Державний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«РОЗРОБКА ІНТЕГРАТОРА ДАНИХ ПО ЦІНАМ ТОВАРІВ В ОНЛАЙН МАГАЗИНАХ»

Студента 4 курсу, 10 групи — Савка Олександр Ігорович

спеціальності
122 «Комп'ютерні науки» _____ *підпис студента*

Кандидат фізико-математичних наук, доцент _____ *підпис керівника* — Філімонова Тетяна Олегівна

Гарант освітньої програми
кандидат технічних наук, доцент _____ *підпис керівника* — Демідов Павло Георгійович

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
« 20» грудня 2022 р.

Завдання
на випускну кваліфікаційну роботу студенту

Савка Олександр Ігорович
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)
«Розробка інтегратора даних по цінам товарів в онлайн магазинах»
Затверджена наказом ректора від *«09» грудня 2022 р. № 3332*
2. Строк здачі студентом закінченої роботи *30 травня 2023 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: полегшенні процесу пошуку найкращих цін на товари в онлайн-магазинах шляхом розробки інтегрованого веб-ресурсу..
Об'єкт дослідження: процес пошуку цін на товари в онлайн-магазинах, який є складним завданням через розмаїтість магазинів та розподіленість інформації.
Предмет дослідження: розробка інтегрованого веб-ресурсу для пошуку цін на товари в онлайн-магазинах з використанням мапи. Цей ресурс дозволить покупцям зручно та ефективно порівнювати ціни на товари з різних джерел, що значно зекономить їх час та зусилля..
4. Перелік графічного матеріалу_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Філімонова Т.О.	15.12.2022 р.	15.12.2022 р.
2	Філімонова Т.О.	15.12.2022 р.	15.12.2022 р.
3	Філімонова Т.О.	15.12.2022 р.	15.12.2022 р.

6. Зміст випускного кваліфікаційного проекту (перелік питань за кожним розділом)

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*01.10.2020*	*01.10.2020*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*15.12.2022*	*15.12.2022*
3	*Вступ*	*03.02.2023*	
4	*РОЗДІЛ 1 ДОСЛІДЖЕННЯ СУЧАСНИХ ВЕБ-САЙТІВ*	*28.02.2023*	
5	*РОЗДІЛ 2 ЗАСОБИ І МЕТОДИ СТВОРЕННЯ ІНТЕГРОВАНОГО ВЕБ-РЕСУРСУ З КОРИСТУВАЦЬКИМ ІНТЕРФЕЙСОМ*	*06.04.2023*	
6	*РОЗДІЛ 3 РОЗРОБКА ВЕБ-САЙТУ*	*12.05.2023*	
7	*Висновки*	*15.05.2023*	
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*20.05.2023*	
9	*Попередній захист випускної кваліфікаційної роботи*	*26.05.2023*	
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*27.05.2023*	
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*30.05.2023*	
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «15» грудня 2022 р.

9. Керівник випускної кваліфікаційної роботи (проекту)
Філімонова Т.О.
(прізвище, ініціали, підпис)

10. Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник
Савка О. І.
(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи (проекту)

Керівник випускної кваліфікаційної роботи (проекту)
Філімонова Т. О. 30.05.2023 р.
(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу (проект)

Випускна кваліфікаційна робота (проект) студента Савка О. І.
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми Демідов П.Г.
(підпис, прізвище, ініціали)

Завідувач кафедри Пурський О.І.
(підпис, прізвище, ініціали)
«_____»_____2023 р.

АНОТАЦІЯ:

Дипломна робота присвячена розробці інтегратора даних по цінам товарів в онлайн-магазинах з мапою. Головною метою дослідження було полегшення процесу пошуку найкращих цін на товари для покупців та забезпечення зручного та ефективного онлайн-шопінг досвіду. У роботі було проведено аналіз літературних джерел з електронної комерції та пошуку цін на товари. На основі отриманих знань був розроблений ефективний алгоритм інтеграції даних про ціни з різних джерел, забезпечуючи їхню точність та актуальність. Далі був розроблений веб-ресурс, який інтегрує зібрані дані про ціни та відображає їх на мапі. Це дозволяє покупцям зручно порівнювати ціни на товари у різних магазинах та знаходити найкращі пропозиції в різних регіонах. Розроблений веб-ресурс був підданий тестуванню з використанням реальних користувачів. Отримані результати підтвердили ефективність та коректність роботи ресурсу, а користувачі високо оцінили зручність інтерфейсу та швидкість надання інформації про ціни на товари. Результати дослідження свідчать про успішне досягнення поставлених завдань і розробку практичної та корисної системи для пошуку цін на товари в онлайн-магазинах. Розроблений інтегратор даних є цінним інструментом для покупців, що дозволяє економити час та зусилля при прийнятті рішень щодо покупок та сприяє зростанню конкуренції на ринку електронної комерції.

КЛЮЧОВІ СЛОВА: Інтегратор даних, Ціни на товари, Онлайн-магазини, Веб-ресурс, Мапа, Електронна комерція

ANOTATION: The thesis is devoted to the development of a data integrator on the prices of goods in online stores with a map. The main goal of the study was to facilitate the process of finding the best prices for goods for buyers and to provide a convenient and efficient online shopping experience. In the work, an analysis of literary sources on electronic commerce and product price search was carried out. Based on the knowledge gained, an effective algorithm was developed to integrate price data from various sources, ensuring their accuracy and relevance. Next, a web resource was developed that integrates the collected price data and displays it on a map. This allows customers to conveniently compare prices of goods in different stores and find the best deals in different regions. The developed web resource was tested using real users. The obtained results confirmed the efficiency and correctness of the resource, and the users highly appreciated the user-friendliness of the interface and the speed of providing information on product prices. The results of the research indicate the successful achievement of the set objectives and the development of a practical and useful system for finding prices for goods in online stores. The developed data integrator is a valuable tool for buyers, saving time and effort in purchasing decisions and contributing to the growth of

competition in the e-commerce market.

KEY WORDS: Data integrator, Product prices, Online stores, Web resource, Map, E-commerce

ЗМІСТ

ВСТУП

У сучасному світі електронна комерція набуває все більшої популярності, привертаючи увагу широкого кола споживачів. Завдяки зручності та доступності онлайн-магазинів, покупці мають можливість придбати різноманітні товари з комфортом, уникнувши традиційного шопінгу в офлайн-магазинах. Однак, зростання кількості інтернет-магазинів призводить до появи проблеми складного вибору та порівняння цін на товари.

Актуальність теми дослідження полягає в тому, що багато покупців витрачають значну кількість часу на перегляд різних магазинів з метою знайти найкращу ціну на товар, який їх цікавить. Цей процес є часо-, праце- та енергозатратним. Тому нашою метою є розробка інтегрованого веб-ресурсу, який спростить покупцям пошук найкращої ціни на товари в різних онлайн-магазинах.

Об'єктом дослідження є процес пошуку цін на товари в онлайн-магазинах, який є складним завданням через розмаїтість магазинів та розподіленість інформації.

Предметом дослідження є розробка інтегрованого веб-ресурсу для пошуку цін на товари в онлайн-магазинах з використанням мапи. Цей ресурс дозволить покупцям зручно та ефективно порівнювати ціни на товари з різних джерел, що значно зекономить їх час та зусилля.

Основними **завданнями нашої дипломної роботи** є:

- Аналіз існуючих методів та інструментів для пошуку цін на товари в онлайн-магазинах.

- Розробка алгоритму інтеграції даних про ціни товарів з різних джерел.
- Розробка веб-ресурсу, який інтегрує дані про ціни з онлайн-магазинів та відображає їх на мапі для зручного порівняння.

Для досягнення поставлених завдань ми плануємо використовувати наступні **методи та засоби дослідження:**

- Аналіз літературних джерел, що стосуються електронної комерції, пошуку цін та інтеграції даних.
- Вивчення роботи аналогічних систем та інструментів для пошуку цін на товари в онлайн-магазинах.
- Проектування та програмування веб-додатку для інтеграції даних про ціни та їх відображення на мапі.
- Проведення експериментів для перевірки ефективності розробленого веб-ресурсу.

Загальна **мета нашої роботи** полягає в полегшенні процесу пошуку найкращих цін на товари в онлайн-магазинах шляхом розробки інтегрованого веб-ресурсу. Це допоможе покупцям економити час та зусилля, а також забезпечить більш зручне та ефективне онлайн-шопінг досвід.

Продовжуючи наше дослідження, ми прагнемо досягти декількох конкретних **результатів:**

Розробка алгоритму інтеграції даних: Ми плануємо проаналізувати різні джерела даних про ціни товарів в онлайн-магазинах і розробити ефективний алгоритм, який дозволить збирати, обробляти та інтегрувати ці дані. Цей алгоритм повинен бути гнучким та масштабованим, щоб забезпечити актуальність та точність інформації про ціни товарів.

Розробка веб-ресурсу з мапою: Нашим наступним завданням є створення веб-ресурсу, який інтегрує зібрані дані про ціни товарів та відображає їх на мапі. Це дозволить покупцям зручно переглядати розташування магазинів та порівнювати ціни на товари у різних регіонах. Мапа надасть візуальний контекст і допоможе зорієнтуватись покупцям у їхньому пошуку.

Тестування та оцінка ефективності: Ми плануємо провести ретельне тестування розробленого веб-ресурсу з метою перевірки його ефективності та коректності роботи. Це включатиме проведення експериментів з реальними користувачами, збір фідбеку та вдосконалення системи на основі отриманих результатів.

Остаточним результатом нашої дипломної роботи буде готовий веб-ресурс, який забезпечує покупців зручним та ефективним інструментом для пошуку найкращих цін на товари в онлайн-магазинах. Цей ресурс має потенціал полегшити процес прийняття рішень щодо покупок та сприяти ефективному використанню ресурсів покупцями.

Наша дипломна робота є актуальною в сучасному світі, де електронна комерція є все більш поширеною формою покупок. Покупці шукають найкращі ціни на товари, але зустрічаються з проблемою розсіяних даних та неефективним процесом пошуку. Наша робота вирішує цю проблему, надаючи покупцям цінну інформацію та зручний інструмент для здійснення покупок з найкращою вигодою.

Загальна актуальність нашої теми дослідження полягає в практичній вартості та потенційному впливі на економіку та споживачів. Розробка інтегрованого веб-ресурсу для пошуку цін на товари в онлайн-магазинах може полегшити процес покупок, зменшити витрати часу та енергії

покупців, а також сприяти зростанню конкуренції на ринку, що може призвести до зниження цін та поліпшення якості обслуговування.

Отже, розробка інтегрованого веб-ресурсу по цінах на товари в онлайн-магазинах з мапою є актуальною та значущою задачею, яка має потенціал зробити покупки в Інтернеті більш зручними та ефективними для споживачів.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ СУЧАСНИХ ВЕБ-САЙТІВ

1.1 Основні відомості про веб-розробку.

Веб розробка – це процес створення веб-сайту. Загалом цей процес можна поділити на Frontend development(клієнтську розробку), та Backend development(серверну розробку).

Результатом процесу клієнтської розробки є веб-сайт із зручним інтерфейсом користувача. Він вже містить текстову, та графічну інформацію, а інтерактивні елементи роблять процес його використання зручним та цікавим. Але такий сайт є статичним, тобто доповнення та зміна вмісту сайту виконується вручну, а взаємодія з базою даних є неможливою.

Яскравими прикладами статичних сайтів є сайт-візитка або веб-каталог продукції. Головною метою статичних сайтів є реклама товару, послуги чи події.

Соціальні мережі, інтернет-магазини, та відеохостинги є більш складними системами і потребують динамічної зміни свого вмісту, можливості взаємодії з базою даних, чи отримання даних зі сторонніх ресурсів. Створення таких проектів завжди включає в себе серверну розробку.

Динамічні веб-сайти мають значно більші функціональні можливості, та при цьому є більш вимогливими до ресурсів хостингу. Системи управління базами даних, та механізми динамічної зміни вмісту дають змогу користувачу впливати на вміст сайту, та зберігати цей результат в пам’яті бази даних. Таким чином звичайний веб-сайт перетворюється на веб-додаток.

1.2 Складові частини інтерфейсу веб-сайтів супермаркетів.

На етапі проектування важливо виділити загальні структурні блоки, з яких згодом буде скомпоновано інтерфейс сайту. Для цього потрібно дослідити продукти конкурентів і виділити загальновживані елементи. Спираючись на потреби замовника, та отриману в ході дослідження інформацію буде значно більше шансів створити продукт, який задовольнить як замовника, так і користувача.

Для дослідження було обрано наступні веб-сайти: novus.ua, furshet.ua, atbmarket.com, eko.com.ua, kishenya.ua, billa.ua, silpo.ua, varus.ua.

В ході дослідження отримано такі дані:

Структура головної сторінки(рис. 1.1.): банер; акції; особливі пропозиції / новини; форма для email розсилки; мапа магазинів.

Рис. 1.1. Елементи головної сторінки

Банер: елемент веб-сайту, ідея якого була запозичена з сфери зовнішньої реклами. Являє собою блок рекламного, або інформаційного характеру, часто реалізується у вигляді слайдера з метою заощадження простору та створення більш інтерактивного інтерфейсу.

Особливі пропозиції / новини: блок, що часто зображується у вигляді карток з короткими текстовими заголовками, та яскравими зображеннями. Основна мета – привернути увагу користувача. Клік по картці перенаправляє на інформаційну сторінку.

Акції: виглядає як сітка з блоків рівного розміру, що містять інформацію про актуальні акції.

Форма розсилки: невелика форма з одним полем вводу, та кнопкою. Після відправки форми, користувач додається в базу і буде регулярно отримувати рекламні повідомлення на пошту.

Мапа магазинів: інтерактивна мапа, додається завдяки сторонньому API, містить коротку інформацію про конкретні магазини мережі.

Невід'ємними частинами інтерфейсу веб-сайтів є так звані «header», та «footer»(рис. 1.2.). Частіше за все ці блоки містять навігаційні меню, контактну інформацію та логотипи.

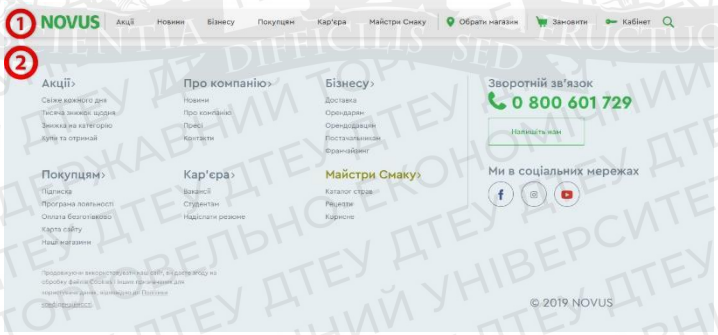

Рис 1.2. Header та footer

Інші сторінки(рис. 1.3.): акції; співпраця/контакти; покупцям; кар'єра; мапа; про компанію; власний бренд.

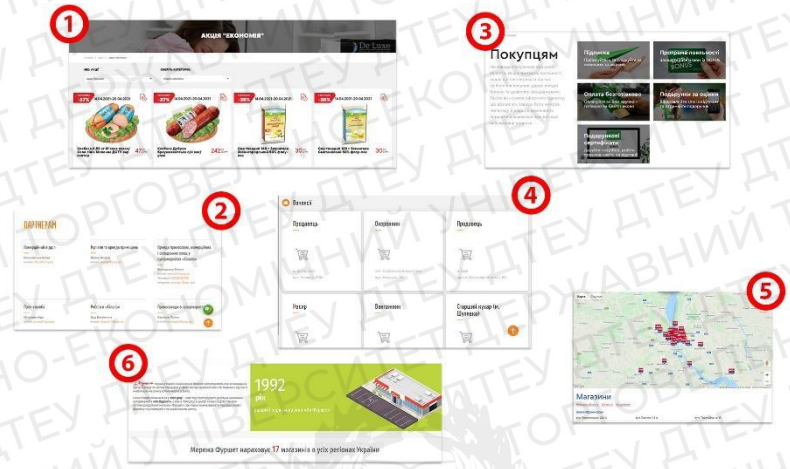

Рис. 1.3. Інші сторінки

Сторінка з контактами містить здебільшого текстову інформацію: телефонні номери, електронні адреси різних відділів компанії. Сторінка «кар'єра» містить список вакансій компанії, та контактні дані.

Сторінка «покупцям» зустрічається рідше и містить інформацію про особливі пропозиції, унікальні опції, та ін… Менша вживаність обумовлюється тим, що іноді її вміст виноситься на головну сторінку, або розподіляється між кількома сторінками. Сторінка «про компанію» містить короткі відомості про компанію, відео і зображення. Якщо у компанії є власне виробництво, то зазвичай на її сайті є окрема сторінка, на якій представлена інформація про власну торгову марку.

1.3 Проектування інтерфейсу. Мапа сторінок.

Спираючись на інформацію отриману в ході дослідження існуючих веб-сайтів супермаркетів можна зібрати діаграму(рис. 1.4.), що складається з макетів сторінок низької деталізації.

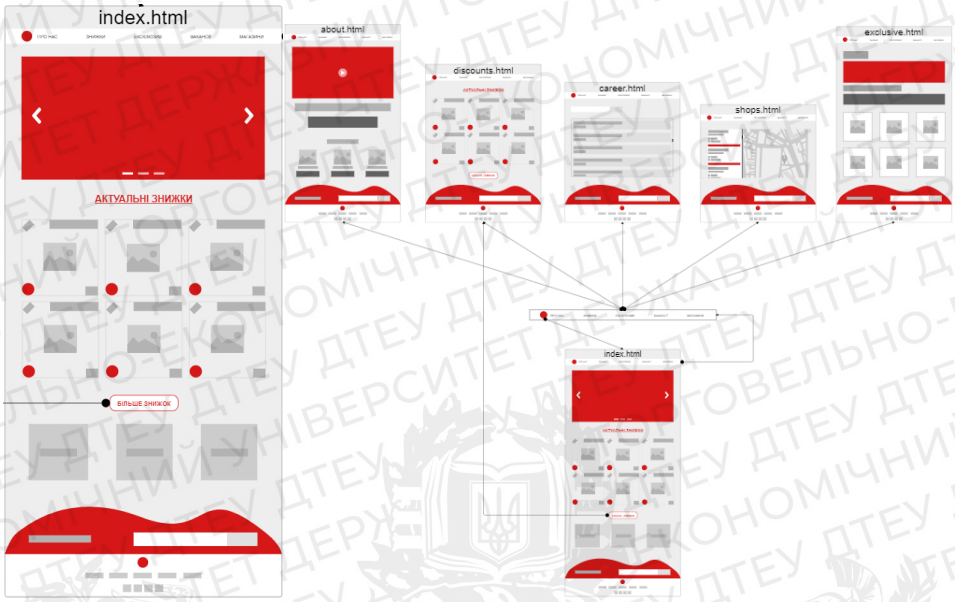

Рис. 1.4. Мапа сторінок.

Діаграма допоможе краще орієнтуватись в процесі розробки, особливо якщо всі етапи розробки виконує одна людина. Всі структурні блоки сторінки легко розрізняються, це сильно допоможе в процесі створення розмітки.

Результати розділу: в розділі були наведені короткі теоретичні відомості про загальний процес розробки веб-сайтів, та описані деталі етапу проектування. На етапі проектування було проведене дослідження існуючих веб-сайтів супермаркетів. На основі отриманих даних вдалось виділити загальновживані блоки, а також корисні прийоми розробки і дизайну. Завдяки цим даним була створена діаграма, так звана «мапа сторінок», що складається з недеталізованих сторінок веб-сайту, з позначенням внутрішніх посилань для пересування між ними. Ця діаграма стане хорошим орієнтиром в процесі розробки клієнтської частини веб-сайту, але звісно не є відображенням фінального результату.

РОЗДІЛ 2 ЗАСОБИ І МЕТОДИ СТВОРЕННЯ ІНТЕГРОВАНОГО ВЕБ-РЕСУРСУ З КОРИСТУВАЦЬКИМ ІНТЕРФЕЙСОМ

2.1 Огляд інструментів для розробки клієнтської частини веб сайту.

З технічної точки зору клієнтська частина веб-сайту створюється в три етапи:

- Налаштування інструменту збірки;
- Верстка сторінок;
- Створення інтерактивних елементів;

Для усвідомлення важливості першого етапу необхідно поглибитись в тонкощі двох інших, адже основним завданням інструментів збірки(або таск-ранерів) є прискорення, та спрощення процесу розробки.

2.1.1 Верстка сторінок.

HyperText Markup Language(HTML) — базовий будівельний блок будь-якої веб-сторінки. Він визначає зміст і структуру веб-контенту. Інші технології окрім HTML, зазвичай використовуються для опису зовнішнього вигляду(CSS) або функціональності (JavaScript).

Гіпертекст – це не що інше, як посилання, які поєднують веб-сторінки або в межах одного веб-сайту, або між веб-сайтами. Посилання є фундаментальним аспектом веб середовища.

HTML використовує розмітку («markup») для відображення тексту, зображень і іншого контенту в браузері. Розмітка утворюється з спеціальних елементів. Всі вони мають свої особливості та призначення.

Елементи виділяються з поміж іншого вмісту сторінки за допомогою тегів(рис. 2.1.), що складаються з імені елемента оточеного «<» і «>». Наприклад: , , <p>, <h1>…<h6> і т. п.

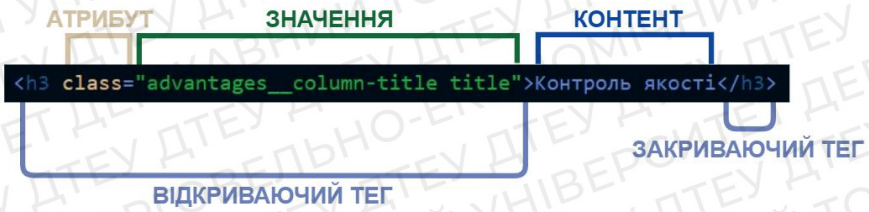

Рис 2.1 Структура HTML-розмітки

Cascading Style Sheets (CSS) — це мова ієрархічних правил (таблиць стилів), що використовується з метою представлення зовнішнього вигляду документу, написаного на HTML чи XML (включаючи різні мови XML, такі як SVG та XHTML). CSS описує, яким чином елемент повинен бути відображеним на екрані, папері, голосом чи з використанням інших медіа засобів.

Основна мета CSS — розповісти движкові браузера, як вимальовувати елементи сторінки — їх колір, позиціонування, форми і т. п. Його синтаксис створено слідуючи цій ідеї, він складається з наступних складових:

- Властивість — ідентифікатор дії, що буде застосована до елементу.
- Значення — описує, як саме властивість буде оброблена браузером. Кожне значення має обмежений набір допустимих значень, визначених формальними правилами, а також семантичний сенс, реалізований движком браузера.

Взаємодія між HTML та CSS здійснюється за допомогою **селекторів** — спеціальних позначень, які вказують на конкретну вибірку HTML елементів до яких потрібно застосувати блок властивостей. Відповідно до різноманіття тегів та їх параметрів існує багато видів селекторів: за назвою тегу(рис. 2.2.), класом, ідентифікатором, тощо. Таким чином утворюється структура схожа на сукупність об’єктів(сутностей загальновживаних в багатьох мовах програмування).

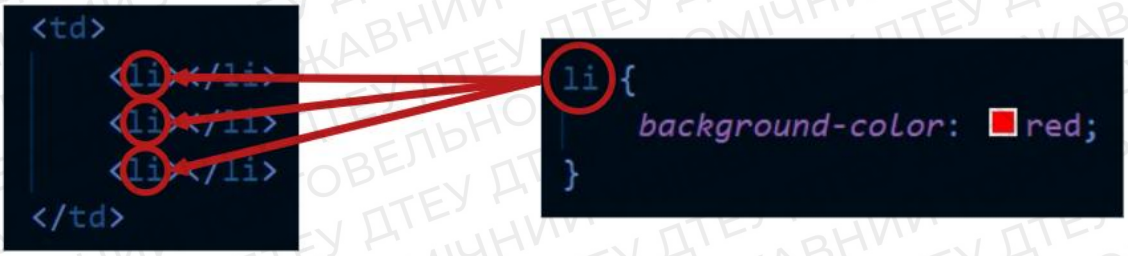

Рис 2.2. Взаємодія CSS та HTML(селектор за тегом).

HTML в поєднанні з CSS має дивовижні можливості, але світ не стоїть на місці і розробники постійно створюють нові інструменти для підвищення своєї ефективності. Яскравими прикладами таких інструменів є CSS препроцесори та CSS бібліотеки або фреймворки.

CSS препроцесор — це програма, яка має свій особливий синтаксис, але може генерувати з нього CSS код. Метою препроцесорів є розширення властивостей чистого CSS, шляхом додавання таких опцій як: домішки, вкладені правила, селектори спадкування та ін. Ці особливості полегшують роботу з CSS: спрощують читання коду і його подальшу підтримку.

Найуживанішими CSS препроцесорами є: Less, Sass та Stylus. Відрізняються вони за синтаксисом і функціональними можливостями. Для використання в проекті ми обрали Sass, зокрема його діалект Scss, що має більш звичний синтаксис. Sass написаний на мові Ruby, має досить велику спільноту розробників і забезпечує наступні базові функції:

- Використання змінних;
- Модульність;
- Вкладений синтаксис;
- Операції з рядками;
- Математичні вирази;

Крім того в нових версіях з’явились логічні вирази, функції, та цикли, які вивели можливості цього препроцесора на новий рівень.

CSS фреймворк — набір готових рішень, створений для спрощення роботи розробника, пришвидшення розробки і зменшення кількості помилок.

Найбільш популярними є: Bootstrap, Foundation, Materialize CSS та інші… Коротко пройдемось по найбільш вагомим з них.

Bootstrap — фреймворк, створений з метою полегшити розробку адаптивних веб-сайтів. В своєму складі має модульну сітку, побудовану на основі flexbox і готові реалізації часто вживаних блоків. Є розробкою компанії Twitter і за сумісництвом найуживанішим CSS фреймворком. Актуальна версія Bootstrap написана з використанням Sass.

Foundation – на даний момент другий за популярністю CSS фреймворк. Підходить для розробки масштабних проектів. Серед його користувачів є Facebook, eBay, Mozilla, Adobe та багато інших відомих компаній. Також побудований на Sass і має потужну JavaScript складову.

Materialize CSS – розробка компанії Google, пропонує набір готових компонентів в стилі Material Design. Використовує сітку Bootstrap.

Для використання в проекті був обраний фреймворк Bootstrap, як перевірений часом, потужний інструмент із зручною і змістовною документацією. З основних його переваг можна відмітити наступне:

- Добре продумана і перевірена часом адаптивна сітка;
- Готові реалізації інтерактивних компонентів з використанням JavaScript;
- Простота вивчення;

2.1.2 Створення інтерактивних елементів.

Крім HTML та CSS в розробці клієнтської частини веб-сторінок активно використовується мова JavaScript.

JavaScript — це мова програмування, що була створена з метою «зробити сторінки живими». Програми на цій мові називаються «скриптами». Вони можуть вбудовуватись в HTML і виконуватись автоматично при завантаженні сторінки. Скрипти розповсюджуються і виконуються як простий текст, їм не потрібна особлива підготовка для запуску.

JavaScript є повноцінною і потужною мовою програмування, що дає змогу робити веб-сайти більш функціональними. Його використання дає змогу створювати складні інтерактивні елементи і навіть SPA — веб-додатки, які працюють без необхідності перезавантаження сторінки, а велика частина логіки міститься на клієнтській стороні.

JavaScript дає змогу розробнику маніпулювати вмістом HTML сторінок завдяки DOM API – об’єктній моделі документу, яка дає змогу звертатись до елементів сторінки як до JavaScript об’єктів з своїми властивостями і методами.

В проекті буде частково використовуватись JavaScript бібліотека jQuery, яка є необхідною залежністю для інтерактивних компонентів Bootstrap.

JQuery – це бібліотека, що фокусується на полегшенні взаємодії з JavaScript та HTML. Вона дозволяє легко отримувати доступ до будь-якого елементу DOM, звертатись до атрибутів і змісту елементів, маніпулювати ними.

2.1.3. Інструменти збірки.

Результатом клієнтської розробки є велика кількість файлів різних форматів та ваги. В процесі використання веб-сайтів важливий кожен кілобайт, адже вага файлів веб-сайту сильно впливає на швидкість його завантаження. Крім того декілька JavaScript файлів корисно поєднувати в один, що дає змогу браузеру зберігати в кеші зкомпільований байт-код, це допомагає прискорити завантаження відвіданих раніше сторінок. І звісно файли препроцесорів не мають ніякого сенсу без етапу компіляції. Всі ці і багато інших рутинних задач допомагають вирішити таск-ранери, до того ж в автоматичному режимі.

Таск-ранер — невеликий додаток, що використовується для автоматизації задач, які доводиться виконувати в процесі розробки. Наприклад: мініфікація файлів CSS та JavaScript, компіляція файлів препроцесорів, оптимізація зображень, та ін.

Для розробки проекту було обрано таск-ранер Gulp, який добре підходить для верстки, та має великий набір корисних утиліт. Його налаштування виконується шляхом написання інструкцій на мові JavaScript. Ці інструкції визначають правила взаємодії утиліт між собою.

2.2 Системи управління базами даних

Всі досліджені в першому розділі веб-сайти супермаркетів мають одну схожу рису — потреба в збереженні, використанні, та зміні даних. Інформацію про знижки та адреси магазинів мережі потрібно десь зберігати, крім того це сховище повинно забезпечувати захищений та зручний доступ до даних. Ці можливості забезпечують системи управління базами даних.

СУБД(система управління базами даних) являє собою комплекс ПЗ, за допомогою якого можливо створювати бази даних і проводити над

ними різноманітні операції: оновлювати, видаляти, вибирати, редагувати і т. п. СУБД гарантує збереження, безпеку та цілісність даних і дозволяє видавати доступ до адміністрування бази даних.

В залежності від моделі даних СУБД бувають:

- мережевими;
- ієрархічними;
- реляційними;
- об’єктно-реляційними;
- об’єктно-орієнтованими;
- нереляційними(NoSQL);

Серед відомих СУБД можна виділити: MySQL, PostgreSQL, MongoDB.

Хорошим вибором для невеликого проекту без складної серверної логіки буде СУБД нереляційного типу. Такі системи легко масштабуються та краще підходять для простих запитів. Яскравим прикладом NoSQL систем є MongoDB.

Структура MongoDB складається з JSON-подібних документів(рис. 2.3.). MongoDB використовується в веб-розробці, зокрема, в рамках MERN(MongoDB Express React Node.js) та MEVN(MongoDB Express Vue Node.js) стеків. В якості мови запитів ця СУБД використовує JavaScript, що теж є значною перевагою.

Рис 2.3 Документ — головна одиниця структури MongoDB.

2.3 Інструменти для серверної розробки.

Для розробки серверної частини є широкий вибір мов та інструментів. Найчастіше в веб-сегменті використовуються наступні мови: PHP, Java, C#, JavaScript(Середовище Node.js), Python та ін. Враховуючи навички отримані в ході розробки клієнтської частини додатку буде доцільно використати середовище, яке потребуватиме меншого часу на вивчення. Тут на передній план виходить Node.js.

Node.js — це JavaScript-оточення побудоване на JavaScript-рушієві Chrome V8, який дозволяє транслювати виклики на мові JavaScript в машинний код. Середовище перш за все призначене для створення серверних додатків, але також існують проекти по створенню додатків для PC(Electron) і програмуванню мікроконтролерів.

Це середовище дає змогу використовувати одну мову для розробки обох частин веб-сайту, що є значною перевагою. А в поєднанні з фреймворком Express, Node э потужним інструментом, який допоможе швидко та якісно вирішити завдання проекту.

Express — мінімалістичний та гнучкий фреймворк для веб-додатків, фактично є стандартним каркасом для Node. Express є лише каркасом, отже сам по собі він має обмежений набір найважливіших функцій. Цей факт не робить його поганим інструментом, додаткові функції забезпечує використання npm пакетів.

NPM — це менеджер пакетів, який входить в склад Node. Складається з двох частин: CLI (інтерфейс командного рядка) — засіб для розміщення, та завантаження пакетів та онлайн-репозиторії, що містять JS пакети.

Процес розробки серверної частини сайту з використанням Node.js та Express поділяється на три головні етапи:

- Створення представлень (views);
- Налаштування маршрутів(routing);
- Створення моделей(models);

Views (Представлення) — особливі сутності з яких згодом генеруються html файли. Керується цей процес так званим «движком представлень»(view engine). У зв'язці з Express часто використовують наступні движки: Pug, Jade, Dust, Nunjucks, EJS, Handlebars, тощо… Відрізняються вони здебільшого за синтаксисом, але принцип роботи мають схожий. Для використання в проекті було обрано движок Handlebars, який має синтаксис більш схожий до звичайного HTML і легко інтегрується з Express(пакет express-hbs).

Routes (Маршрути) — визначають, як додаток відповідає на запит клієнта на конкретну адресу. Express в своєму складі має власний маршрутизатор, який підтримує основні методи маршрутизації, що відповідають методам HTTP: get, post, put, head, delete, options, trace, copy, тощо…

Models (Моделі) — відповідають за зберігання даних та їх структуру. Модель допомагає налаштувати взаємозв'язок між серверною частиною додатку, та базою даних. В проекті буде використано пакет mongoose, який дає змогу створювати об'єкти-моделі на основі схем(схематичне відображення документа, з зазначенням полів, та типів даних) і звертатись до них через налаштовані шляхи.

Результатом використання перелічених в цьому та минулих пунктах інструментів є досить складна система. Її приблизним відображенням є наступна схема(рис. 2.4.).

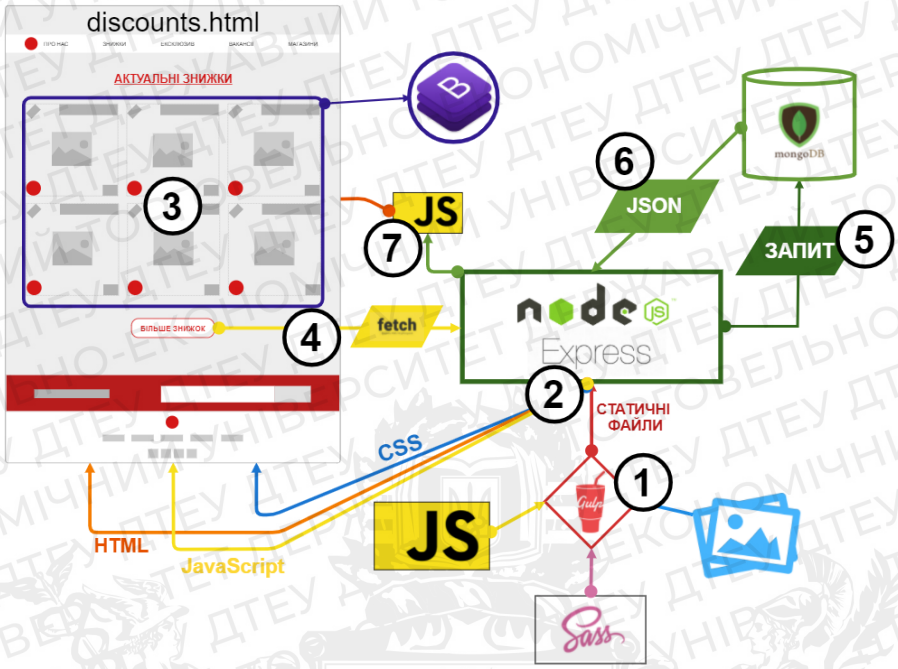

Рис 2.4 Взаємодія всіх складових на прикладі сторінки «Акції».

Опис алгоритму:

1. Створюються статичні файли, обробляються інструментом збірки, і потрапляють до серверної частини проекту;
2. На сервері створюється розмітка, до розмітки підключаються статичні файли і все це прямує на клієнт;
3. Завантажується розмітка і стилі сторінки, встановлюються обробники подій, запускаються необхідні скприпти.
4. Після кліку по кнопці відправляється запит на сервер.
5. З сервера відправляється запит до бази даних.
6. База даних відповідає даними у форматі JSON.
7. Дані відправляються на клієнт, де з використанням JavaScript створюється нова розмітка і додається до існуючої.

Результати розділу: в другому розділі роботи були наведені більш розгорнуті теоретичні відомості про всі етапи розробки веб-сайту, та описані технології обрані для використання в ході розробки. Взаємодія

обраних технологій була зображена на схемі, що відображає процес роботи сторінки «Знижки», а саме – принцип завантаження інформації про нові акційні товари, та її відображення на сторінці.

РОЗДІЛ 3 РОЗРОБКА ВЕБ-САЙТУ

3.1 Налаштування gulp збірки.

Процес розробки почнемо з налаштування збірки, яка буде у фоновому режимі обробляти статичні файли.

3.1.1 Утиліти.

Функціональні можливості збірки напряму залежать від потреб проекту, отже для початку необхідно виділити список завдань, що потребують автоматизації. Спираючись на сучасні реалії розробки, та обрані технології отримаємо наступний перелік завдань з зазначенням обраних утиліт для їх вирішення:

CSS:

- Мініфікація та очищення css — gulp-clean-css;
- Компіляція файлів препроцесора — gulp-sass;
- Автоматичне встановлення вендорних префіксів — gulp-autoprefixer;
- Групування медіазапитів — gulp-group-css-media-queries;

JS:

- Мініфікація файлів скриптів — gulp-uglify-es;
- Адаптація javascript коду для старих браузерів — gulp-babel;
- Обробка медіафайлів:
- Стискання зображень — gulp-tinypng;
- Компонування svg спрайтів — gulp-svg-sprite;
- Призначення деяких з них потребують пояснень:
- Gulp-babel: адаптація відомого інструмента Babel для використання в якості gulp утиліти. Babel — це javascript-транскомпілятор, який

дає можливість перетворювати код нових версій JavaScript (ES6+) у код сумісний з старішими двигунами JavaScript.

- Gulp-svg-sprite: утиліта, яка дає змогу збирати іконки формату svg в один файл, що значно спрощує їх використання. Код кожної іконки записується в окремий тег symbol з унікальним id.
- Gulp-autoprefixer: утиліта для автоматичного встановлення вендорних префіксів — приставок до css властивостей, що забезпечують їх підтримку браузерами, в яких певна функція ще не впроваджена на постійній основі.

Крім того необхідно використати деякі допоміжні утиліти, що забезпечують роботу з файловою системою.

3.1.2 Принцип роботи.

Принцип роботи gulp полягає у постійному слідкуванні за обраними типами файлів у фоновому режимі. Коли файл з зазначеним розширенням змінив свій зміст, до нього застосовується заздалегідь підготована функція, яка автоматично виконує зазначені в її тілі інструкції. Наприклад функція обробки scss файлів(рис. 3.1.).

```
const scss = () => {
    return src(path.src.scss) //Робимо вибірку scss файлів з директорії ресурсів
        .pipe(
            scss({ //Перетворюємо scss код в css
                outputStyle: 'expanded',
            })
        )
        .pipe(groupMedia()) //Групуємо медіазапити
        .pipe(
            autoprefixer({
                overrideBrowserslist: ['last 5 versions'], //Префікси для останіх 5-ти версій браузерів
                cascade: true,
            })
        )
        .pipe(dest(path.build.css)) //Відправляємо css в папку dist
        .pipe(cleanCss()) //Стискаємо та очищуємо css
        .pipe(
            rename({
                extname: '.min.css', //Додаємо до мтиснутого файлу префікс min
            })
        )
        .pipe(dest(path.build.css)) //Відправляємо його в папку dist
        .pipe(browserSync.stream()); //Перезавантаження браузера
};
```

Рис 3.1 Функція обробки scss.

Слідкування за файлами виконується за допомогою метода watch, який приймає в якості аргументів шлях до файлів, та функцію обробки.

Порядок виконання інструкцій регулюють методи series(виконує функції одна за одною), та parallel(виконує функції паралельно).

На етапі проектування було вирішено використовувати інструмент збірки таким чином, щоб скомпільовані файли відправлялись одразу в директорію для статичних файлів серверної частини. Так буде значно зручніше вести розробку обох частин проекту, та тестувати їх роботу. До того ж обробка файлів розмітки буде проводитись за допомогою серверного движка представлень, а отже додаткова їх обробка з використанням gulp не має сенсу.

3.2 Наповнення бази даних.

СУБД MongoDB буде використана для зберігання даних про акційні товари, та магазини мережі. Дані були отримані шляхом парсингу і записані в два окремих JSON файли(рис. 3.2.).

Рис 3.2. Вміст файлів з інформацією про товари та магазини

Для зручності використаємо хмарний сервіс MongoDB Atlas, який дає змогу зберігати дані на віддаленому сервері. Після створення бази

даних, та налаштування доступу до неї залишається лише підключитись і наповнити її.

Для наповнення використаємо додаток MongoDB Compass, який має зручний графічний інтерфейс, та функцію завантаження JSON документів. Для підключення використаємо посилання, зазначене в налаштуваннях бази даних.

В результаті отримаємо дві окремі колекції, що наповнені даними з завантажених JSON файлів. Кожний документ колекції тепер має власний унікальний ідентифікатор.

3.3 Серверна розробка.

3.3.1. Необхідні пакети та точка входу в проект.

Процес розробки серверної частини додатку почнемо з створення точки входу в проект, та завантаження необхідних пакетів. Точкою входу буде файл index.js, саме в ньому будуть міститись загальні налаштування проекту.

Завантажимо необхідні пакети за допомогою CLI пакетного менеджера. Інформація про завантажені пакети зберігатиметься в файлі package.json(рис. 3.3.).

Рис 3.3. Вміст файлу package.json

Дивлячись на структуру файлу paskage.json, можна помітити, що пакети розподілені на дві групи. Залежності(dependencies) і Залежності для розробки(devDependencies). Позначення є лише семантичним і означає, що пакети з групи «devDependencies» не повинні потрапити в готовий продукт. Нижче зазначені короткі відомості про пакети, та команди для запуску проекту.

Пакети, що будуть використовуватись в додатку:

- Express: фреймворк;
- Express-handlebars: движок уявлень;
- Mongoose: пакет для взаємодії з базою даних;
- Nodemailer: пакет для відправки поштових повідомлень;
- Nodemailer-sendgrid-transport: додатковий пакет для використання поштового сервісу Sendgrid;
- Nodemon: пакет для автоматичного перезапуску скриптів;
- Request: пакет для відправки http запитів.

Команди для запуску проекту:

1. Start – запускає проект звичайним способом.

2. Dev – запускає проект за допомогою nodemon, отже при змінах в файлах проект буде запускатись заново.

Завантаживши необхідні пакети можна перейти до процесу налаштування їх взаємодії. Розглянемо вміст точки входу в проект(рис. 3.4.).

Рис 3.4. Вміст файлу index.js

1. Підключення модулів до проекту, шляхом створення констант. Остання константа «keys» посилається на файл, що містить паролі і токени для підключення бази даних, та поштового сервісу.
2. Константа «app» містить об’єкт додатку. А константа «PORT» містить значення, що відповідає значенню змінної «PORT» середовища, в якому запущено додаток. Якщо змінна не встановлена, додаток буде запускатись за портом 3000.
3. Об’єкт, що містить шляхи(routes).
4. Налаштування движка уявлень.
5. Реєструємо в додатку папку з статичними файлами, та налаштовуємо аналізатор запитів з метою розпізнавання об’єкта вхідного запиту в якості рядків або масивів.

6. Реєструємо шляхи(routes) в додатку.
7. Функція запуску додатку, що запускається одразу після створення. Всередині функції асинхронно створюємо підключення до бази даних, після чого запускаємо додаток за вказаним раніше портом.

Налаштування проекту завершимо(рис. 3.5.) створенням структури папок. Необхідно створити папки для уявлень(views), моделей(models), шляхів(routes), статичних файлів, та паролів.

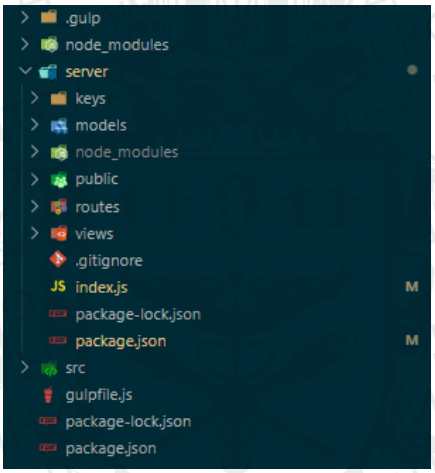

Рис 3.5. Структура папок.

3.3.2. Створення шляхів(routes).

Шляхи визначають поводження веб-додатку залежно від запитів користувача. Принцип роботи схожий до браузерних обробників подій — коли користувач посилає запит за певним посиланням, спрацьовує функція коллбек(функція, яка виконується після завершення роботи іншої функції). Шляхи згрупуємо за сторінками сайту і розділимо на різні файли.

Скрипт починається з підключення маршрутизатора, потім слідує підключення моделі і ініціалізація маршрутизатора. Далі будуть наведені приклади деяких шляхів додатку(рис. 3.6.).

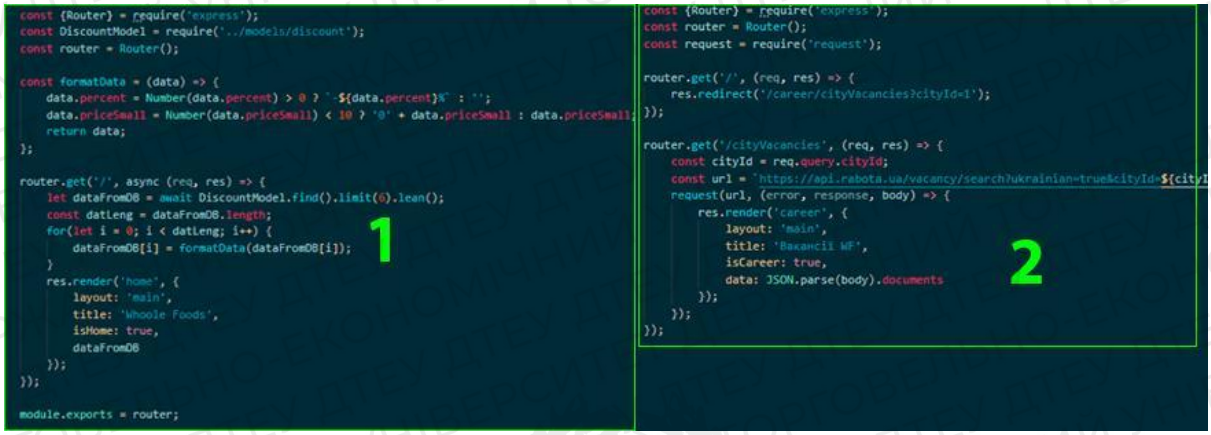

Рисунок 3.6. Приклади файлів шляхів.

1. Запит типу «GET» за шляхом «/» викликає функцію коллбек. В тілі функції ми отримуємо необхідні дані з бази даних(перші шість документів з інформацією про товари). Далі перебираємо отриманий масив об’єктів і застосовуємо до кожного об’єкта функцію форматування. За допомогою метода «render» об’єкта «res» ми відправляємо користувачу сторінку, оброблену движком уявлень. В об’єкті передаємо дані, що будуть використані движком уявлень.
2. Запит типу «GET» за шляхом «/» викликає функцію коллбек. В тілі функції використовується метод «redirect» і користувач перенаправляється на інший шлях(вакансії міста Київ). Запит типу «GET» за шляхом «/cityVacancies» викликає функцію коллбек. В тилі функції отримуємо ідентифікатор міста і відправляємо запит на сторонній API. Отримуємо дані про вакансії і використовуємо функцію «render», передаючи в якості параметра об’єкт, що включає отримані дані і деякі додаткові параметри.

В наведених прикладах чітко прослідковується наступний алгоритм: встановлення обробника(з вказанням типу запиту та шляху), обробка запиту(з використанням коллбєк функції), відправлення відповіді(після обробки дані відсилаються на клієнт у форматах html, json, тощо…).

3.3.3 Представлення(views). Динамічна розмітка з використанням Handlebars.

Handlebars дозволяє використовувати дані відправлені в через метод «render» при верстці шаблонів. Також є можливість виділяти блоки розмітки в окремі файли, та автоматично додавати їх до необхідних сторінок, таким чином значно легше додавати «header», «footer» та інші блоки, що часто повторюються. Крім того в Handlebars є власні реалізації циклу for, та операторів розгалуження, що є досить корисним в процесі верстки динамічних шаблонів.

Всі сторінки створюються на основі головного шаблону «layout». В секцію «body» після процесу рендерингу запишеться весь вміст шаблону, а елементам «head», «header» та «footer» відповідають так званим «partials» — винесені в окремі файли блоки сайту, що часто повторюються.

Використовуючи функціональні особливості Handlebars створимо початкову розмітку сторінок, яка слугуватиме каркасом для наступних етапів клієнтської розробки. Адже саме на основі розмітки будується DOM дерево – об’єктна модель документу, що дає змогу маніпулювати вмістом сторінки з використанням JavaScript. Розмітка буде написана відповідно до стандарту HTML5, з використанням семантичних тегів. Розглянемо процес написання розмітки на прикладі головної сторінки(рис. 3.7.).

Перш за все потрібно поділити сторінку на структурні блоки, спираючись на низькодеталізований макет. Сторінка міститиме наступні блоки: слайдер(банер), блок знижок, особливі пропозиції.

Слайдер: зовнішній контейнер представлений елементом «div» з унікальним ідентифікатором, на цей ідентифікатор будуть посилатись елементи керування. Всередині нього міститься контейнер слайдів, та елементи керування, що представлені елементами «a» з атрибутом «href»,

який відповідає ідентифікатору зовнішнього контейнера. Атрибут «role» дає пристроям зчитування зрозуміти, що даний елемент виконує роль кнопки. Кожен слайд міститься в елементі «a» з посиланням на сторінку з повним переліком акційних товарів. Таким чином клік користувача по тілу слайдера одразу переправить його на цільову сторінку. Всередині слайд складається з контейнера заголовків, та контейнера ціни, ці блоки містять коротку текстову інформацію про акційний товар.

Блок знижок: складається з заголовку(h1), та контейнера(div). Контейнер містить ненумерований список(ul) наповнений елементами «li». Кожен елемент списку містить дані про окремий акційний товар: назва, зображення, ціна, знижка. При заповненні списку використаємо цикл «each», перебираючи дані передані при обробці шляху(route).

Особливі пропозиції: містить контейнер «div», всередині якого міститься список посилань(ul) на сторінки особливих пропозицій(li > a).

Рис. 3.7. Розмітка головнної сторінки.

Загалом процес створення шаблонів не сильно відрізняється від звичайної верстки і полягає у відтворенні структури макету з використанням HTML тегів. Можливість використання динамічних даних забезпечують вбудовані в Handlebars конструкції: цикли, розгалуження, тощо… В процесі верстки важливо не забувати про семантичні теги і aria-атрибути, що покращують сприйняття змісту сайту пристроями зчитування.

3.3.4. Моделі(models).

Моделі створені з використанням пакета Mongoose являють собою об’єкти з великою кількістю методів, що відповідають командам для запитів СУБД MongoDB. Моделі створюються на основі схем, що визначають дані моделі: властивості, типи даних, тощо…

В якості типу даних можливо вказати наступні значення: String, Number, Date, Buffer, Boolean, Mixed, Objectid, Array, Decimal128, Map. За допомогою параметра «default» можемо вказати значення властивості за замовчуванням. Після створення схема передається в метод «model» в якості аргументу, а метод повертає модель.

Отримана модель дає змогу відправляти запити до колекції, зазначеної в схемі. Щоб точно позначити конкретну колекцію необхідно передати в конструктор класу «Schema» об’єкт налаштувань з параметром «collection».

Як приклад нижче наведено код моделі «Discount», що посилається на однойменну колекцію, яка містить дані про акційні товари. Далі модель буде імпортуватись в файли шляхів(routes) і використовуватись для отримання необхідних даних з колекції.

3.4. Стилізація з використанням класів Bootstrap, та препроцесора SASS.

3.4.1. Використання Bootstrap.

Bootstrap вносить деякі зміни в процес верстки, адже більша частина стилізації відбувається на етапі написання розмітки, завдяки різноманіттю готових класів.

Основою фреймворку Bootstrap є модульна сітка, яка дає змогу легко маніпулювати поводженням блоків на різних розмірах екранів. Розміри блоків задаються відносно сітки з дванадцяти колонок і можуть бути різними на декількох розширеннях екранів завдяки брейкпоінтам, що відповідають розмірам екранів п’яти основних видів пристроїв: xs – (ширина 576px і менше), sm – (ширина більше 576px), md – (ширина більше 768px), lg(ширина більше 992px), xl(ширина більше 1200px).

Крім того Bootstrap має в своєму складі готові реалізації типових елементів сайтів: кнопки, навігаційні меню, форми, тощо… Готові компоненти легко модифікуються за допомогою класів-утиліт, та власних стилів. Кольори, тіні, інтервали між елементам і багато іншого можна змінювати лише додавши певний клас до елемента.

Після додавання класів деякі блоки вже мають належний вигляд і не потребують додаткових маніпуляцій, крім того вони добре адаптуються до різних розмірів екранів(рис. 3.8.).

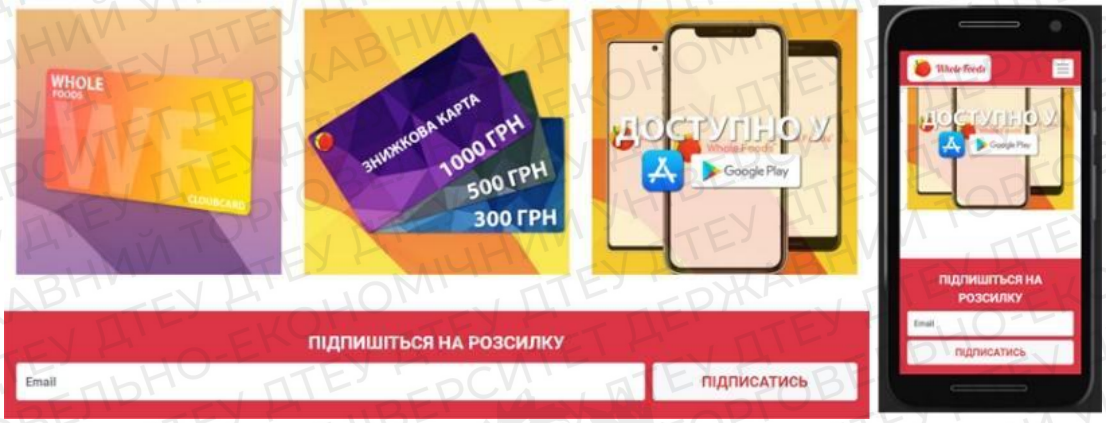

Рис. 3.8. Блоки сайту на різних розширеннях екранів.

Та все ж деякі блоки потребують більш точного налаштування, в цьому допоможе додавання власних стилів.

3.4.2. Власні стилі. Препроцесор SCSS.

Роль власних стилів полягатиме в доведенні фінальних штрихів, тому доцільно буде слідувати методології Atomic CSS, яка використовується у Bootstrap, та схожих до нього бібліотеках і фреймворках. В цій методології стилі не прив’язані до конкретного блоку і існують окремо. Такий підхід дає змогу змінювати вміст сайту без втручання в файли стилів, як і було зазначено в минулому підпункті.

Додамо класи для розмірів шрифтів, фонових кольорів, та тощо... В процесі написання класів використаємо особливості препроцесора SCSS: наслідування, вкладеність, змінні, та ін. Адаптивність текстових елементів реалізуємо за допомогою медіазапитів.

Завершивши стилізацію веб-сайту з використанням власних стилів отримаємо бажаний вигляд сторінок. Приклади сторінок наведено нижче(рис. 3.9.).

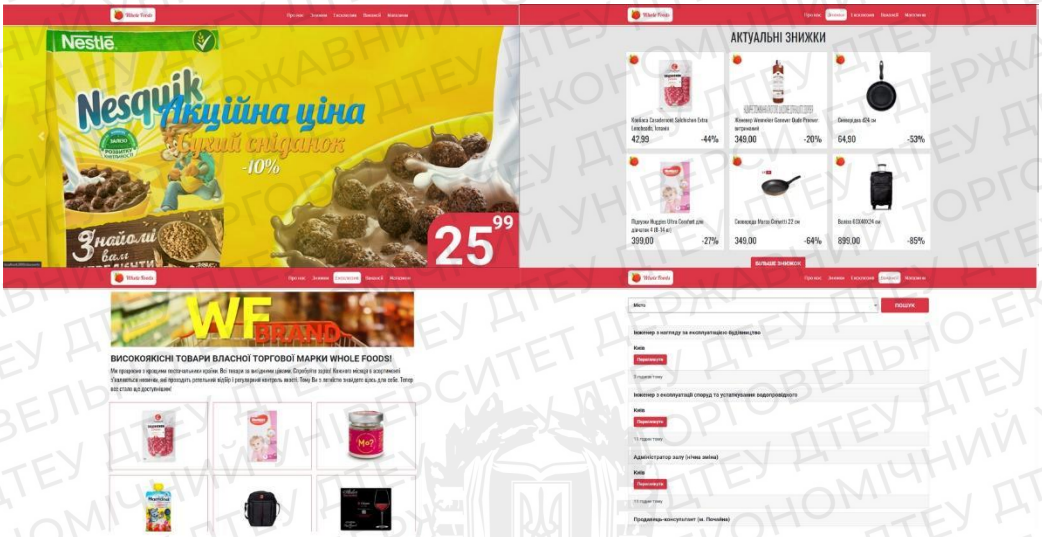

Рис. 3.9. Результат стилізації сторінок.

3.5 Додавання інтерактивних елементів.

Фінальним етапом розробки веб-сайту буде створення інтерактивних елементів з використанням мови JavaScript. На цьому етапі потрібно виконати три основних завдання:

- Список міст для пошуку на сторінці «Вакансії»;
- Завантаження нових товарів по кліку для сторінки «Знижки»;
- Функціонування інтерактивної мапи на сторінці «Магазини»;

3.5.1. Пошук вакансій.

Найпростішим в реалізації буде випадаючий список міст, з нього і почнемо. Принцип роботи сторінки пошуку вакансій полягає у відображенні даних отриманих в результаті відправлення запитів на сторонній API. Для того, щоб отримати список вакансій по конкретному місту посилання запиту має містити ідентифікатор міста. Наприклад: https://api.rabota.ua/vacancy/... cityId=1, де значення параметру «cityId» відповідає ідентифікатору міста Київ в базі даних сервісу з пошуку вакансій. Запити на API відправляються через серверну частину веб-сайту, це зроблено з метою підвищення швидкодії. Такий підхід дає змогу

використовувати серверний движок представлень для відображення отриманих даних, що зазвичай працює значно швидше ніж маніпуляції з DOM на клієнтській стороні додатку. Вигідність цього рішення буде добре помітна при збільшенні масивів даних.

Отже для завантаження нових вакансій буде достатньо переправити користувача на необхідне посилання, іншу частину роботи зроблять серверні скрипти і движок представлень.

Створимо масив об’єктів, кожен об’єкт буде містити ключі name(назва міста) і id(ідентифікатор). Випадаючий список буде заповнюватись за основі цього об’єкту. З точки зору розмітки список буде представлений у вигляді елементу «select», всередині нього будуть міститись елементи «option» з параметром «value», значення якого відповідатиме ідентифікатору, а текстовий вміст елементів буде відображати назви міст.

При відправці форми дані відправляться на сервер, ідентифікатор запишиться в посилання запиту і відправиться запит на сторонній API. Нижче наведено код скрипту, для відображення випадаючого списку(рис. 3.10.).

```
const container = document.querySelector('.js-vac-container'),
    dropdown = document.querySelector('.js-dropdown');

if(container !== null) {
    cities.forEach(city => {
        dropdown.innerHTML += city.name === 'Місто' ?
        `<option disabled selected>${city.name}</option>`:
        `<option value=${city.id}>${city.name}</option>`;
    });

    if(container.children.length < 2) {
        document.querySelector('.js-empty').style.display = "flex";
    }
}
```

Рис. 3.10. Основний код скрипту відображення списку міст.

3.5.2. Завантаження нових товарів.

На сторінці «Знижки» за замовчуванням відображаються перші шість акційних товарів. Для завантаження наступних шести товарів користувач має натиснути на кнопку. Після натискання спрацьовує подія «click», спрацьовує її обробник і запускає функцію зворотного виклику. В тілі функції міститься код для відправки HTTP запиту з використанням fetchAPI – сучасного інтерфейсу для роботи з HTTP запитами. Крім того кожен клік інкрементує змінну-лічильник, яка записується в посилання запиту у вигляді: /discounts/${cnt}.

Після отримання даних запускається функція «render», яка перебирає масив даних і викликає функцію заповнення даними певного HTML шаблону. Код нових шести елементів міститься в змінній «productsHTML», яка по завершенні виконання циклу додається до вмісту контейнера товарів.

Коли дані на сервері закінчились(масив об’єктів з даними має довжину менше шести), кнопка перетворюється на посилання, яке веде на верх сторінки. Код функції «render» наведено нижче(рис. 3.11.).

```
const render = (data, container) => {
    const datLeng = data.length;
    let productsHTML = '';

    if(datLeng < 6) {
        transformButton();
    }
    for(let i = 0; i < datLeng; i++) {
        productsHTML += insertData(data[i]);
    }
    container.innerHTML += productsHTML;

    setTimeout(() => {
        document.querySelectorAll('.js-product').forEach(li => {
            li.classList.remove('faded');
        });
    }, 300);
};
```

Рис. 3.11. Функція «render».

3.5.3. Інтерактивна мапа.

Код інтерактивної мапи є найбільш об’ємним і може бути розділеним на дві частини. Перша – логіка роботи елементу, що блокує можливість взаємодії з мапою для мобільної версії сайту. Оскільки мапа займає майже всю висоту і ширину екрану смартфону, буде доцільно додати функцію запобігання випадковим натисканням. Саме цей елемент запобігатиме випадковим натисканням і може бути прибраний, якщо користувач торкнеться мапи двома пальцями.

Функція «isTouchable» звіряє властивість «userAgent» об’єкту «navigator» з переліком ідентифікаторів мобільних пристроїв у вигляді регулярного виразу і повертає значення типу «bool». Так ми визначаємо чи має пристрій користувача можливість керування дотиком.

Функції «addTouchBlocker» і «removeTouchBlocker» відображають або приховують цей елемент в залежності від того, до якого елементу торкнувся користувач і скільки точок дотику він використав. Дві точки дотику по прихованому блокуючому-елементу – приховати, доторкнувся до елементів за межами мапи – показати.

Друга частина відповідає функціоналу самої мапи. В якості API мапи використовується 2gisAPI, яке було обране через безкоштовність, зручність і доступність документації.

Функція «initMap» створює мапу відцентровану за вказаними координатами(місто Київ) і повертає об’єкт всередині Promis. Це пов’язано з тим, що перед відображенням мапи відправляється асинхронний запит на сервери 2gis і на отримання відповіді потрібен невизначений проміжок часу. Всі необхідні методи для взаємодії з мапою знаходяться саме в цьому об’єкті.

Після відображення полотна мапи необхідно отримати дані про магазини(координати на мапі, адреси, тощо). За це відповідає функція «loadShopsData», яка відправляє на сервер запит і отримує у відповідь масив об’єктів з даними про магазини.

На сторінці вже є список магазинів, заздалегідь відмальований движком представлень і полотно мапи, залишилось додати на мапу маркери і модифікувати дані так, щоб в кожному об’єкті магазину містилась інформація і про елементи сторінки, що йому відповідають.

Функція **«addDOMLinks»** приймає масив даних і список DOM елементів. Вона перебирає масив даних, і додає до кожного об’єкту магазину додаткові параметри: елемент зі списку магазинів, маркер, спливаюча підказка. По завершенню функція повертає масив модифікованих даних.

Для забезпечення інтерактивності створені «функції-слухачі», які додають обробники подій взаємодії з елементами мапи. При натисканні на кнопку «на мапі» мапа буде центруватись на необхідному маркері і показувати спливаючу підказку, натискання на маркер зцентрує мапу на ньому і прогорне список магазинів до пов’язаного з цим маркером, тощо…

Останнім штрихом є функція «filter», яка зчитує дані з поля вводу, обирає об’єкти, що підходять під критерій пошуку і повертає відфільтрований масив даних. Функція спрацьовує кожен раз, коли користувач вводить дані в поле.

Узагальнити алгоритм можна наступним чином:

- Ініціалізація блокуючого елементу(якщо необхідно);
- Відображення полотна мапи;
- Отримання даних про магазини;

- Отримання посилань на необхідні DOM елементи;
- Запуск обробника взаємодії з полем вводу;
- Модифікація масиву даних;
- Відображення інтерактивних елементів та прослуховування взаємодії з ними.

Нижче зображено вигляд мапи на екранах різного розміру(рис. 3.12.).

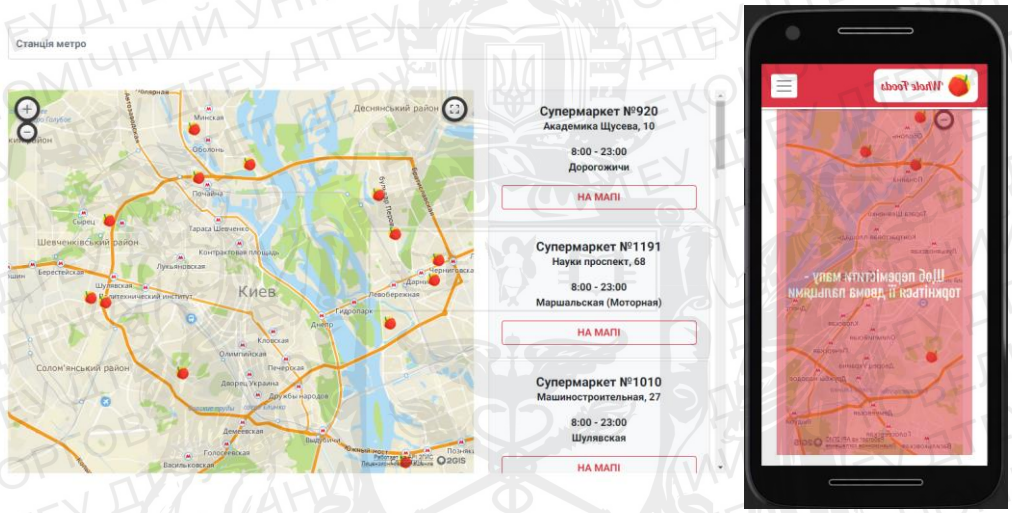

Рис. 3.12. Інтерактивна мапа.

Результати розділу: в цьому розділі було розглянуто всі етапи розробки рекламного веб-сайту супермаркету. Ми розробили серверну частину веб-сайту з використанням фреймворку Express, та зібрали базу даних під управлінням MongoDB. Клієнтська частина додатку була створена з використанням фреймворку Bootstrap, препроцесору SCSS, та чистого JavaScript. Отриманий в результаті виконаної роботи веб-сайт є адаптивним, має привабливий та зручний інтерфейс користувача і хорошу швидкодію.

ВИСНОВКИ

У ході нашого дослідження була проведена робота з розробки інтегрованого веб-ресурсу для пошуку цін на товари в онлайн-магазинах з мапою. Ми успішно досягли поставлених завдань та отримали певні результати, які варто відзначити. Спочатку ми провели аналіз існуючих методів та інструментів для пошуку цін на товари в онлайн-магазинах. Це надало нам необхідну базу знань та розуміння проблематики. Далі ми розробили ефективний алгоритм інтеграції даних про ціни товарів з різних джерел. Цей алгоритм дозволяє збирати, обробляти та інтегрувати інформацію про ціни з різних онлайн-магазинів.

Наступним етапом було розроблення веб-ресурсу, який інтегрує зібрані дані про ціни та відображає їх на мапі. Це дозволяє користувачам зручно порівнювати ціни на товари у різних магазинах та знаходити найкращі пропозиції в різних регіонах.

Ми успішно протестували розроблений веб-ресурс, проведши експерименти з реальними користувачами. Отримані результати підтвердили ефективність та коректність роботи ресурсу. Користувачі оцінили зручність інтерфейсу, швидкість та точність надання інформації про ціни на товари. Завдяки розробленому веб-ресурсу, покупці отримали зручний інструмент для пошуку найкращих цін на товари в онлайн-магазинах. Вони можуть значно економити час та зусилля, отримуючи доступ до актуальної інформації про ціни та розташування магазинів на мапі.

Отже, наш дипломний проект "Розробка інтегратора даних по цінам товарів в онлайн-магазинах" був успішно реалізований, а ми досягли певних результатів. Розроблений веб-ресурс дозволяє полегшити процес

пошуку найкращих цін на товари в онлайн-магазинах та сприяє зручному та ефективному онлайн-шопінг досвіду для користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Стаття про веб-розробку. [Електронний ресурс] – Режим доступу до ресурсу: https://numpy.org/doc/stable/.
2. Стаття про статичні веб-сайти. [Електронний ресурс] – Режим доступу до ресурсу: https://puzzleweb.ru/php/00_teacher.php.
3. Стаття про динамічні веб-сайти. [Електронний ресурс] – Режим доступу до ресурсу: https://puzzleweb.ru/php/00_teacher2.php.
4. Стаття про розробку інтерфейсів. [Електронний ресурс] – Режим доступу до ресурсу: https://clck.ru/L5vZm.
5. Стаття про HTML. [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/HTML.
6. Стаття про CSS. [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/CSS.
7. Стаття про CSS препроцесори. [Електронний ресурс] – Режим доступу до ресурсу: https://clck.ru/NmoQB.
8. Короткий опис відомих CSS фреймворків. [Електронний ресурс] – Режим доступу до ресурсу: https://clck.ru/VA3SL.
9. Вступ до веб-посібника з мови JavaScript. [Електронний ресурс] – Режим доступу до ресурсу: https://learn.javascript.ru/intro.
10. Стаття про основи jQuery. [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/38208/.
11. Стаття про переваги Gulp. [Електронний ресурс] – Режим доступу до ресурсу: https://frontender.info/no-need-to-grunt-take-a-gulp-of-fresh-air/
12. Стаття про СУБД. [Електронний ресурс] – Режим доступу до ресурсу: https://itglobal.com/ru-ru/company/glossary/subd-sistema-upravleniya-bazami-dannyh/.

13. Офіційний сайт Node.js. [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/uk/.
14. Офіційний сайт MongoDB. [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/2.
15. Офіційний сайт Express. [Електронний ресурс] – Режим доступу до ресурсу: https://expressjs.com/ru/.
16. NPM. [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/.
17. Стаття про методологію Atomic CSS. [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/432586/.