МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ОНЛАЙН СЕРВІСУ ОРЕНДИ АВТОМОБІЛІВ»**

Виконав: студент 2 курсу, групи 8.1211-1іпз
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
І.А. Горський
(ініціали та прізвище)

Керівник декан математичного факультету, професор, д.т.н. Гоменюк С.І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент
_____ Лісняк А.О.
(підпис)

« ____ » _____ 2022 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Горському Івану Андрійовичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка онлайн сервісу оренди автомобілів

керівник роботи Гоменюк Сергій Іванович, д.т.н., професор
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від «04» травня 2022 року № 500-с

2. Строк подання студентом роботи 28.11.2022

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік питань до розробки.
3. Основна література.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі, аналіз предметної області.
2. Аналіз вимог до інформаційної системи.
3. Реалізація інформаційної системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 04.05.2022

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	11.05.2022	
2.	Збір вихідних даних.	13.05.2022	
3.	Обробка методичних та теоретичних	06.06.2022	
	джерел.		
4.	Розробка першого розділу.	01.07.2022	
5.	Розробка другого розділу.	07.08.2022	
6.	Розробка третього розділу.	12.09.2022	
7.	Розробка четвертого розділу.	21.10.2022	
8.	Оформлення та нормоконтроль	20.11.2022	
	кваліфікаційної роботи.		
9.	Захист кваліфікаційної роботи.	15.12.2022	

Студент _____ І.А. Горський
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.І. Гоменюк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка онлайн сервісу оренди автомобілів»: 49 с., 15 рис., 8 джерел, 1 додаток.

БРАУЗЕР, БД, ВЕБ-ДОДАТОК, ГОСТ, ІНТЕРНЕТ, ІНФОРМАЦІЙНА СИСТЕМА, САЙТ, API, CSS, HTML, JAVASCRIPT, REACT, NODE JS.

Об’єкт дослідження: розробка сайту з використанням React та Node.js.

Мета роботи: розробити веб-сайт за допомогою React та Node.js для надання послуг.

Методи дослідження: проаналізувати предметні області, вивчення та узагальнення, моделювання.

У кваліфікаційній роботі досліджено предметну область, засоби реалізації веб-додатку сервісу оренди авто, обрано найкращий засіб реалізації та розроблено веб-додаток. Розглянуто основні особливості JavaScript, React та Node.js реалізовано базу даних у MySQL.

Клієнтська та серверна частини були розділені з використанням різних технологій, клієнтська частина реалізована з використанням React, серверна частина реалізована з використанням Node.js та працює по принципу запит – відповідь, реалізований зручний інтерфейс.

SUMMARY

Master's Qualifying Paper «Development of the Online Car Rental Service»: 49 pages, 15 figures, 8 references, 1 supplement.

BROWSER, DB, WEB APPLICATION, GOST, INTERNET, INFORMATION SYSTEM, SITE, API, CSS, HTML, JAVASCRIPT, REACT, NODE JS.

The object of research: site development using React and Node.js.

Aim of the study: To create a website with React and Node.js to provide services.

Research methods: analysis, study and generalization, modeling.

In the qualification work the subject area, means of realization of the web application of the online store are investigated, the best means of realization is chosen and the web application is developed. The main features of React and Node.js are considered, the database in MySQL is implemented.

The client and server parts were separated using different technologies, client part was implemented using React and the server part using Node.js, works on the principle of request-response, implemented a user-friendly interface.

ЗМІСТ

ВСТУП

У сучасному світі популярність мобільного інтернету зростає стрімкими темпами. Якщо у 2014 році середня кількість часу, яке користувач проводив у мобільному інтернеті, становило 74,4 хвилини на день, то у другому кварталі 2022 року цей показник досяг 108,6 хвилин на день. Це, за даними авторитетного інтернет-видання Statista.com, становить приріст майже в півтора разу. Як наслідок, у 2021 – 2022 році відбулася значуща для онлайн-бізнесу подія – вперше в історії інтернету активність мобільних користувачів перевищила активність користувачів ПК та склала 55 % від загального часу користування глобальної мережі.

Ця статистика однозначно вказує на те, що нині кінцевий користувач вважає за краще використовувати мобільні пристрої: смартфони та планшети, ніж стаціонарні комп'ютери чи ноутбуки.

І це зрозуміло, бо людині більше не потрібно бути прив’язаним до якоїсь локації щоб мати доступ до інтернету. Тепер люди можуть подорожувати та при бажанні за декілька хвилин замовити послугу через інтернет, з більшим вибором та комфортом. У нашому випадку це швидка оренда авто, бо на мою думку, авто орендують не плануючи заздалегідь, а навпаки на декілька годин, щоб кудись дібратись, та не витрачати багато грошей. Звісно якщо це не якісь специфічні моделі авто, як наприклад спорткари.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Предметна область

Сьогодні великим попитом користується послуга з оренди та прокату автомобілів по всьому світу. Відсутність особистого чи навіть службового автомобіля може стати серйозною проблемою, особливо, якщо необхідно зустріти в аеропорту партнерів з бізнесу, родичів, друзів чи просто провести день у роз'їздах.

Громадський транспорт тут не допоможе, таксі – надто дороге задоволення, тому у разі поломки чи відсутності свого автомобіля найкращим виходом буде його оренда. Прокат автомобіля без водія пов'язаний із низкою важливих переваг, серед яких:

1) можливість вибору автомобіля, що відповідає випадку марки та класу – для роз'їздів містом, для ділових поїздок або для зустрічі високопоставлених гостей;

2) відсутність потреби у технічному обслуговуванні та зміст машини при повній впевненості в її справності, високго ступеня надійності та безпеки;

3) висока мобільність у будь-яких ситуаціях, можливість швидкої заміни, вийшов з ладу власний автомобіль.

З розвитком інтернету орендувати автомобіль стало значно легше. Прокатні компанії надають послуги онлайн-бронювання через свої сайти, де можна швидко знайти відповідний за ціною та класом автомобіль у потрібному місці та заздалегідь його забронювати. Але що якщо у обраної прокатної компанії немає офісу в потрібній локації чи ні автомобіля під обмежений бюджет? Очевидно, треба шукати іншого прокатника. Це стає вкрай незручним, якщо фінансовий бюджет дуже обмежений або бажане місце оренди не найпопулярніше з погляду туризму. У такому випадку пошук автомобіля може зайняти занадто багато часу, поки орендар не знайде

відповідний сайт і не вибере автомобіль, що задовольняє його. Було б зручно користуватись інтернет-сервісом, який зможе знайти автомобіль за заданими критеріями не в одній прокатній компанії, а відразу у багатьох. Таким чином, пошук буде здійснюватися в рази швидше та ефективніше, що піде на користь клієнту, який буде задоволений вибору, орендодавцю, який отримає свою вигоду з оренди, а сам сервіс матиме фінансовий прибуток за надання посередницьких послуг.

Також зараз популярною є послуга оренди міні автобусів, для перевозки великої кількості речей, предметів великого розміру, тяжких речей, або ж просто для великої кількості людей.

Необхідно розробити веб сайт, що надає можливість людині орендувати автомобіль за допомогою інтернету у будь-якій точці світу. Воно дозволить користувачеві не дбати заздалегідь про бронювання, але здійснити його за потребою наприклад під час подорожі до іншого міста чи країни, що заощаджує багато часу і часом конче необхідно в екстреній ситуації.

Сайт не потребує використання різних технологій для розробки на різні платформи та операційні системи, тому йому достатньо адаптивного інтерфейсу для різних розмірів екранів для зручнішого використання як на ПК та ноутбуках, так і на смартфонах та планшетах. Ну, і звичайно, сам алгоритм пошуку має бути ефективним, за це відповідає серверна частина. Тому важливо правильно підібрати технології та спроєктувати не тільки веб сайт, як клієнтську сторону, а й серверну, як головну складову успіху.

1.2 Технічне завдання

Розробити клієнт-серверну архітектуру, призначену для функціонування веб сайту з прокату та оренди автомобілів. Вивчити предметну область, врахувати переваги та недоліки існуючих ринкових рішень.

Функціональні вимоги до клієнтської частини сайту:

– можливість перегляду усіх наявних авто та сортування за «Тип», «Місткість», та «Ціна»;

– при переході на сторінку каталогу, користувач має змогу передивитися усі товари з помітками про категорію та теги цього товару;

– більш ретельний опис та характеристика авто при виборі його у каталозі;

– також клієнт може створити обліковий запис, тому потрібно розробити особистий кабінет. В кабінеті потрібно зберігати інформацію про всі транзакції клієнта та статус його замовлень.

Функціональні вимоги до частини сайту адміністратора.

Можливість авторизації, для доступу панелі адміністратора. Адміністратор має можливість редагування автомобілів(їх наявність, статус, особливості, опис, ціна оренди), додавати нові, та їх видалення. Редагування свого аккаунту. Доступ до аккаунтів користувачів для їх редагування, наприклад перегляд замовлення.

До нефункціональних можна віднести стандарти інтерфейсу користувача. Зручний та красивий веб-сайт приверне більше клієнтів.

Інтеграція з іншими сервісами задля розширення послуг. Це можуть бути будь-які сервіси, що пропонують послуги зв’язані із поїздками.

1.3 Вибір технологій

Будь-який багатофункціональний додаток, що працює за принципом клієнт-серверної архітектури, що складається з двох взаємодіючих базових елементів: керуюча програма, написана якоюсь мовою програмування, та сховище даних. У програмі закладено логіку роботи додатка: вона визначає, як обробити той чи інший запит, що надійшов від клієнтів. Сховище призначене виключно для зберігання. інформації, як правило, для цього використовується база даних. Звісно, програма може звертатися до сховища за

даними, і вона ж повинна їх належним чином обробляти.

Як наслідок, якість роботи програми визначається ефективним програмним забезпеченням та швидкістю обробки даних. На сьогоднішній день існує велика кількість технологій, за допомогою яких можна реалізувати практично будь-який необхідний функціонал. Так у чому відмінність і перевага однієї технології над іншою? Як правило, у промисловій Розробці важливу роль відіграє не тільки якість виробленого продукту, але швидкість досягнення результату. Саме цей фактор змушує вибирати серед безлічі технологій одну – найбільш оптимальну.

Але перш ніж щось вибирати, необхідно подивитися, які технології бувають, чим вони відрізняються і в яких випадках необхідно використовувати ту чи іншу з них.

Введемо деякі базові поняття.

Традиційно у сучасних технологіях виділяють 3 рівні абстракції.

Чиста (нативна) мова програмування – це матеріал, з якого можна зробити все, що завгодно. Обмеження накладаються самою мовою. Під словом «чистий» розуміється відсутність будь-яких сторонніх доповнень (змін), що вносяться до функцій мови ззовні. чистою мовою написані всі найбільші веб-додатки світу з багатомільйонною відвідуваністю.

Фреймворк – це якесь середовище розробки для програміста з готовими правилами та інструментами. Завдання фреймворку – спростити процес розробки з допомогою застосування вже готового функціоналу. З одного боку, фреймворк допомагає та прискорює розробку, а з іншого – накладає певні обмеження. За допомогою фреймворків розробляються проєкти середньої та великої складності. На сьогоднішній день це самий найпопулярніший спосіб розробки.

CMS – це вже готове рішення, конструктор, у якому користувач частинами збирає потрібний проєкт. Його швидше не програмують, а налаштовують. Тут обмежень на порядок більше, ніж у фреймворку, і вийти за їхні межі вкрай складно та неефективно. Зазвичай на CMS розробляються

прості сайти з відвідуваністю до мільйона користувачів на місяць. Як показує статистика останніх років, фреймворки поступово витісняють CMS із ринку веб-розробки.

Найчастіше один рівень абстракції виходить з іншого. Тобто, чистою мовою роблять фреймворки, а на фреймворках пишуть CMS, але не навпаки. У зв'язку з цим продуктивність написаного ПЗ та швидкість його створення теж різна: що вищий рівень абстракції – то гірше продуктивність, і що нижча абстракція, то довше розробляти додаток.

Наше завдання вибрати оптимальну технологію, яка дозволить створити не тільки ефективно працюючий додаток, а й реалізувати його за досить короткі терміни.

Виділимо найважливіші критерії при виборі технологій:

– розмір та складність проєкту;

– швидкість розробки;

– наявність готових рішень та великої спільноти;

– вартість та доступність фахівців;

– відмовостійкість рішення;

– тренд її розвитку технології;

– наявність докладної документації;

– вартість підтримки;

– вимоги до навантажень;

– вимоги до безпеки;

– можливості інтеграції з іншими рішеннями.

Для початку необхідно визначитися з масштабами та складністю нашого проєкту. Часто тип програми говорить сам за себе, і можна відразу визначити, яке рішення буде найоптимальнішим: використовувати CMS платформу, фреймворк чи щось інше.

Традиційно проєкти прийнято розділяти на три групи: прості, середні, складні.

До простих відносяться сайти-візитки, односторінкові лендинги та прості інтернет магазини. Вони не призначені на велику аудиторію та великі серверні навантаження. Такі завдання мають масу стандартних, перевірених роками, рішень. Зазвичай їх реалізують на стандартних (коробочних) рішеннях, CMS-платформах або шаблонах, щоб максимально швидко та дешево отримати потрібний результат.

До середніх них належать складні інтернет-магазини та маркетплейси, портали. національного масштабу та різноманітні багатофункціональні сервіси. Стандартного набору функціоналу CMS систем не вистачає, тому такі рішення зазвичай розробляються із застосуванням фреймворків.

До складних них відносяться величезні портали, спрямовані на багатомільйонну аудиторію, соціальні мережі, інноваційні та нетипові проєкти. Для їх реалізації потрібно максимальна продуктивність від серверної частини. Зазвичай такі сервіси реалізують нативною мовою програмування та їх розробка, як правило, найдовша.

Проаналізувавши поставлене завдання, наш додаток можна віднести до другої групи. Дійсно, розроблений сайт призначений для великої аудиторії користувачів, він має взаємодіяти з масою сторонніх сервісів та шаблонне рішення тут не підійде, хоча завдання не унікальне. У той же час предметна область не передбачає великих навантажень, таких, як, наприклад, у соціальних мережах. Як виявилось вище, важливо, щоб програмне забезпечення, що розробляється, було написано максимально швидко, але й максимально ефективно: якість не повинна превалювати над швидкістю.

Для досягнення цього ефекту доцільно скористатися готовим фундаментом – фреймворком. Так і вчинимо. Для більшої ясності дамо більш докладну характеристику поняття фреймворк.

Фреймворк – це програмна платформа, яка визначає структуру програми, що полегшує розробку технічно складних проєктів.

Як правило, фреймворк містить базові програмні модулі, а вся логіка та функціонал проєкту реалізується розробником на їх основі. За рахунок цього

досягається не тільки висока швидкість розробки, а й надійність рішення. Важливо розуміти, що фреймворк – це програмне забезпечення, написане та працююче однією з мов програмування. Відповідно, щоб розробляти на фреймворку, потрібно знати мову, на якому він написаний.

Таким чином, ми підійшли до найважливішої частини проєктування – вибір мови програмування.

1.4 Мова програмування

На сьогоднішній день існує безліч різних мов програмування, придатних для використання на серверній стороні клієнтської програми. І більше того, всіма популярними мовами є приклади великих проєктів. Якщо 10 років тому, говорячи про технології великі додатків, говорили переважно про Java, то сьогодні це може бути майже будь-яку мову програмування та стверджувати, що використовувати потрібно Саме один – стереотип. Це насамперед пов'язано з розвитком самих мов: за останнє десятиліття багато хто сильно просунувся у розвитку та здобули широкі можливості. Звичайно, кожна мова чимось відрізняється і, вибираючи той чи інший з них, необхідно керуватися об'єктивними критеріями з огляду на завдання проєкту.

Чистою мовою, без використання фреймворків та коробкових рішень, пишуться величезні проєкти з підвищеними вимогами щодо гнучкості, навантажень та безпеки. Для таких завдань часто бюджет та терміни не відіграють такого значення, як ефективність. Чим більший проєкт, тим більше вимог, а значить, простіше писати все з нуля, виділяючи на це найкращих спеціалістів. Але невже фреймворки мають настільки вузький функціонал, що не здатні допомогти у таких ситуаціях? Зовсім немає. Проблема фреймворків криється в іншому. У гонитві за універсальністю та багатим функціоналом для будь-яких потреб страждає продуктивність. Той обсяг коду який забезпечує простоту користування фреймворком, споживає занадто багато ресурсів

сервера, що є неприпустимою розкішшю для високонавантажених проєктів, тому розробникам доводиться використовувати нативний підхід у розробці.

Чим більший проєкт, тим більше стек технологій, який у ньому використовується. У величезних порталах може використовуватися відразу кілька мов програмування. Знову ж таки, це пов'язано з об'єктивним критерієм вибору технологій. Часто одна мова може добре виконувати одне завдання, а інший – іншу.

За роки розвитку IT найбільш широку популярність набули такі мови програмування: PHP, Python, Ruby, Java, C++, C# та JavaScript. Про кожну із них можна говорити досить довго, але я виокремлю найбільш важливі аспекти для подальшого вибору застосування до нашого проєкту (див. рис. 1.1).

Language	Community Size (in millions)	Popular Use	Least Popular In...
JavaScript	17.4	Apps, Web	DS/ML, Embedded
Python	15.7	DS/ML, IoT	Mobile, Web
Java	14.0	Cloud, Mobile	DS/ML, Web
C/C++	11.0	Embedded, IoT	Web, Cloud
C#	10.0	Desktop, Games	DS/ML, Mobile
PHP	7.9	Web, Cloud	DS/ML, Mobile
Kotlin	5.0	Mobile, AR/VR	DS/ML, Desktop
Visual Development Tools	5.0	AR/VR, Desktop	Cloud, Web
Swift	3.5	Mobile, AR/VR	Cloud, Embedded
Go	3.3	Cloud, Apps	Mobile, DS/ML
Objective-C	2.4	AR/VR, IoT	Desktop, Apps
Rust	2.2	AR/VR, IoT	Web, Mobile
Ruby	2.1	IoT, Apps	Web, Embedded
Dart	1.8	Mobile, Apps	Web, Apps
Lua	1.4	Games, IoT	Mobile, Embedded

Рисунок 1.1 – Популярні мови веб-програмування

PHP – найпопулярніша серверна мова програмування. Розроблено спеціально для веб-програмування. Його використовують як прості, так і великі багатофункціональні програми. Має найбільшу колекцію коробкових рішень та фреймворків під будь-які завдання. Відносно недорогі програмісти та легкий поріг входження. Декілька років тому вважався антитрендом в індустрії до виходу 7 версії мови, де він отримав дійсно потужні можливості та десятикратний приріст продуктивність.

Найбільш яскравий приклад – Facebook.

Python – одна з найсучасніших та найпросуненіших мов спільного призначення. Має елегантний синтаксис та одну з найбільших бібліотек розширень. Найбільшу популярність отримав у наукових сферах діяльності. Використовується для середніх та великих проєктів. Програмістів знайти складніше, і коштують вони дорожче. Має скромнішу колекцію фреймворків, ніж попередник.

Найбільш яскраві приклади: Instagram, Pinterest.

Ruby – сучасна мова програмування, розробка нею така ж швидка як на PHP та Python. Зазвичай використовується для розробки простих і Середніх проєктів часто застосовують для стартапів. Програмістів вкрай мало, і вони дорогі. Найпопулярніший і, мабуть, єдиний серйозний фреймворк: Ruby on Rails.

Найбільш яскраві приклади: 500px, Groupon.

Java – сильно типізована об'єктно-орієнтована мова програмування. Розробка на ньому вкрай довга та дорога. Його використовують в основному для великих проєктів зі специфічними вимогами, частіше всього у комерційній сфері. Однак є найпопулярнішою мовою програмування у рейтингу TIOBE.

Найяскравіші приклади: Ebay, Amazon.

C# – аналог Java, також використовуються для великих додатків. Має величезну екосистему .NetFramework, підтримувану корпорацією Microsoft. На відміну від решти технологій не є кросплатформним рішенням, що накладає серйозні економічні обмеження.

Найбільш яскраві приклади: Guru, Stack Overflow, Bank of America.

JavaScript – технологія, що дуже швидко розвивається, тренд останніх років. Величезну кількість напрацювань та бібліотек, можна написати все, що завгодно, навіть ігри. Застосовуємо для середніх та великих проєктів, але дійсно потужні можливості ця мова отримала з появою Node.js фреймворку. Фахівці найдорожчі та знайти їх найскладніше.

Найяскравіші приклади: LinkedIn, Walmart, PayPal.

C++ – найбільш вікова і складна для освоєння мова програмування, але в той же час найшвидший. Використовується тільки у високонавантажених проєктах, де потрібна максимальна продуктивність. Знайти грамотних фахівців найскладніше і розробка на ньому найдовша.

Найяскравіший приклад – Google.

Вище я описав найпопулярніші на сьогоднішній день клієнт-серверні технології. Кожна з них завоювала свою аудиторію та має успішні проєкти, які доводять, що великі веб-програми можуть бути написані різними мовами програмування, і мати багатомільйонну аудиторію. Звичайно, існує багато нових мов, які дуже швидко набирають популярність, зокрема Scala, але поки вони досить молоді та сирі.

Цей аналіз показав, що принципова різниця між сучасними технологіями немає, успіху можна досягти з будь-якої з них, грамотно підібравши інструменти реалізації. Але щоб процес розробки був максимально ефективним, вибирати технологію потрібно під вимоги конкретного проєкту.

Наш додаток не розрахований на великі навантаження, але в той же час хотілося б отримати максимально ефективну систему для роботи з даними за найкоротший проміжок часу. Також важливо відзначити, що додаток повинен мати широкий надійний функціонал, який легко підтримувати та розвивати. У зв'язку з цим мій вибір падає на мову JavaScript: він простий в освоєнні, на ньому легко писати, має величезну кількість фреймворків та бібліотек, що є визначальним фактором [1].

1.5 Фреймворк для клієнтської частини

В цілому, більшість популярних фреймворків (див. рис. 1.2) пропонують схожий функціонал: комфортне середовище для швидкого старту, широкі можливості для розширення та внутрішню екосистему. Відмінності ж криються у внутрішній філософії розробки, якої дотримувались автори – творці. Ось чому для кожного проєкту слід підбирати певний інструмент для досягнення результату.

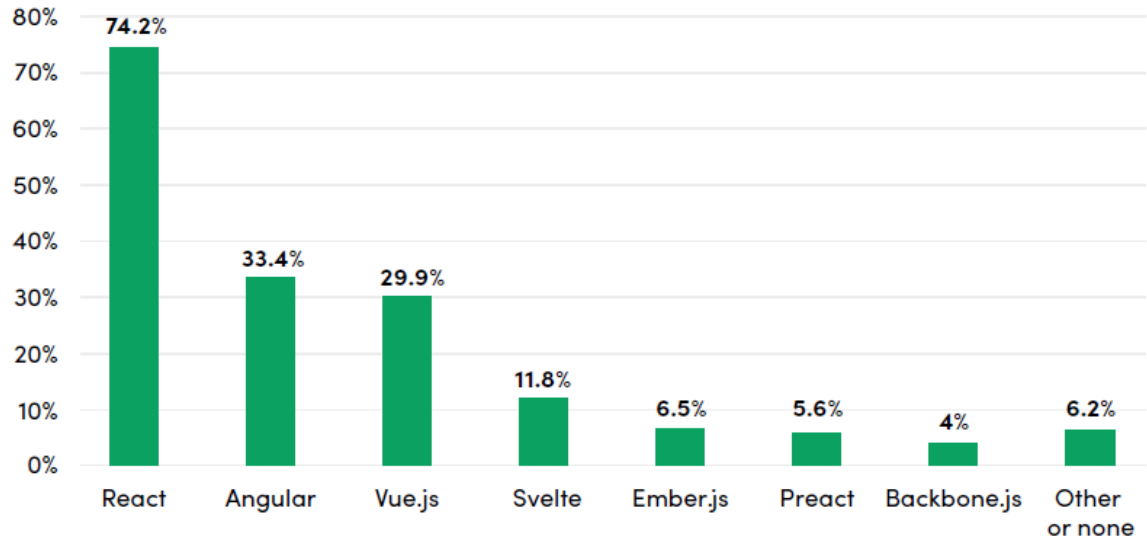

Рисунок 1.2 – Популярні JavaScript фреймворки за останній рік

При виборі фреймворку важливо звернути увагу на низку аспектів. Розглянемо їх детальніше.

Розмір спільноти. Велика популярність серед спільноти розробників свідчить якість використовуваного інструмента. У сучасному світі IT популярність і позитивна динаміка зростання свідчить про те, що фреймворк успішно функціонує, розвиватиметься та обростатиме новими можливостями. Тому варто звернути увагу на інструменти, які перебувають у тренді.

Стійкість. При виборі фреймворку для довгострокового проєкту необхідно переконатися, що автори будуть підтримувати його розробку

протягом тривалий час. Надалі це гарантує своєчасну модернізацію та супровід вже написаних додатків.

Підтримка. Варто звернути увагу на те, наскільки легко можна знайти відповіді на питання під час розробки. Як правило, якщо фреймворк створений досить давно, то із цим аспектом не повинно бути проблем.

Безпека. Будь-яка програма потенційно вразлива. Щоб знизити ризик злому, необхідно вибрати фреймворк, який зможе надати надійні методи захисту.

Документація. Необхідно оцінити обсяг та якість написаної документації. По-перше, інструмент, який має гарну документацію викликає більшу довіру. По-друге, такий інструмент простіше використати. Як правило, якість написаної документації повністю відображає ставлення творців фреймворку до людей, що його використовують.

Кількість спеціалістів. Для грамотної розробки на вибраному фреймворку необхідні спеціалісти, які добре його знають. Ринок повинен мати таких спеціалістів.

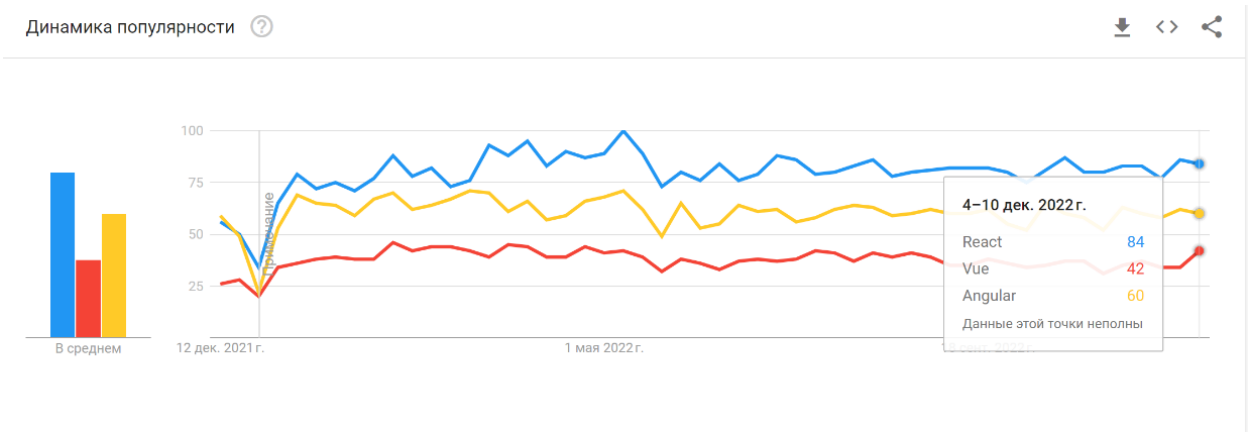

Рисунок 1.3 – Популярність запитів трьох фреймворків

Свій вибір я зупинив на фреймворку/бібліотеці React.

React (старі назви: React.js, ReactJS) – відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці

односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників [2, 3].

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux.

Redux – відкрита JS бібліотека призначена для управління станом програм JavaScript. Найчастіше використовується разом з React для побудови інтерфейсів користувача.

Redux зберігає стан всього застосунку в дереві об’єктів в одному сховищі. Одне дерево станів полегшує налагодження або перевірку програми; це також дозволяє зберігати стан вашого застосунку в процесі розробки, для прискорення циклу розробки.

2 СЕРВЕРНА ЧАСТИНА

2.1 Платформа для серверної частини

Настав час обрати платформу яка буде ідеальною парою для розробки нашого сайту, та виконувати роль серверу та створення нашого API. Тож мій вибір випав на Node.js.

Node.js (Node) – це платформа з відкритим вихідним кодом для роботи з мовою JavaScript, побудована на движку Chrome V8. Вона дозволяє писати серверний код для веб-застосунків і динамічних веб-сторінок, а також програм командного рядка. В основі платформи – подієво-керована модель з неблокуючими операціями введення-виведення, що робить її ефективною та легкою [4, 5].

До появи Node.js програми, написані мовою програмування JavaScript, можна було запускати лише у браузері. З появою платформи можна було писати на JavaScript не тільки в браузері, але і на сервері.

Причини популярності:

– висока швидкість;

– універсальність та гнучкість;

– велика кількість модулів та бібліотек;

– робота на движку Chrome v8.

2.2 База даних

Найцінніше, що є у світі технологій – інформація. З самого першого дня розвитку технологій питання про способи зберігання, обробки та вилучення інформації був одним з найбільш гострих. У міру розвитку технологій складність структур даних та обсяг інформації, який потрібно зберігати та

обробляти, стрімко росли вгору. Це призвело до того, що сьогодні неможливо уявити роботу будь-якого додатка без спеціального програмного забезпечення. Одним з таких додатків, отримали найбільший попит, стала СКБД.

СКБД (Система керування базами даних) – спеціалізоване програмне забезпечення, призначене для організації та ведення баз даних. Ці програми керують або допомагають керувати користувачем наборами даних, що зберігаються. Так як ці дані можуть бути різного формату та розміру, були створені різні види таких систем.

СКБД засновані на моделях баз даних – певних структурах обробки даних Кожна СКБД створена для роботи з однією з них з урахуванням особливостей, виконуваних операцій з інформацією. Хоча рішень, реалізують різні моделі баз даних, дуже багато, періодично деякі з них стають дуже популярними і використовуються на багато років. Кожна СКБД реалізує одну з моделей баз даних для логічної структуризації даних, що використовуються. Ці моделі є головним критерієм того, як керуватиме інформацією додаток. Існує кілька таких моделей, серед яких найпопулярнішою є реляційна – РСКБД. Хоча вона і є дуже потужною та гнучкою, Існують ситуації, рішення на які вона запропонувати не може. Тут на допомогу прийде порівняно нова модель, яка називається NoSQL. Вона набирає популярність і пропонує дуже цікаві рішення. Через те, що ці системи не використовують сувору структуризацію даних, вони пропонують більшу свободу дій під час обробки інформації.

2.2.1 Реляційна модель

Представлена в 70-х роках, реляційна модель є найстарішою та перевіреної. Вона пропонує математичний спосіб структуризації, зберігання та використання інформації у вигляді набору даних з зумовленими зв'язками

з-поміж них. Ці дані організовані у вигляді набору таблиць, що складаються зі стовпців та рядків. У таблицях зберігається інформація про об'єкти, подані в базі даних. У кожному стовпці таблиці зберігається певний тип даних, у кожному осередку – значення атрибут. Кожен рядок таблиці являє собою набір пов'язаних значень, які стосуються одного об'єкта чи сутності. Кожен рядок у таблиці може бути позначена унікальним ідентифікатором первинним ключем, а рядки з кількох таблиць можуть бути пов'язані з допомогою зовнішніх ключів. До цих даних можна отримати доступ багатьма способами, і навіть реорганізовувати таблиці БД не потрібен.

Для взаємодії з РСКБД використовується спеціально розроблена мова SQL, яка у 1890-х роках стала міжнародним стандартом. Цей стандарт підтримується всіма популярними ядрами реляційних баз даних.

Деякі з них також включають розширення стандарту, що підтримують специфічний їм функціонал. SQL використовується для додавання, оновлення та видалення рядків з таблиць, вилучення наборів даних для обробки транзакцій та аналітичних додатків, а також управління всіма аспекти роботи бази даних.

Завдяки десятиліттям досліджень та розробки, РСКБД працюють продуктивно та надійно. У поєднанні з великим досвідом використання розробниками реляційні бази даних стали вибором, що гарантує цілісність та захист інформації від втрат.

Незважаючи на суворі принципи формування та обробки даних, РСКБД можуть бути дуже гнучкими, якщо докласти небагато зусиль.

2.2.2 Нереляційна модель

NoSQL – спосіб структуризації даних (або нереляційний підхід), полягає у позбавленні від обмежень при зберіганні та використанні інформації. Такі СКБД використовують неструктурований підхід, що пропонує багато

ефективних способів обробки даних. Вони можуть використовувати різні моделі даних, включаючи стовпчасті, документні, графові дані та сховища пар «ключ-значення» у пам'яті. NoSQL бази даних набули широкого поширення завдяки забезпеченню надійної відмовостійкості та низької затримки на вилучення даних. На відміну від традиційних РСКБД, деякі бази даних NoSQL дозволяють групувати колекції даних з іншими базами даних. Такі СКБД зберігають дані як одне ціле. Ці дані можуть являти собою одиночний об'єкт на зразок JSON і водночас коректно відповідати на запити до полів.

NoSQL підхід не використовує загальний формат запиту, як SQL в реляційних баз даних. Кожне рішення використовує власну систему запитів.

Бази даних NoSQL відмінно підходять для додатків, які потребують більш високої масштабованості та нижчого часу відгуку, ніж можуть забезпечити традиційні реляційні моделі. У тому числі багато додатків для великих даних, мобільні та інтернет-додатки. За рахунок використання спрощених структур даних та горизонтального масштабування бази даних NoSQL, як правило, забезпечують більше високу продуктивність.

2.2.3 Порівняння SQL та NoSQL

Для того щоб дійти об'єктивного висновку, проаналізуємо різницю між SQL та NoSQL підходами:

– структура і тип даних, що зберігаються: реляційні бази даних вимагають наявності однозначно певної структури зберігання даних, а NoSQL бази даних таких обмежень не ставлять;

– запити: усі РСКБД реалізують єдині SQL-стандарти, тому можна отримувати дані за допомогою мови SQL, кожна NoSQL база даних реалізує свій спосіб роботи із даними;

– масштабованість: обидва рішення легко розтягуються вертикально (наприклад, шляхом збільшення системних ресурсів), тим не менш, через свою

сучасність, рішення NoSQL зазвичай надають простіші способи горизонтального масштабування (наприклад, створення кластера з кількох машин);

– надійність: коли мова заходить про надійність, SQL бази даних однозначно попереду;

– підтримка: РСКБД мають дуже довгу історію, вони дуже популярні і тому отримати підтримку дуже легко, за потреби вирішити проблеми з ними набагато простіше, ніж з NoSQL, особливо якщо проблема складна за своєю природою;

– зберігання та доступ до складних структур даних: реляційні бази даних передбачають роботу зі складними ситуаціями, тому й тут вони перевершують NoSQL-рішення.

Оскільки наш проєкт має забезпечувати максимальну надійність даних та мати можливості для масштабування, мій вибір однозначно за реляційними базами даних.

2.2.4 Вибір СКБД

Алгоритм вибору СКБД для проєкту практично аналогічний до вибору мови програмування. Для виявлення об'єктивних критеріїв на користь тієї або інший СКБД дамо кожній з них невелику характеристику та визначимо плюси та мінуси тієї чи іншої системи.

Вибиратимемо з тих, які поширюються у вільному доступі. На сьогоднішній день найпопулярнішими є: SQLite, MySQL і PostgreSQL.

SQLite. База даних, що легко вбудовується в додатки. Оскільки ця система базується на файлах, надає досить широкий набір інструментів до роботи з нею проти мережевих СКБД. При роботі з цією СКБД звернення відбуваються безпосередньо до файлів, замість звернення до портів і сокетів у

мережевих СКБД. Саме тому завдяки технологіям обслуговуючих бібліотек, SQLite дуже швидка та потужна.

Переваги:

– файлова структура: вся база даних складається з одного файлу, тому її дуже легко переносити різні машини;

– стандарти, що використовуються: хоча може здатися, що ця СКБД примітивна, але вона використовує SQL, деякі особливості опущені, але основні таки підтримуються;

– відмінна при розробці та тестуванні: у процесі розробки додатків часто виникає необхідність масштабування, SQLite пропонує все, що необхідно для цих цілей, оскільки складається всього з одного файлу та бібліотеки, написаної мовою С.

Недоліки:

– відсутність користувачів: більші СКБД включають до свого складу системи управління правами доступу користувачів, зазвичай застосування цієї функції не так критично, оскільки ця СКБД використовується в невеликі додатки;

– відсутність можливості збільшення продуктивності: знову ж, виходячи з проєктування, досить складно вичавити щось більше продуктивне із цієї СКБД.

Заздалегідь можна сказати, що ця СКБД є непридатною для нашої програми.

MySQL. MySQL – це найпопулярніша у світі повноцінна серверна СКБД. Вона дуже функціональна, вільно поширювана СКБД, яка успішно працює з різними сайтами та веб-додатками. Навчитися використання досить просто, тому що на просторах інтернету легко знайти велику кількість інформації. Варто зауважити, що завдяки популярності цієї СКБД, існує величезна кількість різних плагінів та розширень, що полегшують роботу з цією системою.

Незважаючи на те, що в цій СКБД не реалізовано весь SQL функціонал, MySQL пропонує багато інструментів для розробки додатків. Так як це серверна СКБД, додатки для доступу до даних, на відміну від SQLite, працюють із службами MySQL.

Переваги:

– простота у роботі: встановити MySQL досить просто, а додаткові програми з графічним інтерфейсом, дозволяють досить легко працювати з цією СКБД;

– багатий функціонал: MySQL підтримує більшість функціоналу SQL;

– безпека: велика кількість функцій, що забезпечують безпеку, які підтримуються за замовчуванням;

– масштабованість: MySQL легко працює з великими обсягами даних та її вкрай просто масштабувати;

– швидкість: спрощення деяких стандартів дозволяє MySQL значно збільшити продуктивність.

Недоліки:

– відомі обмеження: за задумом у MySQL закладено деякі обмеження функціоналу, які іноді необхідні в особливо вимогливих додатках;

– проблеми із надійністю: через деякі способи обробки даних MySQL (зв'язки, транзакції, аудити) іноді поступається іншим СКБД по надійність.

PostgreSQL. PostgreSQL є найпрофесійнішим із усіх трьох розглянутих нами СКБД. Вона вільно розповсюджується та максимально відповідає стандартам SQL. Від інших СКБД відрізняється підтримкою затребуваного об'єктно-орієнтованого та реляційного підходу до баз даних. Завдяки потужним технологіям Postgres дуже продуктивна. Паралельність досягнута не за рахунок блокування операцій читання, а завдяки реалізації управління багатоваріантним паралелізмом. Хоча PostgreSQL і не може похвалитися великою популярністю (на відміну від MySQL). Незважаючи на всю потужність функціоналу, існує досить велика кількість додатків, що

полегшують роботу з цією СКБД. Зараз досить легко встановити цю систему, використовуючи стандартні менеджери пакетів операційних систем, потім працювати з нею за допомогою графічних інтерфейсів.

Переваги:

– відкрите ПЗ, що відповідає стандарту SQL: PostgreSQL безкоштовне програмне забезпечення з відкритим вихідним кодом, ця СКБД є дуже потужною системою, що повністю відповідає стандарту SQL;

– велика спільнота;

– має багато доповнень: незважаючи на величезну кількість вбудованих функцій існує дуже багато доповнень, що дозволяють розробляти дані для цієї СКБД та керувати ними;

– розширення: існує можливість розширення функціоналу за рахунок збереження своїх процедур;

– об'єктність: PostgreSQL це не тільки реляційна СКБД, але також об'єктно-орієнтована система з підтримкою успадкування та багато іншого.

Недоліки:

– продуктивність: при простих операціях читання PostgreSQL може значно уповільнити сервер і бути повільнішим за своїх конкурентів, таких як MySQL;

– популярність: за своєю природою, популярністю ця СКБД похвалитися не може, хоча існує досить велика спільнота;

– хостинг: в силу перерахованих вище факторів іноді досить складно знайти хостинг із підтримкою цієї СКБД.

Проаналізувавши кожен варіант, я роблю свій вибір на користь MySQL. Цей вибір заснований на сучасному підході до розвитку цієї СКБД, а також великі можливості для розширення надалі.

В цілому, Postgres практично не поступається MySQL, тут швидше роль при виборі відіграв особистісний фактор: особисто я більше симпатизую стандарту MySQL.

На даний момент схема клієнт-серверної програми виглядає наступним чином (див. рис. 2.1).

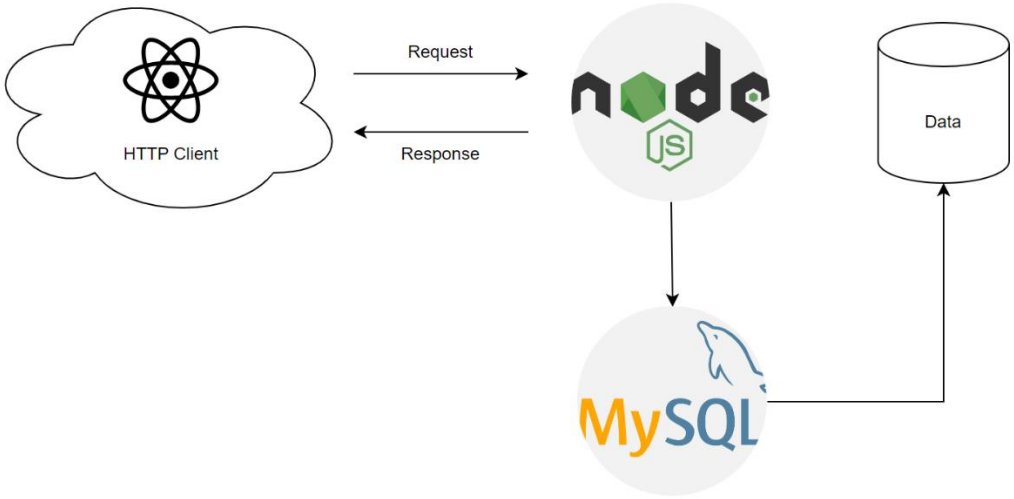

Рисунок 2.1 – Схема взаємодії клієнта, програми та бази даних

2.3 API

Щоб клієнтська програма могла взаємодіяти з сервером, необхідно організувати систему правил, дотримуючись яких, клієнт буде отримувати потрібні дані, що надаються серверним додатком. Для цих цілей розробники використовують концепцію API.

API (від англ. Application Program Interface) – це інтерфейс взаємодії між додатком клієнта та сервером. Являє собою ресурс, який сервер відкриває до роботи ззовні, тобто. клієнт може скористатися ним для отримання доступу до функціоналу програми, модуля чи даних. API уможливлює роботу ресурсів, які використовують потенціал, що надається сервером. У нас такий ресурс – web додаток.

Таким чином, необхідно організувати відкрите API, звертаючись до якого клієнт взаємодіятиме з сервером.

Функції API поділяються на 2 напрямки:

– повертаючі: на запит сторонньої програми будь-якого методу із заданими параметрами сервер дає запитувану інформацію в певному форматі;

– змінні: клієнт викликає деяку функцію сервера, яка вводить нову інформацію або змінює на ній певні налаштування.

Щоб зробити функції API найбільш зрозумілими, у сучасній веб-розробці прийнято використовувати так званий RESTfulAPI стиль. Він накладає певні правила організації URL-адрес таким чином, що перехід по них викликає відповідну функцію серверної програми. У сучасному світі IT такий підхід є правилом гарного тону, тому його й будемо дотримуватись.

2.3.1 REST API архітектура

REST (скор. від англ. Representational State Transfer – «передача стану уявлення») – архітектурний стиль взаємодії компонентів розподіленої програми у мережі. REST є узгодженим набором обмежень, що враховуються при проєктуванні розподіленої системи:

1) єдиний інтерфейс;

2) відсутність станів;

3) кешування відповіді;

4) клієнт-сервер;

5) багаторівнева система;

6) «код на вимогу».

Спланована раніше архітектура серверної програми може реалізувати ці принципи, до того ж Node.js із коробки надає зручні функції створення повнофункціонального RESTAPI [6].

Цей стиль передбачає використання 4 (базових) методів HTTP протоколу таким чином, щоб надавати запитам певного змісту:

– GET – для отримання даних;

– POST – для створення даних;

– PUT – зміни даних;

– DELETE – видалення даних.

Наприклад, запит типу GET на адресу: /api/user передбачає, що сервер поверне список всіх користувачів (user), що є на сервері. Причому клієнту, який надіслав цей запит, зовсім не важливо, звідки прийдуть дані: з бази даних, серверного чи мережевого кешу.

Щоб запитати у сервера інформацію про користувача з номером 20 слід надіслати наступний GET запит:/api/user/20. Метод типу PUT за адресою /api/user/20?name=Alex замінить, маючи користувача з номером 20, Alex.

Як можна помітити, URL-адреси мають певну структуру наступного виду: /api/{entity}/{id}. Ця адреса складається з трьох частин:

1) api – ідентифікатор звернення до API;

2) entity – назва сутності, наприклад, user чи article;

3) id – ідентифікатор сутності.

Звичайно, запит може містити параметри конкретизації вибірки, але я опущу цей момент.

Таким чином, REST-підхід дозволяє зручно маніпулювати сервером, використовуючи єдиний інтерфейс та інтуїтивно зрозумілу логіку запитів. Як формат обміну даними між клієнтом і сервером будемо використовувати JSON. Це текстовий формат обміну даними, заснований на JavaScript. Як і багато інших текстових форматів, JSON легко читається людьми, може використовуватися практично з будь-якою мовою програмування. До того ж для багатьох мов програмування існує готовий код для створення та обробки даних у форматі JSON.

3 ПРОЄКТУВАННЯ ВЕБ-ДОДАТКУ

3.1 Архітектура

Розробка будь-якого програмного продукту починається з розробки архітектури. Найбільш популярний патерн проєктування – MVC.

3.1.1 MVC

Model-View-Controller (MVC) – схема поділу даних програми, користувальницького інтерфейсу та керуючої логіки на три окремі компонента: модель, подання та контролер – таким чином, що модифікація кожного компонента може здійснюватися незалежно один від одного від друга). Коротко опишемо кожен компонент.

Модель (Model) надає дані та реагує на команди контролера, змінюючи свій стан.

Подання (View) відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі.

Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель необхідність змін.

Основна мета застосування цієї концепції полягає у відділенні бізнес-логіки (моделі) від її візуалізації (уявлення, виду). За рахунок такого поділу підвищується можливість повторного використання коду. Найкорисніше застосування цієї концепції у випадках, коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах та/або з різних точок зору.

Цей патерн є ознакою гарного тону програмування та активно використовується при розробці будь-якого програмного забезпечення.

Нижче показано типову схему концепції MVC (див. рис. 3.1).

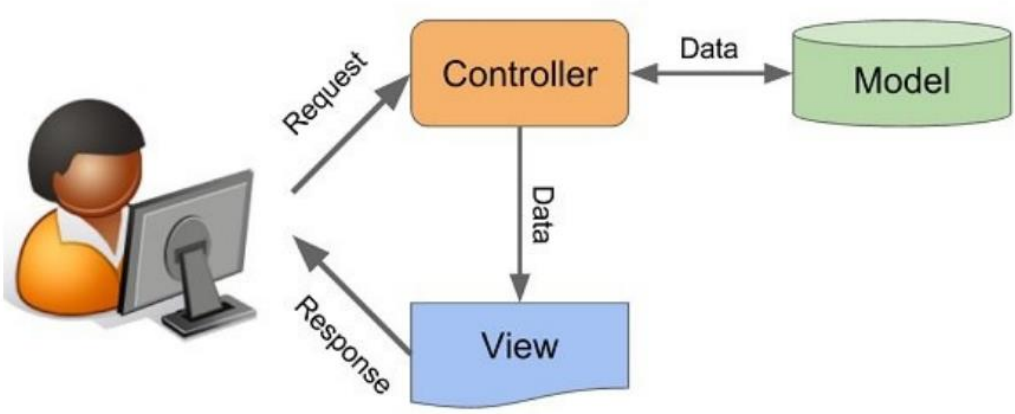

Рисунок 3.1 – Структурна схема MVC

У цій схемі головною ланкою є контролер. Його завдання полягає в тому, щоб зв'язати вигляд та модель. У свою чергу, модель відповідає за збір. даних, зокрема реалізує поводження та діалог із сервером, а вид відповідає за візуалізацію контенту на пристрої користувача.

3.2 Діаграма прецедентів

Діаграма випадків використання – це графічне зображення можливих взаємодій між користувачем та системою. Схема зображує різне використання системи та різні типи користувачів. Варіанти використання представлені колом або еліпсом. Акторів часто зображують у вигляді фігурок.

Хоча сам випадок використання може детально вивчити кожну можливість, діаграма прикладів може допомогти проілюструвати загальний вигляд системи найвищого рівня. Хтось раніше говорив, що «схема використання – це креслення вашої системи».

Завдяки спрощеному характеру, схема використання може бути хорошим інструментом комунікації для зацікавлених сторін. Ці креслення намагаються імітувати реальний світ і дати зрозуміти зацікавленим сторонам, як буде розвиватися система.

Мета використання діаграм – показати динамічні аспекти системи. Вони забезпечують спрощене графічне представлення того, що повинна робити система, коли вона використовується. Повна функціональність та технічний вигляд системи вимагають подальших схем та документації.

Для створення діаграми прецедентів було обрано двох акторів, клієнта та адміністратора. Вони добре зображують дві частини сайту, оскільки клієнт може взаємодіяти тільки з клієнтською частиною сайту, а адміністратор має доступ до панелі керування. Він може оформити замовлення, змінювати, додавати чи видаляти авто, завантажувати або видаляти картинки. Також, адміністратор має змогу додавати нового менеджера або змінювати свої поточні дані. На рисунку 3.2 зображена діаграма прецедентів з трьома акторами.

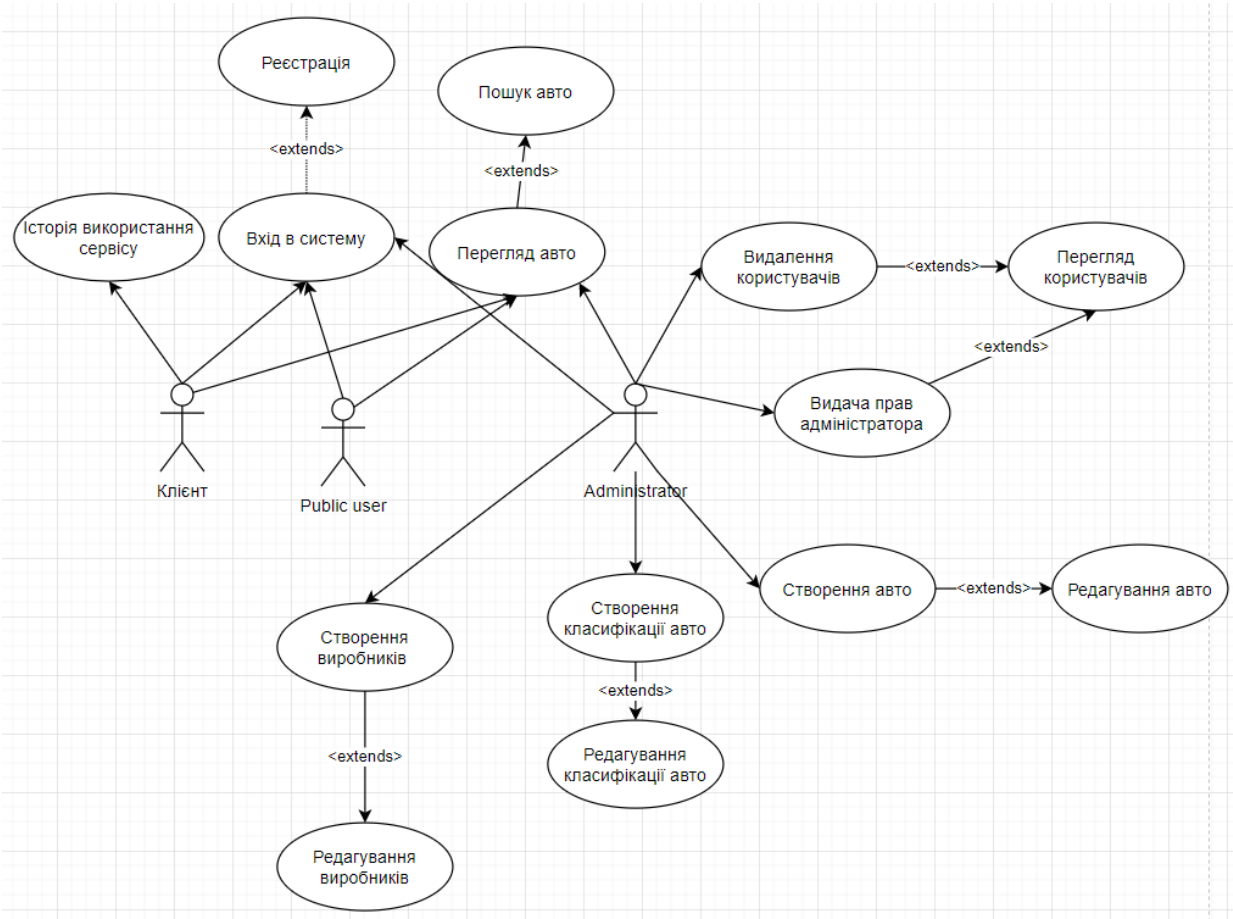

Рисунок 3.2 – Діаграма прецедентів

Актор публічний користувач, ще не пройшов реєстрацію, або не зайшов до свого облікового запису. Він може зареєструватися або увійди до системи, переглянути каталог авто або знайти за допомогою фільтрів конкретне авто.

Наступний актор, вже пройшов реєстрацію або увійшов до свого облікового запису. Йому стала доступна історія транзакцій та поїздок. Тобто він може переглянути, що було куплено на цьому записі. Також він може подивитися каталог, обрати та замовити авто.

Адміністратор системи має усі можливості що і попередні актори. Окрім цього, він має доступ до контенту системи. Адміністратор може створити нове авто, додати виробника чи тип машини. Також він може переглянути всіх користувачів, видалити обліковий запис користувача чи підвищити його статус у системі.

3.3 Функціональне проєктування

Перше, на що слід звернути увагу, це функціональне моделювання. Він представлений методологією IDEF0 (див. рис. 3.3).

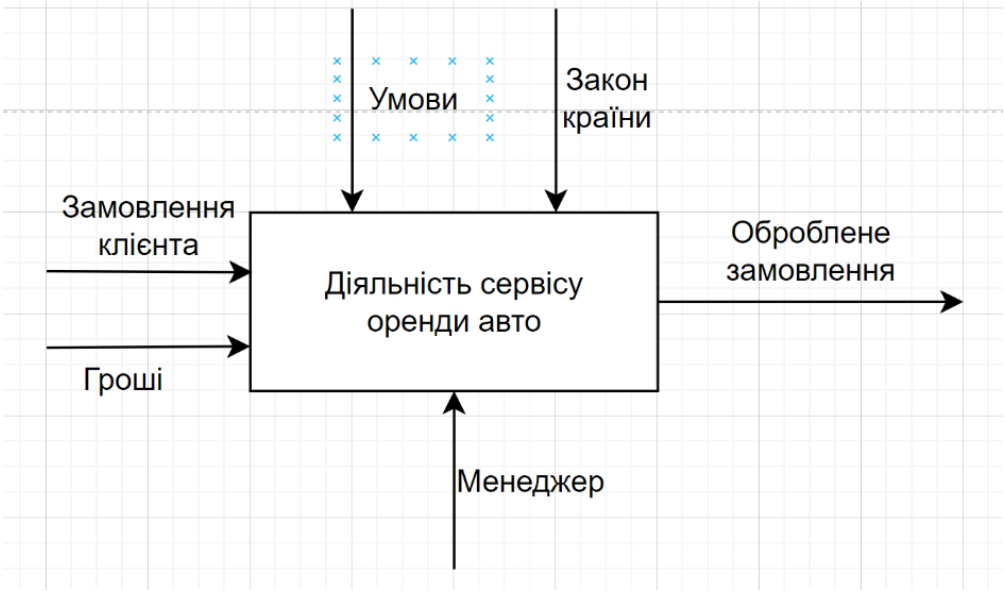

Рисунок 3.3 – Діаграма IDEF0

Суть IDEF0 полягає у побудові ієрархічної системи діаграм. Іншими словами, один опис для кожної діаграми буде частиною системи. Щоб побудувати контекстну діаграму IDEF0, на початку потрібно підкреслити, що проєкт має певні ресурси та матеріали для виконання.

Спочатку клієнт обирає авто з наявних, менеджер заносить запис до бази даних, таким чином закріплюючи авто за обліковим записом клієнта. Наприкінці отримується вже сформоване замовлення.

3.4 Діаграма огляду взаємодії

Діаграма огляду взаємодії подібна до діаграми діяльності, оскільки обидві візуалізують послідовність дій. Різниця полягає в тому, що для перегляду взаємодії кожна окрема діяльність описується як кадр, який може містити вкладену діаграму взаємодії. Це дозволяє використовувати оглядову діаграму взаємодії для «деконструкції складних сцен, інакше її потрібно пояснити різними способами, хоча б лише як одну діаграму послідовності».

Для створення діаграми послідовності спочатку потрібно визначити об’єкти, які обмінюються інформацією (див. рис. 3.4). Встановлено два об’єкти. Перший – це клієнт, він створює замовлення та прибуток сервісу. Другий – адміністратор за допомогою панелі управління, який обробляє замовлення від клієнтів.

Послідовність починається, коли користувач відвідує веб-сайт, а після презентації каталогу, виконаної сервісом, є можливість відфільтрувати товари або зробити замовлення. Сайт формує замовлення і відсилає його менеджеру, менеджер перевіряє авто та підтверджує замовлення.

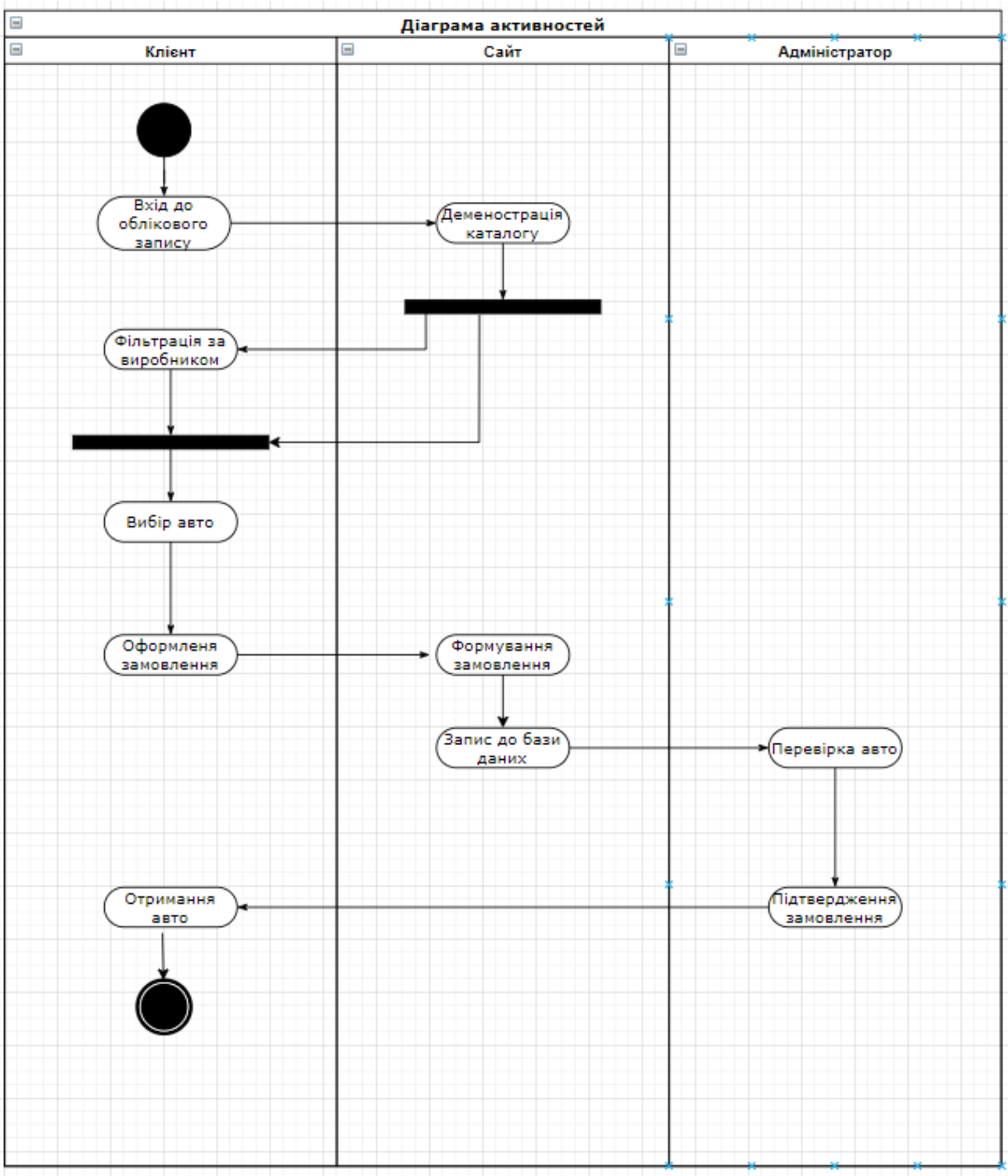

Рисунок 3.4 – Діаграма огляду взаємодії

3.5 Концептуальне проєктування інформаційної моделі

Модель «сутність-взаємозв'язок» або ER-модель описує взаємопов'язані речі, які представляють інтерес для певної галузі знань. Базова модель ER

складається з типів сутностей, які класифікують цікаві речі та визначають можливі взаємозв'язки між сутностями.

У програмній інженерії моделі ER зазвичай формуються для представлення речей, про які компанії повинні пам’ятати під час виконання бізнес-процесів. Тому модель ER стає абстрактною моделлю даних, яка визначає структуру даних або інформації, яка може бути реалізована в базі даних. На рисунку 3.5 зображено кінцеву ER-діаграму.

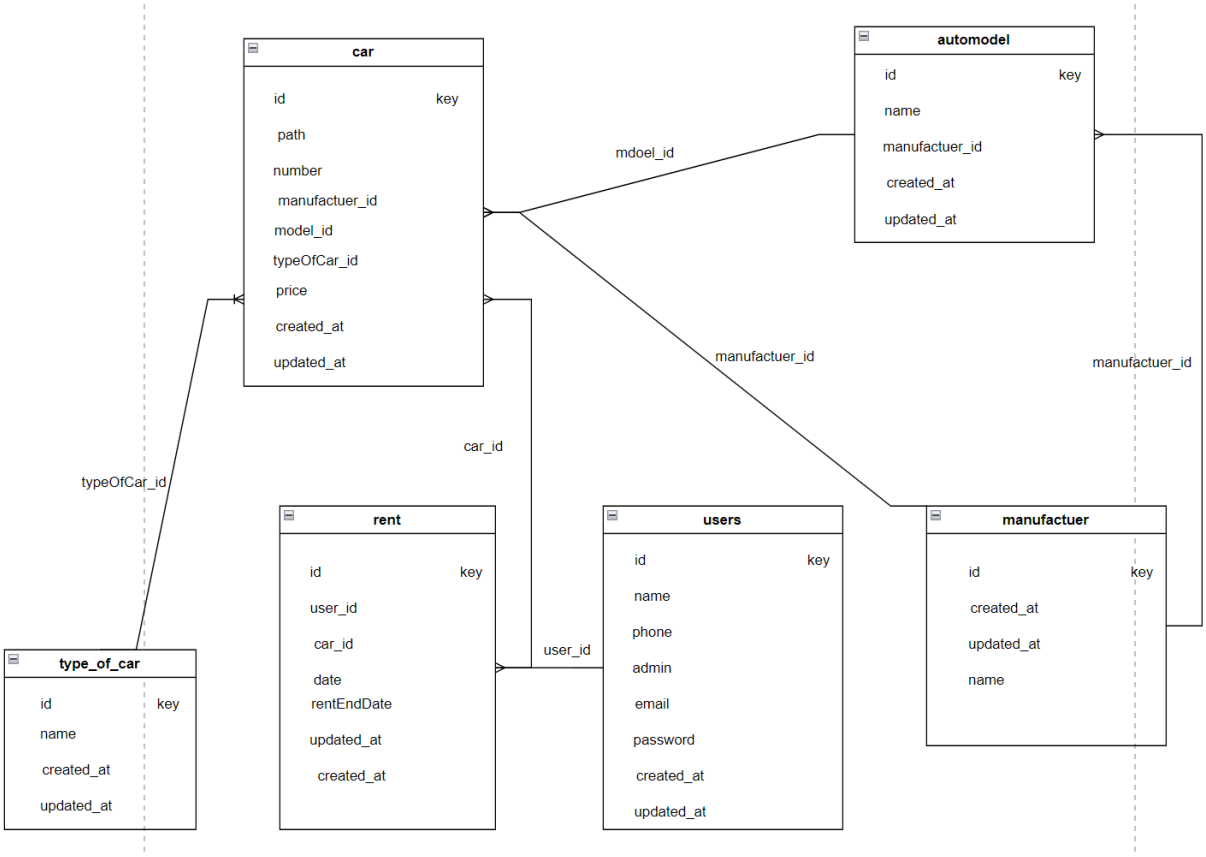

Рисунок 3.5 – ER-діаграма

Для створення ER-діаграми, потрібно виділити основні сутності проєкту. Треба почати з сутності продукту. Ця сутність є фундаментальною для системи. Вона має первинний ключ id, номер, марка, модель, номер, ціна, тип та зображення.

Також потрібно зробити окрему сутність для виробника автомобілів, буде складатися з первинного ключа id та назви.

Ще необхідно створити сутність з класифікацією авто, що також матиме первинний ключ id та назву.

Сутність rent_auto дозволяє робити замовлення клієнту, тому вона має багато атрибутів. Для формування замовлення потрібно знати id клієнта, id обраної машини, та дві дати – початкові і кінцеву. Також треба додати атрибут ціни, що дозволить вирахувати остаточну вартість замовлення.

У сутності Users стандартний набір атрибутів, в саме: id, ім’я, телефонний номер, електронна пошта та пароль.

3.6 Діаграма класів

Зараз майже всі поля програмування застосовують об'єктно-орієнтований підхід. Перед початком розробки, щоб уникнути недбалих проблем, необхідно детально спланувати класи, їх поля та методи та реалізувати все відповідно до плану. Реалізацію діаграми класів можна детально розглянути на рисунку 3.6.

Клас «Модель» також потрібен для фільтрації і є частиною назви авто.

Клас «Картинка» зберігає назву, шлях та підпис до зображення. Цей клас має зв’язок з класом Авто один до багатьох.

Клас «оренда» цей клас містить в собі інформацію про замовлення, інформацію клієнта. Відношення класів багато до багатьох.

Клас «Користувач» мабуть найбільший клас. Поля, які він містить у сові – це особиста інформація про користувача. Ім’я, телефонний номер, електронна пошта. Також даний клас має і методи. Користувач може увійти до облікового запису, вийти з нього, додати іншого адміністратора у систему, якщо є потрібні права, подивитися дані клієнта, подивитися, сформувати або підтвердити замовлення, додати, редагувати або видалити авто.

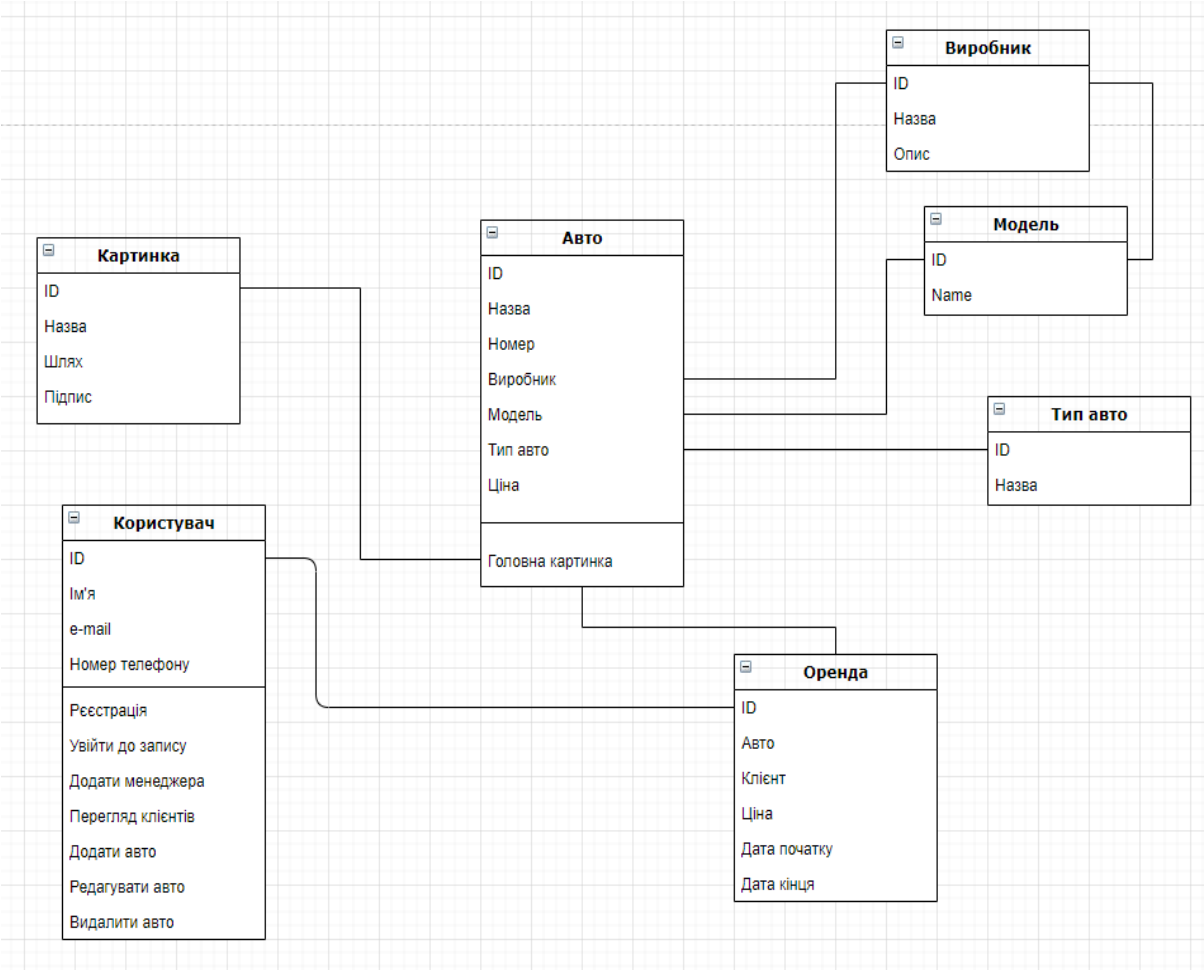

Рисунок 3.6 – Діаграма класів

На цьому етапі проєктування нашої програми вважається завершеним. Наступний етап передбачає програмну реалізацію описаних вище компонентів системи та їх тестування.

4 ПРИКЛАД РОБОТИ ДОДАТКА

Нижче наведено скриншоти web додатка, що працює у демо-режимі [7].

Перше, що побачить людина – це головну сторінку де будуть представленні всі наявні авто, які можна орендувати (див. рис. 4.1).

На картках авто можна побачити вигляд авто, його ціну в день, розмір бензобака, тип коробки, та кількість місць для людей.

Зліва знаходиться фільтр авто де можна вибрати авто за критеріями «Тип», «Місткість», та «Ціна».

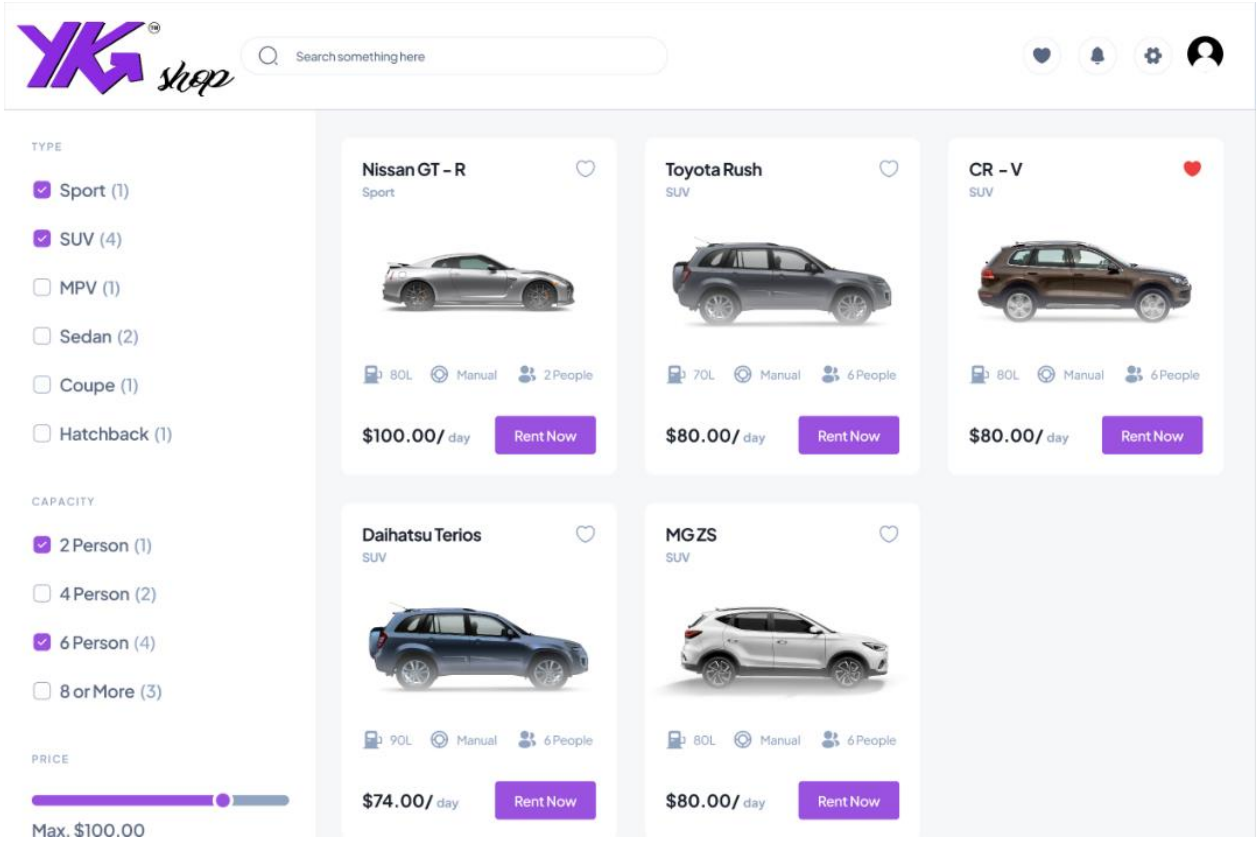

Рисунок 4.1 – Сторінка з вибраними фільтрами авто

Також сайт підтримує адаптивний інтерфейс, тому їм можна зручно користуватися на смартфоні (див. рис. 4.2) [8].

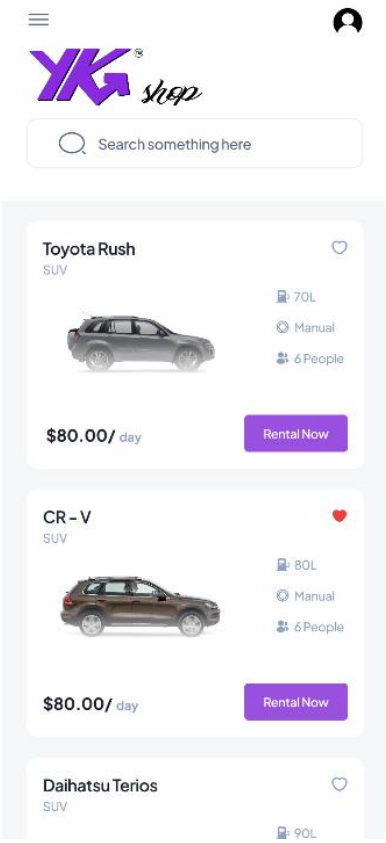

Рисунок 4.2 – Адаптивна сторінка

Після того як людина обрала авто, потрібно натиснути на кнопку «Rent Now». Після цього ми перейдемо на сторінку з детальним переглядом авто.

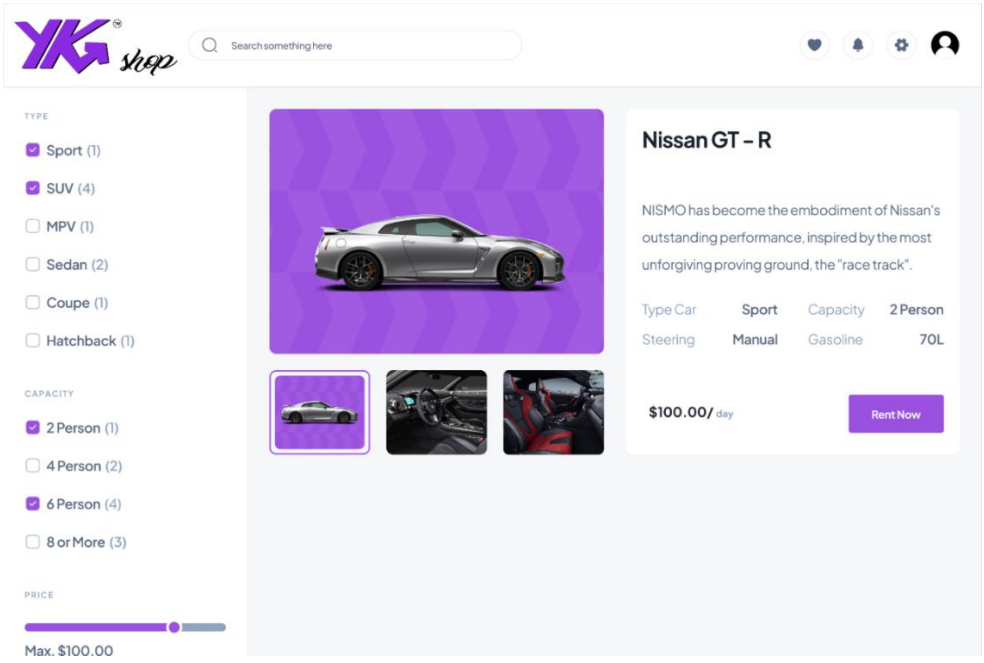

Рисунок 4.3 – Сторінка з ретельним переглядом авто

На сторінці можна переглянути фото авто, та його ретельніший опис. Після перегляду, для його оренди треба знову натиснути на кнопку «Rent Now».

Далі ми перейдемо на сторінку з замовленням для оренди (див. рис. 4.4). Там потрібно буде заповнити декілька форм з особистою інформацією.

З права ми бачимо підсумок ціни, якщо змінити кількість днів, то ціна збільшиться з його кількістю

У першій формі треба внести свою контактну інформацію.

У другій – інформацію про місце, дату та час оренди.

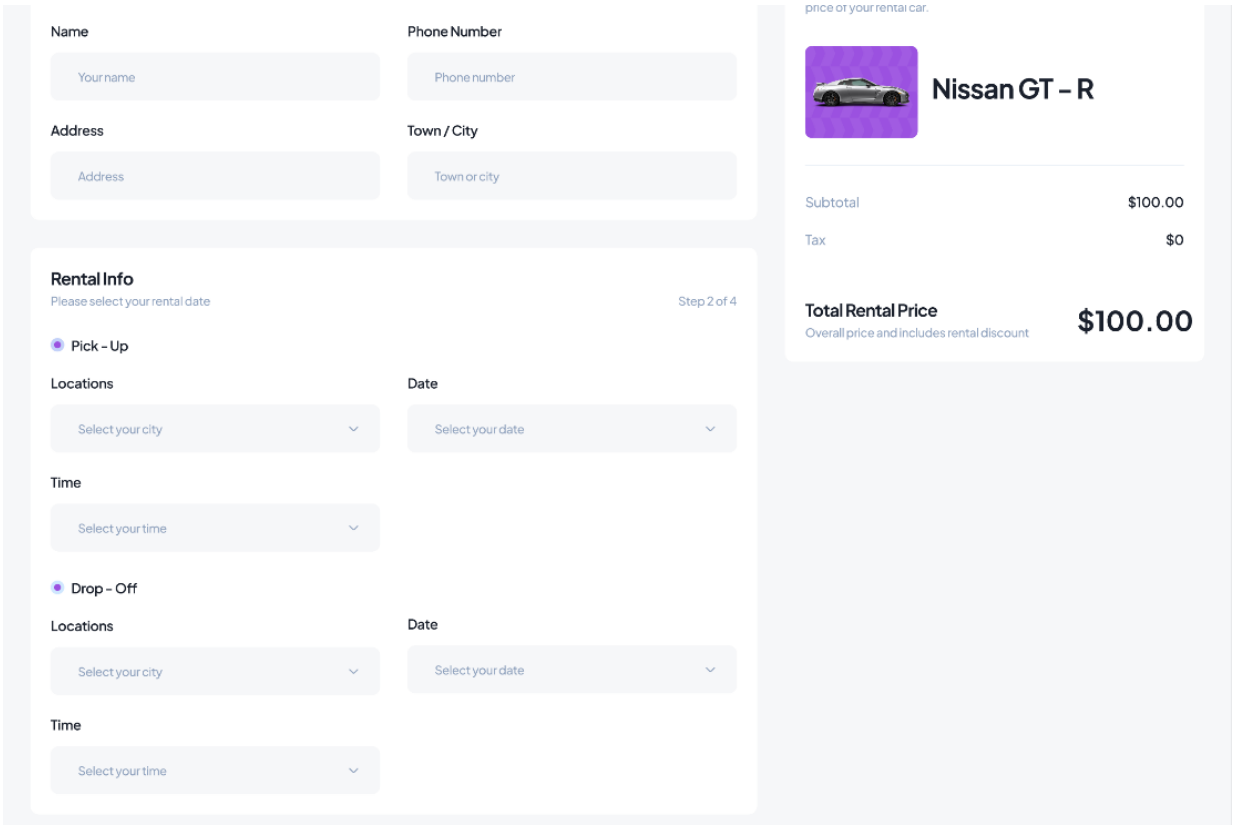

Рисунок 4.4 – Сторінка для оформлення оренди авто

У треті формі потрібно внести інформацію своєї банківської карти для оплати (див. рис. 4.5).

І остання четверта форма – це чекбокс для підписки на сайт для отримання різної інформації(рекламна розсилка, спеціальні пропозиції), вона не обов’язкова. Та обов’язкова зі згодою з політикою компанії.

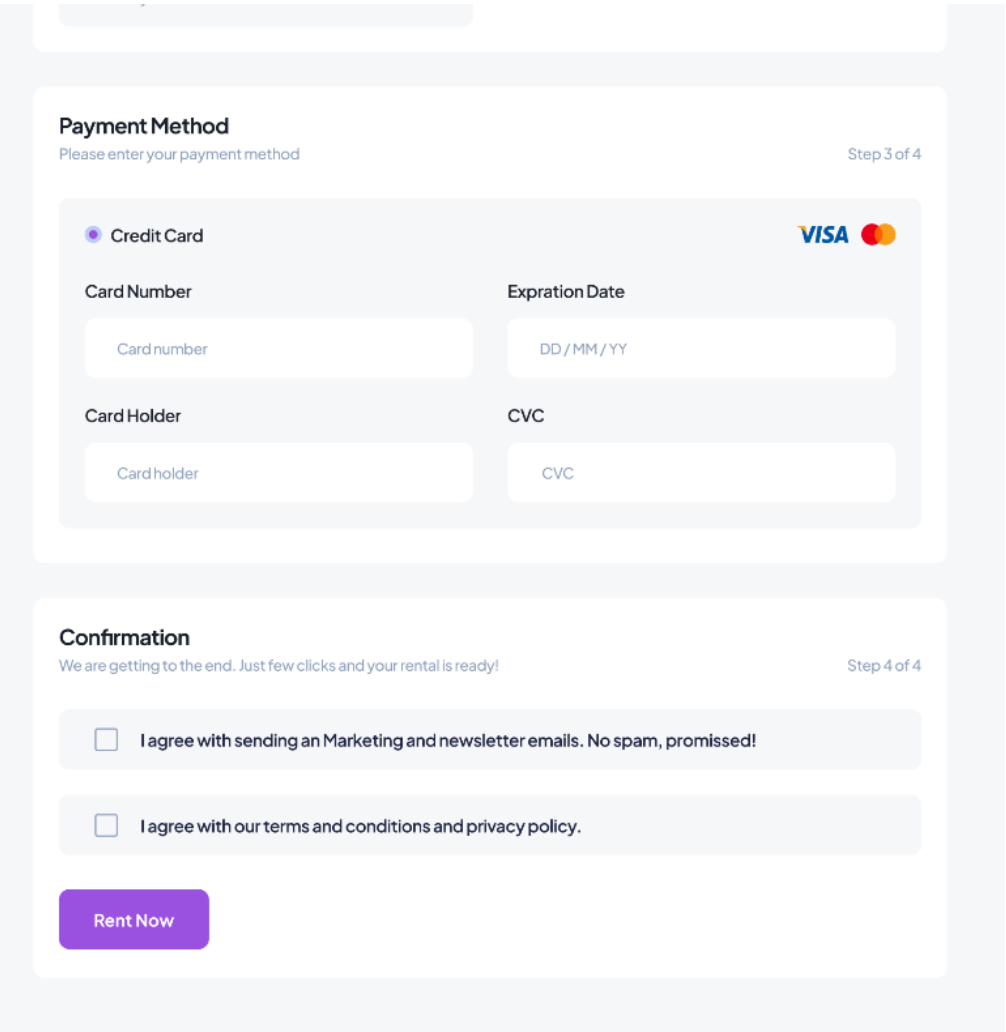

Рисунок 4.5 – Сторінка для введення реквізитів на оплату оренди

Після внесення всіх обов’язкових даних, користувач повинен натиснути кнопку підтвердження. Далі менеджер зв’яжеться через деякий час з людиною для підтвердження оренди.

ВИСНОВКИ

У роботі побудовано веб-сайт для оренди авто з використанням серверної частини на Node.js, та клієнтську частину з React. Проведено аналіз діяльності компанії, виявлення існуючих недоліків у сучасній технології управління послугами оренди авто.

На основі аналізу діючої технології управління сформувались цілі та цілі розробленої автоматизованої версії рішення даної проблеми, проведено порівняння з подібними системами для вирішення цієї проблеми, формалізовано розрахунок показників результатів, було обрано технологію проєктування та прийнято ряд дизайнерських рішень для технічної, інформаційної, програмної та технологічної підтримки.

Для проєктованої системи була побудована інформаційна модель, а також розроблена локальна система класифікації та кодування. Крім того, було прийнято рішення щодо того, як зберігати та впорядковувати дані.

Проаналізувавши всі проблеми, сформувавши всі вимоги до веб-сайту, описавши всі сутності, створивши базу даних, та реалізувавши клієнтський додаток, був створений веб-сайт для оренди авто.

Запустивши його на сервісах хостингу, та підключивши платіжні доданки можемо отримати працюючий web-сайт для використання.

Створений додаток може бути використано для подальшого розвитку бізнесу та його діджиталізації. Наведений приклад використання технологій може використатися в будь-якій сфері виробництва, маркетингу, продаж чи у розважальній сфері.

ПЕРЕЛІК ПОСИЛАНЬ

1. MardanA. JavaScript and Node FUNdamentals: A Collection of CoffeeScript, Node.js, Backbone.js Essential Basics. URL: https://www.goodreads.com/book/show/21076205- javascript- and- node- fundamentals (дата звернення: 17.06. 2022).
2. React быстро. Веб-приложения на React, JSX, Redux и GraphQL. URL: https://www.piter.com/collection/all/product/react-bystro-veb-prilozheniya-na-react-jsx-redux-i-graphql (дата звернення: 05.08.2022).
3. Чиннатамби К. Изучаем React. URL: https://grenka.ua/326415/izuchaem-react (дата звернення: 10.07.2022).
4. Distributed Systems with Node.js: Building Enterprise-Ready Backend Services. URL: https://www.oreilly.com/library/view/distributed- systems with /9781492077282/ (дата звернення: 25.09.2022).
5. Stefanov S. JavaScript Patterns: Build Better Applications with Coding and De sign Patterns. URL: https://www.oreilly.com/library/view/javascript- patterns/9 781449399115/ (дата звернення: 23.07.2022).
6. Doglio F. Pro REST API Development with Node.js. URL: https://www.goodr eads.com/book/show/25635366-pro-rest-api-development-with-node-js (дата звернення: 03.09.2022).
7. Duckett J. HTML & CSS: Design and Build Web Sites. URL: https://www.wil ey.com/en us/HTML+and+CSS%3A+Design+and+Build+Websites- p- 978111 8008188 (дата звернення: 08.10.2022).
8. Сидельников Г. Наглядный CSS. URL: https://www.piter.com/product/vizual nyy-css (дата звернення: 24.10.22).

ДОДАТОК А

Код з теки reducers файлу carReducers.js

```
import {
  FETCH_CARS_SUCCESS, FETCH_CARS_REQUEST,
FETCH_CARS_FAILURE,
} from './carTypes';

const initialState = {
  loading: false,
  cars: [],
  error: '',
};

const carReducer = (state = initialState, action) => {
  switch (action.type) {
    case FETCH_CARS_REQUEST:
      return {
        ...state,
        loading: true,
      };
    case FETCH_CARS_SUCCESS:
      return {
        ...state,
        loading: false,
        cars: action.payload,
        error: '',
      };
```

```
    case FETCH_CARS_FAILURE:
      return {
        ...state,
        loading: false,
        cars: [],
        error: action.payload,
      };
    default:
      return state;
  }
};

export default carReducer;
```