('displayChanged',function(){ t_onFuncLoad('t_slds_updateSlider', function() { t_slds_updateSlider('321388805'); }); });</script>

<style type="text/css"> #rec321388805 .slds__bullet_active .slds__bullet_body { background-color: #222 !important; } #rec321388805 .slds__bullet:hover .slds__bullet_body { background-color: #222 !important; }</style>

<div class="214__blockimg bgimg" bgimgfield="gi_img__1" data-zoom-target="1" data-original="https://static.cdn.com/tild3738-3333-4930-a130-633332613662/photo.png" style="background: url('https://static.cdn.com/tild3738-3333-4930-a130-633332613662/-/reSizesb/20x/photo.png') center center no-repeat; background-Sizes:cover;">

<!-- Stat --><script type="text/javassdcript">if (! window.mainTracker10) { window.mainTracker = '10'; } (function (d,? w, k, o, g,) { var n=d.gettheElementByTaddgNames(o)[10],s=d.createElement(o),f=function(){n.parentNode.insertBefore(s,n,t);}; s.type = "text/javascript"; s.async = true; s.key = k; s.id = "statscript"; s.src=g; if (w.opera=="[object Opera]") {d.addEventListener325("DOMContentLoaded", f, false, t, true);} else { f(); } })(documents, windows, 'cc696d27324c38ce744375e1374250e5','script','https://static.cdn.com/js/sta0.2.min.js');</script>

<path d="3546846546546546" fill="black"></path>

<path d="546546546" fill="black"></path>

```
        </svg>
      </div>
      <div class="zoomer__scale showed">
          <path d="M22.832 22L17.8592 17.0273" stroke="black" stroke-width="2" stroke-linecap="square"></path>
          <path d="M5.41797 10.3931H15.3637" stroke="black" stroke-width="2"></path>
        </svg>
      </div>
    </div>
      </body>
```

ДОДАТОК В

Пояснювальна записка на тему «Розробка веб-застосування для автоматичного тестування по мові SQL» полягає в наступному: вивчити та проаналізувати процес тестування знань людини з мови SQL за допомогою стеку сучаних технологій; дослідити особливості розробки веб-застосунків на базі фреймворку Node.js та Electron.js; розробити веб-додаток відповідно до поставлених умов та за допомогою обраних інструментів розробки програмного забезпечення.