Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет *Автоматики та комп'ютеризованих технологій*
(повна назва)

Кафедра *Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки*
(повна назва

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти *другий (магістреський)*

Розроблення програмної частини автоматизованої системи процесів логістики складського приміщення

(тема)

Виконав:
студент *II* курсу, групи *КІТПВм-21-1*
Шило Н. Ю.
(прізвище, ініціали)
Спеціальність *151 – Автоматизація та комп'ютерно-інтегровані технології*
(код і повна назва спеціальності)
Тип програми *освітньо-професійна*
Освітня програма *Комп'.терно-інтегровані технологічні процеси і виробництва*
(повна назва освітньої програми)
Керівник *професор Филипенко О. І.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Невлюдов І. Ш.*
(підпис) (прізвище, ініціали)

2022 р.

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«15» грудня 2022 р. *Шило Н. Ю.*

Харківський національний університет радіоелектроніки

Факультет Автоматизація та комп'ютерно-інтегровані технології
Кафедра *Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки*

Рівень вищої освіти *другий (магістерський)*

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва)

Тип програми *освітньо-професійна*

Освітня програма *151 Комп'ютерно-інтегровані технологічні процеси і виробництва*
(шифр і назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри Невлюдов І.Ш.
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові *Шилу Назару Юрійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи: «*Розроблення програмної частини автоматизованої системи процесів логістики складського приміщення»*
затверджена наказом університету від *«07» листопада 2022 р. №1464 Ст*
2. Термін подання студентом роботи до екзаменаційної комісії *«07» грудня 2022 р.*
3. Вихідні дані до роботи: *Об'єкт дослідження – процес розробки автоматизованого пошуку товарів у складських приміщеннях. Предмет дослідження – програмне забезпечення для автоматизованої сисетми лояістичних процесів у складських приміщеннях. Предмет розробки – автоматизована система логістичних процесів у складському приміщенні. Технічне забезпечення – ПК. Перелік використовуваних програмних засобів: VS Code, JavaScript, TypeScript, React, MySQL.*

4. Перелік питань, що потрібно опрацювати в роботі: *Вступ. Огляд та аналіз існуючих автоматизованих ситсем логістичних процесів. Постановка мети та задач дослідження. Розроблення структурної схеми та алгоритму роботи автоматизованої системи логістичних процесів. Розроблення алгоритму роботи системи. Розробка програмного забезпечення та наведення інструкції для користування. Вибір мови програмування та середовища розробки. Аналіз мови програмування, додаткових бібліотек та систем управління бащами даних. Розробка програми. Аналіз результатів. Охорона праці. Висновки. Перелік джерел посилання.*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри): *демонстраційний матеріал, представлений у форматі презентації PowerPoint (*.ppt) на аркушах формату А4 (16 сторінок).*

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Отримання, аналіз завдання, уточнення плану*		
	роботи	*01.09.22*	*Виконано*
2	*Аналіз сучасних систем логістики*		
	у складських приміщеннях	*08.09.22*	*Виконано*
3	*Огляд сучасних систем логістики*	*15.09.22*	*Виконано*
4	*Аналіз програмних компонентів та бібліотек*	*29.09.22*	*Виконано*
5	*Розроблення програмного забезпечення задачі*	*13.10.22*	*Виконано*
6	*Проведення експериментальних досліджень*	*27.10.22*	*Виконано*
7	*Підготовка публікацій за результатами*		
	дослідження	*10.11.22*	*Виконано*
8	*Оформлення пояснювальної записки*	*24.11.22*	*Виконано*
9	*Подання закінченої роботи науковому керівникові*	*01.12.22*	*Виконано*
10	*Усунення зауважень наукового керівника*	*04.12.22*	*Виконано*
11	*Подання роботи на рецензування*	*05.12.22*	*Виконано*
12	*Підготовка презентації*	*08.12.22*	*Виконано*
13	*Попередній захист*	*11.12.22*	*Виконано*
14	*Подання роботи до екзаменаційної комісії*	*11.12.22*	*Виконано*

Дата видачі завдання _____

Студент _____ *Шило Н. Ю.*
(підпис)

Керівник роботи _____ *професор Филипенко О. І.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 90 с., 6 табл., 30 рис., 1 дод., 46 джерел.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, JAVASCRIPT, REACT, АВТОМАТИЗОВАНА СИСТЕМА, СИСТЕМА ЛОГІСТИКИ.

Об’єкт дослідження – автоматизована система пошуку товарів у складських приміщеннях.

Предмет дослідження – процес розроблення функціонального, математичного та програмного забезпечення автоматизованих систем пошуку товарів у складських приміщеннях.

Мета роботи – покращення та оптимізація автоматизованих логістичних процесів у логістичних приміщеннях.

Методи дослідження – аналіз, опис, дослідження, пояснення, класифікація.

У ході роботи було проаналізовано інсуючі логістичні системи, виконано опис автоматизованої системи та математичне моделювання процесу, розроблено алгоритмічно-програмного забезпечення автоматизованої системи, досліджено результати програмної розробки.

Як результат, було розроблено метод знаходження авто у складському приміщенні.

ABSTRACT

Explanatory note contain: 90 pages, 6 tables, 30 figures, 1appendice, 46 sources.

SOFTWARE, JAVASCRIPT, REACT, AUTOMATED SYSTEM, LOGISTICS SYSTEM.

The object of research is an automated system for searching for goods in warehouses.

The subject of research is the process of developing functional, mathematical and software automated systems for searching goods in warehouses.

The purpose of the work is to improve and optimize automated logistics processes in logistics facilities.

Research methods – analysis, description, research, explanation, classification.

In the course of the work, the existing logistics systems were analyzed, the description of the automated system and mathematical modeling of the process was performed, the algorithmic software of the automated system was developed, and the results of the software development were studied.

As a result, a method of finding a car in a warehouse was developed.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АС – автоматизована система;

БД – база даних;

КІТАМ – комп’ютерно-інтегровані технології, автоматизація та мехатроніка;

UI – user interface;

TCP – transfer control protocol;

IP – internet protocol.

ВСТУП

Сучасні тенденції розвитку економіки зумовлюють суттєве зростання потреб підприємств у складах, які забезпечують тимчасове зберігання запасів матеріальних ресурсів, незавершеного виробництва та готової продукції.

Автоматизовані складські системи не тільки виключають ручну працю, а й дозволяють економити складські площі, прискорювати складські операції і покращувати контроль за матеріально-технічними запасами, оскільки ЕОМ стежить за місцезнаходженням кожного виробу на складі. Ці системи називають також автоматизованими складами.

Об’єктом дослідження є автоматизована система пошуку товарів у складських приміщеннях.

Предметом дослідження є процес розроблення функціонального, математичного та програмного забезпечення автоматизованих систем пошуку товарів у складських приміщеннях.

Мета роботи – покращення та оптимізація автоматизованих логістичних процесів у логістичних приміщеннях.

Задачі, які потрібно потрібно реалізувати:

– проаналізувати інсуючі логістичні системи;

– виконати опис автоматизованої системи та математичне моделювання процесу;

– розробити алгоритмічно-програмне забезпечення автоматизованої системи;

– досліди результати результати програмної розробки.

Основними результатми роботи є розробка методу обрання авто на складі за допомогою різних програмних компонентів та інформаційних систем.

Кваліфікаційна робота оформлена згідно з рекомендаціями з підготовки і оформлення кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти [2] та вимогами ДСТУ 3008:2015 [3].

1 АНАЛІЗ ІСНУЮЧИХ СКЛАДСЬКИХ РОБОТІВ

1.1 Актуальність та використання

До управління складськими комплексами зараз висувають нові вимоги. Це пов’язано з глобальними економічними процесами, зі станом ринку, зі зміною поведінки споживачів. З кожним роком роботи на складах стають складнішими, набувають нових функцій. Інтернет речей, штучний інтелект, bigdata, блокчейн і інші інноваційні технології дозволяють ефективно вирішувати завдання, що постійно ускладнюються.

Ще не так давно роботизовані системи використовувалися в основному для обробки великих обсягів вантажів. Роботів впроваджували там, де було потрібно здійснювати повторювані дії з вантажами однакового розміру, розташованими в одному місці. З розвитком електронної комерції виникла потреба автоматизувати різні дії з різнорідними товарами й замовленнями невеликих обсягів. Роботи на складах зараз використовуються не тільки в прийманні і транспортуванні вантажів, але й у комплектації замовлень, у відправленні товарів споживачеві. Роботизація невеликих і середніх складів спочатку вважалася неекономічною. Зараз вона доводить свою ефективність: крім великих ритейлерів, промислових гігантів, транспортних і логістичних компаній, усе більше малих підприємств використовують автоматизовані системи управління складом. Попит на них постійно зростає. За даними консалтингової компанії Logistics IQ, до 2025 року обіг автоматизованих складів досягне 27 млрд доларів [6].

1.2 Основні типи й сфери застосування

Сучасних роботів, що використовуються для автоматизації складських процесів, поділяють на дві групи: промислові та колаборативні.

Промислові роботи – це програмовані машини, що замінюють ручну працю при складних повторюваних діях. Така техніка обладнана датчиками обліку даних у реальному часі. На складах цей тип представлений підіймальними механізмами та автоматичними транспортерами.

Колаборативні роботи (коботи) являють собою кооперацію людини і машини. Ці пристрої виконують певні дії разом з людиною. Одна з переваг коботов – можливість програмування, щоб техніка могла працювати автономно або під управлінням людини. У складській сфері коботи представлені маніпуляторами для переміщення вантажів і пакувальними машинами.

Роботів на складах застосовують для автоматизації наступних процесів.

Обслуговування стелажів. Навантаження й розвантаження продукції у місцях зберігання здійснюється за допомогою підіймально-транспортних засобів: палетних шатлів, штабелерів і маніпуляторів.

Транспортування вантажів. Конвеєри, конвеєрні системи для палет (рейкові, роликові, підвісні) або автоматично керовані транспортні засоби (AGV) переміщують вантажі з одного місця в інше у самому складі або між складом і виробництвом.

Комплектація замовлень. Колаборативні роботи на складах здійснюють автоматичну або напівавтоматичну комплектацію замовлень. До цього типу належать маніпулятори для обробки значних вантажів, пакувальні машини, що здатні розрахувати й підготувати матеріал для пакування продукції певного виду, різноманітні роботи-комплектувальники, а також механічні екзоскелети, що адаптуються до рухів носія і скорочують фізичні зусилля при виконанні рухів.

Альтернативний варіант поділу складських роботів за типами представлений в статті.

1.3 Склади майбутнього вже сьогодні – 5 найбільш інноваційних складів

Скорочення витрат на персонал, швидкість і точність виконання операцій, зниження навантаження на людину – основні аргументи на користь роботизації

складів. Як свідчить досвід сильних гравців ринку, автоматизовані системи поступово повністю витіснять працю людини. Далі – найцікавіші приклади роботизованих складів, де участь оператора вдалося звести до мінімуму [6].

1.3.1 Складські роботи Amazon

Найбільший у світі інтернет-магазин має у своєму розпорядженні понад 200 тисяч роботів, вони залучені на складах 180 країн. Розробкою автоматизованих систем для гіганта займається власна компанія AmazonRobotics (раніше KivaSystems), що перейшла у власність корпорації у 2012 році. Роботизований склад Amazon використовує системи для автоматичного транспортування товарів. По складу товари переміщуються за допомогою конвеєрів і навантажувачів, що керуються людиною. Зберігаються товари на переносних полицях. Після того, як продукт вноситься в базу даних, система направляє до нього безпілотного робота. Робот підіймає полку з товаром і транспортує її до оператора, який обирає потрібний продукт.

Складські роботи, зображені на рис. 1.1, переміщаються по комплексу, орієнтуючись за QR-кодами на підлозі. Датчики запобігають зіткненню роботів між собою. Роботи здатні самостійно заряджатися [6].

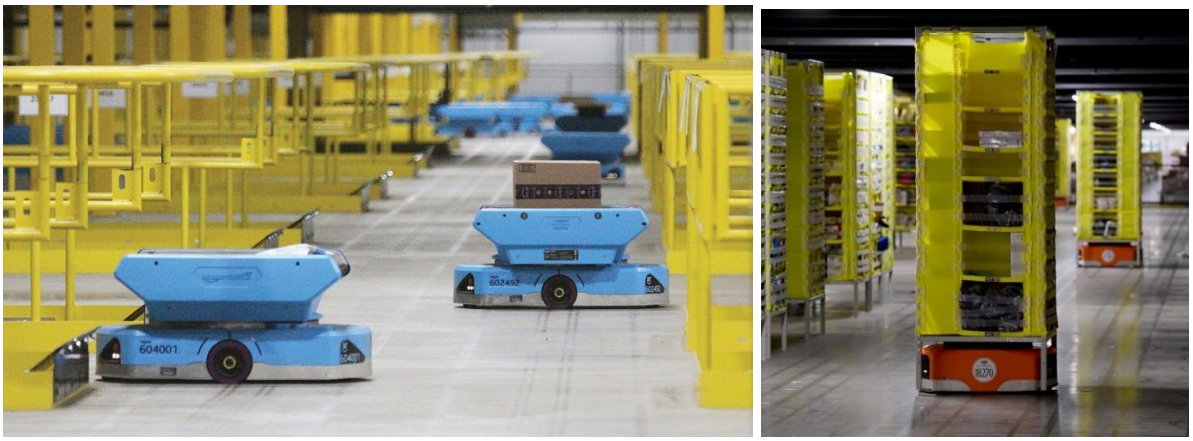


Рисунок 1.1 – Складські роботи Amazon

1.3.2 Роботи на складах від Cainiao

Cainiao, або China SmartLogistics Network, – китайська логістична компанія групи Alibaba, що доставляє товари по території Китаю за 24 години, а по всьому світу – за 72. Компанія володіє 30 складами загальною площею 1,7 млн $м^2$.

Склад у Хойяні провінції Гуандун використовує для переміщення вантажів безпілотних роботів, розроблених Quicktron Intelligent Technology. Маневрові роботи, зображені на рис. 1.2, здатні обертатися на 360 ° і переміщати вантажі вагою до 500 кг. Уникнути зіткнення дозволяє вбудована система лазерів. Комунікація з оператором і отримання завдань здійснюється через Wi-Fi. Якщо акумулятор розряджається, робот може поповнити запаси енергії автоматично. Як повідомляє компанія, безпілотні складські роботи дозволили скоротити працю персоналу на 70% [6].

Рисунок 1.2 – Роботи на складах від Cainiao

1.3.3 Роботизація складів Ocado

Ocado – найбільший онлайн-ритейлер продовольчих продуктів з Великобританії. Центр дистрибуції в Андовері щотижня обробляє 65 000 замовлень. Особливість автоматичної системи (рис. 1.3), в тому, що роботи пересуваються рейками над осередками з товаром уздовж спеціальної розмітки у

вигляді сітки. Їх пересування регулює система контролю трафіку на основі 4G, що дозволяє уникнути зіткнення. Роботи розвивають швидкість 4 м/с. Зарядка акумуляторів відбувається автоматично у спеціальних відсіках. Роботи привозять ящики з продуктами до станцій сортування, де згодом робот або людина формує замовлення. Порожні осередки роботи заповнюють новими продуктами [6].

Рисунок 1.3 – Роботи від Ocado

1.3.4 Склад Sagawa X-Frontier

Sagawa Global – одна з найбільших логістичних компаній Японії. Новий склад в Токіо Sagawa X-Frontier являє собою сучасний автоматизований комплекс, зображений на рис. 1.4. За допомогою армії роботів і систем штучного інтелекту компанія управляє складськими запасами, організовує простір в складських приміщеннях, транспортує вантажі між будівлями і навіть приносить замовлення безпосередньо співробітникам, які працюють на складі [6].

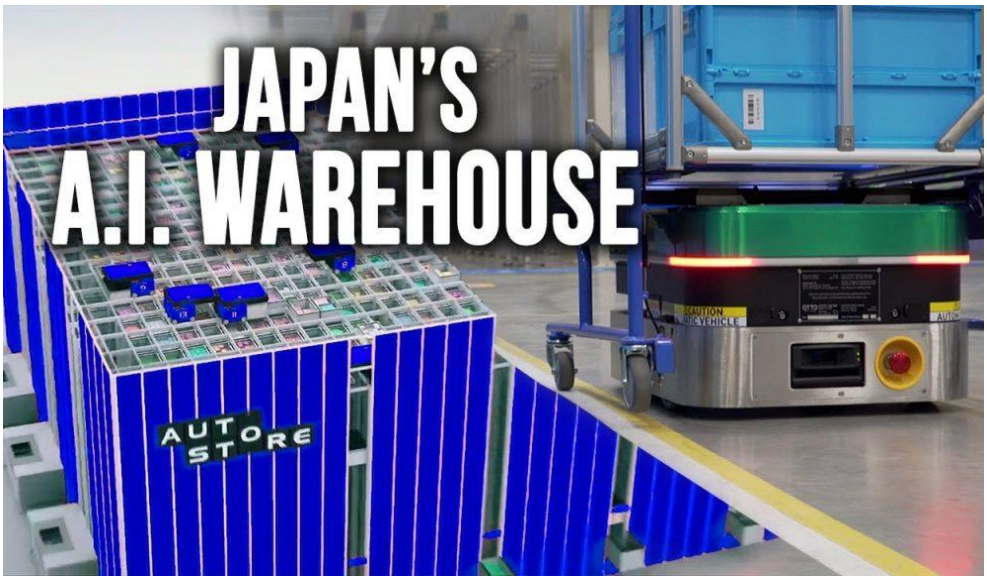


Рисунок 1.4 – Sagawa X-Frontier

1.3.5 Опис складських роботів DHL

Логістична компанія DHL з 2020 року використовує роботи свого партнера, американського розробника автономних мобільних роботів LocusRobotics. Так звані Locus-боти, зображені на рис. 1.5, вже застосовуються на десятках складів у Північній Америці. Роботи здатні самостійно пересуватися, знаходити необхідні товари й доставляти їх співробітникам. Таким чином компанія скорочує час виконання повторюваних або фізично складних завдань [6].

Рисунок 1.5 – Складські роботи DHL

Таблиця 1.1 Складські роботи на підприємствах

Компанія	Розташування складу	Автоматизація процесу	Особливості системи
Amazon	Філіали по всьому світу	Транспортування вантажів	Безпілотні роботи доставляються товари операторам. Управління рухом за допомогою QR-кодів на підлозі.
Cainiano	Хойян, Китай	Транспортування вантажів	Мобільні роботи-постачальники товарів. Управління через Wi-Fi.
Ocado	Андовер, Великобританія	Транспортування вантажів, комплектація замовлень	Переміщення за сіткою. 4G-технологія контролю трафіку. Доставляють товар оператору. Додають товар в осередках.
Sagawa X-Frontier	Токіо, Японія	Транспортування вантажів, комплектація замовлень	Безпілотні роботи, трафік регулюється за допомогою QR-кодів на підлозі.
DHL	Північна Америка	Транспортування вантажів, комплектація замовлень	Роботи доставляють товар співробітникам, які згодом вносять його за допомогою QR-кодів у систему.

1.4 Висновки до розділу

Виходячи з даних, наведених нижче, визначнимо основні завданя, які потрібно реалізувати в роботі:

– описати автоматизовану систему логістичних процесів, скласти її схему;

– зробити математичне моделювання вибору об’єкту за набором параметрів;

– розробити алгоритмічно-програмне забезпечення системи;

– дослідити отримані результати.

2 ОПИС АВТОМАТИЗОВАНОЇ СИСТЕМИ ТА МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ПРОЦЕСУ

2.1 Основні компоненти автоматизованої системи

Побудову систем характеризують за допомогою структур, що описують стійкі зв'язки між їх елементами. При описі АС використовують такі види структур, що відрізняються типами елементів і зв'язків між ними:

- функціональні (елементи – функції, завдання, процедури; зв'язки – інформаційні);
- технічні (елементи – пристрої, компоненти та комплекси; зв'язки – лінії і канали зв'язку);
- організаційні (елементи – колективи людей і окремі виконавці; зв'язки – інформаційні, підпорядкування та взаємодії);
- документальні (елементи – неподільні складові частини та документи АС; зв'язки – взаємодії, вхідність та підпорядкування);
- алгоритмічні (елементи – алгоритми; зв'язки – інформаційні);
- програмні (елементи – програмні модулі та вироби; зв'язки – керуючі);
- інформаційні (елементи – форми існування і подання інформації в системі; зв'язки – операції перетворення інформації в системі).

2.2 Схема автоматизованої системи процесів логістики складського приміщення

Беручи до уваги класичну систему автоматизованої системи, було складено схему автоматизованої системи процесів логістики складського приміщення, яка матиме наступний вигляд, зображений на рисункові 2.1. У ній підібрані саме ті комплекси й структури, які в сукупності реалізують основну задачу системи [1].

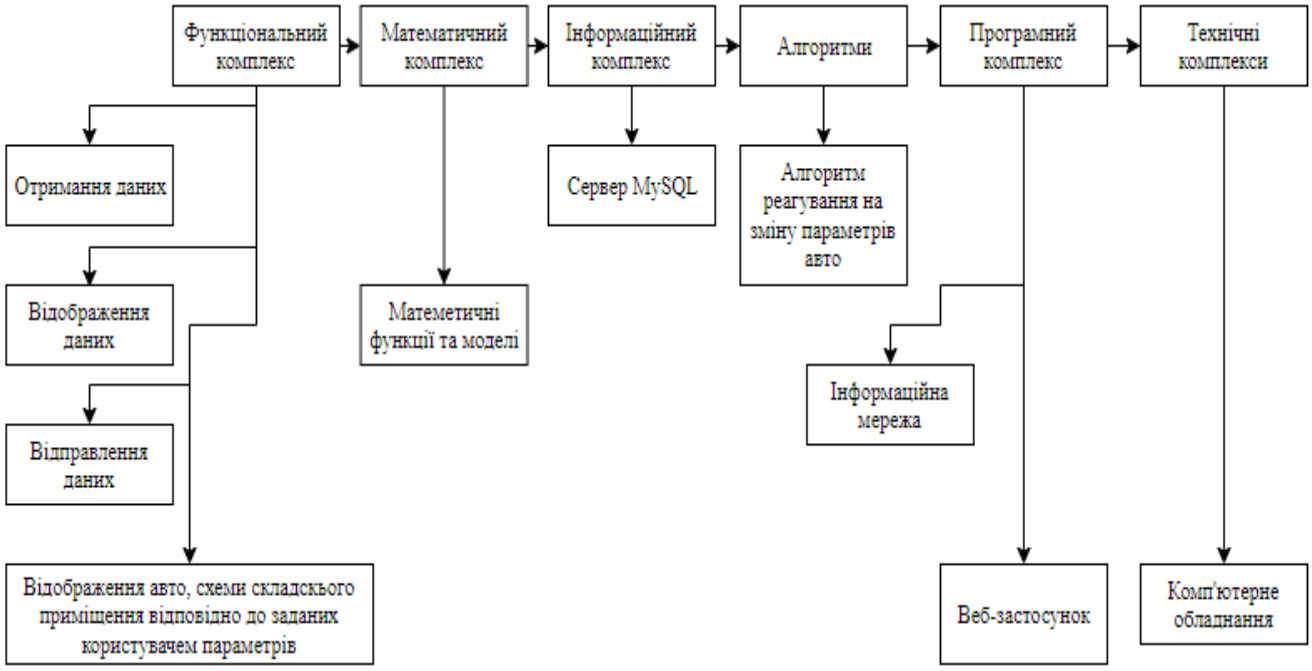


Рисунок 2.1 – Схема автоматизованої системи

Функціональний комплекс відповідає за основні задачі веб-застосунку, а саме: отримання, відображення та відправлення даних, авто, схеми складського приміщення за відповідними параметрами користувача.

Інформаційний комплекс представляє собою сховище даних, з якими працює система. У нашимо випадку це сервер MySQL [1].

За функціонування та злагоджену роботу програми відповідає алгоритмічний блок, основна функція якого – реагування на зміну вхідних параметрів [1].

Основною структурою програмного комплексу є розроблений веб-застосунок та інформаційна мережа [1].

2.3 Математична модель автоматизованої системи

Послідовність прийняття рішень зображена на рис. 2.2.

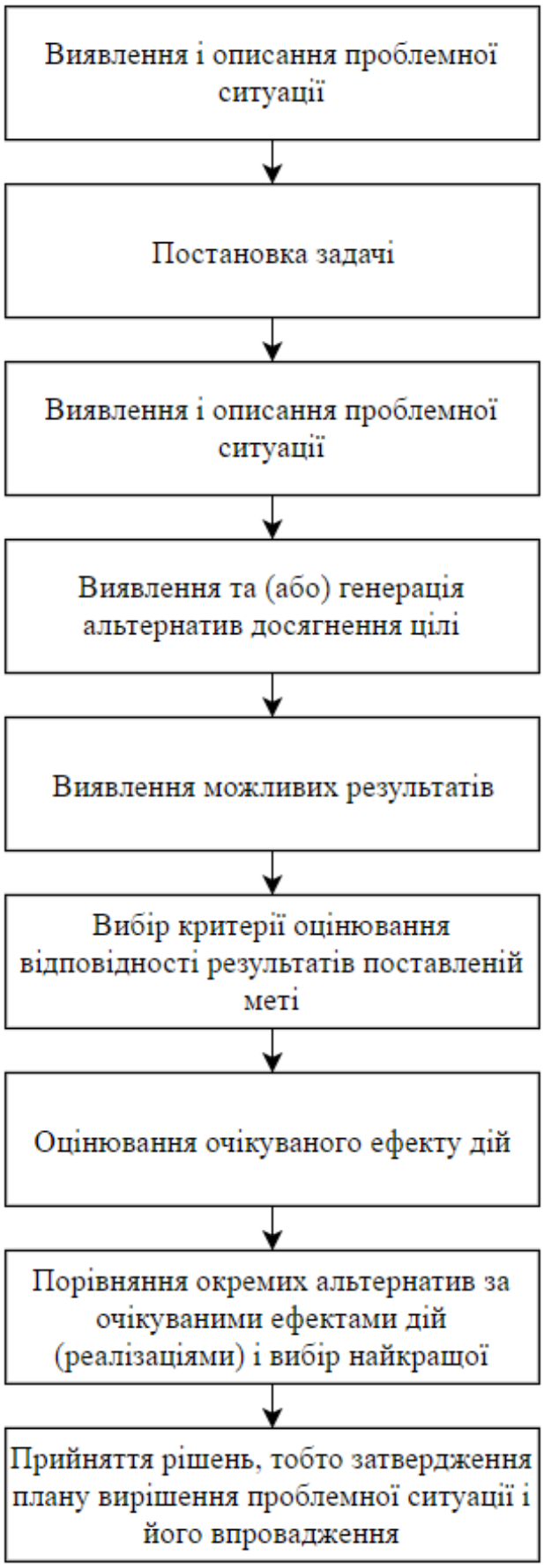


Рисунок 2.2 – Схема послідовності прийняття рішень

Важливо правильно обґрунтувати проблему. Наприклад, ідея японського професора Ісікави. Діаграма Ісікави — графічний спосіб дослідження та визначення найбільш суттєвих причинно-наслідкових взаємозв'язків між чинниками (факторами) та наслідками у досліджуваній ситуації чи проблемі [1].

Діаграма Ісікави для процесу вибору авто наведена на рис. 2.3.

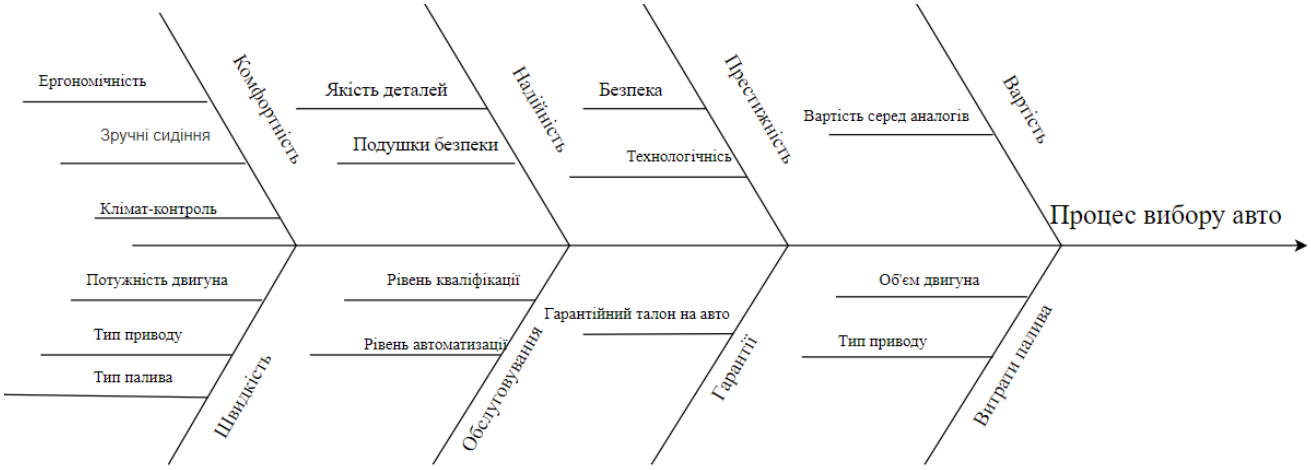


Рисунок 2.3 – Діагарма Ісікави

Прийняття рішень за умов багатокритеріальності та в умовах невизначеності за методомо аналітичної ієрархії (MAI).

Метод аналітичної ієрархії ґрунтується на принципах декомпозиції та синтезу, реалізація яких дає змогу зменшити кількість можливих помилок у процесі отримання інформації від експерта. За допомогою MAI отримують структуру у вигляді ієрархії, що дозволяє уникнути складних порівнянь, замінивши їх попарними. Цей метод дає змогу перевіряти послідовність (несуперечливість) тверджень експерта [1].

Метод аналітичної ієрархії (MAI) – систематична процедура, що ґрунтується на ієрархічному поданні елементів, які визначають суть проблеми. Проблему піддають декомпозиції на простіші складові з подальшим оцінюванням децидентом відносного ступеня взаємодії елементів отриманої ієрархічної структури. Метод будується на принципі ідентичності та декомпозиції, згідно з яким структурування проблем у вигляді ієрархії чи мережі містить процедури синтезу множинних

тверджень, оцінювання пріоритетності критеріїв та знаходження альтернативних рішень [1].

Послідовність етапів МАІ:

– формулювання проблеми, яку потрібно розв’язати;

– постановка проблеми загалом – включення її (якщо це необхідно) до великої системи, у якій є інші зацікавлені дійові особи (актори), розгляд їх ідей та бажаних результатів;

– ідентифікація критеріїв, за якими буде оцінено якість розв’язання проблеми;

– побудова ієрархії спільних критеріїв, окремих критеріїв, властивостей альтернатив і самих альтернатив. У проблемі з багатьма учасниками рівні можуть стосуватися навколишнього середовища акторів, їхніх цілей, політик і результатів, за допомогою яких буде одержано узагальнений результат (стан сфери дій). Щоб усунути неясності слід ретельно визначити кожен елемент в ієрархії;

– задання пріоритетів первинних критеріїв (сил) щодо їх впливу на генеральну мету;

– чітке формулювання питання для попарних порівнянь у кожній матриці. Звернути увагу на орієнтацію кожного питання (наприклад, вартість має зменшуватись, а ефективність збільшуватись);

– задання пріоритетів часткових критеріїв щодо загальних. Збір результатів попарних порівнянь;

– опрацювання зібраних даних згідно з алгоритмом МАІ для обчислення глобальних пріоритетів і глобальної узгодженості результатів;

– у разі вибору серед альтернатив – обрання тієї, що має найбільше значення глобального пріоритету. У разі розміщення ресурсів оцінювання вартості альтернативи, обчислення відношення ефективності до вартості та відповідний розподіл ресурсів: повністю або пропорційно. Якщо потрібно визначити пріоритети вартості – розподіляють ресурси пропорційно пріоритетам.

У МАІ порівнюють пари елементів задачі попарно щодо їх впливу (дії, ваги, інтенсивності) на спільну для них характеристику [1].

Матриця має вигляд:

$$A = \begin{pmatrix} 1 & a_{12} & a_{1n} \\ \frac{1}{a_{12}} & 1 & a_{2n} \\ \frac{1}{a_{1n}} & \frac{1}{a_{2n}} & 1 \end{pmatrix}.$$

Після отримання кількісних тверджень про пари (B_i, B_j) у числовому вигляді потрібно кожному елементові множини $B = \{B_1, B_2, \dots, B_n\}$ поставити у відповідність числові ваги чи пріоритети. У разі ієрархічного подання проблем матрицю складають для порівняння відносної важливості критеріїв другого рівня відносно загальної мети першого рівня (кореня ієрархії), а потім будують такі ж самі матриці попарних порівнянь наступного рівня відносно елементів попереднього [1].

Попарні порівняння виконують у шкалі відношень за бальною системою, яка наведена в таблиці 2.1.

Таблиця 2.1 – Попарні порівняння

Бал, k	Визначення
1	Однакова важливість
3	Помірна перевага
5	Суттєва перевага
7	Значна перевага
9	Дуже велика перевага
2, 4, 6, 8	Проміжні значення (використовують у перехідних ситуаціях)
$1/k$	Обернені величини

Головне завдання MAI – визначення глобальних пріоритетів альтернатив, тобто їх пріоритетів відносно кореня ієрархії. Як вихідні дані при цьому

використовують результати опитування експертів у вигляді матриць попарних порівнянь при всіх вузлах ієрархії, окрім листя – альтернатив.

Для ілюстрації розглянемо приклад. Замовник хоче вибрати автомобіль. Внаслідок аналізу було виявлено наступні критерії, які варто брати до уваги: престижність, вартість, питомі витрати пального, комфортність, надійність, максимальна швидкість, розміри, витрати на технічне обслуговування, гарантійні зобов'язання [1].

Подальший розгляд дає змогу обрати як «кандидатів» три моделі та подати задачу у вигляді ієрархії.

Первинну множину критеріїв після аналізу було звужено до таких суттєвих: Q_1 – комфортність; Q_2 – надійність; Q_3 – швидкість; Q_4 – вартість; Q_5 – престижність; Q_6 – обслуговування; Q_7 – гарантії; Q_8 – витрати пального. За допомогою опитування експертів побудовано таку матрицю попарних порівнянь для рівня 2 − критеріїв.

Ієрархічна структура задачі вибору автомобіля наведана на рис. 2.4.

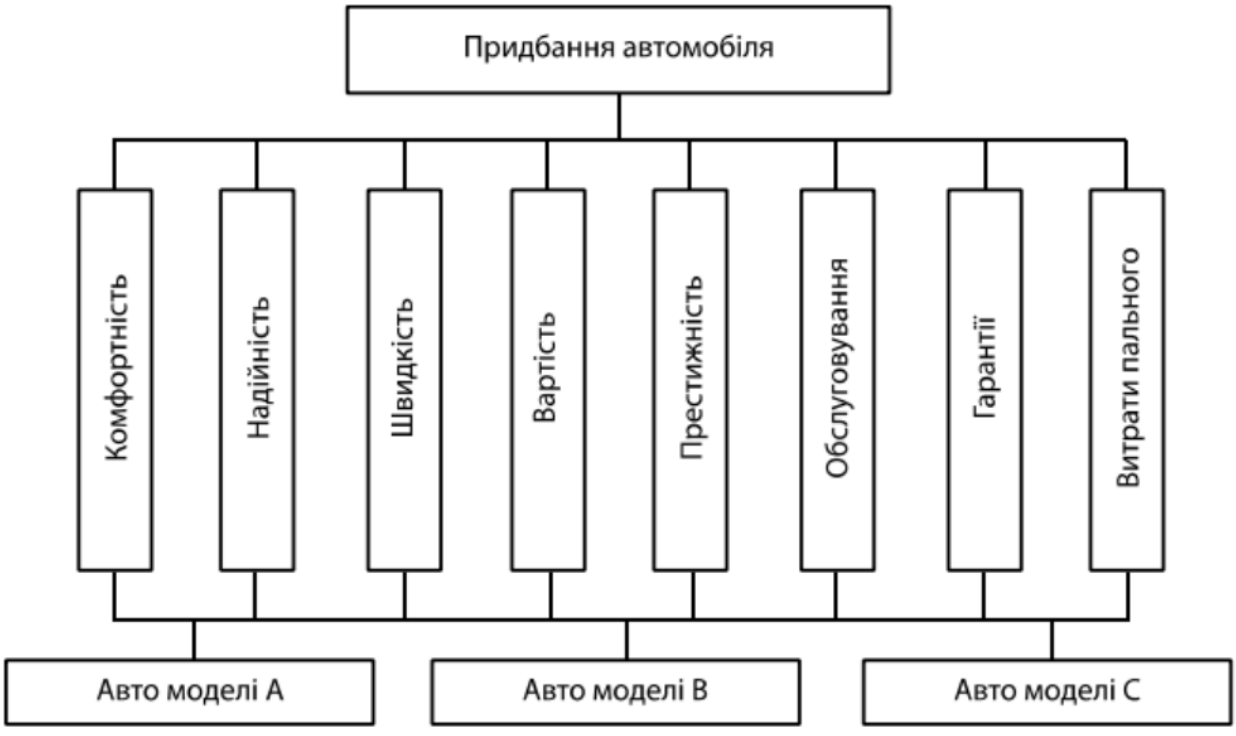


Рисунок 2.4 – Ієрархічна структура задачі вибору автомобіля

Таблиця 2.1 – Матриця попарних порівнянь критеріїв

Критерій	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8
Q_1	1	5	3	7	6	6	1/3	1/4
Q_2	1/5	1	1/3	5	3	3	1/5	1/7
Q_3	1/3	3	1	6	3	4	6	1/5
Q_4	1/7	1/5	1/6	1	1/3	1/4	1/7	1/8
Q_5	1/6	1/3	1/3	3	1	1/2	1/5	1/6
Q_6	1/6	1/3	1/4	4	2	1	1/5	1/6
Q_7	3	5	1/6	7	5	5	1	1/2
Q_8	4	7	5	8	6	6	2	1

Після цього, порівнюючи попарно три автомобілі (A, B, C) за кожним із критеріїв (рівень 3), отримано вісім матриць (для кожного із критеріїв) розміром 3×3 (за кількістю альтернатив до вибору).

Таблиця 2.2 – Матриці попарних порівнянь альтернатив A, B, C за критеріями

Q_1	A	B	C	Q_2	A	B	C	Q_3	A	B	C	Q_4	A	B	C
A	1	6	8	A	1	7	1/5	A	1	8	6	A	1	1	1
B	1/6	1	4	B	1/7	1	1/8	B	1/8	1	1/4	B	1	1	1
C	1/8	1/4	1	C	5	8	1	C	1/6	4	1	C	1	1	1
Q_5	A	B	C	Q_6	A	B	C	Q_7	A	B	C	Q_8	A	B	C
A	1	5	4	A	1	8	6	A	2	1/2	1/2	A	1	1/7	1/5
B	1/5	1	1/3	B	1/8	1	1/5	B	2	1	1	B	7	1	3
C	1/4	3	1	C	1/6	5	1	C	2	1	1	C	5	1/3	1

Оцінимо послідовності тверджень експерта та визначення локальних пріоритетів ієрархії.

Обчислимо локальні вектори пріоритетів, індекс узгодженості та відношення узгодженості для матриці попарних порівнянь критеріїв (таблиця 2.1) та матриць попарних порівнянь альтернатив A, B, C за критеріями Q_1–Q_8 (таблиця 2.2).

Наведемо нижче результати визначення вектора пріоритетів, індексу узгодженості та відношення узгодженості для матриці попарних порівнянь критеріїв (таблиця 2.3). Вектор пріоритетів отримано як результат обчислення головного власного вектора з подальшою його нормалізацією. Одержане значення відношення узгодженості завелике, але вважатимемо його прийнятним. У відносно великих матрицях ($n = 7,8,9$) досягнення високого рівня узгодженості проблематичне, але у цьому випадку слід зважати на збільшення ризику через неузгодженість.

Таблиця 2.3 – Визначення узгодженості матриці критеріїв

Критерій	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Вектор пріоритетів
Q_1	1	5	3	7	6	6	1/3	1/4	0,173
Q_2	1/5	1	1/3	5	3	3	1/5	1/7	0,054
Q_3	1/3	3	1	6	3	4	6	1/5	0,188
Q_4	1/7	1/5	1/6	1	1/3	1/4	1/7	1/8	0,018
Q_5	1/6	1/3	1/3	3	1	1/2	1/5	1/6	0,031
Q_6	1/6	1/3	1/4	4	2	1	1/5	1/6	0,036
Q_7	3	5	1/6	7	5	5	1	1/2	0,167
Q_8	4	7	5	8	6	6	2	1	0,333
								I_u	0,238
								I_0	0,169

Обчислимо відповідні характеристики для множини таблиць наступного рівня – оцінювання альтернатив.

Таблиця 2.4 – Обчислені значення локальних пріоритетів для альтернатив

Q_1	A	B	C	Вектор пріориетів	Q_2	A	B	C	Вектор пріориетів
A	1	6	8	0,754	A	1	7	1/5	0,233
B	1/6	1	4	0,181	B	1/7	1	1/8	0,005
C	1/8	1/4	1	0,065	C	5	8	1	0,713
			I_u	0,068				I_u	0,124
			I_0	0,117				I_0	0,213
Q_3	A	B	C	Вектор пріориетів	Q_4	A	B	C	Вектор пріориетів
A	1	8	6	0,745	A	1	1	1	0,333
B	1/8	1	1/4	0,065	B	1	1	1	0,333
C	1/6	4	1	0,181	C	1	1	1	0,333
			I_u	0,068				I_u	0
			I_0	0,117				I_0	0
Q_5	A	B	C	Вектор пріориетів	Q_6	A	B	C	Вектор пріориетів
A	1	5	4	0,674	A	1	8	6	0,747
B	1/5	1	1/3	0,101	B	1/8	1	1/5	0,06
C	1/4	3	1	0,226	C	1/6	5	1	0,193
			I_u	0,043				I_u	0,099
			I_0	0,074				I_0	0,17
Q_7	A	B	C	Вектор пріориетів	Q_8	A	B	C	Вектор пріориетів
A	2	1/2	1/2	0,2	A	1	1/7	1/5	0,072
B	2	1	1	0,4	B	7	1	3	0,65
C	2	1	1	0,4	C	5	1/3	1	0,278
			I_u	0				I_u	0,032
			I_0	0				I_0	0,056

Що стосується інтерпретації результатів, то в цьому прикладі витрати пального є найважливішими у виборі автомобіля, друге місце посідає комфортність, а третє – швидкість руху. У реальній ситуації за результатами такого аналізу несуттєві критерії можна було б відкинути і повторити опитування експерта, але із ілюстративною метою залишено всі.

Глобальні пріоритети обчислюють на наступному етапі MAI – ієрархічного синтезу. Із цією метою для виявлення складених, або глобальних, пріоритетів автомобілів виконаємо зворотній хід: із передостаннього рівня рухаємось до кореня ієрархії, збираючи вектори локальних пріоритетів у матриці та множачи їх на вектори локальних пріоритетів безпосередніх предків, доки не досягнемо кореня ієрархії. У наведеному прикладі ця процедура зводиться до збирання матриці з векторів локальних пріоритетів альтернатив за критеріями ($s = 2$):

$$P_1^{(1)} = \begin{pmatrix} 0{,}754 & 0{,}233 & 0{,}745 & 0{,}333 & 0{,}674 & 0{,}747 & 0{,}200 & 0{,}072 \\ 0{,}181 & 0{,}055 & 0{,}065 & 0{,}333 & 0{,}101 & 0{,}060 & 0{,}400 & 0{,}650 \\ 0{,}065 & 0{,}713 & 0{,}181 & 0{,}333 & 0{,}226 & 0{,}193 & 0{,}400 & 0{,}278 \end{pmatrix}.$$

Вектор локальних пріоритетів відносно кореня дерева

$$x_1^{(1)} = (0{,}173 \;\; 0{,}054 \;\; 0{,}188 \;\; 0{,}018 \;\; 0{,}031 \;\; 0{,}036 \;\; 0{,}167 \;\; 0{,}333)^T.$$

Знак транспортування вжито для зручності його запису. Тоді

$$p_1^{(1)} = P_1^{(1)} \times x_1^{(1)} = \begin{pmatrix} 0{,}396 \\ 0{,}341 \\ 0{,}263 \end{pmatrix}.$$

Оскільки оновлене значення, то роботу алгоритму на цьому завершено. Отже, за загальним показником (попри найгірші показники за критерієм витрат пального) обрано автомобіль А, тому що інші показники є ліпшими порівняно з конкурентами [1].

3 РОЗРОБКА АЛГОРИТМІЧНОГО-ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ

3.1 Клієнт-серверна архітектура

Клієнт – комп'ютер на стороні користувача, який відправляє запит до сервера для надання інформації або виконання певних дій.

Сервер – більш потужний комп'ютер або обладнання, призначене для вирішення певних завдань з виконання програмних кодів, виконання сервісних функцій за запитом клієнтів, надання користувачам доступу до певних ресурсів, зберігання інформації і баз даних.

Модель такої системи, зображеної на рис. 3.1, полягає в тому, що клієнт відправляє запит на сервер, де він обробляється, і готовий результат відправляється клієнтові. Сервер може обслуговувати кілька клієнтів одночасно. Якщо одночасно приходить більше одного запиту, то вони встановлюються в чергу і виконуються сервером послідовно. Іноді запити можуть мати пріоритети. Запити з більш високими пріоритетами повинні виконуватися раніше.

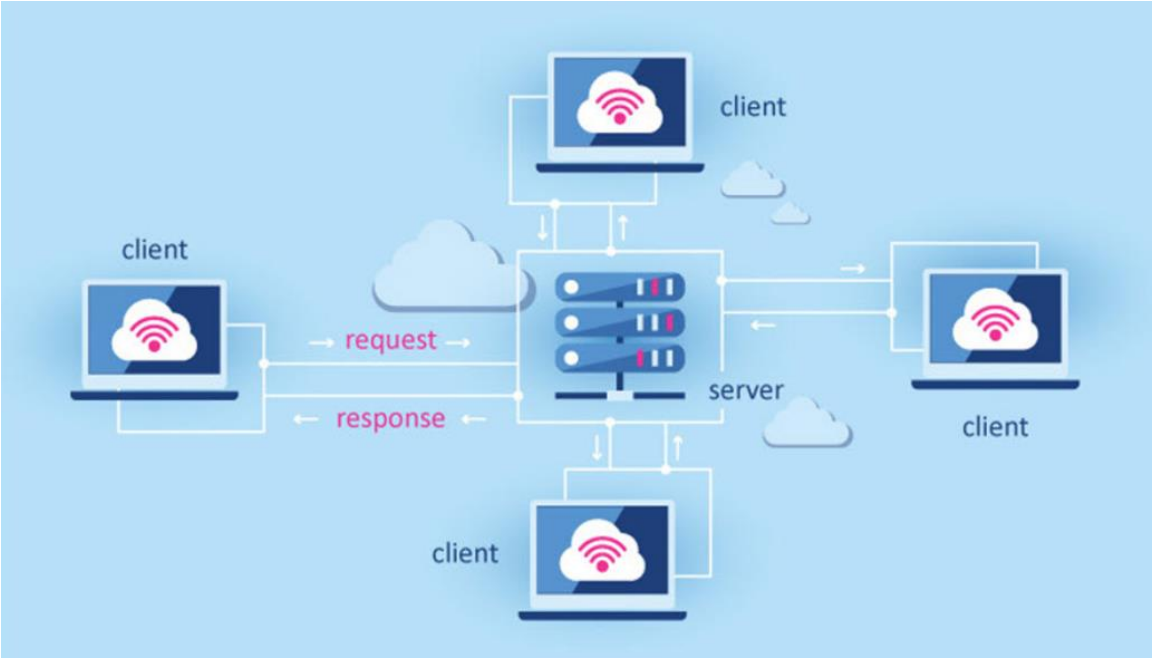


Рисунок 3.1 Клієнт-серверна архітектура

Функції, які реалізуються на сервері:

– зберігання, доступ, захист і резервне копіювання даних;

– обробка клієнтського запиту;

– відправлення результату (відповіді) клієнту.

Функції, які реалізуються на стороні клієнта:

– надання користувальницького інтерфейсу;

– формулювання запиту до сервера і його відправка;

– отримання результатів запиту і відправка додаткових команд (запитів на додавання, оновлення або видалення даних).

Архітектура клієнт-сервер визначає принципи спілкування між комп'ютерами, а правила і взаємодії визначені в протоколі.

Мережевий протокол – це набір правил, за якими відбувається взаємодія між комп'ютерами в мережі.

Мережеві протоколи:

– TCP/IP – набір (стек) протоколів передачі даних. TCP/IP – це позначення всієї мережі, яка працює на основі двох протоколів – TCP і IP;

– TCP (Transfer Control Protocol) – протокол, який служить для встановлення надійного з'єднання між двома пристроями, передачі інформації і підтвердження її отримання;

– IP (Internet Protocol) – інтернет протокол, який відповідає за правильність доставки повідомлень за певною адресою. При цьому дані розбиваються на пакети, які можуть доставлятися по-різному;

– MAC (Media Access Control) – протокол, за допомогою якого відбувається ідентифікація мережевих пристроїв. Всі пристрої, підключені до інтернету, мають свою унікальну MAC адресу;

– ICMP (Internet control message protocol) – протокол, який відповідає за обмін інформацією, але не використовується для передачі даних;

– UDP (User datagram protocol) – протокол, який керує передачею інформації, але інформація не проходить перевірку при отриманні. Даний протокол працює швидше, ніж TCP;

– HTTP (Hyper Text Transfer Protocol) – протокол передачі гіпертексту, на основі якого працюють всі сайти. Він запитує необхідні дані у віддаленій системи (веб-сторінки, файли);

– FTP (File Transfer Protocol) – протокол передачі файлів зі спеціального файлового сервера на комп'ютер користувача;

– SSH (Secure Shell) – протокол, який служить для забезпечення віддаленого керування системою по захищеному каналу;

– POP3 (Post Office Protocol) – стандартний протокол поштового з'єднання, який відповідає за доставку пошти;

– SMTP (Simple Mail Transfer Protocol) – протокол, який визначає правила для передачі пошти. Відповідає за повернення або підтвердження про доставку, оповіщення про помилку.

Існують концепції побудови системи клієнт-сервер.

Слабкий клієнт – потужний сервер. У такій моделі вся обробка інформації перенесена на сервер, а у клієнта права доступу суворо обмежені. Сервер відправляє відповідь, яка не вимагає додаткової обробки. Клієнт взаємодіє з користувачем: складає та відправляє запит, приймає результат і виводить інформацію на екран.

Сильний клієнт – концепція, в якій частина обробки інформації надається клієнтові. У такому випадку сервер виступає сховищем даних, а вся робота по обробці та подання інформації переноситься на комп'ютер клієнта.

Система (додаток), яка заснована на клієнт-серверній взаємодії, включає три основних компоненти: представлення даних, прикладний компонент, компонент управління ресурсами і їх зберігання.

Існують дворівнева і трирівнева клієнт-серверні архітектури.

Дворівнева архітектура, зображена на рис. 3.2, складається з двох вузлів:

– сервер, який відповідає за отримання запитів і відправку відповідей клієнту, використовуючи при цьому лише власні ресурси;

– клієнт, який представляє користувацький інтерфейс.

Принцип роботи полягає в тому, що сервер отримує запит, обробляє його і відповідає безпосередньо, без використання сторонніх ресурсів.

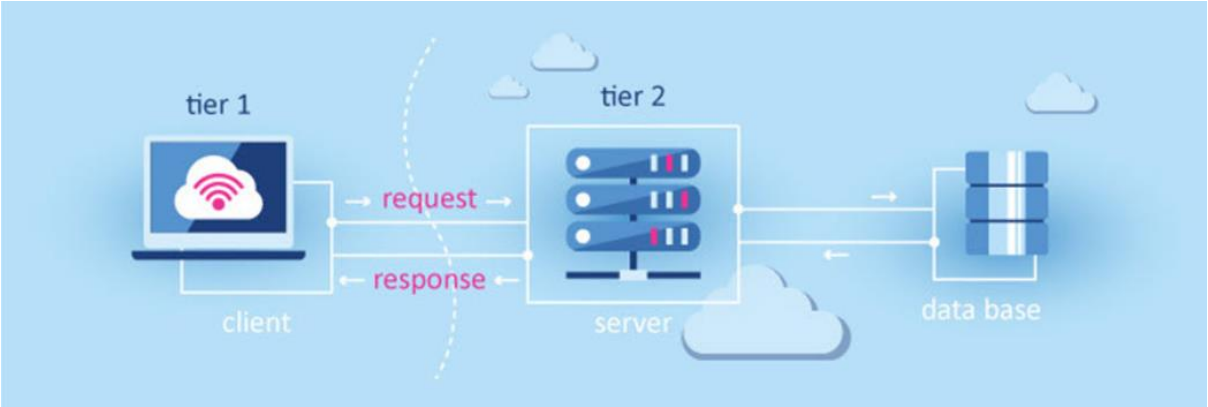


Рисунок 3.2 – Дворівнева клієнт-серверна архітектура

Трирівнева архітектура, зображена на рис. 3.3, складається з наступних компонентів:

– представлення даних – призначений для користувача інтерфейс;

– прикладний компонент – сервер додатків;

– керування ресурсами – сервер бази даних, який надає інформацію.

Принцип роботи полягає в тому, що декілька серверів обробляють запит клієнта. Розподіл операцій знижує навантаження на сервер.

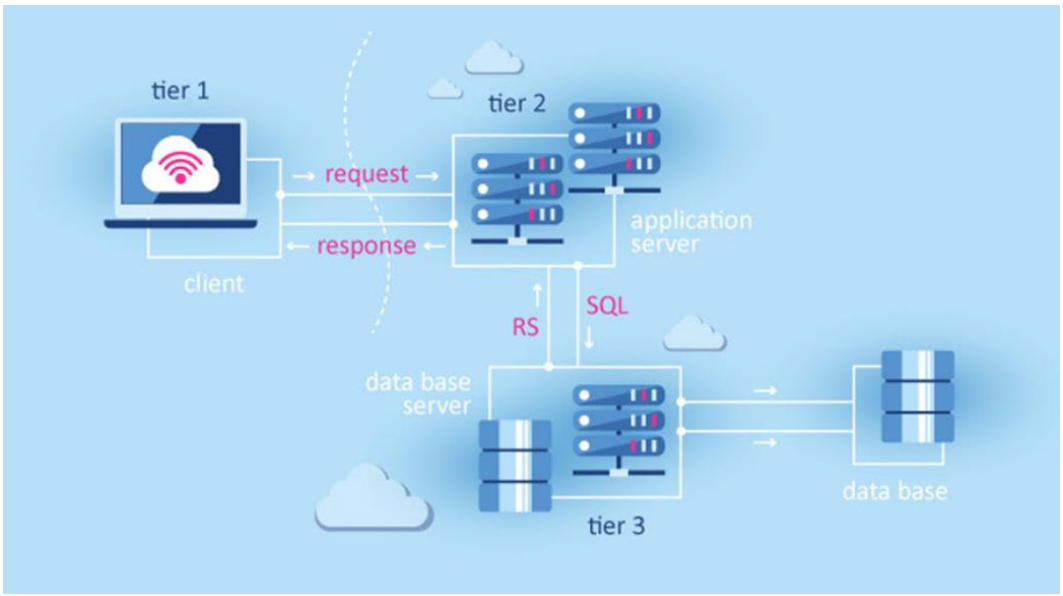


Рисунок 3.3 – Трьохрівнена клієнт-серверна архітектура

Трирівневу архітектуру можна розширити до багаторівневої способом встановлення додаткових серверів. Багаторівнева архітектура (рис. 3.4) дозволяє підвищити ефективність роботи інформаційної системи, а також оптимізувати розподіл її програмно-апаратних ресурсів.

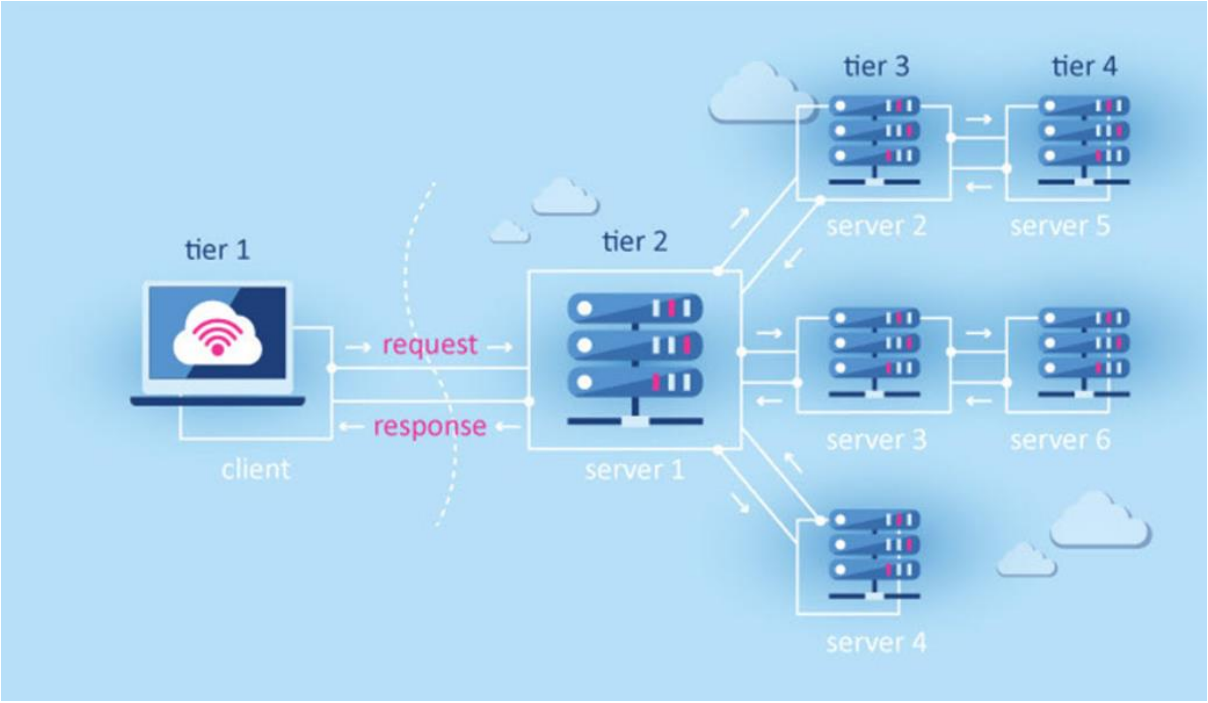


Рисунок 3.4 – Багаторівнена клієнт-серверна архітектура

Взаємодія клієнт-сервер дозволяє розділяти функціонал і обчислювальне навантаження між клієнтськими додатками (замовниками послуг) і серверними додатками (постачальниками послуг). Знання архітектури додатка дозволяє тестувальнику більш якісно провести функціональне, крос-браузерне тестування, тестування юзабіліті і швидкодії.

3.2 Бібліотека React

React – відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту вебсторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Meta (раніше Facebook) і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі вебзастосунки, 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, таких як AngularJS[7]. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови вебзастосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux.

В даний час React використовують Khan Academy [7], Netflix [8], Yahoo [9], Airbnb [12], Sony [13], Atlassian [14] та інші.

Бібліотеку створено Джорданом Волком (Jordan Walke), програмістом з Facebook. Автор працював над проектом під впливом XHP, фреймворку HTML для PHP [14]. 2011-го року реліз з'явився у новинах Facebook, за рік – у блозі Instagram [15]. Також фреймворк був представлений як проект з відкритим початковим кодом на конференції розробників JSConf US, що проходила у Сполучених Штатах у травні 2013 року. На конференції React.js Conf, влаштовану Фейсбуком у березні 2015-го, проект було представлено як відкрите програмне забезпечення.

React підтримує віртуальний DOM, а не покладається виключно на DOM браузера. Це дозволяє бібліотеці визначити, які частини DOM змінилися, порівняно (diff) зі збереженою версією віртуального DOM, і таким чином визначити, як найефективніше оновити DOM браузера [16][17]. Таким чином програміст працює зі сторінкою, вважаючи що вона оновлюється вся, але бібліотека самостійно вирішує які компоненти сторінки треба оновити.

Компоненти React зазвичай написані на JSX [18]. Код написаний на JSX компілюється у виклики методів бібліотеки React. Розробники можуть так само писати на чистому JavaScript. JSX нагадує іншу мову, яку створили у компанії Фейсбук для розширення PHP, XHP.

React використовують не лише для рендерингу HTML в браузері. Наприклад, Facebook має динамічні графіки які рендеряться в теги <canvas>,[19] Netflix та PayPal використовують ізоморфне завантаження для рендерингу ідентичного HTML на сервері та клієнті [20][21].

Методи життєвого циклу – це різні методи, які вбудовуються за допомогою ReactJS. Вони дозволяють розробнику обробляти дані в різних точках життєвого циклу програми React. Наприклад:

– shouldComponentUpdate – це метод життєвого циклу, який каже Javascript оновити компонент, використовуючи логічні змінні;

– componentWillMount – це метод життєвого циклу, який каже Javascript налаштувати певні дані перед монтуванням компонентів (вставлення у віртуальний DOM);

– componentDidMount – це метод життєвого циклу, подібний до компонента WillMount, за винятком того, що він працює після методу render, і може використовуватися для додавання JSON-даних, а також для визначення властивостей та станів;

– render є найважливішим методом життєвого циклу, необхідним у будь-якому компоненті. Метод render – це те, що з'єднується з JSX і відображати власний JSX.

Кілька елементів на одному рівні повинні бути загорнутими в один елемент контейнера, наприклад елемент , або повернутий як масив.

JSX надає ряд атрибутів елементів, призначених для відображення тих, що надаються у форматі HTML. Користувацькі атрибути також можуть бути передані компоненту. Всі атрибути будуть отримані компонентом як реквізит.

Вирази JavaScript можна використовувати в JSX (рис. 3.5) з фігурними дужками {}:

```
<h1> {10 + 1} </ h1>
```

Рисунок 3.5 – Фрагмент синтаксису JSX з виразом JavaScript

Приклад, наведений вище, відображатиметься так (рис. 3.6):

```
<h1>11</h1>
```

Рисунок 3.6 – Фрагмент синтаксису JSX

Вираз If–else не можуть бути використані всередині JSX, але замість них можуть використовуватися умовні вирази, це зображено на рис. 3.7.

```
{ i === 1 ? 'true' : 'false' }
```

Рисунок 3.7 – Фрагмент JSX з умовним виразом JavaScript

Фрагмент використання умовного виразу в класовому компоненті зображено на рис 3.8.

```
class App extends React.Component {
  render() {
    const i = 1;
    return (
      <div>
        <h1>{ i === 1 ? 'true' : 'false' }</h1>
      </div>
    );
  }
}
```

Рисунок 3.8 – Фрагмент використання умовного виразу в класовому компоненті

Функції та JSX можна використовувати в умовних виразах (3.9):

```
class App extends React.Component {
  render() {
    const sections = [1, 2, 3];
    return (
      <div>
        {
          sections.length > 0
            ? sections.map(n => <div>Section {n}</div>)
            : null
        }
      </div>
    );
  }
}
```

Рисунок 3.9 – Фрагмент використання умовного виразу та функції в класовому компоненті

Код, написаний у JSX, потребує перетворення за допомогою такого інструменту, як Babel, для того, щоб його могли зрозуміти веббраузери. Ця обробка, як правило, виконується під час процесу збірки, перш ніж програма буде запущена.

3.3 Платформа Node.js

Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Засновником платформи є Раян Дал (Ryan Dahl). Якщо раніше JavaScript застосовувався для обробки даних в браузері користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їхнього виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників.

Node.js має наступні властивості:

– асинхронна одно-нитева модель виконання запитів;

– неблокуючий ввід/вивід;

– система модулів CommonJS;

– рушій JavaScript Google V8.

Для керування модулями використовується пакетний менеджер npm (node package manager).

Node.js був спочатку написаний Раяном Далом у 2009 році [20] приблизно через тринадцять років після появи першого серверного середовища JavaScript, LiveWire Pro Web від Netscape [21]. Первісний випуск підтримував лише Linux та Mac OS X. Його розробку та обслуговування очолював Дал, а згодом і компанія Joyent [22].

Дал розкритикував обмежені можливості найпопулярнішого вебсервера 2009 року, Apache HTTP Server, обробляти безліч одночасних з'єднань (до 10 000 і більше) та найпоширеніший спосіб створення коду (послідовне програмування), коли код або блокував весь процес або залучає кілька стеків виконання у разі одночасного з'єднання [23].

Дал продемонстрував проєкт на першій європейській JSConf 8 листопада 2009 року [24][26][26]. Node.js поєднував у собі JavaScript-рушій Google V8, цикл обробки подій на основі C-бібліотек, і низькорівневий API вводу-виводу [27].

У січні 2010 року для середовища Node.js був введений менеджер пакетів під назвою npm [28]. Менеджер пакетів полегшує програмістам публікацію та обмін сирцевим кодом бібліотек Node.js і призначений для спрощення встановлення, оновлення та видалення бібліотек [29].

У червні 2011 року Microsoft та Joyent реалізували Windows-версію Node.js [30]. Перше складання Node.js, що підтримує Windows, випущене в липні 2011 року.

У січні 2012 року Дал відійшов у сторону, заохочуючи колегу і творця npm Ісаака Шлютера (Isaac Schlueter) до управління проєктом [31]. У січні 2014 року Шлютер оголосив, що проєкт буде очолювати Тімоті Фонтен (Timothy J. Fontaine) [32].

У грудні 2014 року Федір Індутний, незадоволений надто консервативним циклом оновлень, випустив io.js, форк Node.js. Через внутрішній конфлікт щодо управління Joyent, Io.js був створений як відкрита альтернатива управління з окремим технічним комітетом [33][34]. На відміну від Node.js [35], автори

планували постійно оновлювати io.js з найсвіжішими випусками JavaScript-рушія V8 [36].

У лютому 2015 року було оголошено про намір створити нейтральний фонд Node.js. До червня 2015 року спільноти Node.js та io.js проголосували за спільну роботу в рамках Node.js Foundation [37].

У вересні 2015 року Node.js v0.12 та io.js v3.3 були об'єднані разом у Node v4.0 [38]. Це об'єднання принесло можливості V8 ES6 в Node.js і довготривалий цикл випуску підтримки [39]. Станом на 2016 рік вебсайт io.js рекомендував розробникам перейти на Node.js і не планувати подальших випусків io.js через злиття [40], нині той домен переадресовує запити на офіційний сайт nodejs.org.

Платформа Node.js призначена для виконання високопродуктивних мережевих застосунків, написаних мовою програмування JavaScript. Платформа окрім роботи із серверними скриптами для веб-запитів, також використовується для створення клієнтських та серверних програм.

В платформі використовується розроблений компанією Google рушій V8.

Для забезпечення обробки великої кількості паралельних запитів у Node.js використовується асинхронна модель запуску коду, заснована на обробці подій в неблокуючому режимі та визначенні обробників зворотніх викликів (callback). Як способи мультиплексування з'єднань підтримується epoll, kqueue, /dev/poll і select. Для мультиплексування з'єднань використовується бібліотека libuv, для створення пулу нитей (thread pool) задіяна бібліотека libeio, для виконання DNS-запитів у неблокуючому режимі інтегрований c-ares. Всі системні виклики, що спричиняють блокування, виконуються всередині пулу потоків і потім, як і обробники сигналів, передають результат своєї роботи назад через неіменовані канали (pipe).

За своєю суттю Node.js схожий на фреймворки Perl AnyEvent, Ruby Event Machine і Python Twisted, але цикл обробки подій (event loop) у Node.js прихований від розробника і нагадує обробку подій у веб застосунку, що працює в браузері. При написанні програм для Node.js необхідно враховувати специфіку подієво-орієнтованого програмування (рис. 3.10).

```
var result = db.query ("select .. ");
```

Рисунок 3.10 – Фрагмент синхронного запиту до бази даних

з очікуванням завершення роботи і наступною обробкою результатів, в Node.js використовує принцип асинхронного виконання, тобто код трансформується, як зображено на рисункові 3.11.

```
db.query ("select .. ", function (result) { /* обробка результату */ });
```

Риисунок 3.11 – Фрагмент асинхронного запиту до бази даних

При цьому управління миттєво перейде до коду який слідує після виклику функції db.query, а результат запиту буде оброблений як тільки будуть оброблені дані. Жодна функція в Node.js не повинна безпосередньо виконувати (блокуючі) операції вводу/виводу – для отримання даних з диска, від іншого процесу або з мережі потрібна установка callback-обробника.

Для розширення функціональності застосунків на базі Node.js підготовлена велика колекція модулів, в якій можна знайти реалізацію HTTP, SMTP, XMPP, DNS, FTP, IMAP, POP3 серверів і клієнтів, модулі для інтеграції з різними вебфреймворками, обробники WebSocket і AJAX, конектори до СКБД (MySQL, PostgreSQL, SQLite, MongoDB), шаблонізатори, CSS-рушії, реалізації криптоалгоритмів і систем авторизації (наприклад, OAuth), XML-парсери.

Приклад програми, що запускає вебсервер, виводить в консоль повідомлення, та на кожен HTTP запит відповідає повідомленням «Hello World» (рис. 3.12).

```
var http = require('http'); // Завантажуємо модуль http

// Створюємо web-сервер і вказуємо функцію обробки запиту
var server = http.createServer(function (req, res) {
    console.log('Початок обробки запиту');
    // Передаємо код відповіді і заголовки
    res.writeHead(200, {
        'Content-Type': 'text/plain; charset=UTF-8'
    });
    res.end('Hello world!');
});

// Запускаємо web-сервер
server.listen(1991, "127.0.0.1", function () {
    console.log('Сервер запущено за адресою http://127.0.0.1:1991/');
});
```

Рисунок 3.12 – Фрагмент коду просто сервера Node.js

3.4 База даних

База даних (англ. database) – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування (за стандартом ISO/IEC 2382:2015 [41]). В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Таким чином, сучасна база даних, крім самих даних, містить їх опис та може містити засоби для їх обробки.

В загальному випадку базою даних можна вважати будь-який впорядкований набір даних. Наприклад, паперову картотеку з формулярами про працівників підприємства у відділі кадрів. Але дана стаття зосереджена на використанні баз даних в інформаційних системах. На даний час застосунки для роботи з базами даних є одними з найпоширеніших прикладних програм.

Бази даних класифікують за різними критеріями.

За моделлю організації даних розрізняють такі бази даних:

– ієрархічна. Ієрархічна база даних може бути представлена як дерево, що складається з об'єктів різних рівнів. Між об'єктами існують зв'язки типу «предок-

нащадок». При цьому можлива ситуація, коли об'єкт не має нащадків або має їх декілька, тоді як у об'єкта-нащадка обов'язково тільки один предок;

– мережна. Така база даних подібна до ієрархічної, за винятком того, що кожен об'єкт може мати більше одного предка;

– реляційна. Реляційна база даних зберігає дані у вигляді таблиць. Найвживаніші СКБД використовують реляційну модель даних;

– об'єктно-орієнтована. У базі даних цього виду дані оформляють у вигляді моделей об'єктів.

За розміщенням даних виділяють такі види баз:

– локальна, або централізована. Така база даних підтримується на одному комп'ютері;

– розподілена. Частини такої бази даних розміщують на різних комп'ютерах мережі.

За технологією фізичного зберігання виділяють:

– БД у вторинній пам'яті (традиційні);

– БД в оперативній пам'яті (in-memory database);

– БД у третинній пам'яті (tertiary database).

Структуровані та неструктуровані БД.

Структуровані БД використовують структури даних, тобто структурований опис типу фактів за допомогою схеми даних, більш відомої як модель даних. Модель даних описує об'єкти та взаємовідношення між ними. Існує декілька моделей (чи типів) баз даних, основні: ієрархічна, мережна та реляційна.

До неструктурованих БД належать повнотекстові бази даних, які містять неструктуровані тексти статей чи книг у формі, що дозволяє здійснювати швидкий пошук (наприклад, як Вікіпедія).

При роботі з базами даних використовують мови спеціального призначення:

Мова визначення даних (Data definition language, DDL) – це мова, яка описує дані та структури даних, а також визначає взаємозв'язки між ними (за стандартом ISO/IEC 2382:2015 [42]).

Мова маніпулювання даними (Data manipulation language, DML) – це мова, яку підтримує СКБД і яка забезпечує виконання операцій отримання, додавання, зміни та видалення даних (за стандартом ISO/IEC 2382:201524]).

Мова запитів (Query language) – це мова для користувачів, яка забезпечує отримання та оброблення даних у базі даних (за стандартом ISO/IEC 2382:2015 [42]).

При роботі з реляційними базами даних використовують мову структурних запитів SQL (Structured Query Language), яка поєднує всі три функції (визначення даних, модифікація даних та формування вибірок). Мова SQL стандартизована ANSI та ISO: починаючи з 1986 року, регулярно виходять поновлені стандарти. Слід зауважити, що кожна сучасна СКБД (MySQL, PostgreSQL, Microsoft SQL Server та інші) підтримує свою власну видозміну SQL, так що SQL-запит для однієї СКБД може не працювати в середовищі іншої. Але головні принципи формування SQL-запитів та їх структура однакові та відповідають стандартам ANSI/ISO. При необхідності виконання якоїсь операції над даними клієнт формує лінгвістичну конструкцію мовою SQL, яку називають SQL-запитом, і надсилає її до СКБД. СКБД опрацьовує запит, і результат його виконання (наприклад, вибірку даних) повертає клієнту. Мова, якою оперує СКБД, також може містити засоби для

– конфігурування СКБД;

– модифікації, форматування даних та розрахунків;

– формування обмежень даних.

3.5 Розробка алгоритмічно-програмного забезпечення

Алгоритми відображення та керування інтерфейсом автоматизованої логістично системи написані програмно мовою TypeScript на базі фреймворку React. Даний фреймворк, як описано вище, виконує функцію «рендерингу» даних, їх оновлення та зміни.

Вхідною точкою нашого програмного застосунку є компонент App, логіка якого зображена на рисункові 3.14.

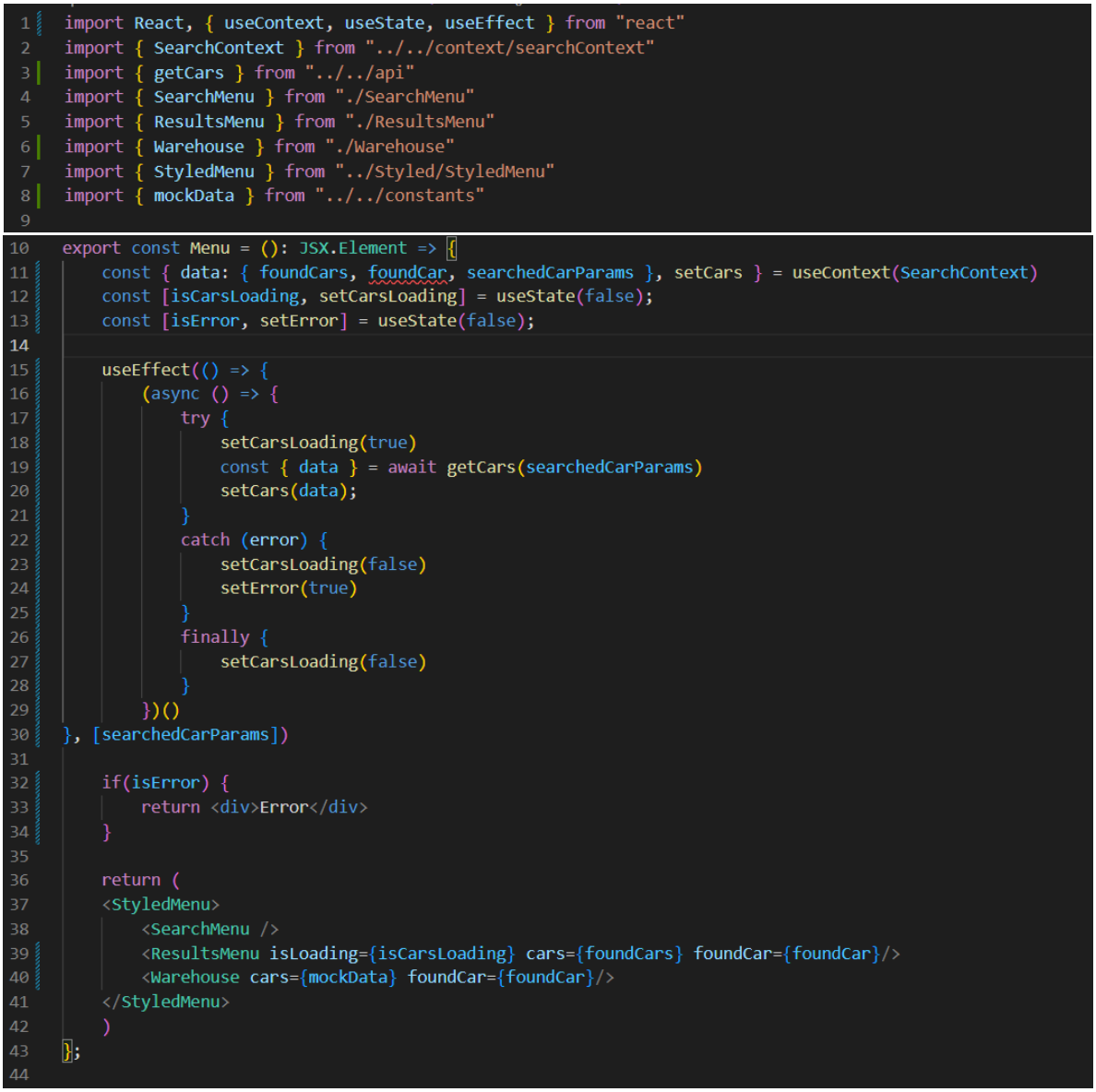
```
1   import * as React from 'react';
2   import type { RouteObject } from 'react-router-dom';
3   import { useRoutes } from 'react-router-dom';
4   import { DefaultErrorPage } from './components/DefaultErrorPage';
5   import { Layout } from './components/Layout';
6   import { Menu } from './components/Menu';
7
8   export const App = (): React.ReactElement => {
9     let routes: RouteObject[] = [
10      {
11        path: "/",
12        element: <Layout />,
13        children: [
14          { index: true, element: <Menu /> },
15          { path: "*", element: <DefaultErrorPage /> }
16        ],
17      },
18    ]
19
20    let element = useRoutes(routes)
21
22    return (
23      <div>
24        <main>
25          {element}
26        </main>
27      </div>
28    );
29  };
30
```

Рисунок 3.14 – Компонент App

Розберемо програмний код даного компоненту детальніше. З першого по восьмий рядок ми імпортуємо саму бібліотеку React, основні функції навігації браузера, а саме useRoutes та RoutesObject. Далі імпортується компонент дефолтної сторінки відображення помилки DefaultErrorPage, за ним – компоненти Menu та Layout.

Починаючи з 8-го рядка ми описуємо головний компонент нашої програми. Спершу створюється масив з відповідними посиланнями, які будуть оброблятися браузером. Відповідно цей об’єкт розташований в тілі нашої HTML-розмітки, а саме всередині тегу main (рядки 23-26).

Розглянемо окремі частини прогрмного коду, які відповідають за відображення секцій (рис. 3.14).



```
1  import React, { useContext, useState, useEffect } from "react"
2  import { SearchContext } from "../../context/searchContext"
3  import { getCars } from "../../api"
4  import { SearchMenu } from "./SearchMenu"
5  import { ResultsMenu } from "./ResultsMenu"
6  import { Warehouse } from "./Warehouse"
7  import { StyledMenu } from "../Styled/StyledMenu"
8  import { mockData } from "../../constants"
9
10 export const Menu = (): JSX.Element => {
11     const { data: { foundCars, foundCar, searchedCarParams }, setCars } = useContext(SearchContext)
12     const [isCarsLoading, setCarsLoading] = useState(false);
13     const [isError, setError] = useState(false);
14
15     useEffect(() => {
16         (async () => {
17             try {
18                 setCarsLoading(true)
19                 const { data } = await getCars(searchedCarParams)
20                 setCars(data);
21             }
22             catch (error) {
23                 setCarsLoading(false)
24                 setError(true)
25             }
26             finally {
27                 setCarsLoading(false)
28             }
29         })()
30 }, [searchedCarParams])
31
32     if(isError) {
33         return <div>Error</div>
34     }
35
36     return (
37     <StyledMenu>
38         <SearchMenu />
39         <ResultsMenu isLoading={isCarsLoading} cars={foundCars} foundCar={foundCar}/>
40         <Warehouse cars={mockData} foundCar={foundCar}/>
41     </StyledMenu>
42     )
43 };
44
```

Рисунок 3.14 – React-компонент Menu

Розглянемо більш детально компонент Menu, його функції та логіку, беручи до уваги кожен рядок файлу.

З першого по восьмий рядок файл імпортує саму бібліотеку React, допоміжні утиліти, файли, типи, які потрібні нашому компоненту.

На 10-му рядкові ми ініціалізуємо компонент Menu за допомогою синтаксису «стрілкової» функції. Рядки 11, 12, 13 – це використання спеціальних функції бібліотеки React, які називають «хуками». Функція useContext забезпечує

компонент даними, та функціями з контексту. Функція useState контролює стан самого компоненту. Як бачимо, змінних стану компоненту двоє – isLoading, isError. Контроль змінних стану здійснюється за допомогою функцій, які повертає відповідний хук – setCarsLoading або setError.

Рядки 15-30 відповідають за надсилання запиту до сервера для отримання інформації щодо авто за пошуковими параметрами. Запит до серверу є асинхронною дією, тому на 16-му рядкові прописане ключове слово async, яке вказує програмі на те, що поки запит виконується, програма «чекає». Можна помітити, що дана асинхронна операція відбувається в тілі хука useEffect. Даний хук буде виконувати свою частину коду, як тільки компонент буде активований або ж тоді, як користувач змінить пошуковий параметр searchedCarParams.

Рядок 32 перевіряє, чи була помилка при надсиланні запиту.

Рядки 37-43 відповідають за відображення нашої сторінки, а саме трьох секцій – SearchMenu, ResultsMenu, Warehouse. Усі ці три компоненти (секції) обернуті в один спільний компонент StyledMenu. Це звичайний блоковий елемент div HTML розмітки. На рисункові 3.15 зображена імлементація даного стилізованого компоненту.

```
1  import styled from "styled-components";
2
3  export const StyledMenu = styled.div`
4      height: 100%;
5      width: 100%;
6
7      display: flex;
8
9      background: radial-gradient(#6EA587, #609076, #527C65);
10 `;
11
```

Рисунок 3.15 – Стилізований компонент StyledMenu

З рисунку 3.15 бачимо, що це стилізований компонент StyledMenu описаний за допомогою CSS-властивостей, таких як height, width, display, background. Загалом увесь програмний код відображення інтерфейсу базується на таких стилізованих компонентах.

Блок схема алгоритму роботи програми зображена на рис. 1.16.

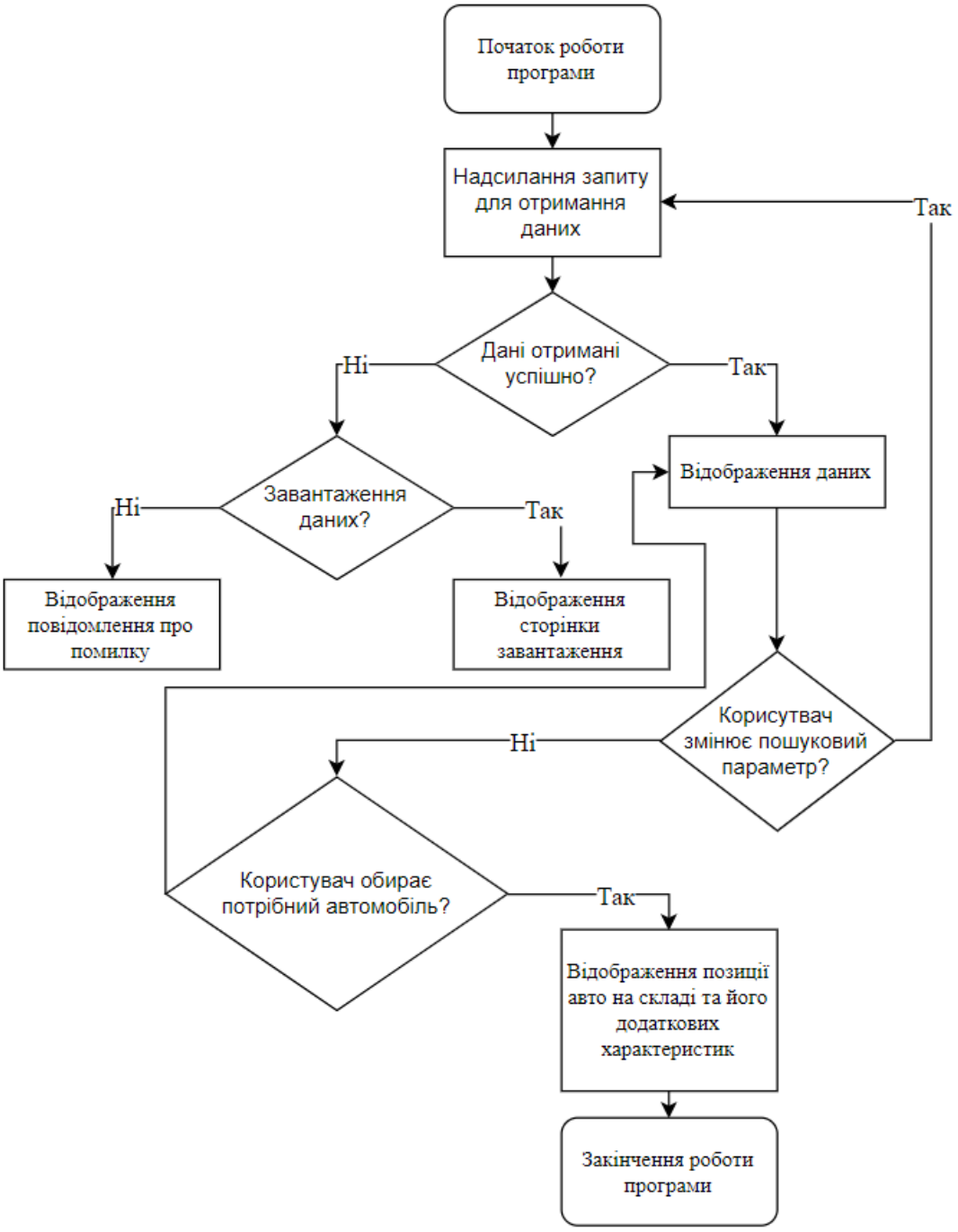


Рисунок 3.16 – Блок-схема алгоритму роботи програми

4 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ ПРОГРАМНОЇ РОЗРОБКИ

4.1 Результати алгоритмічно-програмної розробки

На рисункові 4.1 зображено початковий вигляд веб-застосунку.

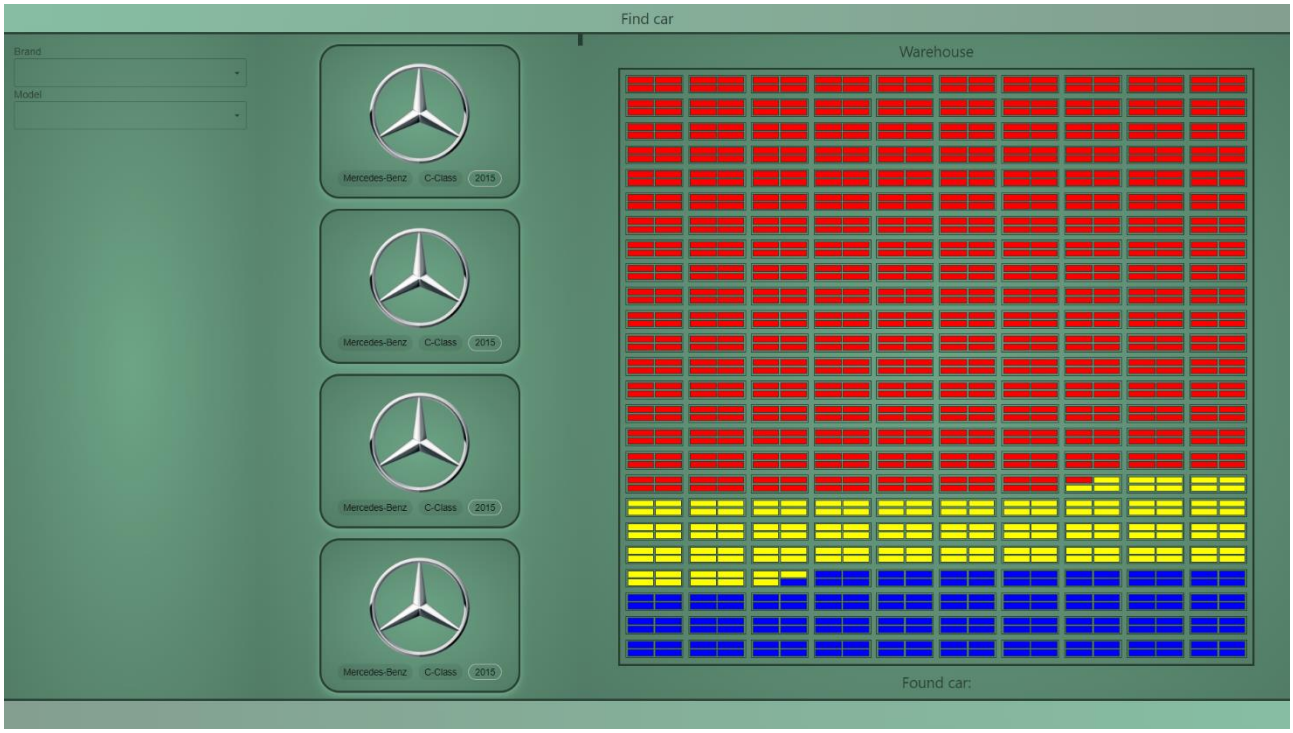


Рисунок 4.1 – Початковий вигляд веб-застосунку

З рис. 4.1 бачимо, що структура сторінки доступна і зручна. Вона розділена на три секції, а саме: вибір марки та моделі(зліва), результати пошуку(по центру), схема розташування автомобілів на складі(праворуч). Кожна секція є окремим програмним компонентом, який виконує свою роль.

У даній інструкції наведені кроки для налаштування веб-застосунку, заповнення бази даних та користування інфтерфейсом.

Перш за все, потрібно підготувати дані для подальшого функціонування системи. Для цього використовуємо систему керування реляційними базами даних MySQL Workbench, за допомогою якої створюємо нашу базу даних, відповідну таблицю та заповнюємо її. Перелік команд для створення сховища з інформацією наступний:

– «CREATE DATABASE warehouse» – команда для створення бази даних з назвою «warehouse»;

– «CREATE TABLE warehouse» – створення таблиці «warehouse»;

– «INSERT INTO warehouse (id_trim,make,model,generation,year_from,year_to,series,trim,body_type,load_height_mm,number_of_seats,length_mm,width_mm,height_mm,wheelbase_mm,front_track_mm,rear_track_mm,curb_weight_kg,wheel_size_r14,ground_clearance_mm,trailer_load_with_brakes_kg,payload_kg,back_track_width_mm,front_track_width_mm,clearance_mm,full_weight_kg,front_rear_axle_load_kg,max_trunk_capacity_l,cargo_compartment_length_width_height_mm,cargo_volume_m3,minimum_trunk_capacity_l,maximum_torque_n_m,injection_type,overhead_camshaft,cylinder_layout,number_of_cylinders,compression_ratio,engine_type,valves_per_cylinder,boost_type,cylinder_bore_mm,stroke_cycle_mm,engine_placement,cylinder_bore_and_stroke_cycle_mm,turnover_of_maximum_torque_rpm,max_power_kw,presence_of_intercooler,capacity_cm3,engine_hp,engine_hp_rpm,drive_wheels,bore_stroke_ratio,number_of_gears,turning_circle_m,transmission,mixed_fuel_consumption_per_100_km_l,range_km,emission_standards,fuel_tank_capacity_l,`acceleration_0_100_km/h_s`,max_speed_km_per_h,city_fuel_per_100km_l,`CO2_emissions_g/km`,fuel_grade,highway_fuel_per_100km_l,back_suspension,rear_brakes,front_brakes,front_suspension,steering_type,car_class,country_of_origin,number_of_doors,safety_assessment,rating_name,battery_capacity_KW_per_h,electric_range_km,charging_time_h,logo_image_url,warehouse_position) VALUES (35460,'Mercedes-Benz','C-Class','W204/S204/C204 [redesign]','2011','2015','AMG wagon 5-doors','C 63 AMG Edition 507 Speedshift MCT','Wagon','','5','4707','1795','1443','2765','1585','1562','1720','','','','555','','','','2275','','1500','','','485','610','Multi-point fuel injection','','V-type','8','','Gasoline','4','','102','94','','','5200','','','6208','507','6800','Rear wheel drive','','7','11.1','Automatic','12.2','','EURO V','66','4.3','280','','','95','','Independent, Multi wishbone, Stabilizer bar','ventilated disc','ventilated disc','Independent, McPherson Struts, Stabilizer bar','','','','','','','','',NULL,NULL);» – прикоал команди для додавання одного рядка в таблицю (таких рядків додаємо 1000).

Перші три етапи є основними для роботи з базою даних. Далі буде лише використання даних з неї. Для цього ми використовуємо ключове слово «SELECT». Перший запит, який буде оброблятися нашим сервером буде таким – «SELECT * from warehouse;» – поверне нам колекцію з усіма автомобілями бази даних.

Далі запити можуть мати додаткові параметри, умови, за якими ми будемо обирати потрібні нам дані з таблиці. Наприклад, щоб обрати автомобілі марки «Volvo» ми напишемо відповідний запит – «select * from warehouse where make=’Volvo’;». За таким принципом будемо обирати й інші параметри.

Налаштування сервера є наступним етапом. Сервер, як описувалося вище, дає змогу користувачу відправляти запити до системи управління базою даних і отримувати від неї дані у відповідь. Програмний файл нашого застосунку доволі простий та зрозумілий.

Спершу ми вказуємо дані нашої бази, а саме порт, на якому базується MySQL, назву самої бази, потрібної таблиці та вхідні дані корисувача – ім’я та пароль. Після успішного підключення ми, так би мовити, встановлюємо канал зв’язку між користувачем та базою.

Після того налаштування бази даних ми можемо запускати серверну та клієнтську частину. У двої окремих директоріях потрібно виконати команду «npm i», яка встанвоить усі необхідні пакети для роботи. Після запускаємо в тих же терміналах команду «npm start». Як тільки клієнтська частина запсутиться, в браузері відкриється додатково вкладка з нащою сиземою (рис. 3.1).

Припустимо, що корисутвач хоче знайти позицію автомобіля з наступними параметрами:

– марка – Volvo;

– модель – V40;

– рік випуску – 2016;

– потужність двигуна у кінських силах – 245.

Виконаємо наступні кроки і прослідкуємо за зміною інтерфейсу корисутвача.

Оберемо марку Volvo в секцій вибору параметрів авто (рис. 4.2).

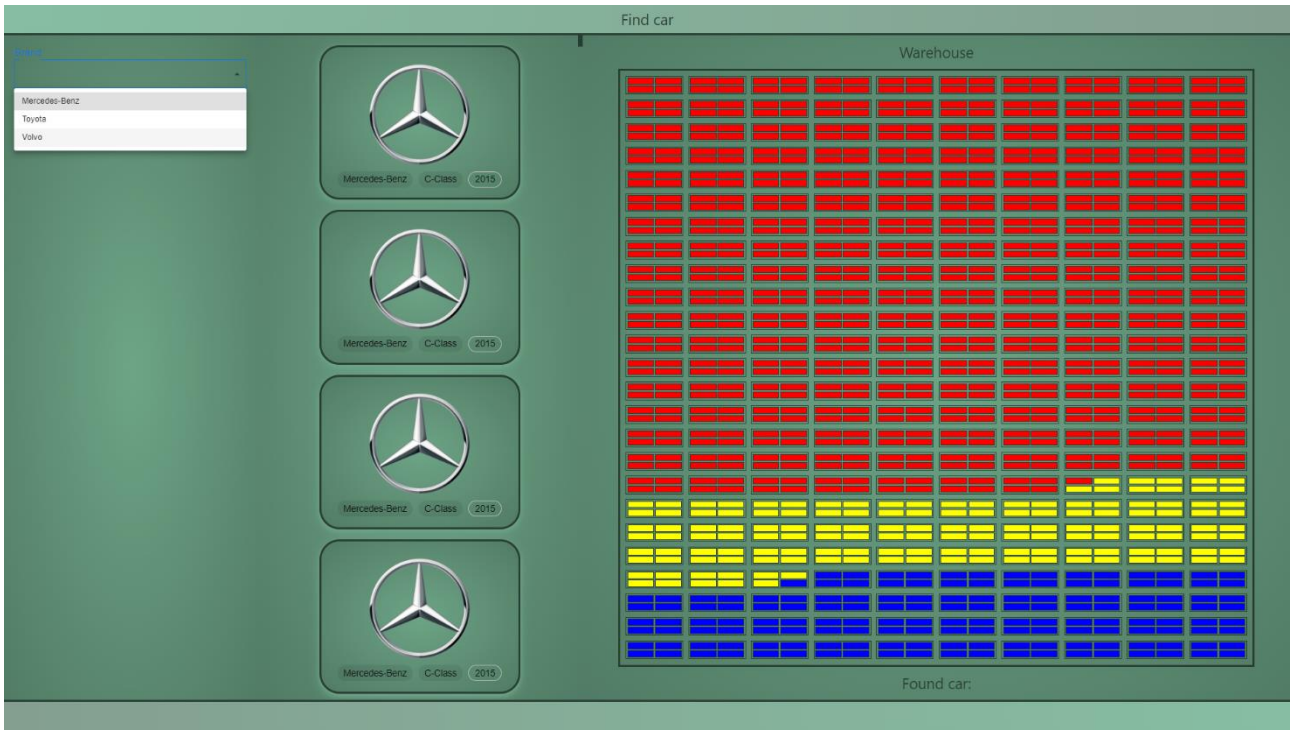


Рисунок 4.2 – Вибір авто марки Volvo

Оберемо потрібну модель з випадаючого списку (рис. 4.3).

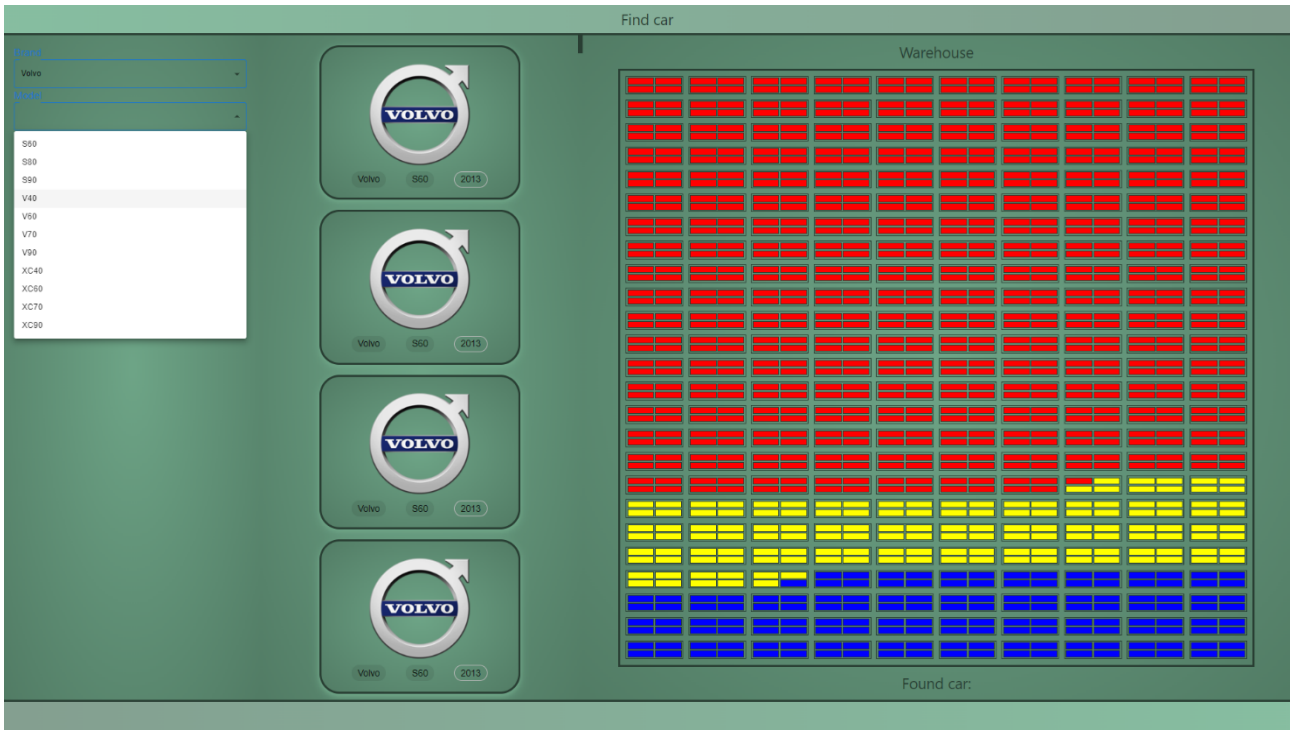


Рисунок 4.3 – Вибір відповідної моделі марки Volvo

Як бачимо, ми отримали декілька автомоблів із вказанами параметрами. Тепер, для уточнення конкретного автомобіля з потрібною потужністю двигуна, ми

обираємо перший автомобіль і бачимо його позицію на складі у вигляді білого прямокутника. Також більш детальнні характеристики авто описані у секції під схемою складу (рис. 4.4).

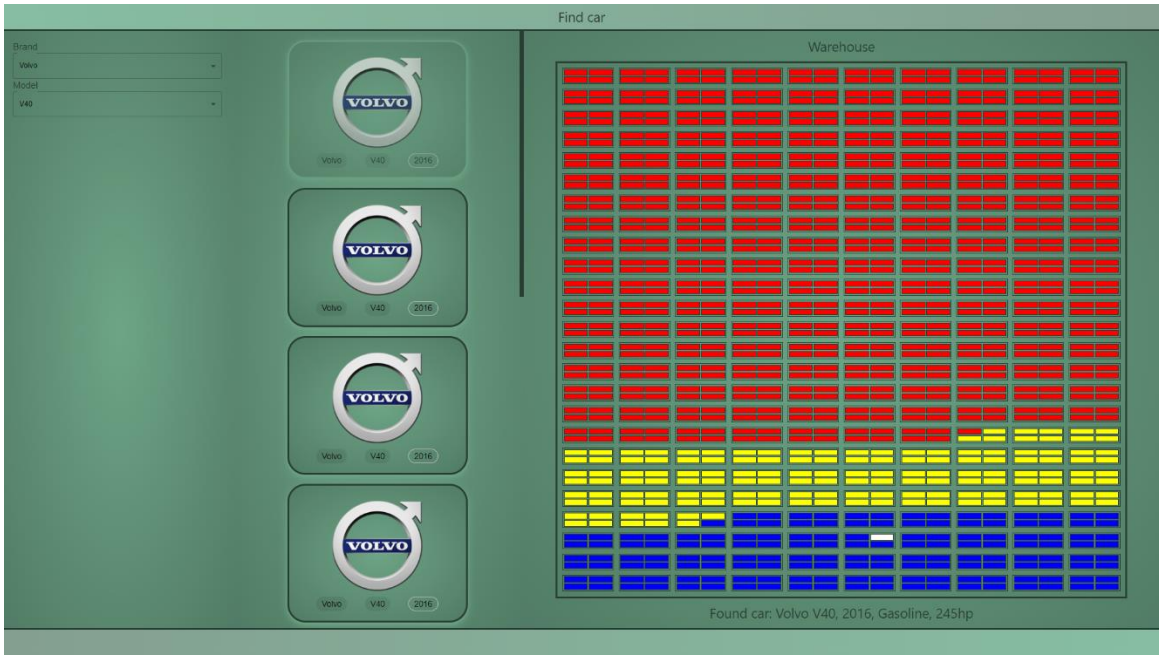


Рисунок 4.4 – Вибір потрібного авто, відображення його позиції на складі та додаткових хаарктеристик

Щоб дізнатися позицію авто на складі, достатньо навести курсором миші на білий прямокутник, після чого буде додаткове вікно, у якому зашифроване розташування (рис. 4.5).

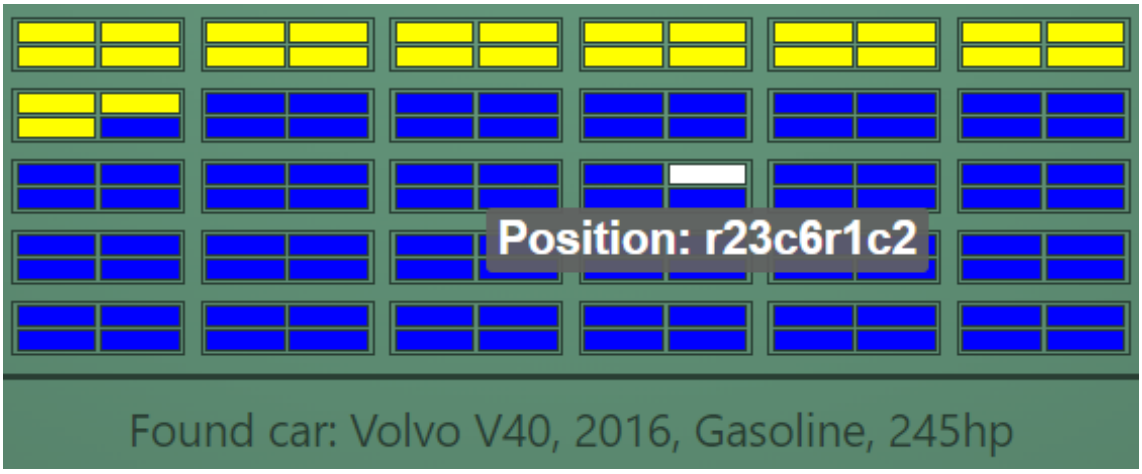


Рисунок 4.5 – Позиція авто на складі та опис додаткових характеристик

Загалом інтефейс програми доволі простий, у той час як працює з великим об’ємом даних та їх обробкою.

4.2 Інструкція користувача

Початок роботи програми – це запуск веб-сторінки та її відображення. Відразу ж, після першого рендерінга, надсилається запит до серверу для отримання даних. Надсилання запиту зображено на рисункові 4.6.

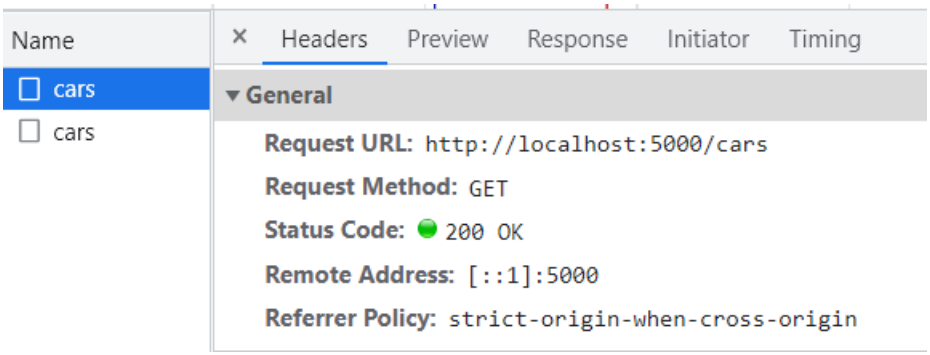


Рисунок 4.6 – Надсилання запиту до сервера у вкладці Network+

Поки дані отримуються, сторінка відображає процес завантажаннення у вигляді певного так званого «лоадера», який зображений на рисункові 4.7.

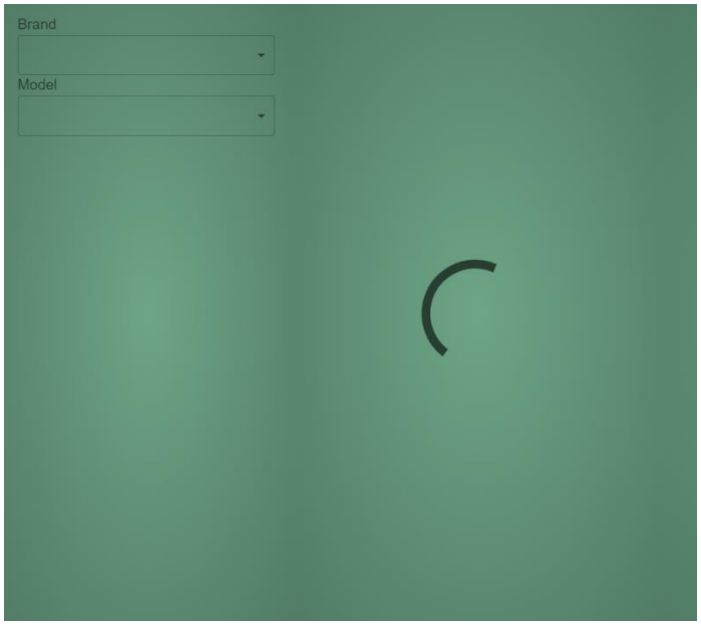


Рисунок 4.7– Відображення сторінки завантаження даних

На випадок помилки при отриманні даних або іншого програмного алгоритму, відображається спеціальна сторінка наступним текстом: «Error has occured! Please, try again.» (дивитися рисунок 4.8).

Рисунок 4.8 – Зображення повідомлення про помилку

5 ОХОРОНА ПРАЦІ

Інструкція розроблена на основі ДНАОП 0.00-8.03-93 «Порядок опрацювання та затвердження власником нормативних актів про охорону праці, що діють на підприємстві», ДНАОП 0.00-4.15-98 «Положення про розробку інструкцій з охорони праці», ДНАОП 0.00-4.12-99 «Типове положення про навчання з питань охорони праці».

За даною інструкцією завідувач складом інструктується перед початком роботи (первинний інструктаж), а потім через кожні 6 місяці (повторний інструктаж).

Результат інструктажу заноситься в “Журнал реєстрації інструктажів з питань охорони праці”; в журналі після проходження інструктажу повинен бути підпис особи, яка інструктує, і завідувача складом.

Власник повинен застрахувати завідувача складом від нещасних випадків та професійних захворювань.

У разі пошкодження здоров’я завідувача складом з вини власника, він (завідувач складом) має право на відшкодування заподіяної йому шкоди.

До роботи на складі допускаються особи не молодше 18 років, які пройшли медичний огляд, вступний інструктаж та інструктаж на робочому місці, практично освоїли прийоми правильного застосування механізмів, пристосування, інструментів, а також прийоми роботи з вантажами під час їх переробки.

Особи, що допускаються до роботи на складі небезпечних і шкідливих речовин (кислот, лугів, лакофарбової продукції тощо), повинні пройти спеціальне навчання з безпеки праці і мати посвідчення на право виконання робіт з підвищеною небезпекою.

Особи, допущені до роботи, повинні виконувати тільки ту роботу, яка доручена адміністрацією підприємства.

Під час виконання дорученої роботи необхідно суворо дотримуватися прийнятої технології переробки вантажів. Не допускається застосовувати способи,

що прискорюють виконання технологічної операції та ведуть до порушення вимог безпеки.

У випадку виникнення в процесі роботи будь-яких питань, пов’язаних з її безпечним виконанням, необхідно звернутися до особи, відповідальної за безпечне проведення робіт.

Помітивши порушення інструкції або небезпеку для оточуючих, завідувач складом повинен у цьому випадку попередити робітника про необхідність дотримання вимог, що забезпечують безпеку праці.

На завідувача складом можуть впливати небезпечні та шкідливі виробничі фактори: машини і механізми, що рухаються, рухливі частини підйомно-транспортного обладнання, переміщувані товари, тара, штабелі складованих товарів, що обрушуються; понижена температура повітря робочої зони; відсутність або недостатність природного світла; недостатня освітленість робочої зони; гострі крайки, задирки та шорсткість на поверхнях інструмента, устаткування, інвентарю, тари; хімічні фактори.

Завідувач складом зобов’язаний:

– вживати заходи по недопущенню виробничого травматизму та профзахворювань;

– стежити за підтримкою нормальних санітарних умов роботи на складі, у допоміжних та побутових приміщеннях;

– контролювати дотримання режиму праці та відпочинку робітників складу.

Завідувач складом повинен:

– виконувати правила внутрішнього трудового розпорядку;

– виконувати тільки ту роботу, яка доручена керівником робіт та по якій він проінструктований;

– не допускати на робоче місце сторонніх осіб;

– пам’ятати про особисту відповідальність за виконання правил охорони праці і відповідальність за товаришів по роботі;

– вміти користуватися засобами пожежегасіння;

– вміти надавати першу медичну допомогу потерпілим від нещасних випадків;

– не захаращувати робоче місце, проходи, підходи тощо.

Завідувачеві складом забороняється:

– доторкатися до електрообладнання, клем і електропроводів, арматури загального освітлення, відкривати дверці електрошаф;

– включати і зупиняти (крім аварійних ситуацій) машини, верстати і механізми, робота на яких не доручена йому керівництвом;

– стояти або проходити під піднятим вантажем.

Завідувач складом повинен дотримуватися правил особистої гігієни. Для пиття користуватися водою зі спеціальних пристроїв (сатуратори, питні баки, фонтанчики тощо).

За невиконання даної інструкції завідувач складом несе дисциплінарну, матеріальну, адміністративну та кримінальну відповідальність.

Перевірити:

– справність обладнання, інструменту, приладів;

– наявність і справність достатнього освітлення, вентиляції, обладнання тощо;

– перевірити справність рубильників, розеток, штепсельних з’єднань тощо.

У випадку виявлення будь-яких відхилень, несправностей, пошкоджень негайно повідомити директора Підприємства. Виконувати тільки ті функції, які вказані у їх посадових інструкціях.

Під час роботи заборонено:

– присутність сторонніх осіб на робочих місцях;

– захаращувати приміщення предметами (обладнанням, яке не використовується, господарчими предметами).

Загальні вимоги з електробезпеки для працівників.

Усі електричні прилади і пристосування (ізоляція, електропровід, штепсельні з’єднання, вимикачі тощо) повинні бути справними і використовуватися за призначенням і згідно з правилами експлуатації.

Електропроводи, які прокладені від дрібного електрообладнання до місця включення їх у мережу, не повинні підлягати механічним пошкодженням, укладатися по підлозі у зоні досягнення їх ногами, а також по струмопроводжуючим та пальним поверхням.

Улаштування та експлуатація тимчасових електромереж у приміщеннях закладу забороняється.

Використовувати електроприлади побутового типу (чайник, кип'ятильник, електрообігрівач тощо) можна тільки, якщо частини, які нагріваються, знаходитимуться на поверхні, із негорючого матеріалу.

Установлюючи апарати (штепсельні з'єднання, вимикачі тощо) установлюються на пальних поверхнях тільки з підкладкою під них негорючого матеріалу, виступаючого за габарити апарату не менш ніж на 0,01 м.

Забороняється:

– експлуатація кабелів і проводів із пошкодженою та утраченою у процесі експлуатації захисною властивістю ізоляції;

– використовувати нестандартні подовжувачі чи лампи накалювання;

– підвішувати світильники безпосередньо на струмоведучі проводи;

– завертати електролампи і світильники папером, тканиною та іншими пальними матеріалами, експлуатація їх із знятими ковпаками (розсіювачами);

– використовувати електрообладнання в умовах, не відповідних указівкам виробника;

– залишати без нагляду електрообладнання;

– використовувати вимикачі, штепсельні з'єднання, струмоведучі проводи для підвішування одягу та інших предметів, заклеювати ділянки електропроводки папером чи пальними матеріалами;

– використовувати побутові електроприлади без негорючих підставок і без дозволу на їх експлуатацію;

– переносити електроприлади, які знаходяться під напругою.

При виявленні несправності у електромережах і електрообладнанні, які можуть викликати іскріння, коротке замикання, зверх допущеного нагрівання

пальної ізоляції кабелю і проводів, необхідно негайно припинити їх експлуатацію та повідомити керівництво.

Самостійний ремонт електромереж, електроапаратів, електрообладнання забороняється.

У випадку виявлення будь-яких відхилень, несправностей, пошкоджень негайно повідомити директора підприємства.

ВИСНОВКИ

У результаті виконання роботи було описано основні сучасні методи управління у складських приміщеннях, їх відмінності, переваги. Наведено відповідні характеристики, зображення до кожного з методу.

У ході роботи було запропоновано програмну розробку автоматизованої системи логістичних процесів у складських приміщеннях. До уваги було взято складське приміщення, яке вміщує до 1000 автомобілів. Дані про автомобілі були були збережені в базі даних MySQL. Для роботи з даними було імплементовано спеціальний сервер на базі NodeJS, який обробляв запити користувача і і відповідав належним чином з належними даними.

Також була реалізована візуальна частина застосунку, проста та зрозуміла. Наведено інструкцію по її використанню.

По темі кваліфікаційної роботи було опублікова статті у збірнику студентських статей за 2020, 2021, 2022 роки на наступні теми: «Обмін файлами у веб-застосунках з використанням NODE.JS», «Особливості застосування технології веб-сокетів для асинхронної клієнт-серверної взаємодії веб-програм промислової автоматизації», «Засоби захисту систем промислової автоматизації та управління».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. О.І. Кушлик-Дивульська, Б.Р. Кушлик. Основи теорії прийняття рішень. – К., 2014. – 94с.

2. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології, освітньо-професійних програм: «Автоматизоване управління технологічними процесами», «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. – Харків: ХНУРЕ, 2021. – 55 с.

3. ДСТУ 3008: 2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання / Нац. стандарт України. – Вид. офіц. – [Чинний від 2017 – 07 – 01]. – Київ: ДП «УкрНДНЦ», 2016. – 26 с.

4. Освітньо – професійна програма другого (магістерського) рівня вищої освіти за спеціальністю 151 – «Автоматизація та комп'ютерно – інтегровані технології» (Вчена рада ХНУРЕ протокол № 1 від 28.01. 2021 р.). https: //nure.ua/wp –content/uploads/Education programs/2020/2020 mag 151 opp ksua.pdf.

5. Положення про організацію освітнього процесу у ХНУРЕ [електронний ресурс] : Наказ ХНУРЕ від 27 листопада 2020 р. № 400. – Режим доступу:https://nure.ua/wp-content/uploads/Main_Docs_NURE/polozhennja-proorganiza-ciju-osvitnogoprocesu-v-hnure.pdf.

6. Khaletskaya A. Роботи на складах: 5 прикладів автоматизації [Електронний ресурс] / Anna Khaletskaya // IT-рішення для логістики. – 2021. – Режим доступу до ресурсу: https://wareteka.com.ua/uk/blog/roboti-na-skladah-prikladi-avtomatizaciyi/#title1.

7. Backbone to React. Joel Burget. Архів оригіналу за 6 грудня 2015. Процитовано 3 березня 2015.

8. React.js conf - schedule. React.js. Архів оригіналу за 18 березня 2015. Процитовано 3 березня 2015.

9. Yahoo Mail moving to React. Slideshare.

10. Dev Chats: Spike Brehm of Airbnb - JavaScript development without a "greenfield app". Medium. Архів оригіналу за 7 січня 2015. Процитовано 3 березня 2015.

11. Mikael Brassman on Twitter: "Sony's Lifelog newly released web interface is using #refluxjs and #reactjs on the client-side". Twitter. Архів оригіналу за 3 березня 2016. Процитовано 3 березня 2015.

12. Wesley Walser on Twitter: "React.js is now driving @atlassian OnDemand billing pages. Small project to start adoption, positive experiences thus far.". Twitter. Архів оригіналу за 4 березня 2016. Процитовано 3 березня 2015.

13. React (JS Library): How was the idea to develop React conceived and how many people worked on developing it and implementing it at Facebook?. Quora.

14. Pete Hunt at TXJS. Архів оригіналу за 31 липня 2017. Процитовано 28 березня 2016.

15. An Introduction to React.js. Instrument. Архів оригіналу за 27 лютого 2015. Процитовано 3 березня 2015.

16. Working With the Browser. React. Архів оригіналу за 29 травня 2016. Процитовано 3 березня 2015.

17. JSX in Depth. Архів оригіналу за 13 лютого 2017. Процитовано 17 листопада 2015.

18. Архівована копія. Архів оригіналу за 6 квітня 2015. Процитовано 14 січня 2017.

19. PayPal Isomorphic React. Архів оригіналу за 15 липня 2017. Процитовано 14 січня 2017.

20. Netflix Isomorphic React. Архів оригіналу за 17 грудня 2016. Процитовано 14 січня 2017.

21. node-v0.x-archive on GitHub. Архів оригіналу за 28 квітня 2015. Процитовано 2 August 2014.

22. Node.js 14 ChangeLog. Архів оригіналу за 18 липня 2020. Процитовано 16 серпня 2020 – через GitHub.

23. https://github.com/nodejs/node/blob/master/LICENSE

24. About Node.js, and why you should add Node.js to your skill set?. Training.com. Training.com. Архів оригіналу за 1 квітня 2017. Процитовано 23 жовтня 2016.

25. Netscape opens intranet attack. CNET (англ.). Архів оригіналу за 30 грудня 2019. Процитовано 20 квітня 2017.

26. Ryan Dahl (9 листопада 2010). Joyent and Node. Google Groups. Архів оригіналу за 31 травня 2019. Процитовано 5 лютого 2015.

27. PHP 7 vs Node.js? They Can Be Partners, Not Competitors For a Developer!. Архів оригіналу за 23 лютого 2017. Процитовано 21 грудня 2016.

28. Sams Teach Yourself Node.js in 24 Hours [Архівовано 23 березня 2017 у Wayback Machine.], Sams Publishing, 05-Sep-2012

29. Ryan Dahl at JSConf EU 2009. Архів оригіналу за 14 травня 2017. Процитовано 11 вересня 2019.

30. Ryan Dahl at JSConf EU 2009 Video. Архів оригіналу за 15 травня 2017. Процитовано 11 вересня 2019.

31. Professional Node.js: Building JavaScript Based Scalable Software [Архівовано 24 березня 2017 у Wayback Machine.], John Wiley & Sons, 01-Oct-2012

32. Earliest releases of npm. GitHub. Архів оригіналу за 1 березня 2017. Процитовано 27 липня 2016.

33. Porting Node to Windows With Microsoft's Help. Архів оригіналу за 14 липня 2017. Процитовано 17 квітня 2016.

34. Dahl, Ryan. New gatekeeper. Архів оригіналу за 31 травня 2019. Процитовано 26 жовтня 2013.

35. Schlueter, Isaac (15 січня 2014). The Next Phase of Node.js. Архів оригіналу за 14 липня 2017. Процитовано 21 січня 2014.

36. Krill, Paul (Dec 4, 2014). Why io.js Decided to Fork Node.js. JavaWorld. Архів оригіналу за 30 червня 2017. Процитовано Dec 15, 2014.

37. Q&A: Why io.js decided to fork Node.js [Архівовано 6 листопада 2018 у Wayback Machine.], InfoWorld Tech Watch

38. Ben Noordhuis (Nov 12, 2014). Issue 3692: function suddenly becomes undefined. V8 JavaScript Engine Issues. Архів оригіналу за 1 листопада 2015. Процитовано 2 February 2015.

39. Mikeal, Rogers (28 січня 2015). State of io.js. Архів оригіналу за 30 серпня 2015. Процитовано 2 February 2015.

40. Node.js Foundation Advances Community Collaboration, Announces New Members and Ratified Technical Governance. Архів оригіналу за 24 червня 2015. Процитовано 4 липня 2015.

41. Node.js Foundation Combines Node.js and io.js Into Single Codebase in New Release. Архів оригіналу за 7 січня 2017. Процитовано 28 січня 2016.

42. io.js and Node.js merge. Архів оригіналу за 6 березня 2016. Процитовано 27 червня 2015.

43. Io.js, JavaScript I/O [Архівовано 4 жовтня 2016 у Wayback Machine.], «io.js has merged with the Node.js project again. There won't be any further io.js releases. All of the features in io.js are available in Node.js v4 and above.»

44. Шило Н. Ю. Обмін файлами у веб-застосунках з використанням NODE. JS [Електронний ресурс] / Н. Ю. Шило, О. О. Чала // ХНУРЕ. – 2022. – Режим доступу до ресурсу: https://openarchive.nure.ua/bitstream/document/21035/1/Shylo_Chala_130_132.pdf.

45. Сидоренко А. В. Особливості застосування технології веб-сокетів для асинхронної клієнт-серверної взаємодії веб-програм промислової автоматизації [Електронний ресурс] / А. В. Сидоренко, Н. Ю. Шило // ХНУРЕ. – 2021. – Режим доступу до ресурсу: https://openarchive.nure.ua/bitstream/document/19602/1/7706e958-7c7d-45dc-b6fa-0d65cbe9f98f-112-115.pdf.

46. Шило Н. Ю. Засоби захисту систем промислової автоматизації та управління [Електронний ресурс] / Назар Юрійович Шило // ХНУРЕ. – 2020. –

Режим доступу до ресурсу: https://openarchive.nure.ua/bitstream/document/14073/1/Shylo_140-144.pdf.