

Детектор Плагиата v. 2867 - Отчёт оригинальности: 15.06.2025 11:50:01

Проанализированный документ: ВКР Дворніков.docx Лицензия: ВОЛОДИМИР МАТІЄВСЬКИЙ

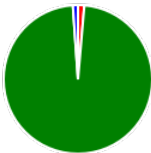
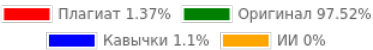
Тип поиска: Поиск переписанного Язык: Uk

Тип проверки: Интернет

ТЕЕ и кодировка: DocX n/a

Детальный анализ тела документа:

Диаграмма соотношения частей:



Граф распределения зон:



Источники плагиата: 16

→ 0,7% 40	1. https://lib.iitta.gov.ua/id/eprint/6472/1/БАЗОВІ_ПОНЯТТЯ.pdf
→ 0,5% 41	2. https://essuir.sumdu.edu.ua/bitstream/123456789/90967/1/Viunnik_mag_rob.pdf
→ 0,4% 32	3. https://krs.chmnu.edu.ua/jspui/bitstream/123456789/3738/1/Кваліфікаційна_магістерська_робота_Лосицький_П..pdf

Детали обработанных ресурсов: 213 - ОК / 21 - Ошибок

Важные замечания:

Википедия: Обнаружена Wiki!	Google Книги: [не обнаружено]	Сервисы платных работ: [не обнаружено]	Античит: Обнаружено сокрытие!
---	--------------------------------------	---	---

Античит-отчет UACE:

- 1. Статус: Анализатор**Включен** Нормализатор **Включен** сходство символов установлено на **100%**
- 2. Обнаруженный процент загрязнения UniCode: **12,7%** с лимитом: 4%
- 3. Процент нераспознанных символов после нормализации: **7,3%**
- 4. Все подозрительные символы будут отмечены фиолетовым цветом: **Abcd...**
- 5. Найдены невидимые символы: 0

Рекомендации по оценке:

Особое внимание следует уделить анализу этого отчета! Предполагается, что этот документ содержит значительное количество символов, чуждых языку документа. Это прямое указание на то, что автор документа использовал специальное программное обеспечение\онлайн-веб-сервис, чтобы эффективно скрыть текст в попытке избежать обнаружения потенциального плагиата. Настоятельно рекомендуется передать это дело на более высокий уровень! В случае сомнений обращайтесь в службу поддержки Детектора плагиата!

Алфавитная статистика и анализ символов:

Активные ссылки (URL-адреса, извлеченные из документа):

URL не найдены

Исключённые ресурсы:

URL не найдены

Включённые ресурсы:

URL не найдены

Детальный анализ документа:

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ДЕРЖАВНИЙ ЗАКЛАД	id: 1
Цитування: 0,07%	
„ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА”	
Навчально-науковий інститут математики та інформаційних технологій Кафедра математики та інформатики Дворніков Дмитро Юрійович РОЗРОБКА ВЕБОРІЄНТОВАНОЇ СИСТЕМИ ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ АВТОСАЛОНА	
Обнаружен Плагиат: 0,1% https://er.knugd.edu.ua/bitstream/123456789/29...	id: 2
кваліфікаційна робота здобувача вищої освіти першого (бакалаврського) рівня освітньої програми	
Цитування: 0,06%	id: 3
«Комп’ютерні науки та інформаційні технології»	
за спеціальністю 122 Комп’ютерні наук Особистий підпис _____ Дмитро ДВОРНИКОВ Науковий керівник _____ Юрій КОЗУБ, д.т.н., професор Завідувач кафедри _____ Юрій КОЗУБ, д.т.н., професор Полтава – 2025 Міністерство освіти і науки України Державний заклад	
Цитування: 0,07%	id: 4
„Луганський національний університет імені Тараса Шевченка”	
Інститут математики та інформаційних технологій Кафедра Математики та інформатики Освітній ступень Бакалавр Напрямок підготовки (спеціальність) Галузь знань 122	
Цитування: 0,02%	id: 5
«Комп’ютерні науки»	
(код, назва) 12	
Цитування: 0,02%	id: 6
«Інформаційні технології»	
(код, назва) ЗАТВЕРДЖУЮ Завідувач кафедри М. Ю. Г. Козуб (підпис) (ініціали, прізвище)	
Цитування: 0,01%	id: 7
“ ”	
_____ 2025	
Обнаружен Плагиат: 0,5% https://essuir.sumdu.edu.ua/bitstream/1234567... + 3	id: 8
р. ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ Дворнікова Дмитра Юрійовича (прізвище, ім’я, по батькові) 1. Тема Розробка веборієнтованої інформаційної системи для підтримки діяльності автосалона Керівник кваліфікаційної роботи Козуб Ю.Г (прізвище, ім’я, по батькові, науковий ступінь, вчене звання) затверджена наказом по університету ві	
Д	
Цитування: 0,01%	id: 9
“ ”	
Обнаружен Плагиат: 0,16% https://essuir.sumdu.edu.ua/bitstream/1234567... + 3	id: 10
ресурс 2025 року № 2. Строк подання студентом проекту (роботи) 3. Вихідні дані до роботи (проекту) у результаті виконання роботи повинна бути детально описана і розроблена веборієнтована система для підтримки діяльності автосалона (визначаються кількісні або (та) якісні показники, яким повинен відповідати об’єкт розробки) 4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Розробка технічного завдання на створення веборієнтованої системи для підтримки діяльності автосалона Вибір інструментів для написання майбутньої програми Розробка інформаційного забезпечення веб-орієнтованої системи Розробка інформаційного забезпечення веб-орієнтованої системи (визначаються назви розділів або (та) перелік питань, які повинні увійти до тексту ПЗ) 5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень) 6. Консультанти розділів проекту (роботи) Розділ Прізвище, ініціали та посада Консультанта Підпис, дата завдання видав завдання прийняв 7. Дата видачі завдання	
Цитування: 0,01%	id: 11
“ ”	
2025 р. КАЛЕНДАРНИЙ ПЛАН № з/п Назва етапів дипломного проекту (роботи) Строк виконання етапів проекту (роботи) Примітка 1 Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника. до 1 лютого 2 Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно-експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень. до 20 березня 3 Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником. до 1 квітня 4 Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання. до 15 квітня 5 Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи. до 30 квітня 6 Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації. до 15 травня 7 Попередній захист роботи на кафедрі До 30 травня 8 Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії За 10 днів до державної атестації 9 Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом. За 5 днів до державної атестації Здобувач вищої освіти Дмитро ДВОРНИКОВ підпис (ініціали, прізвище) Керівник проекту	

(роботи) Юрій КОЗУБ підпис Анотація Дворніков Д.Ю. Тема: Розробка веборієнтовної системи для підтримки діяльності автосалона. Спеціальність: 122

Цитування: 0,02%

id: 12

„Комп’ютерні науки”

Установа: ДЗ ЛНУ імені Т.Шевченка, 2025р. Кваліфікаційна робота містить: 71 с., 36 рис., 4 табл., 3 додат., 32 джерела. Об’єкт дослідження – веб-орієнтована система. Предмет дослідження – технологія розробки клієнт-серверних **CRUD** систем. Мета роботи – розробка інформаційної системи для підтримки діяльності автосалону, що дозволяє здійснювати заходи з підбору та продажу автомобілів обраних марок та моделей. Результат роботи. Продукт, що складається із сайту з сучасним дизайном та інтуїтивно-зрозумілим інтерфейсом, що дозволяє менеджерів ефективно співпрацювати з клієнтами, бо, по-перше, містить доволі великий каталог автомобілів, а по-друге, може безпосередньо вносити деякі зміни у поєднану реляційну базу даних через відповідну клієнт-серверну частину, функціонал якої може відрізнитися в залежності від прав того чи іншого користувача. Висновки. Інформаційна система складається з багатосторінкового сайту (**HTML** + **CSS**), клієнтської та серверної частин, що написані на **JavaScript**, а також реляційної бази даних на мові **SQL**. Ключові слова: сайт, інформаційна система, реляційна база даних, **JavaScript**, **SQL**. **ABSTRACT Dvornikov, D. Theme: Development of a web-based system to support the activities of an autohouse. Specialty: 122**

Цитування: 0,02%

id: 13

"Computer Science"

Institution: Taras Shevchenko Luhansk National University, 2025. Qualification work contains: 71 pages, 36 figures, 4 tables, 3 appendices, 32 sources. Object of research: web-oriented system. Subject of research: technology of development of client-server CRUD systems. Purpose of the study: development of information system to support the activities of an autohouse, which allows to carry out activities on selection and sale of cars of selected brands and models. Result of research. The product consists of a site with a modern design and an intuitive interface that allows the manager to effectively collaborate with clients, because, firstly, it contains a fairly large catalog of cars, and secondly, it can directly make some changes to the combined relational database through the corresponding client-server part, the functionality of which may differ depending on the rights of a particular user. Conclusions. The information system consists of a multi-page site (HTML+CSS), client and server parts written in JavaScript, as well as a relational database in the SQL language. Keywords: site, information system, relational database, JavaScript, SQL. ЗМІСТ Анотація5 Вступ9 1 Розробка технічного завдання на створення веб-орієнтованої системи підтримки діяльності автосалону10 1.1 Обґрунтування актуальності використання веб-орієнтованих інформаційних систем10 1.2 Огляд існуючих аналогів та формування концепції для вирішення поставленого завдання.11 Висновок по розділу 114 2 Вибір інструментів для написання майбутньої програми15 2.1 Особливості веб-технологій для розробки інформаційних систем15 2.2 Існуючі рішення у веб-технології та їх призначення15 2.3 Вибір інструменту для розробки веб-орієнтованої системи автосалону20 2.3.1 **React.js** та його основні складові24 2.3.2 Що таке **Node.js** і як вона влаштована27 2.4 Вибір типу СУБД29 Висновок по розділу 231 3 Розробка інформаційного забезпечення веб-орієнтованої системи33 3.1 Складання концептуальної схеми предметної галузі33 3.2 Побудова даталогічної схеми бази даних36 3.3 Характеристики різних СУБД **SQL**.39 Висновок по розділу 345 4 Розробка програмного забезпечення програмного комплексу46 4.1 Підготовка до реалізації, налаштування необхідних утиліт та компонентів46 4.2 Завантаження бази даних

Цитування: 0,01%

id: 14

«Автосалон»


у **MySQL**47 4.1 Проектування та розробка сайту49 4.2 Створення структури та опис **React**-компонентів панелі адміністратора56 4.3 Забезпечення зв'язку між компонентами шляхом використання стандартної бібліотеки маршрутизації **React - React Router**58 4.4 Конфігурування складових компонентів59 Висновок по розділу 463 5 Висновок64 6 Список використаних джерел65 Додаток А68 Додаток Б69 Додаток В71 Вступ Підтримка діяльності автосалону – це комплексне рішення, яке дозволяє упорядкувати наявні модельні пропозиції та надати зазначити їх характеристики, забезпечити зворотній зв'язок від клієнта, запропонувавши йому ряд послуг та проконтролювати їх виконання – розв'язанням поставлених проблем може виступити спеціально розроблена веб-орієнтована система. Зазначена система повинна спиратися на перевірені програмні рішення задля зручності та надійності операцій з даними. Для цього підійде класична клієнт-серверна архітектура, що поєднана з базою даних. Тому комплексному визначенню завдань щодо побудови необхідної веб-орієнтованої системи підтримки діяльності автосалону через співставний аналіз наданих нам вимог та ознайомлення з вже існуючими системами задля вибору найбільш привабливого для нас варіанту, а також пошуку та вибору необхідного інструментів та визначення з типом бази даних буде й присвячений даний звіт, який, у свою чергу, буде прологом до реалізації вже безпосередньо дипломної роботи. 1 Розробка технічного завдання на створення веб-орієнтованої системи підтримки діяльності автосалону 1.1. Обґрунтування актуальності використання веб-орієнтованих інформаційних систем На сьогоднішній день важко уявити наше життя без усіляких програмних додатків, інтернет-магазинів, електронних довідників та ін. Обробка великих об'ємів інформації, її зберігання чи корегування, процес формування звітності чи сам процес життєдіяльності будь-якої організації неможливо уявити без роботи з базами даних та поєднаною з нею клієнт-серверною архітектурою. Для цих цілей – **CRUD**-операцій та зручної взаємодії менеджера та клієнта, структурування даних та їх зрозумілого відображення – ідеально підходить веб-орієнтована система[13]. Однак, виникає логічне питання про те, яку технологію використовувати, щоб втілити в життя необхідний функціонал та задовольнити всі поставлені вимоги. Тут доцільно розглянути можливість створення веб-орієнтованої програми[4]. Його відмінні риси видно в наступному: Кросплатформність. Як клієнт виступає веб-браузер, який входить до складу будь-якої

операційної системи. Оновлення та супровід браузера лежить на його розробнику. Мобільність. Працювати із системою ви можете, перебуваючи у будь-якому місці, де є Інтернет та з будь-якого пристрою, в якому є інтернет-браузер. Таким чином, користувач не обмежений вимогами до апаратної частини. Низька сукупна вартість володіння. Вартість володіння веб-орієнтованою системою фактично укладена у розробці, підтримці та розвитку серверної частини (веб-сервера та сервера додатків) та володінні сервером бази даних системи. Рис. 1.1 – Загальна структура веб-програми. Значна кількість сучасних напрацювань та технологій присвячена Інтернету. Це можна легко довести. Озираючись довкола та проаналізувавши обстановку, в якій зараз живуть суспільства та держави, ми можемо побачити, що більший відсоток усіх банківських операцій, процес стягування податків та контроль інших сфер громадської діяльності, онлайн-продажі, а також підтримка розподілених транснаціональних компаній та здійснення документообігу дедалі тісніше переплітаються з веб-технологіями[8]. 1.2 Огляд існуючих аналогів та формування концепції для вирішення поставленого завдання. На сучасному ринку представлено доволі багато програмних рішень стосовно нашої проблематики. Тут є як просто рекламні сайти-візитки, так і цілі розгалужені системи, що підтримують безліч функцій новинового порталу, дошки оголошень, майданчика дистриб'юторів різноманітних автовиробників, професійних оглядів новинок тощо. Нам потрібне більш медіанне рішення, бо у надпотужній системі немає потреби, але й доволі примітивні лендінги теж не підходять[8]. Почнемо аналізувати один із найпопулярніших сайтів про автомобілі – [Infocar.ua](http://infocar.ua). Тут представлені всі авто, які коли-небудь продавалися чи виготовлялися на території України. Також доступний форум, де власники чи просто зацікавлені особи обмінюються думками, новини автосвіту, відео огляди авто, контакти автосалонів тощо. Рис. 1.2 – Загальний вигляд сайту infocar.ua. Але й інші типи сайтів на автомобільну тематику, наприклад, сайти з продажу авто, наприклад auto.ria. Цей сайт може бути цікавим, перш за все, як розміщується інформація про конкретний автомобіль, його комплектації тощо з метою подальшого продажу. Рис. 1.3 – Загальний вигляд сайту auto.ria. Також можна розглянути сайт будь-якого комерційного автодилера, наприклад [Hyundai](http://hyundai) [7; 24]. Рис. 1.4 – Загальний вигляд сайту olimpmotor.com.ua. Отже, у проміжному висновку, ми можемо зазначити, що окрім представлення автомобілів та опису їх технічних характеристик, необхідно ще й підтримувати зворотній зв'язок з клієнтом, забезпечити адміністрування як представлених пропозицій, так і збереження можливостей до додачі нових[8]. Висновок по розділу 1. Проаналізовані вже готові впровадження рішень з підтримки діяльності автосалону, виявлені необхідні компоненти та функціонал. 2. Вибір інструментів для написання майбутньої програми 2.1. Особливості веб-технологій для розробки інформаційних систем Web-

 Обнаружен Плагіат: **0,18%** https://lib.iitta.gov.ua/id/eprint/6472/1/БАЗОВИ_... id: **15**

технології – комплекс технічних, комунікаційних, програмних методів вирішення завдань організації спільної діяльності користувачів із застосуванням

Інтернету. Привабливість Web-технологій як засоби доставки інформації багато в чому визначає універсальний інтерфейс між людиною та комп'ютером. Кожній людині зрозумілі написи, заголовки, посилання, картини. Веб-інтерфейс як доступу до інформації інтуїтивно зрозумілий. Наслідком простоти веб-інтерфейсу є широке використання Інтернету як каналу комунікації. Браузер – програма для перегляду веб-сторінок та роботи з інформацією у веб-інтерфейсі. Браузери – програми, якими забезпечені всі майже всі сучасні електронні пристрої, так звані

 Цитування: **0,01%**

id: **16**

«гаджети».

Теоретично всі браузери повинні відображати всі сайти, зроблені за стандартами, однаково, хоча на практиці є багато особливостей. Найбільш популярні браузери: [Internet Explorer](#), [Firefox](#), [Opera](#), [Safari](#), [Chrome](#). Інструмент для написання сайту для підтримки діяльності автосалону повинен бути гнучким, швидко працювати і займати якнайменше ресурсів мережі та комп'ютера, а також мати можливість ефективно взаємодіяти із зовнішніми джерелами[8]. Існуючі рішення у веб-технології та їх призначення. Величезна кількість Інтернет-технологій в сучасному світі може змусити розгубитися навіть досвідченого розробника. Однак, існують найбільш використовувані та затребувані інструменти, а також способи їх застосування, які дозволяють у короткий термін і головне ефективно вирішувати поставлені завдання. Основою всесвітньої павутини є мова розмітки гіпертексту **HTML** – [Hyper Text Markup Language](#). Він служить для логічного (змістового) розмітки документа (веб-сторінки)[5: 18]. Найчастіше у веб-дизайнера виникає необхідність застосувати у процесі створення **html**-документа складне форматування – від абзацу до абзацу змінювати шрифт, розташування тексту, його колір, формувати різні таблиці даних[4]. Можна вирішити цю проблему за допомогою стандартних засобів **HTML**: описувати кожен абзац окремим набором команд, але можна піти іншим шляхом: включити до сторінки опис **CSS** або підключити зовнішній файл, виконаний у стандарті **CSS** – [Cascading Style Sheets](#) (каскадні таблиці стилів), в якому за допомогою спеціальної макромови один раз жорстко. Іншими словами, файл **CSS** виконує роль деякого шаблону, що використовується для форматування тексту, таблиць та інших елементів у документі **HTML**. Є можливість підключати той самий фізичний файл **CSS** до різних веб-сторінок сайту. **CSS** можна використовувати на будь-якому сервері без будь-яких обмежень, оскільки команди **CSS** виконуються безпосередньо на комп'ютері користувача. До недоліків цієї технології можна лише віднести відсутність підтримки **CSS** старими браузерами ([Internet Explorer](#) і [Nescape Navigator](#) нижче 4-ої версії) і трохи різний набір властивостей **CSS**, що підтримується останніми версіями цих двох браузерів[16: 17]. Для надання веб-сторінкам динамізму (випадаюче меню, анімація) використовуються мови написання скриптів. Стандартною скриптовою мовою у всесвітньому павутинні є [JavaScript](#). Ядром [JavaScript](#) є [ECMAScript](#). [JavaScript](#) – це мова програмування, яка використовується у складі сторінок **HTML** для збільшення можливостей. Він був розроблений фірмою [Netscape](#) на базі мови [Sun's Java](#) корпорації [Sun](#). [JavaScript](#) є ніби надбудовою стандарту **HTML** і значно розширює можливості **html**-документа, створеного з використанням цієї технології.

[JavaScript](#) інтегрується у файл [HTML](#) у вигляді кількох рядків коду (наприклад, це може бути функція, що викликається на виконання спеціальною командою). Вбудований у браузер інтерпретатор [JavaScript](#) сприймає і скрипт, і сам [HTML](#)-код як єдиний документ, обробляючи і ті, й інші дані одночасно[2] Модуль [Java](#) на відміну від [JavaScript](#) інтегрується в сторінку, що його використовує, тільки після завантаження і виконання самостійної програми (програми) з розширенням [.class](#), такі програми називаються аплетами. Аплет також викликається з [html](#)-файлу відповідною командою, але завантажується, ініціалізується та запускається на виконання у вигляді окремої програми, у фоновому режимі, а до виконання аплету на його місці ви можете лише споглядати сірий прямокутник. Підтримка цієї технології здійснюється за допомогою так званої

» Цитування: 0,04%

id: 17

"Віртуальної машини [Java](#)".

Аплети [Java](#) в основному використовувалися для надання інтерактивності та візуальної краси [web](#)-сторінкам. Але оскільки аплети завантажувалися досить повільно (через невеликого розміру [class](#) файлів) і після написання коду необхідно було створити безпосередньо аплет за допомогою спеціального компілятора, а також можливість створювати ці інтерактивні елементи з використанням того ж [JavaScript](#), а також [DHTML](#) і [CSS](#), зумовили досить рідкісне застосування технології [Java](#) у вигляді аплетів сьогодні. За допомогою технології [Java/JavaScript](#) можна надати своїй сторінці елементи інтерактивності, формувати, компоувати та повністю контролювати формат спливаючих вікон та вбудованих фреймів, організовувати такі активні елементи, як

» Цитування: 0,01%

id: 18

"годинник",

» Цитування: 0,04%

id: 19

"рядки, що біжать"

та іншу анімацію, створити чат. Більшість [web](#)-камер, що передають на сайт

» Цитування: 0,01%

id: 20

"живе"

зображення, також працюють на базі відповідних програм [Java](#). Використання цих технологій не потребує встановлення та налаштування на сервері будь-яких додаткових модулів, оскільки скрипти та аплети виконуються безпосередньо на комп'ютері користувача. Браузери старих версій ([Internet Explorer](#) і [Netscape Navigator](#) нижче з 4-ої версії), що не підтримують [Java](#) / [JavaScript](#), не зможуть правильно відображати веб-сторінки, створені за допомогою цих технологій. Але зараз таких браузерів лише близько 3-4%. [HTML](#), [CSS](#), [JavaScript](#) - є мовами, за допомогою яких можна створювати складні веб-сайти. Але це лише лінгвістичне забезпечення, тоді як у браузерах документи подаються у вигляді набору об'єктів, безліч типів яких є об'єктною моделлю браузера ([BOM](#)). Об'єктна модель браузера є унікальною для кожної моделі і таким чином виникають проблеми при створенні міжбраузерних додатків. Тому веб-консорціум запропонував об'єктну модель документа ([DOM](#)), яка є стандартним способом представлення веб-сторінок за допомогою набору об'єктів[4]. На відміну від об'єктної моделі браузера [DOM](#) містить набір об'єктів лише для вмісту документа і не має об'єктів, що дозволяють керувати вікнами та рамками вікон. При написанні програм з метою підтримки міжбраузерної переносимості необхідно дотримуватися стандартів [DOM](#), а об'єктної моделі браузера вдаватися лише за крайньої необхідності. Така потреба може виникнути, наприклад, при керуванні вікнами та рядком стану[4]. Сукупність [HTML](#), [CSS](#), [JavaScript](#) та [DOM](#) часто називають динамічним [HTML](#) - [Dynamic HTML](#) або [DHTML](#)[5: 18]. [DHTML](#) ([Dynamic Hyper Text Markup Language](#), динамічна мова розмітки гіпертексту) є розширенням стандарту [HTML](#) і дозволяє створювати [web](#)-сторінки, що включають такі інтерактивні елементи, як рухомий фон, розташований під статичним вмістом документа, рухомі об'єкти, випадають меню, кнопки, що підсвічуються при наведенні курсору. За великим рахунком [DHTML](#) є

» Цитування: 0,02%

id: 21

"середнім арифметичним"

між технологіями [HTML](#) та [JavaScript](#). Цей стандарт використовує прості сценарії, підготовлені за допомогою макромови, що інтерпретується, оброблюваного браузером спільно з кодом [HTML](#). Такі сценарії називаються

» Цитування: 0,01%

id: 22

"скриплетами".

Для створення скриплетів використовуються стандартні розширення [DHTML](#) та будь-яка макромова, що підтримує директиви інтерфейсу [ActiveX](#). [DHTML](#) розпізнається браузерами [Microsoft Internet Explorer](#), починаючи з версії 4.0 та вище. [PHP](#) ([Personal Home Page tools](#)) - це ще одна мова, що інтерпретується, призначена для надання [web](#)-сторінкам елементів інтерактивності. Код, написаний мовою [PHP](#), вбудовується в документ [HTML](#) подібно до підпрограми: в ту ділянку документа, де необхідно розмістити інтерактивний елемент, просто вставляється сценарій [PHP](#). Мнемоніка цієї мови базується на синтаксисі [PERL](#), [Java](#) і [C](#), завдяки чому не викликає будь-яких труднощів щодо. Методики, які дозволяють серверам коректно розпізнавати файли, що містять скрипти [PHP](#), різні і залежать передусім від типу конкретного сервера. Як правило, достатньо призначити такому файлу розширення [.php](#), іноді - з додаванням номера версії мови, наприклад [.php3](#) або [.php4](#). Технологія [PHP](#) дозволяє організовувати на [web](#)-сторінці лічильник відвідувань, підраховувати статистику звернень до тих чи інших розділів сайту, захистити доступ до будь-якого [html](#)-документа паролем та багато іншого. Серед недоліків [PHP](#) слід зазначити, що дана технологія підтримується далеко не всіма серверами Інтернету. Технологія [CGI](#) ([Common Gateway Interface](#)) передбачає використання у складі ресурсу Інтернету інтерактивних елементів з урахуванням додатків, щоб забезпечити передачу потоку даних від об'єкта до об'єкта. Саме так організовано у Всесвітній мережі більшість чатів, конференцій (форумів), дощок оголошень, гостьових книг, пошукових машин та

рейтингових систем. Спрощено принцип роботи [CGI](#) виглядає так: наприклад, користувач заповнює на [web](#)-сторінці ту чи іншу форму і натискає на кнопку, після чого інформація з форми передається в [CGI](#)-скрипт, який запускається на виконання та обробляє отриману інформацію. Серед переваг [CGI](#) слід відзначити їхню незалежність від клієнтського програмного забезпечення - цю технологію зможе застосовувати кожен користувач, який переглядає вміст сервера за допомогою браузера практично будь-якої версії. Рис. 2.1 - Статистика [TIOBE](#) щодо використання веб-технологій Як видно з вищесказаного, всі веб-технології тісно взаємопов'язані. Розуміння цього факту дозволить легше усвідомити призначення того чи іншого механізму, який використовується під час створення веб-додатків. Технології розвиваються та удосконалюються з кожним роком[1]. 2.3 Вибір інструменту для розробки веб-орієнтованої системи автосалону Зважаючи на статистику застосування різних веб-технологій у розробці та, враховуючи їх призначення, а також сильні та слабкі сторони, було прийнято рішення подивитися у бік [JavaScript](#). Однак, відомі ресурси, такі як (вставити щось), сучасні тенденції і власні напрацювання можна прийти до висновку про недоцільність використання класичного [JavaScript](#), а подивитися в бік фреймворків і бібліотек, створених на його основі, так як вони мають більш просунутий і в той же час полегшений функціонал і менше граничний завдань. Насамперед розглянемо особливості проектування з допомогою фреймворків, як фронтенду так і бекенду, з'ясуємо, у чому полягає основна різниця між ними та введемо основні принципи, які у кожному з цих інструментів проектування[2]. Фреймворки надають чітку структуру докладання та реалізуються з використанням так званих

Цитування: 0,02%

id: 23

«патернів проектування».

Найбільш широко поширені такі патерни: [MVC \(Model-View-Controller\)](#), [MVP \(Model-View-Presenter\)](#) та [MVVM \(Model-View-ViewModel\)](#). Переваги побудови програми на [JS](#)-фреймворку: дозволяє легко реалізувати [SPA \(Single Page Application\)](#); підтримує структуру програми точний та добре структурований код; модульність програми, можливість одночасного використання кількох способами; можливість швидко створити мобільний та/або настільний кросплатформовий додаток з веб-версії за допомогою систем типу [PhoneGap](#) або [Apache Cordova](#). [Frontend](#) — це те, що відображається, а також взаємодіє з користувачем у частині програми. Головними завданнями фронтенду є створення інтерфейсу, забезпечення взаємодії користувача, створення дизайну, забезпечення доступності програм різних пристроїв. Програмісти, що працюють у фронтенді, використовують мови програмування плюс засоби верстки сторінок — [HTML](#), [CSS](#), [JavaScript](#). Для прискорення процесу написання коду використовуються фреймворки [React](#), [Angular](#), [Vue.js](#), також багато інших[1: 27]. [React](#) (більше 110 тисяч зірок на [GitHub](#)) - це ефективна та гнучка декларативна бібліотека [JavaScript](#) для складання [UI](#) від команди [Facebook](#). Вона дозволяє без зусиль створювати інтерактивний інтерфейс користувача. [Vue.js](#) - це один фреймворк для збирання інтерфейсів користувача. Він включає доступну кореневу бібліотеку, яка в першу чергу вирішує завдання рівня представлення, та екосистему додаткових бібліотек, що дозволяє створювати складні та об'ємні односторінкові програми ([Single-Page Applications](#)). [Angular](#) - фреймворк від компанії [Google](#), який отримав майже 44 тисячі зірок на [GitHub](#). Являє собою платформу, що спрощує складання додатків у Інтернет. У [Angular](#) поєднуються декларативні шаблони, впровадження залежності, двостороннє зв'язування даних та найкращі практики вирішення проблем розробки. Ця платформа дозволяє збирати програми для веб, мобільних пристроїв та настільних ПК. Серверна частина, тобто [backend](#), є невід'ємною частиною візуального оформлення, але працює на задньому плані у фоновому режимі. Завданням цієї служби є обробка, зберігання запитів, а також забезпечення безпеки даних, виконання бізнес-логіки програми[1; 2; 27; 6]. При створенні [backend](#)-програм використовуються мови програмування [Python](#), [Ruby](#), [Java](#), [PHP](#), [.NET](#) та [JavaScript](#) фреймворки, такі як [node.js](#). Також використовуються серверні технології, мережеві бази даних, віддалені сховища. Таблиця 2.1 Фреймворки для [backend](#)-програм.

Технологія Тип Основне використання Фреймворки Плюси Мінуси [Python](#) Мова Веб, наука про дані, автоматизація, [ML Django](#), [Flask](#) Легка, велика спільнота, популярна в [AI/ML](#) Повільніша за [Java](#). [.NET Ruby](#) Мова Веб-розробка [Ruby on Rails](#) Простий синтаксис, швидка розробка Менша популярність, не дуже гнучкий для інших задач [Java](#) Мова Ентерпрайз, [Android](#), сервери [Spring](#), [Hibernate](#) Надійна, масштабована, стабільна Об'ємний код, складніша в навчанні [PHP](#) Мова Веб-розробка [Laravel](#), [Symfony](#) Простий старт, багато [CMS \(WordPress\)](#) Негативна репутація, застарілий синтаксис (у старих проектах) [.NET](#) Платформа Веб, десктоп, мобільні застосунки [ASP.NET Core](#), [Blazor](#) Висока продуктивність, підтримка [Microsoft](#), [C#](#) Залежність від екосистеми [Microsoft](#) (частково) [Node.js](#) Рантайм ([JS](#)) Серверний [JS](#), [API](#), реального часу застосунки [Express](#), [NestJS](#) Асинхронність, одна мова для фронту і беку [Callback hell](#) (якщо не використовувати [async/await](#)), трохи нижча продуктивність у [CPU](#)-важких задачах Програмісти [backend](#) фокусуються на серверній логіці, базах даних, архітектурі, управлінні [API](#), забезпечуючи стабільну роботу написаних програм. Взаємодія фронт/бек [Frontend](#) та [backend](#) взаємодіють за допомогою [API \(Application Programming Interface\)](#), які дозволяють передавати дані між клієнтом та сервером. Бекенд обробляє запити, що надходять від фронтенду, повертає необхідні дані, які потім відображаються користувачеві[2; 27; 6; 9]. Хорошим прикладом такої взаємодії може бути онлайн-магазин, де фронт відповідає за відображення товарів, інтерфейс кошика, а бек обробляє інформацію про товари, замовлення, виконує операції з базою даних. Різниця фронтенду та бекенду Відмінності у завданнях та відповідальності Інтерфейс і серверна частина є частинами одного процесу, що взаємодоповнюють, але існують відмінності в завданнях, у відповідальності фахівців. Автор інтерфейсу фокусується на тому, що бачить користувач: макет, дизайн, анімація. Його відповідальність полягає у створенні інтуїтивно зрозумілого, гарного інтерфейсу[27]. Серверна частина обробляє

Цитування: 0,01%

id: 24

"невидиму"

частину програми. Програміст відповідає за серверну логіку, базу даних, управління [API](#),

безпеку, виконання серверних скриптів. Він гарантує, що всі дані обробляються, зберігаються правильно, а програма працює стабільно, безпечно без збоїв. Приклади та кейс-стаді для наочності Прикладом, що ілюструє різницю між [backend](#) та [frontend](#), може бути процес створення сайту. Фронтенд-програміст створює дизайн сторінки, має в своєму розпорядженні елементи управління – кнопки, форми введення. Бекенд-програміст забезпечує коректну обробку даних, що вводяться користувачем у форми, а також щоб коректно працювали всі необхідні серверні взаємодії[27]. Рис. 2.2 –Статистика завантажень 3 фреймворків за даними [GitHub React.js](#) та його основні складові [React](#) — це інструмент для створення інтерфейсів користувача. Його головне завдання – забезпечення виведення на екран того, що можна побачити на веб-сторінках. [React](#) значно полегшує створення інтерфейсів завдяки розбиттю кожної сторінки на невеликі фрагменти компонентами. Кожен компонент – це [JavaScript](#)-функція, яка повертає ділянку коду, який представляє фрагмент сторінки. Здійснюється у певному порядку виклик компонентів та збираються разом отримані результати – так формується сторінка, яку бачить користувач. Для оптимізації швидкодії компоненти [React](#) спочатку перетворюються на керовану модель [Virtual DOM](#), а потім за допомогою спеціальних інструментів написаних на [JSX](#) код на зрозумілий браузеру [JavaScript](#)[1; 2; 9; 27]. Рис. 2.3 – Механізм роботибібліотеки [React.js](#)

Взагалі, щоб використовувати [React](#) необхідно знати та освоїти наступне: компоненти [React](#). Рендеринг [ReactDOM](#). Класи компонентів та функціональні компоненти. [JSX](#). Стан ([state](#)). Опрацювання подій. Асинхронний метод [setState](#). Властивості ([props](#)). Посилання ([refs](#)) [7]. Ось наочний приклад [React](#)-компонента. `function OurFirstComponent(){ return (<h1 Hello, I am a React Component! />); }` Для того, щоб відобразити (відрендерити) компонент на сторінці вдаємося до допомоги [ReactDOM](#). `const placeWeWantToPutComponent = document.getElementById('hook'); ReactDOM.render(OurFirstComponent(), placeWeWantToPutComponent);` Функція `getElementById('hook')` використовується для прив'язки рендеринг компонента до певної ділянки [html](#)-коду. Компоненти можуть бути інкапсульовані інші компоненти і т.д. Щоб викликати весь ланцюжок у [ReactDOM](#), потрібно вказати назву верхнього компонента. Такий підхід називається композицією. Однак компоненти можна писати і інакше, у вигляді класів [JavaScript](#) - класи компонентів. Продовжимо огляд бібліотеки, щоб зрозуміти всі механізми проектування та побудови додатків на [React](#), а також побачити всі можливості для їх здійснення. Стан Стан — це інструмент, що дозволяє оновлювати інтерфейс користувача, ґрунтуючись на подіях. Іншими словами – це спеціальний об'єкт усередині компонента. `class Container extends React.Component{ constructor(props) { super(props); this.state = {isMusicPlaying: false}; } Оновити (змінити) стан можна за допомогою єдиної функції this.setState(). Після її викликом слід перемальовування всього компонент, якого воно належить. Це одна з головних особливих бібліотек React. handleClick(){ if(this.state.isMusicPlaying){ this.setState({isMusicPlaying: false}); } else { this.setState({isMusicPlaying: true}); } } [15]` Властивості Під властивостями можна розуміти опції компонента. Вони надаються як аргументи компонента і виглядають так само, як атрибути [HTML](#). Події Вони також можуть служити каталізатором для оновлення стану і, як наслідок, перемальовки компонента. Посилання Якщо нам потрібно звернутися до певного тегу та викликати його методи. Звичайно, можна використовувати традиційний [JavaScript](#)-підхід - написати `document.getElementById(тег).метод()`. Хоча краще звернутися до функціоналу [React](#). Ми призначаємо елементу атрибут, який називається [ref](#), який приймає функцію. Ця функція, як перший аргумент, приймає елемент і надає його `this`. елемент . Далі до цієї конструкції через крапку пишеться ім'я функції, яку слід викликати[14]. Таким чином вдалося познайомитися з новою для нас [js](#)-бібліотекою і навіть зробити перші кроки в її освоєнні. Простота і гнучкість коду, легкість написання та підтримки зв'язків між компонентами, а також хороша поєднання з популярною [No-SQL](#) БД зробили [React](#) найкращим інструментом для вирішення поставленого завдання – але про вибір бази даних та його обґрунтування розповідають наступні розділи. Що таке [Node.js](#) і як вона влаштована [Node.js](#) - це середовище виконання [JavaScript](#), побудоване на движку [V8](#) від [Google](#), яке: Компілює та виконує [JavaScript](#)-код, перетворюючи його на машинний код, зрозумілий для процесора. Завдяки цьому [JavaScript](#) працює дуже швидко та ефективно. Не тільки використовує движок [V8](#), але й розширює його функціональність, додаючи різні модулі та [API](#) для роботи з файлами, мережею, базами даних та іншими ресурсами. Надає подієвий цикл ([event loop](#)) – механізм, який дозволяє обробляти асинхронні операції без блокування основного потоку виконання. Подієвий цикл реєструє функції-обробники ([callback](#)) для різних подій (наприклад, завершення читання файлу або отримання відповіді сервера) і викликає їх по черзі, коли події відбуваються. Переваги [node.js](#) перед [Python](#) та [PHP](#) Єдина мова: полегшує розробку та підтримку коду. [Fullstack](#)-розробнику не потрібно вивчати та перемикатися між різними мовами та середовищами розробки. [JavaScript](#)-фреймворки: пропонують інструменти та функції для обробки [HTTP](#)-запитів, маршрутизації, обробки даних та інших завдань, пов'язаних із [backend](#)-розробкою. Асинхронне програмування: дозволяє ефективно обробляти безліч запитів за допомогою асинхронної моделі виконання коду. Це особливо корисно для розробки високопродуктивних програм, таких як чати, стрімінгові сервіси, де потрібна одночасна обробка великої кількості запитів. Інтеграція з фронтендом: ви можете використовувати [JSON](#) для представлення даних та легко маніпулювати ними як на клієнтській, так і серверній стороні[6; 11; 23]. Як виглядає код на [Node.js](#) Код на [Node.js](#) виглядає так само, як код [JavaScript](#), за винятком того, що він використовує спеціальні модулі і [API](#) для роботи з сервером. Давайте створимо простий сервер: Рис. 2.4 -творення простого сервера на [node.js](#) Цей код створює сервер, який слухає порт 3000 та надсилає вітальне повідомлення кожному, хто звертається до нього. Щоб запустити цей код, потрібно зберегти його у файлі з розширенням `.js` (наприклад, `server.js`) і виконати команду `node server.js` у терміналі. Після цього можна відкрити браузер та перейти за адресою `http://localhost:3000`. Там має з'явитися повідомлення `Hello, Node.js!`. 2.4 Вибір типу СУБД На сьогоднішній день існують два найбільш популярні типи баз даних – реляційні та нереляційні. Давайте розберемося, у чому різниця. Основні відмінності ховаються в деяких характерних рисах: як вони спроектовані, які типи даних підтримують і як зберігають інформацію. Реляційні БД зберігають структуровані дані, які зазвичай становлять об'єкти реального

світу. Це можуть бути відомості про людину або вміст кошика для товарів у магазині, згруповані в таблицях, формат яких заданий на етапі проектування сховища. Нереляційні БД улаштовані інакше. Наприклад, документоорієнтовані бази зберігають інформацію як ієрархічних структур даних. Йдеться про об'єкти з довільним набором атрибутів. Те, що у реляційній БД буде розбито кілька взаємозалежних таблиць, в нереляційній може зберігатися як цілісної сутності. Реляційні бази даних складаються з кількох ключових компонентів: Таблиці: Основні структури для зберігання даних. Кожна таблиця складається з рядків і стовпців, де кожен рядок представляє окремий запис, а кожен стовпець — атрибут даних. Первинний ключ: Унікальний ідентифікатор для кожного запису в таблиці. Він гарантує, що кожен запис може бути однозначно ідентифікований. Зовнішній ключ: Поле в одній таблиці, яке посилається на первинний ключ іншої таблиці, створюючи зв'язок між таблицями. Індеси: Структури, що підвищують швидкість доступу до даних у таблиці. Індеси дозволяють швидко знаходити рядки за певними критеріями. Запити: Використовуються для отримання даних з таблиць. Найпоширенішою мовою запитів для реляційних баз даних є [SQL \(Structured Query Language\)](#)[31; 32]. Рис. 2.5 –творення простого сервера на [node.js](#) Таблиця 2.2 Порівняння реляційних та нереляційних баз даних Критерій Реляційні БД (SQL) Нереляційні БД (NoSQL) 1. Структура зберігання Таблиці з рядками і стовпцями Документи (JSON, BSON), графи, колоночні сховища, ключ-значення 2. Схема Жорстка, чітко визначена. Схема має бути визначена перед зберіганням даних Гнучка, динамічна. Схема може змінюватися

» Цитування: 0,02%

id: 25

"на ходу"

3. Типи даних Переважно структуровані дані Підтримка структурованих, напівструктурованих і неструктурованих даних 4. Запитна мова [SQL \(Structured Query Language\)](#) Залежить від СУБД: [MongoDB Query Language](#), [CQL \(Cassandra\)](#), власні [API](#) 5. Підтримка транзакцій Повна ([ACID](#): атомарність, узгодженість, ізоляція, надійність) Часткова ([BASE: Basically Available, Soft state, Eventually consistent](#)) 6. Зв'язки між даними Використання зовнішніх ключів, [JOIN](#)-операцій Найчастіше — денормалізація, зв'язки описуються вручну або через вкладені об'єкти 7. Масштабування Вертикальне (додавання потужності одному серверу) Горизонтальне (додавання нових серверів або вузлів) 8. Продуктивність Висока при складних запитах і транзакціях Висока при великих обсягах даних, реальному часі, високій паралельності 9. Надійність при збоях Висока. Зазвичай має реплікацію та резервне копіювання Також висока, але залежить від конкретної реалізації 10. Гнучкість зміни структури Низька: потребує [ALTER TABLE](#) та ретельного перепроєктування Висока: зміна структури на рівні одного документа чи колекції 11. Швидкість розробки Повільніша через необхідність визначення схеми Швидша, особливо на ранніх етапах 12. Підтримка великих обсягів Обмежено (масштабування ускладнене) Добре працює з [Big Data](#), великими навантаженнями 13. Приклади СУБД [MySQL](#), [PostgreSQL](#), [Oracle](#), [Microsoft SQL Server](#), [MariaDB](#), [MongoDB](#), [Cassandra](#), [Redis](#), [Couchbase](#), [Neo4j](#), [DynamoDB](#) 14. Кращі випадки використання Банки, фінанси, [CRM](#), [ERP](#), транзакційні системи Соцмережі, [IoT](#), [Big Data](#), веб-програми, системи реального часу 15. Складність підтримки Висока (потрібні [DBA](#), оптимізація запитів) Залежить від типу, але зазвичай простіше для початкового адміністрування 16. Реплікація та шардінг Реплікація можлива, шардінг складний Часто вбудована підтримка шардінгу і реплікації 17. Безпека Високий рівень контролю доступу, аудит Варіюється — може бути менш контрольований, але часто має вбудовані механізми 18. Зрілість технології Дуже зріла, багаторічне використання в індустрії Відносно нові, але активно розвиваються й набирають популярності Висновок по розділу 2 Було складено концептуальну модель предметної області, проведено її опис. Визначено види міжтабличних зв'язків. Розглянуто основні алгоритми з проектування даталогічної схеми та методи щодо її практичної реалізації. Проведено характеристику різних реляційних СУБД, обґрунтовано вибір [MySQL](#), завантажена спроектована база даних 3.Розробка інформаційного забезпечення веб-орієнтованої системи 3.1 Складання концептуальної схеми предметної галузі Виходячи з вже отриманих результатів та проведення оцінки потреб докладання в ресурсах, приступимо до побудови моделі БД на основі аналізу її основних властивостей та компонентів. Рис. 3.1 – Інфологічна схема бази даних, що розробляється для проекту Зважаючи на велику кількість поставлених завдань, база даних сайту

» Цитування: 0,01%

id: 26

«Автосалон»

вийшла досить об'ємною. Після ретельного аналізу даних та тестування чітко позначалися 16 таблиць бази. Вони мають такі назви і містять певні поля: Передня підвіска (код п.п., назва); Тип кузова (код т.к., найменування); Марка (код марки, назва, логотип); Автомобіля (код авто, довжина, ширина, висота, маса, вартість, кількість дверей, кількість місць для сидіння, об'єм багажника, ємність паливного бака, максимальна швидкість, час розгону до 100 км/год, витрата палива, клімат-контроль, розмір шин, розмір дисків) ; Країна (код країни, назва); Привід (код приводу, тип приводу); Електронні асистенти (код ел. асистенту, назва); Двигун(код двигуна, тип, кількість циліндрів, об'єм двигуна, потужність, момент оберт); Коробка передач (код к.п, назва); Модель (код моделі, назва моделі, зображення, модельний рік, ціна) Зображення (код зображення., посилання); Підсилювач керма (код підсилювача керма, тип); Тест-драйв(код тест-драйву); Продажі(код продажі, категорія); Покупець(код покупця, ПІБ); Менеджер(код менеджера, ПІБ). Беручи за основу теоретичні висновки, зроблені раніше, вдалося приступити до побудови інфологічної схеми бази даних

» Цитування: 0,01%

id: 27

«Автосалон».

Оскільки більшість таблиць є довідниками, то зв'язки в основному будуть представлені відношенням 1 : М, наприклад Рисунок 3.2 – приклад відношення 1:М Відношення М:М також є, між таблицями електронні помічники і властивості автомобіля. Рис. 3.3 – приклад відношення М:М Атрибути здебільшого прості, статичні та динамічні, зв'язки мають

обов'язковий клас власності[8; 10; 25]. Ось підсумкова таблиця зв'язків бази даних:
Таблиця 3.1 Зв'язок між таблицями бази даних

Цитування: 0,01%	id: 28
«Автосалон»	
Таблиця 1 Таблиця 2 Вид зв'язку Підвіска Автомобіль 1: М Тип кузова Автомобіль 1: М Марка Модель 1: М Країна Модель 1:М Привід Автомобіль 1: М Електронні асистенти Автомобіль М: М Двигун Автомобіль 1: М Коробка передач Автомобіль 1: М Зображення Модель М:1 Підсилювач керма Автомобіль 1: М Модель Автомобіль 1: М Автомобіль Тест-драйв 1:М Менеджер Продажі 1:М Покупець Продажі 1:М Покупець Тест-драйв 1:М Менеджер Тест-драйв 1:М 3.2 Побудова даталогічної схеми бази даних Згідно з побудованою інфологічною моделлю ми отримали 18 таблиць, у тому числі зі зв'язком М:М. Тому необхідно додавання до бази ще однієї таблиці, яка міститиме первинні ключі обох таблиць, тобто замінимо цей зв'язок на 2 зв'язку 1:М[8; 10; 25]. Рис. 3.4 – приклад відношення М:М Якщо дві таблиці пов'язані ставленням 1:М, то первинний ключ таблиці 1 поміщається в таблицю М як зовнішній. <code>suspender(id_suspender, kind_of_suspender) carcass_type(id_carcass, type_of_carcass) car_brend(id_brend, brend_name, logo) car(car_code, id_carcass, id_model, car_length, car_width, car_height, weight, count_of_doors, count_of_seats, boot_volume, id_drunit, id_engine, fuel_tank_capacity, max_speed, acceleration_time, fuel_consumption, id_gearbox, id_suspender, id_power_steering, climate_control, wheel_size, disc_size); chassis_suspender(id_chassis_suspender, kind_of_cs_suspender) country(id_country, c_name) drive_unit(id_drunit, dr_unit) electronic_control_systems(id_assist, kind_of_elassist) engine(id_engine, type_of_engine, count_of_cylinders, engine_power, engine_capacity, engine_moment) gearbox(id_gearbox, type_of_gearbox) model(id_model, model_name, id_brend, main_view_link, id_country, model_year, price) pictures(id_picture, link, id_model) power_steering(id_power_steering, kind_of_powsteering) select_el_systems(id_list_of_assist, id_assist, car_code) test_drive(test_drive_code, date, time, id_manager, id_customer, id_car); sales(sale_code, date, id_manager, id_customer, car_code); manager(id_manager, name); customer(id_customer, name).</code> Розглянемо загальний вигляд таблиць даталогічної моделі беручи за основу таблицю <code>model</code> . Таблиця 3.2 Таблиця <code>model</code> бази даних	

Цитування: 0,01%	id: 29
"Автосалон"	
<p><code>Name Type Len Dec Screen Description id_model smallint</code> 4 -- Первинний ключ. <code>model_name varchar</code> 20 - <code>InpLine</code> Не пусте <code>id_brend smallint</code> 4 - <code>DDLБ</code> Зовнішній ключ <code>main_view_link varchar</code> 150 - <code>InpLine</code> Не пусте <code>id_country tinyint</code> 4 - <code>DDLБ</code> Зовнішній ключ <code>model_year date</code> Не пусте <code>price int</code> 10 - <code>InpLine</code> Не пусте Рис. 3.5 – Даталогічна схема БД 3.3 Характеристики різних СУБД <code>SQL</code>. Відомі</p>	

Обнаружен Плагіат: **0,17%** <https://www.dreamhost.com/blog/uk/postgresq...> + 2 id: **30**
ресурсів

системи управління реляційними базами даних Існує кілька популярних систем управління реляційними базами даних (`RDBMS`)

), кожна з яких має свої особливості та переваги: 1. `MySQL` Загальний опис: Реляційна СУБД з відкритим кодом. Розроблена спочатку компанією `MySQL AB`, зараз належить `Oracle`. Часто використовується в `LAMP`-стеку (`Linux`, `Apache`, `MySQL`, `PHP/Python`). Переваги: Широко підтримується хостингами. Добре підходить для веб-застосунків. Швидка обробка `SELECT`-запитів. Простота у використанні та налаштуванні. Підтримує реплікацію та кластеризацію. Недоліки: Обмежена підтримка складної аналітики порівняно з деякими іншими СУБД. Тривалий час `Oracle` тримає деякі можливості у платній версії[12; 28; 30]. Рис. 3.6 – Приклад інтерфейсу `MySQL` 2. `PostgreSQL` Загальний опис: Реляційна СУБД з відкритим кодом, орієнтована на відповідність стандартам `SQL`. Відомий як

Цитування: 0,05%	id: 31
"найбільш просунутий об'єктно-реляційний СУБД".	
<p>Переваги: Підтримка складних запитів, <code>CTE</code>, <code>window</code>-функцій. Повноцінна транзакційність (<code>ACID</code>). Підтримка <code>JSON</code>, <code>XML</code>, розширень (наприклад, <code>PostGIS</code> для геоданих). Вища відповідність стандартам <code>SQL</code>. Недоліки: Дещо складніше у налаштуванні для новачків. Може бути менш швидким для простих <code>SELECT</code>-запитів порівняно з <code>MySQL</code>. Коли краще <code>PostgreSQL</code>? Якщо важлива складна аналітика, розширення, відповідність стандартам. Системи з великими транзакціями, або потребою у зберіганні напівструктурованих даних (<code>JSON</code>) [28; 29] Рис. 3.7 – Приклад інтерфейсу <code>PostgreSQL</code> 3. <code>MariaDB</code> Загальний опис: Форк <code>MySQL</code>, створений творцями оригінального <code>MySQL</code> після покупки <code>Oracle</code>. Повністю сумісна з <code>MySQL</code>, але з відкритим управлінням спільнотою. Переваги: Більш відкритий розвиток (<code>community-driven</code>). Часто швидше впроваджує нові функції. Підтримка альтернативних движків зберігання (<code>Aria</code>, <code>ColumnStore</code>). Повністю сумісна з <code>MySQL</code> (у більшості випадків). Недоліки: Деякі зміни можуть порушити повну сумісність з <code>MySQL</code> у довгостроковій перспективі. Менше поширена, ніж <code>MySQL</code>. Коли краще <code>MariaDB</code>? Коли потрібна максимальна сумісність з <code>MySQL</code>, але без контролю <code>Oracle</code>. Коли важлива відкритість розробки. Рис. 3.8 – Приклад інтерфейсу <code>MariaDB</code> 4. <code>SQLite</code> Загальний опис: Надлегка реляційна СУБД, яка зберігає всю базу в одному файлі. Не вимагає сервера, дуже проста у використанні. Переваги: Ідеально для мобільних застосунків, вбудованих систем. Не потребує інсталяції або адміністрування сервера. Дуже швидка для невеликих обсягів даних. Недоліки: Не підходить для багатокористувацьких середовищ або великих систем. Обмежена функціональність у порівнянні з серверними СУБД. Коли краще <code>SQLite</code>? Для локальних застосунків, мобільних апікацій, вбудованих систем. Коли потрібна проста, файлова БД без серверної частини. Рис. 3.9 – Приклад інтерфейсу <code>SQLite</code> 5. <code>Microsoft SQL Server</code> Загальний опис: Комерційна СУБД від <code>Microsoft</code>. Добре інтегрується з <code>Windows</code>-інфраструктурою та продуктами <code>Microsoft</code>. Переваги: Потужні аналітичні можливості. Хороша інтеграція з <code>Power BI</code>, <code>.NET</code>, <code>Excel</code>. Зручне адміністрування через <code>SQL Server Management Studio</code>. Недоліки: Платна (є безкоштовна версія <code>Express</code> з обмеженнями). Переважно для <code>Windows</code>-середовищ. Коли краще <code>SQL Server</code>? Для підприємств, які працюють в екосистемі <code>Microsoft</code>. Коли потрібна глибока інтеграція з бізнес-аналітикою (<code>BI</code>).</p>	

[30] Рис. 3.10 – Приклад інтерфейсу [Microsoft SQL Server 6. Oracle Database](#) Загальний опис: Потужна комерційна СУБД, що використовується у великих підприємствах. Висока масштабованість, підтримка складних транзакцій. Переваги: Надійність, масштабованість, потужна реплікація. Гнучкість у проектуванні та адмініструванні. Недоліки: Дуже дорога. Складне адміністрування, високий поріг входу. Коли краще [Oracle](#)? У великих корпоративних системах з великим обсягом транзакцій. У системах з високими вимогами до надійності та безпеки. Рис. 3.11 – Приклад інтерфейсу [Oracle Database](#) Зупинимо свій вибір на [MySQL](#), оскільки це доволі зрозуміла та розповсюджена СУБД з відкритим кодом, до якої є багато документації, як офіційної, так і користувацької, до того ж вона сумісна з переважною більшістю технологій розробки програмних продуктів. Висновок по розділу 3 Було складено концептуальну модель предметної області, проведено її опис. Визначено види міжтабличних зв'язків. Розглянуто основні алгоритми з проектування даталогічної схеми та методи щодо її практичної реалізації. Проведено характеристику різних реляційних СУБД, обгрунтовано вибір [MySQL](#), завантажена спроектована база даних 4.2 Розробка програмного забезпечення програмного комплексу 4.1 Підготовка до реалізації, налаштування необхідних утиліт та компонентів Зараз нам слід розглянути, як встановити та налаштувати головний інструмент для реалізації проекту – бібліотеку [React.js](#), оскільки набагато зручніше працювати з ним локально[9; 26]. Для початку потрібно перевірити, чи є на робочій машині [node.js](#) командою [node -v](#). оновлення різних компонентів програмного забезпечення Системи управління пакетами активно використовуються в різних дистрибутивах операційної системи [Linux](#) та інших [Unix](#)-подібних операційних системах. Іншими словами – це інструмент [cmd](#), що дозволяє взаємодіяти з браузером, встановлювати або видаляти компоненти проектів зрозумілий браузеру [JavaScript](#). Зробити це можна за допомогою утиліти [create-react-app](#) - вона створить характерну структуру папок і файлів, скачає всі необхідні утиліти і налаштує їх таким чином, що можна буде відразу приступати до написання коду. встановлену утиліту, додавши в кінці назву програми. Після завершення процесу створення програми слід перевірити його працездатність[9; 14; 26]. Рис. 4.1 – Стартове вікно [react](#)-програми Таким же чином налаштуємо й фреймворк [node.js](#). Усе працює, отже можна його можна відкривати в [IDE](#) і розпочинати розробку власного програмного продукту. 4.2 Завантаження бази даних

» Цитування: 0,01% id: 32

«Автосалон»

у [MySQL](#) Відкриємо [MySQL Workbench](#) та натиснемо

» Цитування: 0,05% id: 33

«Create a new schema»

та додамо наш файл

» Цитування: 0,02% id: 34

«[autohouse.sql](#)».

Він має наступний вигляд: Рис. 4.2 – Фрагмент коду з файлу

» Цитування: 0,02% id: 35

«[autohouse.sql](#)»

У результаті отримаємо наступне[10; 12; 21; 22; 25]: Рис. 4.3 Завантажена база

» Цитування: 0,02% id: 36

«[autohouse.sql](#)»

4.3 Проектування та розробка сайту Розробка веб-орієнтованої системи з підтримки діяльності автосалону починається з написання сайту. Сайт повинен мати зрозумілий дизайн зі зручним інтерфейсом, бути адаптованим під різні розміри екрану й, звісно, нести всю необхідну інформацію про компанію та асортимент її товару – автомобілі. Отже, знайшовши у вільному доступі [figma](#)-макет, почали написання сайту, що являє собою один із найважливіших компонентів в системі, що розробляється, поряд із БД та панеллю адміністратора. Рис. 4.4 – Фрагмент макету сайту у [figma](#) Для кожної [html](#)-сторінки спільними є дві частини, [header](#) та [footer](#), перша містить у собі логотип, навігацію по сайту та деяку контактну інформацію, як-от посилання на сторінки в соціальних мережах, друга, у свою чергу, повні контактні дані, час роботи, адресу розташування з міткою на [google](#)-мапі, [footer](#) і [header](#) адаптовані за допомогою технології [flex](#) та медіа-запитів, як і увесь сайт[15; 16; 17; 19; 20]. Рис. 4.5 – Вигляд [header](#) і [footer](#) (у тому числі й адаптований) Рис. 4.6 – Заходи з адаптації [header](#) та [footer](#) (частково притаманні до всього сайту) Основна сторінка [index.html](#) містить у собі окрім тегів, зазначених вище, ще й теги [section](#) з інформацією про наявні акційні пропозиції, короткий опис про діяльність компанії, переваги співробітництва саме з цим автосалоном, приклади відгуків задоволених клієнтів. Кожен розділ включає в себе кнопку

» Цитування: 0,02% id: 37

«Дізнатися більше»,

що переносить вас на відповідну тематичну сторінку сайту, що являє собою, по суті, альтернативну навігацію[15; 16; 17; 19; 20]. Рис. 4.7 – Сторінка [contacts.html](#) На сторінці [contacts.html](#) маємо форму для зворотнього зв'язку чи відправки коментаря. В останньому випадку свою електронну поштову адресу вказувати не потрібно. Рис. 4.8 – Сторінка [contacts.html](#) Центральними сторінками у розробці сайту є [catalog.html](#) та пов'язана з нею [model.html](#). На [catalog.html](#) представлені карточки автомобілів обраної марки (перша стоїть за замовчуванням) з відповідними зображеннями та коротким описом технічних характеристик. При натисканні на зображення чи кнопку

» Цитування: 0,02% id: 38

«Дізнатися більше»

відбувається перехід на [model.html](#), де вже містяться більш розширений список технічних характеристик та детальних зображень, реалізованих за допомогою каруселі. Рис. 4.9 – Сторінки [catalog.html](#), [model.html](#) 4.4 Створення структури та опис [React](#)-компонентів

панелі адміністратора Згідно з усіма правилами побудови [React](#)-додатків, вся програма має бути розбита на низку окремих взаємопов'язаних між собою компонентів, які, у свою чергу, і формують структуру програми. Виходячи з цієї концепції, а також ґрунтуючись на результатах проектування та інших вимог, можна скласти певну ієрархію компонентів. У загальному вигляді попередня структура виглядає так: Рис. 4.10 – Структура програми У кореневому файлі [index.js](#) відображається лише імпортований компонент-менеджер [app.js](#), який взаємопов'язаний з усіма іншими і є керуючим[9]. Кожен компонент представлений у вигляді папки, в якій знаходяться 2 одноименні файли з розширеннями [.js](#) і [.css](#). У першому міститься код компонента, а другий несе у собі необхідні стилі для забезпечення дизайнерської реалізації його зовнішнього вигляду, на думку розробника. І завершує послідовність файл із розширенням [.js](#) - [index.js](#). Справа в тому, що у [webpack](#) є механізм, який полягає в наступному: якщо ми імпортуємо не файл, а папку, то всередині цієї папки веб-пакет спробує знайти файл, який називається [index.js](#). За його наявності здійснюється імпорт за промовчанням. Це робиться для зручнішого запису адреси файлу, який потрібно імпортувати[9; 14] Рис. 4.11 – Список файлів в одній із папок структури 4.5 Забезпечення зв'язку між компонентами шляхом використання стандартної бібліотеки маршрутизації [React](#) - [React Router](#) Сполучною ланкою між усіма компонентами виступає інструмент [React Router](#). Він також забезпечує їх виклик як реакцію на будь-яку подію. Для встановлення потрібно прописати в терміналі команду [npm-install react-router-dom](#). Роутинг [UI](#)-додатків досить складний, тому краще його розбити на окремі блоки, так звані сторінки. випадку, всі

Цитування: **0,01%**

id: **39**

«сторінки»

вимагають ідентифікації. Як ідентифікатор може бути [URL](#)-подібна структура[26]. [export const START_PAGE = '';](#)
[export const SIGN_IN = '/signin';](#)
[export const MIDDLE_PAGE = '/mainpage';](#)
[export const TEST = '/ Test';](#) Крім механізму включення та вимкнення певних компонентів, бібліотека [React Router](#) оновлює поточну [URL](#)-адресу, щоб новий [URL](#) вказував саме на вибрану сторінку, [Route](#) – складова частина [Router](#) - порівнює поточну адресу і значення [path](#), а потім вирішує, чи слід прямо зараз відобразити вміст чи ні [Route](#)-ами. Перевірку на точний збіг шляху реалізує параметр [exact](#). В окремих випадках, для реалізації складної навігації, доцільно використовувати функцію-компонент вищого порядку з [route](#), здатним передавати в інший компонент об'єкти реактивного [router](#): [history](#), [location](#) і [match](#). Перший об'єкт додає новий елемент до історії браузера. Останній компонент буде використовуватися під час роботи з базою після подальшої інтеграції з сервісом [Firebase](#).
[div AuthUserContext.Provider value={this.state.authUser} Router Route exact path={](#)
[ROUTES.START_PAGE} component={MiddlePage} / Route exact path={ROUTES.SIGN_IN}](#)
[component={Form} / Navigation aut exact path={ROUTES.MIDDLE_PAGE} component={Table} /](#)
[Route exact path={ROUTES.TEST} component={MainPage} / /Router /AuthUserContext.Provider](#)
[/div](#) 4.6 Конфігурування складових компонентів Як уже було зазначено в розділі 3.1, додаток буде складатися з трьох основних компонентів, а також окремої технічної частини, що налаштовується. Найпростішим буде так званий стартовий компонент, куди користувач потрапляє відразу після запуску програми в браузері[4; 14]; . На окрему увагу заслуговує форма авторизації – наступний за ієрархією та складністю компонент. Саме художнє оформлення форми, поля та правила заповнення є стандартними. Проте є деякі особливості. Насамперед – це наявність перевірки від [Google](#) - [reCAPTCHA](#) для [React](#). Вона встановлюється за допомогою команди [npm i react-recaptcha](#). [reCAPTCHA](#) - це безкоштовний сервіс, який захищає сайт або веб-орієнтовану програму від спаму та зловживань. Він використовує вдосконалений механізм аналізу ризиків, щоб відрізнити людей і ботів - поставляється у вигляді віджету, який ви можете легко додати до свого блогу, форуму, реєстраційної форми тощо. Рис. 4.12 -[ReCAPTCHA](#) у роботі Рис. 4.13 – Форма авторизації Також зобразимо код клієнтської частини, її код наведений нижче. Вона містить у собі можливості для редагування полів таблиць та відправляти результат на сервер через форму: Рис. 4.14 – Код клієнтської частини Фронтенд створено на [React](#) із використанням [CDN](#) ([React](#), [ReactDOM](#), [Babel](#)) для простоти. Для стилізації використано [Tailwind CSS](#) через [CDN](#), адаптоване до дизайну сайту[9; 27]; . Структура адмін-панелі Адмін-панель має вкладки для кожної таблиці. Кожна вкладка містить: Таблицю із даними (з можливістю редагування та видалення). Форму для додавання/оновлення записів із випадковими списками для зовнішніх ключів (напр., [id_brend](#), [id_model](#)). Кнопки у стилі [.btn-default](#). Рис. 4.15 – Вигляд адмін панелі (таблиця [test_drive](#)) Вона отримує дані з серверу, обробляє їх, виводить на екран, надаючи користувачеві змогу їх змінювати, далі усі зміни зберігаються в стейт та передаються на сервер[6]. Також серверна частина, що обробляє запити, вносить зміни, якщо вони є, до бази даних та відправляє код відповіді, наприклад, до таблиці моделі: Рис. 4.16 – Код серверної частини Висновок по розділу 4 Спроектовано та розроблено веб-орієнтовану систему підтримки діяльності автосалону. Описано основні [React](#)-компоненти, налагоджено забезпечення взаємодії між ними за допомогою бібліотеки [React Routing](#), створено сайт, реалізована серверна частина через [Node.js](#), а також реалізована та під'єднана база даних через СУБД [MySQL](#). Висновок Результатом виконаної бакалаврської роботи стала розроблена веб-орієнтована системи для підтримки діяльності автосалону. Першим етапом ми саме проаналізували функціонали вже існуючих та впроваджених систем та сайтів, що дозволяють виконувати аналогічну роботу, згідно з вимогами до нашого проекту. До цих систем можна віднести великі інформаційні та провітницькі рішення, такі як [Infocar](#), сайти-агрегатори оголошень [avto.rja](#) та переважна частина сайтів комерційних автосалонів. Таким чином, на основі узагальнення зібраних даних, були сформовані вимоги до нашої системи, що поєднують в собі необхідний функціонал, дизайн та ефективність. Стек технологій, визначений для розробки проекту ґрунтується на клієнт-серверній архітектурі та реалізовуватиметься через фреймворки та бібліотеки [JavaScript](#), як на стороні фронтенду так і бекенду, зокрема [react.js](#) та [node.js](#). Зберігання та обробку даних забезпечує реляційна БД на мові [SQL](#), що відповідає третій нормальній формі. Необхідні пояснення про доцільність вибору саме описаних рішень

детально обґрунтовуються другому розділі звіту. Також, варто зазначити, що розмірковування по даній роботі та деякі рішення лягли в основу статті Розробка веборієнтованої системи для підтримки діяльності автосалону". Отже, за підсумками виконаної роботи та після неодноразових консультацій з керівником, побудована вб-орієнтована система, що повністю відповідає поставленим вимогам, усі технічні рішення реалізовані з використанням сучасних веб-технологій. Список використаних джерел Найкращі мови програмування 2020, які варто вивчати: веб-сайт. URL: <https://merehead.com/ru/blog/popular-programming-languages-2020/> [JavaScript](#) фреймворки. Що це таке? Навіщо? Чому? : веб-сайт. URL: <https://www.reclamare.ua/blog/javascript-frejmorki/> Гайна Г.А. Основи проектування баз даних. : Кондор, 2018. 208 с. Мельник Р. А. Програмування веб-застосунків (фронт-енд та бек-енд) : Львівська політехніка, 2018. 246 с. Джон Дакетт. [HTML](#) і [CSS](#). Розробка та дизайн веб-сайтів. : К., 2013. 246 с. [Node.js](#) + [MySQL](#) : веб-сайт. URL: https://www.w3schools.com/nodejs/nodejs_mysql.asp CRM система для автосалонів та автотранспорту : веб-сайт. URL: <https://wezom.com.ua/ua/blog/crm-sistema-dlja-avtosalonov-i-avtodilerov> Дворніков Д., Козуб Ю.

Цитування: **0,09%**

id: **40**

"Розробка веборієнтованої системи для підтримки діяльності автосалону".

[Grail of Science](#), no. 52. 2025, pp. 686-691, doi:10.36074/grail-of-science.23.05.2025.093 [Robert Wieruch](#). [The Road to learn React](#) – M-Publishing, 2025. – 260 с. Алан Бол'є. Вивчаємо [SQL](#) : Науковий світ. 2023. 402 с. [Jonathan Wexler](#). [Get programming with Node.js](#) : [Manning](#), 2019. 250 с. [MySQL documentation](#) : веб-сайт. URL: <https://dev.mysql.com/doc/> Що таке [CRUD](#) простими словами: функції, переваги та приклади : веб-сайт. URL: <https://highload.tech/uk/shho-take-crud-prostimi-slovami-funktsiyi-perevagi-ta-prikladi/> [Creating a React, Node, and Express App](#) : веб-сайт. URL: <https://dev.to/techcheck/creating-a-react-node-and-express-app-1ieg> [flex](#) - [CSS](#): [Cascading Style Sheets](#) - [MDN Web Docs](#) : веб-сайт. URL: <https://developer.mozilla.org/en-US/docs/Web/CSS/flex>. [CSS Flexbox Layout Guide](#) : веб-сайт. URL: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>. [CSS](#) Медіа запити - Приклади - [W3Schools](#) українською : веб-сайт. URL: https://w3schoolsua.github.io/css/css3_mediaqueries_ex.html [Jon Duckett](#). [HTML and CSS: Design and Build Websites](#) : [Wiley](#), 2011. 512 с. Що таке адаптивний дизайн сайту та як його зробити : веб-сайт. URL: <https://hostiq.ua/blog/ukr/adaptive-design/> Що таке медіа запити [CSS](#) і для чого вони потрібні : веб-сайт. URL: <https://freehost.com.ua/uk/fag/articles/chto-takoe-media-zaproci-css-i-dlja-chego-oni-nuzhni/>. Повний посібник (v1): Конструктор запитів : веб-сайт. URL: <https://yiiiframework.com.ua/uk/doc/guide/database.query-builder/> Мулеса О.Ю. Основи мови запитів [SQL](#) : посібник. Ужгород, 2015. 48 с. URL: <https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/8868/1/sql.pdf> (дата звернення: 06.05.2025) [MySQL Connector/Node.js](#) : веб-сайт. URL: <https://dev.mysql.com/doc/dev/connector-nodejs/latest/>. В.В.

Обнаружен Плагиат: **0,16%** https://uk.wikipedia.org/wiki/Біліченко_Віктор_...

id: **41**

Біліченко, В.В. Варчук, О.В. Вдовиченко Менеджмент технічних служб на автотранспортних підприємствах

: посібник. Вінниця: ВНТУ, 2006. 154 с. URL:

<https://atm.vntu.edu.ua/subject/books/MTS%20na%20AT/Posibnyk.pdf> (дата звернення:

06.05.2025) [Anthony DeBarros](#) [Practical SQL: A Beginner's Guide to Storytelling with Data](#) : [No](#)

[Starch Press](#), 2018. 312 с. [React Router Official Documentation](#) : веб-сайт. URL:

<https://reactrouter.com/>. Порівняння популярних фреймворків для фронтенд-розробки: веб-

сайт. URL: <https://it-rating.ua/porivnyannya-populyarnih-freymvorkiv-dlya-frontend-rozrobki/>

[Postgres vs. MySQL: a Complete Comparison in 2025](#) : веб-сайт. URL:

<https://www.bytebase.com/blog/postgres-vs-mysql/>. [SQLite vs MySQL vs PostgreSQL: A](#)

[Comparison Of Relational Database Management System](#) : веб-сайт. URL:

<https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-ma>

[MySQL vs SQL Server: Your Ultimate Comparison Guide](#) (2025) : веб-сайт. URL:

<https://www.astera.com/knowledge-center/mysql-sql-server/>. [SQL vs NoSQL: 5 Critical](#)

[Differences](#) : веб-сайт. URL: <https://www.integrate.io/blog/the-sql-vs-nosql-difference>. [Difference](#)

[between SQL and NoSQL](#) : веб-сайт. URL:

<https://www.geeksforgeeks.org/difference-between-sql-and-nosql/>. Додаток А Додаток Б

Додаток В

Заявление об ограничении ответственности:

Этот отчет должен быть правильно истолкован и проанализирован квалифицированным специалистом, который несет ответственность за оценку!

Любая информация, представленная в этом отчете, не является окончательной и подлежит ручному просмотру и анализу. Пожалуйста, следуйте инструкциям: [Рекомендации по оценке](#)

Детектор Плагиата - Ваше право на оригинальность! © SkyLine LLC