Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра комп’ютерних наук

Пояснювальна записка

до магістерської кваліфікаційної роботи
на тему:

Інформаційна технологія бронювання місць в ресторанах: клієнтська частина

Виконав: студентка 1 курсу, групи 1КН-19м
Спеціальності 122 –
«Комп’ютерні науки»

Янісевич А.В.
(прізвище та ініціали)

Керівник ст. вик. Петришин С.І.
(прізвище та ініціали)

Рецензент д.т.н., проф кафедри ПЗ Ліщинська Л.Б.
(прізвище та ініціали)

Вінниця - 2020 року

АНОТАЦІЯ

Робота присвячена створенню інформаційної технології розробки клієнтської частини сервісу бронювання місць в ресторані. Метою магістерської роботи є покращення роботи додатку за рахунок використання сучасних технологій для розробки клієнтської частини, що дасть змогу підвищити швидкість завантаження додатку та його надійність для клієнтів.

Проаналізовано переваги та недоліки кожного з них, створений список вимог. Розроблена безспосередньо сама клієнтська частина та проведена інтергація з серверною частиною.

Ключові слова: інтерфейс, клієнтська частина, React, React Native, Java Script.

ABSTRACT

The work is devoted to the creation of information technology for the development of the client part of the restaurant reservation service. The purpose of the master's thesis is to improve the application by using modern technologies to develop the client part, which will increase the download speed of the application and its reliability for customers.

The advantages and disadvantages of each of them are analyzed, the list of requirements is created. The client part itself was developed and integrated with the server part.

Keywords: interface, client part, React, React Native, Java Script

ЗМІСТ

ВСТУП

Актуальність теми дослідження. На сьогоднішній день неможливо уявити своє життя без використання мережі інтернету. Людство все більше відкриває для себе нові технології та активно використовує їх. На виробництві, чи медичній сфері, при створені витвору мистецтва, чи редагуванню фотографій, зберіганні документації і найбільше у повсякденному житті.

Раніше, щоб оплатити квитанцію, потрібно було простояти довгий час у черзі в банку. Для перекладу тексту доводилося діставати важкий словник, а іноді і зовсім йти за ним в бібліотеку. Список покупок був обмежений тим, що завозили в магазини міста. Зараз же можна і рахунки оплатити, і переклад зробити, і купити, що тільки не захочеться, - і все це за допомогою інтернету і численних онлайн-сервісів, які значно спрощують наше життя та відкривають нові можливості.

При розробці онлайн-сервісу основна увага приділяється його інтерфейсу, оскільки він повинен бути інтуїтивно зрозумілим користувачам, а також швидко завантажуватися з будь-яких пристроїв. Наприклад, дорожніми сервісами користувачі часто користуються вже перебуваючи в дорозі і завантажують їх на смартфонах і планшетах.

Саме тому, розробка якісної клієнтської частини онлайн-сервісу для бронювання місць в ресторані є дуже актуальною у наш час та і розробка практично будь-якого онлайн-сервісу.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою даної роботи є розширення функціональних можливостей системи для бронювання місць в ресторані та покращення клієнтської частиною шляхом якісної її розробки. Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати існуючі технології, методи і моделі розробки клієнтської частини онлайн-сервісів;

– сформулювати вимоги до програмного додатку для бронювання місць в ресторані та розробити ТЗ.

– спроектувати клієнтську частину програмного додатку для бронювання місць в ресторані;

– Провести тестування додатку та розробити інструкцію користувача.

Об’єкт дослідження – процес розробки клієнтської частини онлайн-сервісу для бронювання місць в ресторані.

Предмет дослідження – засоби розробки клієнтської частини онлайн-сервісу для бронювання місць в ресторані.

Методи дослідження. У роботі використані наступні методи наукових досліджень: методи моделювання та розробки інтерфейсу користувача, методи моделювання архітектури онлайн-сервісу.

Наукова новизна одержаних результатів полягає у тому, що:

буде розроблено клієнтську частину онлайн-сервісу для бронювання місць в ресторані, що дасть змогу бронювати столики, не відволікаючись на дзвінки до адміністрації ресторану, за допомогою браузера чи мобільного додатку.

Практичне значення одержаних результатів полягає у тому, що розроблений онлайн-сервіс для бронювання місць в ресторані допоможе бронювати столик в режимі онлайн з різних девайсів.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням різноманітних методів під час доведення наукових положень, строгим

виведенням аналітичних співвідношень, порівнянням результатів з відомими та збіжністю результатів математичного моделювання з результатами, що отримані під час впровадження розроблених програмних засобів.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗРОБКИ КЛІЄНТСЬКОЇ ЧАСТИНИ ОНЛАЙН-СЕРВІСУ

1.1 Аналіз проблеми розробки клієнтської частини онлайн-сервісу

Веб-додаток складається з клієнтської і серверної частин, тим самим реалізуючи технологію «клієнт-сервер».

Клієнтська частина реалізує користувальницький інтерфейс, формує запити до сервера і обробляє відповіді від нього.

Серверна частина отримує запит від клієнта, виконує обчислення, після цього формує веб-сторінку і відправляє її клієнту через мережу з використанням протоколу HTTP.

Веб-інтерфейс - це сукупність коштів, за допомогою яких користувач взаємодіє з веб-сайтом або веб-додатком через браузер. Веб-інтерфейси набули широкого поширення в зв'язку з ростом популярності всесвітньої павутини і відповідно - бути широко розповсюдженим веб-браузерів [1].

Одним з основних вимог до веб-інтерфейсів є їх однаковий зовнішній вигляд і однакова функціональність при роботі в різних браузерах.

Варіанти реалізації:

Класичним і найбільш популярним методом створення веб-інтерфейсів є використання HTML із застосуванням CSS і JavaScript. Однак різна реалізація HTML, CSS, DOM і інших специфікацій в браузерах викликає проблеми при розробці веб-додатків і їх подальшої підтримки. Крім того, можливість користувача налаштовувати багато параметрів браузера (наприклад, розмір шрифту, кольору, відключення підтримки сценаріїв) може перешкоджати коректної роботи інтерфейсу.

Інший (менш універсальний) підхід полягає у використанні Adobe Flash, Silverlight або Java-аплетів для повної або часткової реалізації призначеного для

користувача інтерфейсу. Оскільки більшість браузерів підтримує ці технології (як правило, за допомогою плагінів), Flash- або Java-додатки можуть виконуватися з легкістю. Так як вони надають програмісту більший контроль над інтерфейсом, вони здатні обходити багато несумісності в конфігураціях браузерів, хоча несумісність між Java або Flash реалізаціями на стороні клієнта може призводити до різних ускладнень.

Поряд з основними технологіями, є також і інші, що прискорюють процес розробки. Вони вже характерні для більш просунутих користувачів, активно застосовуються в топових веб-студіях таких як або при створенні сайтів професійними фрілансерами. В цілому, дозволяють працювати більш ефективно. це:

- SASS - використовується для поліпшення структури CSS-коду і збільшення зручності внесення правок в нього;
- jQuery - бібліотека мови JavaScript, яка допомагає в рази скоротити процес написання програмного коду, при цьому не на шкоду його призначенням;
- Emmet - спеціальний плагін під текстові редактори, за допомогою якого туди додаються готові конструкції для прискорення написання рутинного HTML / CSS коду.

В даний час набирає популярність новий підхід до розробки інтерфейсної частини веб-додатків, званий Ajax. При використанні Ajax інтерфейс не перезавантажуються цілком, а лише довантажують необхідні дані з сервера, що робить їх більш інтерактивними і продуктивними.

1.2 Аналіз переваг та недоліків використання онлайн-сервісів для бронювання місць в ресторані

У США значна частина відвідувачів ресторанів знайомі з сервісами онлайн-бронювання - за даними статистики, більше 37% гостей хоча б пару раз в своєму житті ними користувалися [2].

Ринок подібних додатків в Штатах також вельми розвинений, показником чого, наприклад є угоди з купівлі подібних проектів більшими IT- компаніями. Наприклад, не так давно індійський сервіс пошуку ресторанів Zomato за $ 52 млн викупив NexTable, американську онлайн-платформу бронювання столиків в ресторанах. Це було зроблено для того, щоб скласти конкуренцію сервісам OpenTable від Priceline і SeatMe від Yelp.

В Україні і країнах СНД подібні інструменти поки не користуються такою серйозною популярністю. Для чого ж потрібно ресторанам співпрацювати з онлайн-сервісами бронювання столиків?

1. Залучення додаткової аудиторії

Найбільш очевидна мотивація власників ресторанів, що штовхає їх на співпрацю з системами онлайн-бронювання - це надія на те, що з їх допомогою закладу вдасться залучити нових відвідувачів. Кількість подібних сервісів досить велике навіть на СНГ ринку.

Подібна конкуренція змушує такі проекти уважніше ставитися до вимог ресторанів - наприклад, додавати можливості перегляду аналітичної інформації з бронювання. При цьому такі стартапи часто мають власний маркетинговий бюджет на залучення аудиторії, що дозволяє ресторанам розраховувати на те, що про них дізнаються користувачі, які раніше не чули про даний конкретний заклад.

2. Заповнення залу в непопулярні години

Другий популярний спосіб використання сервісів бронювання - це залучення аудиторії для заповнення закладу в так звані «тихі» годинн, коли велика частина столиків пустує. За допомогою невеликих знижок або компліментів від шеф-кухаря, які пропонуються користувачам таких сервісів, ресторан стимулює їх до здійснення бронювання.

В такому випадку, бізнес практично нічого не втрачає - якщо користувач потім не прийде, це не призведе до збитків, оскільки відвідувачів в цей час і так

майже немає. А ось якщо він прийде і щось замовить, то заклад відразу отримає додатковий дохід.

3. Нові моделі монетизації

У тих же США конкуренція на ринку онлайн-бронювання сприяє розвитку нових моделей монетизації, які в майбутньому можуть перекочувати і на інші ринки. Приклад - так звані «квитки в ресторан», які продають сервіси типу Tock.

Якщо коротко, працює це так - все схоже на покупку квитка на концерт: користувач вибирає дату і час, ресторан, а потім оплачує все, що буде записане в замовлення. У підсумку потім клієнт просто приходить в заклад і отримує те, за що вже заплатив.

4. Зниження ризиків

Поширена проблема при бронюванні - клієнт, «забиває» столик на конкретний час, а потім просто не приходить, а ресторан зазнає збитків. Деякі онлайн-сервіси допомагають звести подібні проблеми до мінімуму - наприклад, при використанні сервісів на зразок Tock клієнт платить заздалегідь, а можливість потім «перепродати» бронювання обмежена, а це знижує ймовірність того, що гість в результаті не прийде.

5. Робота з власною аудиторією на сайті

Крім роботи зі сторонніми сервісами бронювання, деякі заклади йдуть і по шляху розміщення власних модулів на сторінках свого сайту. Плюси тут очевидні - не потрібно ні з ким ділитися даними про своїх клієнтів, умови бронювання також цілком визначаються керівництвом ресторану.

Мінуси також зрозумілі - залучати аудиторію на цю форму бронювання теж потрібно самостійно, і ніхто в цій справі допомагати не буде.

Незважаючи на те, що в певних випадках - особливо, якщо мова йде про популярному заклад - ресторанам може бути вигідне використання систем онлайн-бронювання, тут є і певні ризики. Наприклад, досить велике число таких

проектів являють собою простих спекулянтів, які заздалегідь бронюють столики «на себе», а потім перепродають ці броні користувачам інтернету [3].

За такою схемою працює, наприклад, сервіс Killer Rezzy - він заздалегідь бронює місця в ресторанах на популярні дати, а потім перепродає бронь з націнкою. В такому випадку ресторани не отримують ніякої додаткової вигоди (хоча клієнтам це може бути зручно, оскільки дається шанс потрапити в потрібне заклад в намічений святковий день). Більш того, менеджмент закладу може навіть не знати, що ціна броні була завищена.

1.3 Порівняння аналогів онлайн-сервісів для бронювання місць в ресторані

На сьогоднішній день вже існують сервіси, які реалізують дану проблему. Для прикладу розглянемо сервіс для бронювання столиків - «Максимум». Даний сервіс розрахований лише під одну мережу ресторанів. Додаток включає в себе дуже детальну інформацію про мережу ресторанів «Максимум»: перегляд фото, відгуки, меню, опис, місцезнаходження і також дає змогу забронювати столик. Дизай виконаний краще, ніж у прикладі, який був розглянутий вище, але немає можливості подитись модель залу. Очевидно, що в даному сервісу немає можливості для власника додати самому свій ресторан, тому що це серіс тільки для однієї мережів ресторанів.

Одне з вікон сервісу «Максимум» відображено на рисунку 1.2.

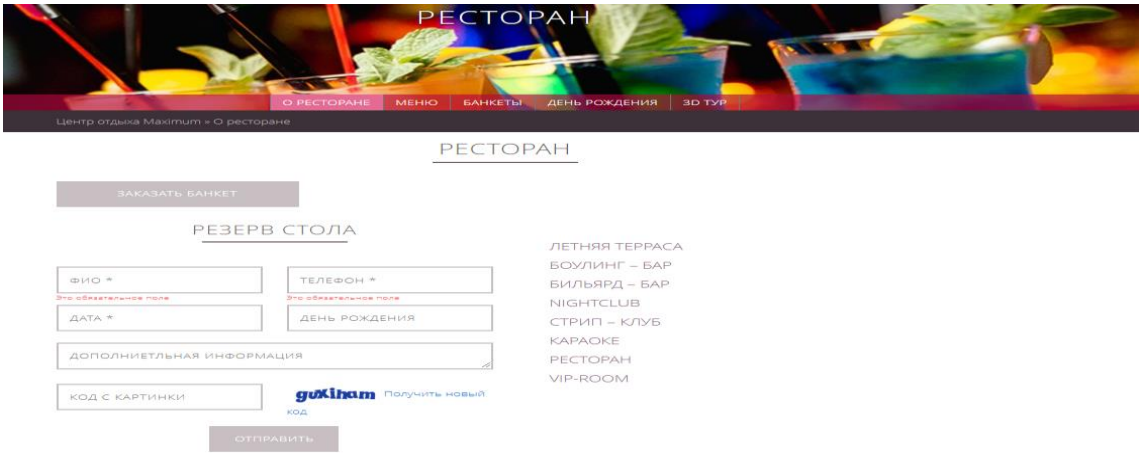

Рисунок 1.2. – Одне з вікон сервісу «Максимум»

Плюси:

– детальний опис;

– user friendly interface;

– облік часу;

– створення декількох проектів.

Мінуси:

– немає моделі залу;

– немає реалізованого функціоналу для показу рекомендацій користувачу;

– неможливо додати новий ресторан самотужки.

Ще одним аналогом для вирішення даної проблеми є сервіс «Letsbar».

«Letsbar» дає можливість клієнтам обрати місця, які они планують відвідати та забронювати місце в онлайн режимі. Даний сервіс працює лише зі сторони клієнта(користувача) та має не найкращий UI та UX. «Letsbar» дозволяє переглядати детальну інормацію про місця, ресторани, паби, також переглядати фото. Головною особливістю даного сервісу є можливість для користувачів бачити розташування столиків у залі.

Плюси:

– моделювання залу;

– можливість перегляду детальної інформації;

– пріоритетність завдань.

Мінуси:

– працює лише з клієнтами, тобто для власника ресторану немає можливості додати свій ресторан самотужки;

– no user friendly interface.

Приклад вікна системи " Letsbar " відображений на рисунку 1.2.

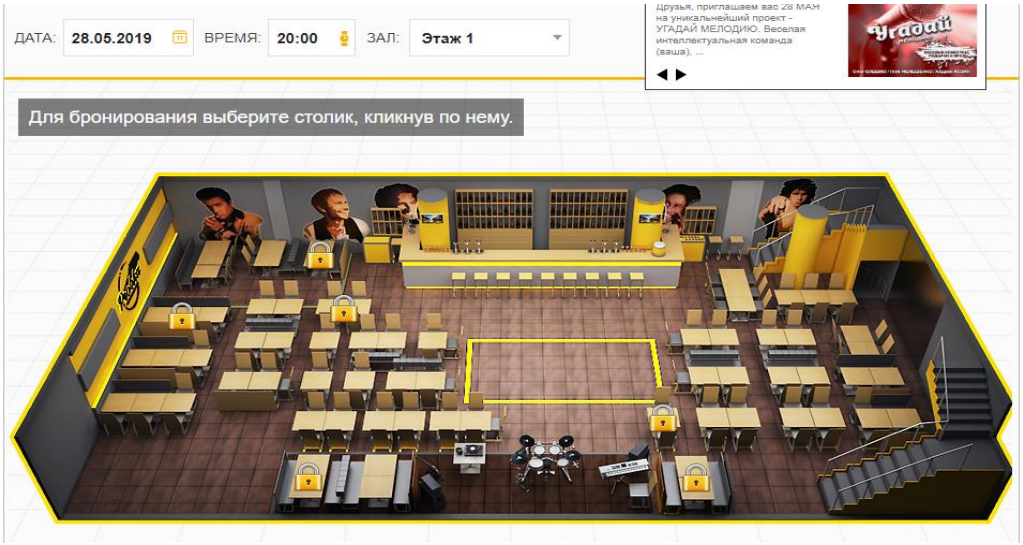

Рисунок 1.2 – Приклад вікна системи «Letsbar»

Таблиця 1.3 – Порівняльна характеристика аналогів

Назва	Сортування ресторанів за інтересами	Історія бронювань	Зручний інтерфейс	Можливість додати ресторан самотужки
Максимум	-	-	+	-
Letsbar	-	+	-	+
Our Service	+	+	+	+

1.4 Висновки

Отже, у данному розділі було проведено аналіз основної проблематики розробки клієнтської частини онлайн-сервісу. Показано основні шляхи для її реалізації та обрано найефективніший з них для подальшої розробки, а саме розробки онлайн-сервісу “з нуля” з урахуванням основних недоліків аналогів. Також було проаналізовано модель розробки клієнтської частини онлайн-сервісу та обрано її в подальшому використанні та встановлено такі необхідні компоненти для UI, та функціоналу онлайн-сервісу:

- Сортування ресторанів за критеріями;
- Історія бронювань;
- Зручний та доступний інтерфейс користувача;
- Можливість взаємодії, як з клієнтами так і з власниками ресторанів.

Додатково було проаналізовано основні переваги та недоліки існування та використовування онлайн-сервісів для бронювання місць в ресторані там доведено, що їх розробка має великі перспективи.

2 РОЗРОБКА АРХІТЕКТУРИ ТА ЕТАПІВ ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Етапи проектування клієнтської частини системи

Кліентська частина - це те, що бачить користувач на екрані браузера. Веб-проект будується з тексту, зображень, посилань, списків, форм введення даних, таблиць і тому подібне По суті, веброзробка інтернет-сайту полягає в розміщенні всіх його текстових і графічних елементів в потрібному місці і надання їм необхідних форм та властивостей [4].

Наприклад, простий текст можна оформити у вигляді списку або заголовка і опублікувати його по середині екрана. Зображення можна збільшити або зменшити, зробити його посиланням і помістити в потрібному місці. Таким чином, створення клієнтської частини веб - це редагування тексту / графіки веб-сторінки, тільки не традиційним способом (як це роблять мишею), а через написання спеціального коду на HTML, CSS і JavaScript. Процедура ділиться на наступні етапи:

- Проектування макета.
- Верстка зовнішнього вигляду веб-ресурсу по макету.
- Програмування інтерактивних можливостей.

1. Підготовка макета здійснюється в спеціальних графічних редакторах - Photoshop, Fireworks та інших. Він являє собою зображення майбутнього сайту в форматі jpg або psd.

2. Далі відбувається процедура верстки. Це, по факту, основа клієнтської розробки, і ділиться вона на кілька кроків:

- Нарізка макета дизайну на окремі шари і вибірка зображень.
- Створення всіх необхідних файлів для проекту в форматах * .html і * .css.
- Запис в них спеціальних конструкцій на мовах HTML / CSS.

3. Наступним (необов'язковим) етапом є програмування на мові JavaScript. Процес дозволяє додати в інтернет-сайти певну інтерактивну функціональність: списки, що випадають, анімацію, спрацьовування кнопок і інших функцій, що виконують певні дії на веб-сторінці (відправка пошти, калькулятори та ін.).

Заключним етапом, в принципі, вважається тестування всіх елементів веб-сайту і перевірка його на відповідність макету і технічним завданням [5].

2.2 Архітектура клієнтської частини онлайн-сервісу

Архітектура сайту - це спосіб організації сторінок, доступу до них і навігації. До неї відносяться: навігація і посилання; URL-адресу; хлібні крихти; сторінки категорій; файл Sitemap [6].

Правильна архітектура допомагає користувачам і пошуковим системам знаходити те, що вони шукають. Крім того, вона говорить системі про значимість і релевантності вашого контенту. Вона спрямовує користувача і пошукових роботів на найважливіші сторінки, розповідає, що собою являє ваш контент.

1. Золоте правило: задовольняти намір користувача

Перед вибором способу зв'язування і організації сторінок, дайте відповідь на три питання: 1) Що шукають люди? 2) Чому це важливо? 3) Як сторінки співвідносяться один з одним?

2. Згладжування архітектури

Гарна архітектура допомагає користувачам швидше і простіше знаходити інформацію, яку вони шукають.

Гладка архітектура - це та, де важливі сторінки можна побачити поруч із головною, а значить, людина до них добереться за менше число кліків.

Оптимізатори і дизайнери дотримуються правила "трьох кліків". Воно говорить, що будь-яка важлива сторінка сайту повинна бути на відстані не більше

3 кліків від домашньої сторінки сайту (або, можливо, іншої сторінки з високим авторитетом).

3. Сторінки-концентратори

Сторінка-концентратор - це оглядова сторінка, що відображає широку тему або категорію. Вона з'єднується з дочірніми категоріями, більш конкретними темами.

Цілі створення сторінок-концентраторів:

- дають короткий огляд теми або розділу;
- відповідають на популярні питання користувача;
- посилання на важливі підтеми і популярні продукти;
- приємніші для користувача, ніж загальні сторінки категорій;
- допомагають виділити предмет.

4. Створення тематичних “воронок”

Сторінки-концентратори добре справляються зі своїм завданням - агрегувати однотипну інформацію. А коли ви до того ж використовуєте структуру SILO, ви покращуєте архітектуру.

SILO - це ієрархічна організація контенту за темами. Концентратор - це агрегований контент за темами, Silo - його ієрархічне представлення.

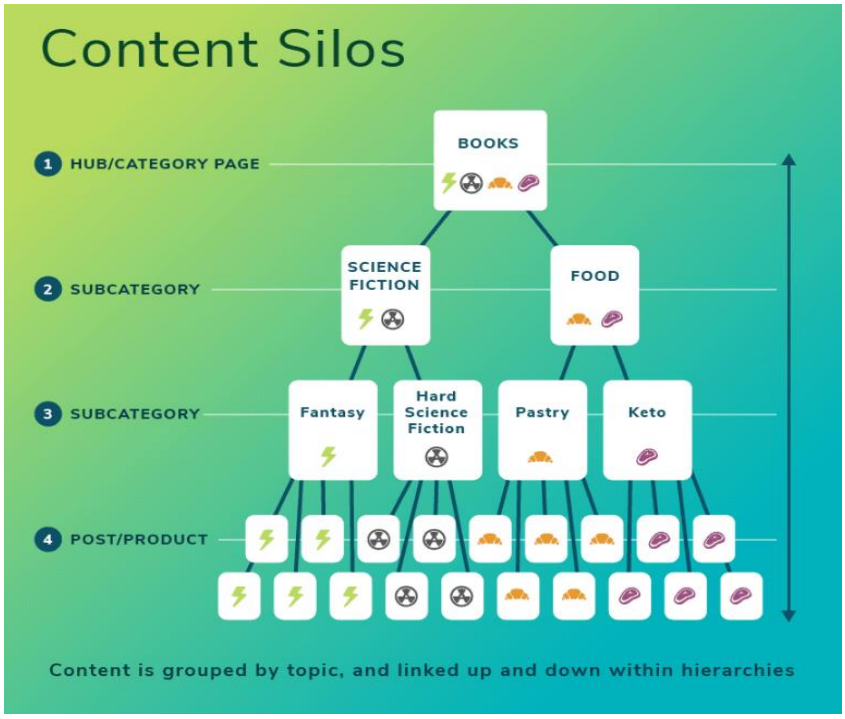

Рисунок. 2.1 –Content Silos

Кожен елемент в ієрархічній драбині пов'язаний з нижнім і верхнім об'єктом. Зв'язок допомагає користувачам орієнтуватися у вмісті сайту, а пошуковим системам - краще розуміти вміст.

Воронка зазвичай передбачає: навігацію, хлібні крихти; контекстні посилання; структуру URL. Ці елементи - основа угруповання для SILO.

5. Перехресні посилання і контекстно-залежні сторінки

Отже тепер у нас є архітектура, яка передбачає сторінки-концентратори і їх ієрархічну угруповання. Наступний крок - зробити їх потужнішими

Ми займаємося зв'язуванням релевантних між собою сторінок. Мова не тільки про перелінкування груп в ієрархії - вертикальному лінкбілдінгу, а й горизонтальному.

Зазвичай оптимізатори перелінковують тісно пов'язані сторінки, у яких в Silo один батьківський елемент. Це ефективно, коли товар, стаття або категорія тісно пов'язана з іншим об'єктом

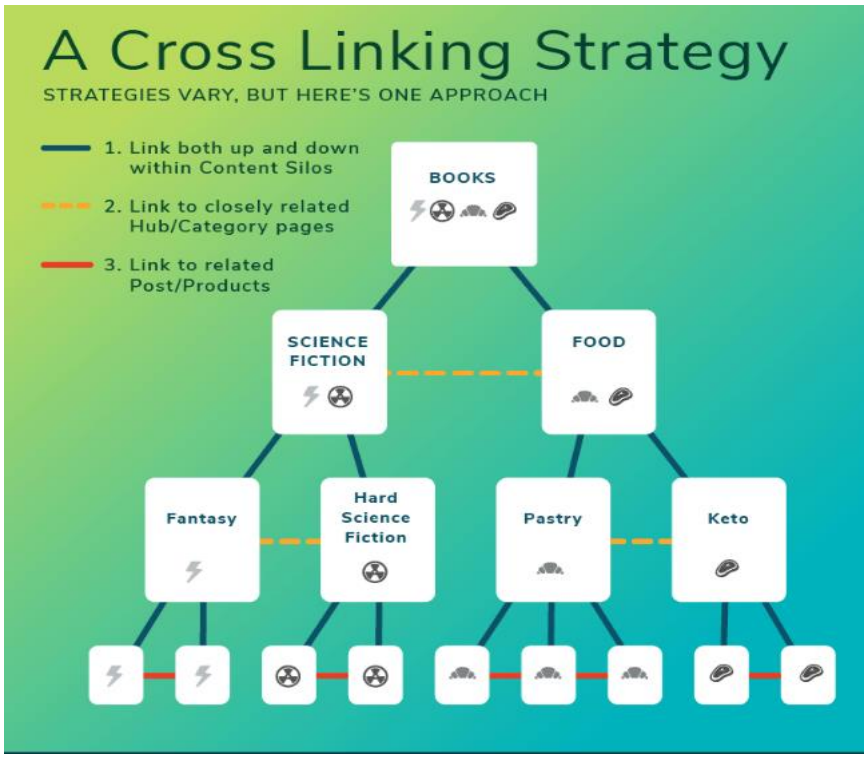

Рисунок 2.2 –“A Cross Linking Strategy”

6. Посилайтеся з авторитетних сторінок на менш авторитетні

Тепер у нас є розділи, у них є ієрархія, ми налаштували внутрішні посилання. Є ще одна область сайту, яку ми можемо використовувати і поліпшити, а саме силочний профіль - цільова сторінка.

Це сторінка для цільових відвідувачів, наприклад: сторінка продажу товарів, на яку користувач потрапляє з головної; сторінка з високою конверсією всередині silo-структури; сторінка конвертації з меншою важливістю, яка не дуже добре пов'язана.

Сторінка з високим авторитетом – це така сторінка, яка добре ранжируется в ПС і привертає великий трафік. Ідеєю покращення є - розставити посилання таким чином, щоб направити відвідувачів з таких сторінок на інші важливі сторінки. Причому, мова про посилання не тільки в панелі навігації.

Методи для того щоб визначити, які сторінки вимагають нових посилань:

Загрузити звіт про внутрішніх посиланнях в Google Search Console. Ви побачите інформацію для кращої тисячі URL-адрес. Якщо на вашому сайті понад 1000 урл-адрес, налаштуйте сегментацію по каталогам для детальної інформації.

Інструменти Moz, Ahrefs, SEMRush повідомляють інформацію про кожне посилання. Ви можете відфільтрувати звіт і побачити як трастові посилання, так і самі рідкі.

Google Analytics розповість, які сторінки отримують найбільший трафік, а які мають високий показник конверсії. Направте трафік на сторінки з високою конверсією.

7. Пагінація, View All і нескінченні екрани прокрутки

Для сторінок категорій, в яких 100, 1000 елементів, є три методи згладжування архітектури:

- пагінація;

- перегляд всіх елементів;

- нескінченний скролінг.

Найпростіше і поширене рішення - пагінація. Ви нумеруєте сторінки, тим самим розбиваючи великий список об'єктів. При правильній реалізації ви згладжуєте структуру і говорите пошуковій системі, що всі об'єкти належать одному розділу. Часто оптимізатори використовують просте розбиття на сторінки - це зручніше і для роботів, і для людей.

Другий метод - "показати все" - також згладжує архітектуру порталу. Деякі оптимізатори вважають, що так Google краще зчитує вміст сайту, адже всі об'єкти пов'язані з однією сторінкою. Метод добре працює, коли у вас багато продуктів або записів. Мінус в тому, що коли у вас 100 або 1000 записів / товарів на сторінці, вона довго завантажується. А як ми пам'ятаємо, швидкість завантаження сайту - наше все.

Третій спосіб поєднує в собі перші два. Результати завантажуються в міру скролінгу сайту вниз. Тим не менш, вони все одно розділені на сторінки, що допомагає пошуковим роботам швидше індексувати вміст.

8. Фасетна навігація

Фасетна навігація дозволяє користувачам сортувати об'єкти, налаштовувати фільтри і звужувати область пошуку по сайту. Цей метод, в основному, використовують інтернет-магазини.

Для користувачів це найзручніший спосіб знайти на сайті потрібний товар або запис. Але з точки зору пошукових систем при фасетной навігації у вас на сайті з'являються сотні однакових URL-адрес, які є повними, або частковими дублями. Найчастіше це обертається головним болем для веб-майстрів.

Рішення є: заохочувати пошукових роботів індексувати унікальні сторінки з високим трафіком, при цьому максимально обмежуючи обхід низькорівневих URL-адрес.

9. Помітні посилання на новий контент

Ви публікуєте нову статтю, виводите в магазин новий товар, запускаєте нову послугу, але це нічого не дає. Відсутність видимості в пошукових системах, відсутність рейтингу - відсутність трафіку.

Перша з проблем - у Google немає сигналів, щоб якось судити ваш контент, навіть якщо ви помістіть сторінку з ним в карту сайту. Рішення досить просте - дати посилання на матеріал на авторитетної сторінці. Ось як це пояснює Джон Мюллер (John Mueller) з Google:

"Як авторитетного джерела підійде головна сторінка. Так ви встановите тісний зв'язок нового контенту з авторитетним джерелом. Тому у багатьох сайтів є, наприклад, бічна панель, на яку виводяться нові матеріали, або продукти, які користуються великим попитом".

Новинні портали та інформаційні блоги - прекрасний приклад відповіді Джона. Щодо сайтів електронної комерції можна посперечатися. "Помітна

посилання" - не завжди означає додавання лінків на головну, т. К. Може відволікати відвідувача і погіршити призначений для користувача досвід.

10. “Хлібні крихти”

Це просто і важливо тому, що:

- визначає місце поточного URL в silo-структурі або ієрархії.
- пов'язує елементи в цій ієрархії;
- дає користувачам найпростіший спосіб навігації на сайті.

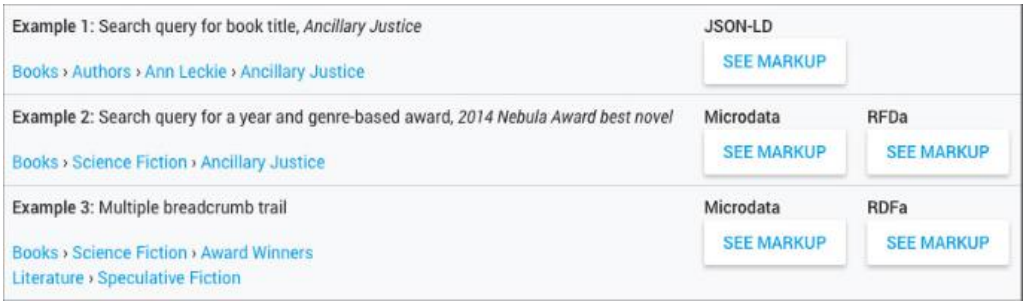

Рисунок 2.3 –“Хлібні крихти”

11. Ієрархічна структура URL

При організації контенту на сайті, використовуйте такі адреси, які відображають суть контенту на даній сторінці.

Наприклад, https://semantica.in/blog/5-istorij-uspekha-v-kontent-marketinge-uchimsya-i-vdokhnovlyaemsya-perevod.html - це адреса статті "5 історій успіху в контент-маркетингу: вчимося і надихаємось" . Користувач відразу бачить, що це запис в блозі студії і розуміє, що вона - містить кейс контент-маркетингу.

Переваги такої системи:

– користувач чітко розуміє, що він знайде за адресою;
– ключові слова в URL покращують ранжування, підвищують CTR;
– Google використовує показники рівня каталогу для оцінки важливості матеріалу та релевантності нових URL-адрес.

Деякі веб-майстри "підробляють" плоску структуру каталогів, обмежуючи папки або розміщуючи всі адреси в кореневому каталозі. Хоча цей метод може і спрацювати, для пошукових систем набагато важливіше, скільки кліків потрібно користувачеві для досягнення контенту, а не тільки кількість міститься слешів в URL [7].

2.3 Розробка архітектури системи

При розробці моделі роботи розроблюваної системи потрібно враховувати моменти, щоб розроблена модель дозволяла робити систему масштабованою та легко керованою. Також такий підхід дозволить під’єднати кілька клієнтів. Система складається з чотирьох компонент: 1. База даних (БД). Тут зберігаються дані про ресторани та вільні столики, його адреса, кухня та дизайн. 2. Сервер. Призначений для забезпечення взаємодії між базою даних та онлайн-сервісом. Містить спеціально розроблене раніше API, яке використовує JSON, як модель обміну даними. 3. Клієнтська частина – розроблюваний онлайн-сервіс для бронювання столиків в ресторані. 4. Кеш. В кеші зберігаються дані про користувача, які отримуються після авторизації. При цьому вони також відправляються на сервер і зберігаються в БД. Кеш використовується для того, щоб при відсутності з’єднання з інтернетом бачити на екрані якісь дані. Онлайн-сервіс повинен підлягати масштабуванню, внесенню змін без зміни основної частини коду тощо, необхідно, щоб його внутрішня архітектура відповідала цим вимогам. Розроблена архітектура системи наведена на рисунку 2.2.

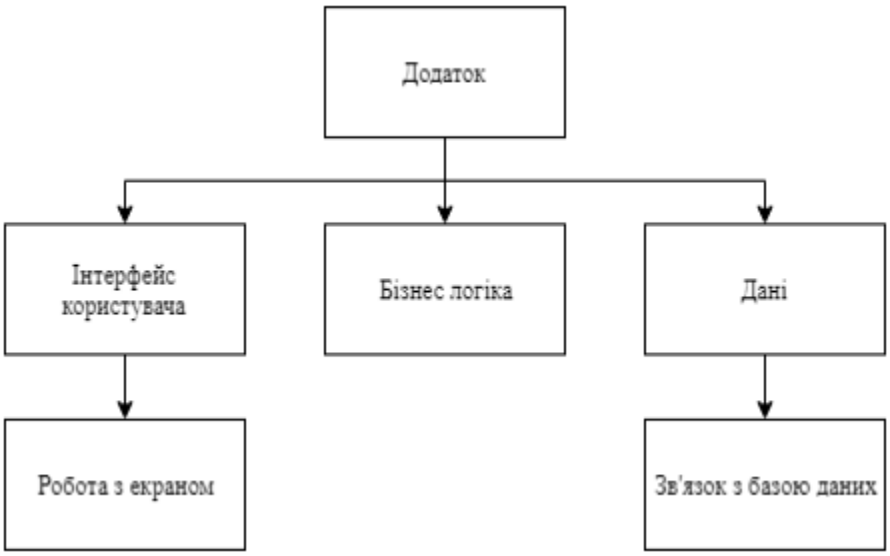

Рисунок 2.4 - Архітектура онлайн-сервісу

Архітектура додатку онлайн-сервісу є деревоподібною, адже це дозволяє легко контролювати процес розробки кожного модуля окремо, дозволяє логіку обробки даних без зміни інтерфейсу. Додаток складається з трьох частин: користувацького інтерфейсу, бізнес логіки та даних. Інтерфейс користувача містить класи для роботи підбором ресторану за критеріями та бронюванням столиків. Також містить спеціально створені хелпери – допоміжні класи, які мають часто використовувані функції. Модуль бізнес логіки призначений для того, щоб відділити інтерфейс від даних, адже дані можуть приходити з різних джерел, але на екрані відображатись однаковим чином. Тому модуль бізнес логіки слугує так званим посередником. Модуль для роботи з даними призначений для керування різними джерелами даних. В даному випадку це серверна частина (яка виконує API запити) та роботу з кешом. Також сюди можна додати локальну БД або інше джерело даних (наприклад Firebase SDK). Створена архітектура системи та клієнтської частини дозволяє створити систему, яка буде ефективно працювати та дозволить легко вносити зміни в систему та мобільний додаток окремо.

2.4 Розробка концепції PWA

На сьогоднішній день можна сказати, що наші дні – це «дні цифрових технологій». Практично всі люди не уявляють свого життя без цифрових гаджетів. Виходячи з цього, кількість різних сайтів і додатків, з кожним днем, збільшується у геометричній прогресії.

Задаючи собі питання: «А який з двох типів додатків буде більш зручніший для використання: веб-додаток чи мобільний додаток?» відповідь 100% буде дуже неоднозначною. Комусь більш до вподоби використовувати браузер на комп'ютері, комусь мобільні додатки. Також є можливість використовувати мобільний браузер і переглядати сайти з нього. Такий підхід доречний, але займає трохи більше часу, ніж просто натиснути на іконку додатку і відкрити його, тому що потрібно спочатку відкрити браузер, ввести в адресному рядку посилання на сайт і тоді вже користуватись.

Щоб вирішувати такі проблеми на допомогу приходять PWA.

PWA – це Progressive Web Application (укр. Прогресивний Веб-додаток). Саме слово «прогресивний» говорить, що воно якось модернізоване і є кращим за звичайні веб-додатки. PWA – це технологія в веб-розробці, яка візуально і функціонально трансформує веб-додаток в мобільний додаток, при цьому зберігаючи повний функціонал.

Всі знають, щоб встановити додаток на свій смартфон потрібно використати певний сервіс. Для iOS – це AppStore, а для Android – Play Market. У випадку з PWA можна про це забути, адже є можливість завантажити цей додаток прямо з сайту, без перелічених вище сервісів. Приклад завантаження додатку зображений на рисунку 2.5.

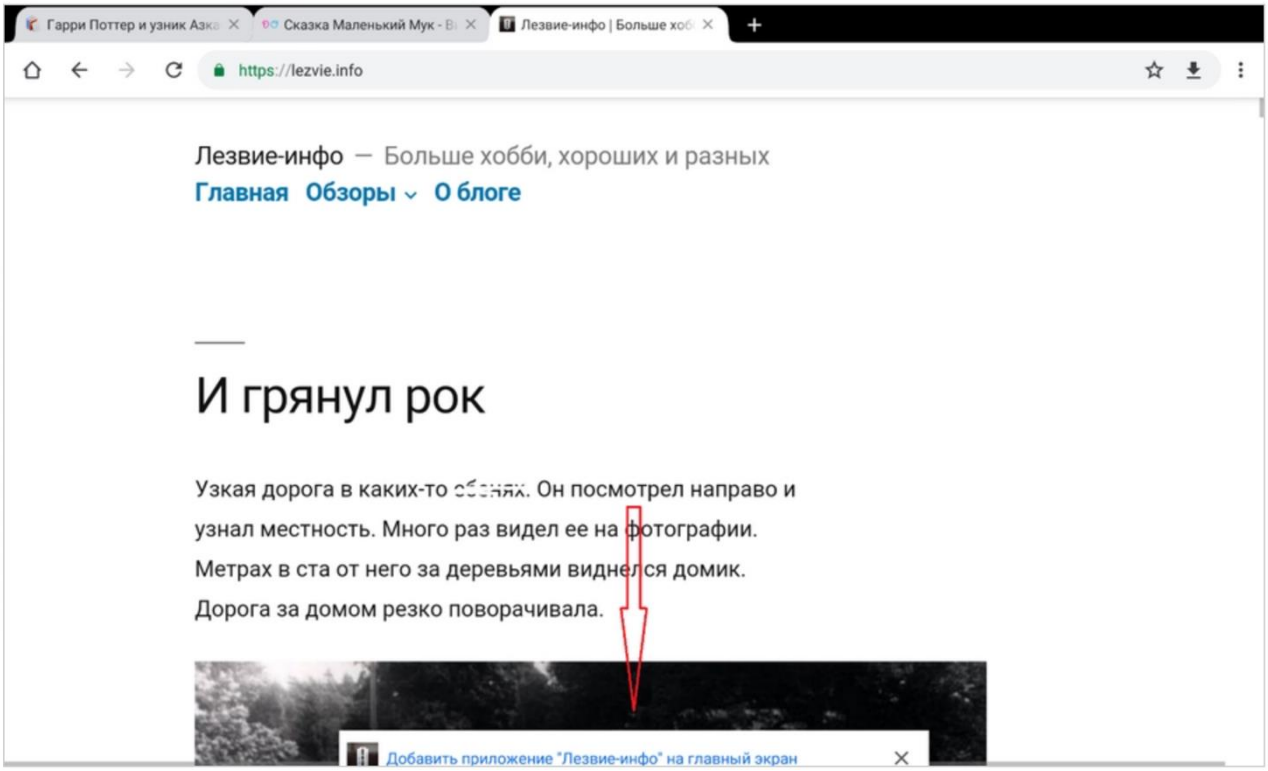

Рисунок 2.5 – Приклад завантаження PWA

Після завантаження додаток з’явиться на головному екрані смартфона.

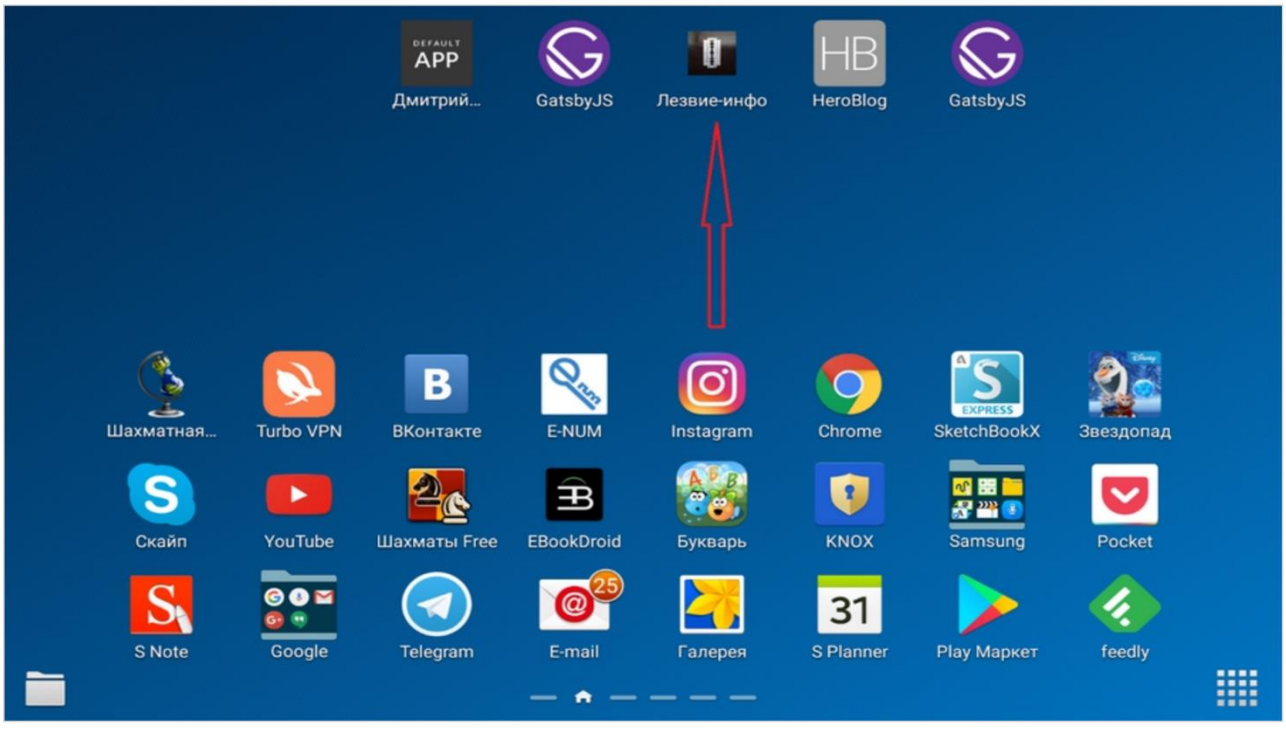

Рисунок 2.6 – Завантажений додаток PWA

Які ще переваги мають PWA?

Надійність – додаток завантажується і показується відразу ж, незалежно від статусу і якості мережевого з'єднання.

Швидкість – взаємообмін даними по мережі відбувається швидко, UI плавний і чуйний.

Привабливість – робить для користувача досвід роботи з додатком комфортним і приємним.

З точки зору Google, саме це відокремлює зараз за зовнішнім виглядом і відчуттями веб-сайти від нативних додатків.

2.5 Висновки

Отже, у другому розділі було проаналізовано основні етапи проектування онлайн-сервісу, серед них вибрано переваги технологій, які були використані при розробці системи. Також було проаналізовано архітектуру клієнтської частини онлайн-сервісу, як допомагає при розташуванні клієнтського інтерфейсу та побудовано архітектуру онлайн-сервісу при взаємодії клієнтської, серверної частини з користувачами. Проаналізовано концепцію PWA.

3 РОЗРОБКА АЛГОРИТМІВ ОСНОВНИХ МОДУЛІВ СИСТЕМИ

3.1 Розробка структури системи

Структура, що демонструє роботу онлайн-сервісу представлена на рисунку 3.1. В цій системі користувач через графічний інтерфейс підбирає ресторан за критеріями та бронює у ньому. При цьому модуль бронювання містить специфічні класи, що відповідають за перевірку наявності вільного столика та виведення результату бронювання.

Рисунок 3.1 – Модуль роботи онлайн-сервісу для бронювання столиків у ресторані

3.2 Розробка структуної схеми бронювання ресторану

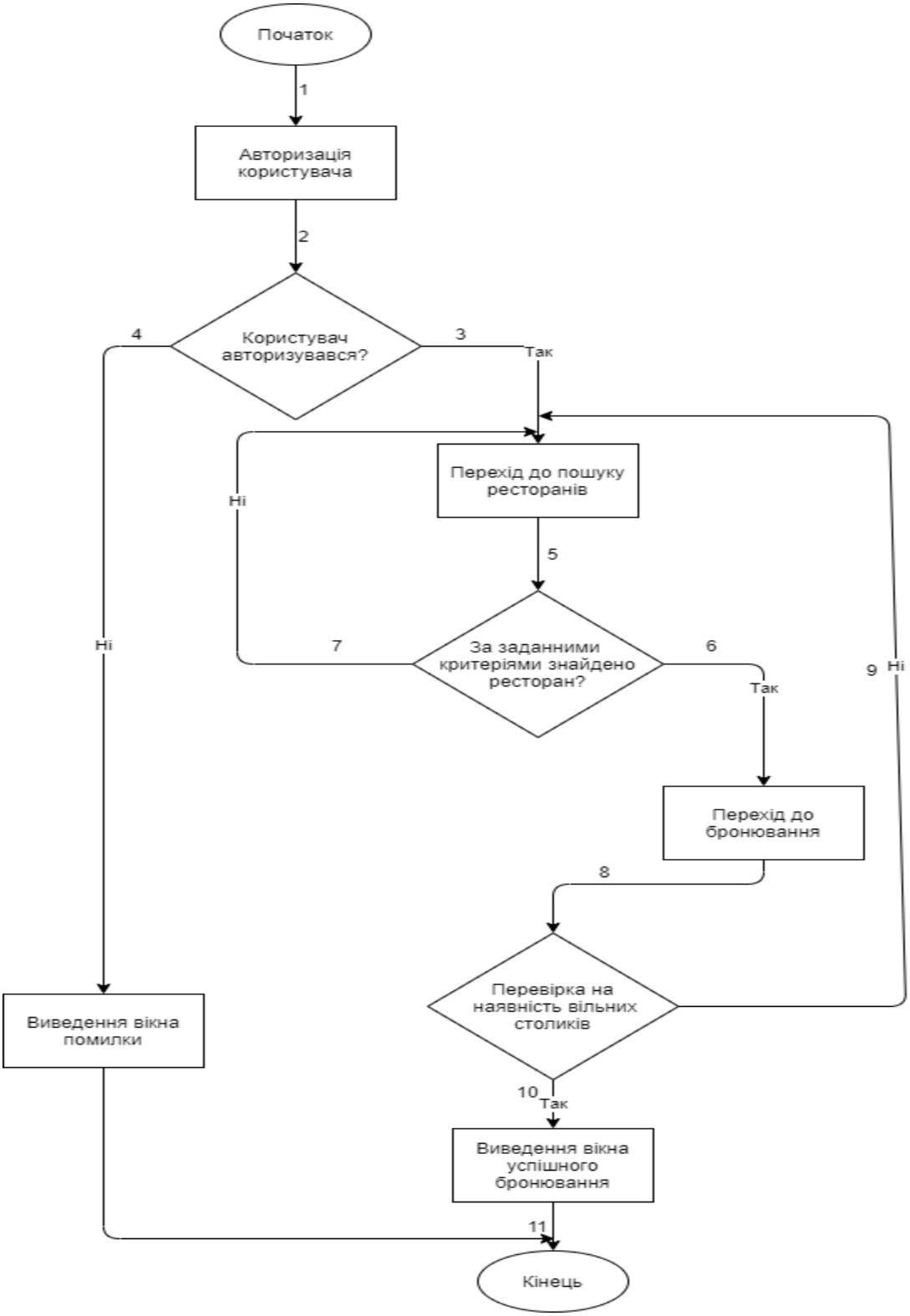

Рисунок 3.2 - Алгорим бронювання столика користувачем

Алгоритм, що демонструє етапи взаємодії користувача з онлайн-сервісом бронювання столиків у ресторані зображено на рисунку 3.2.

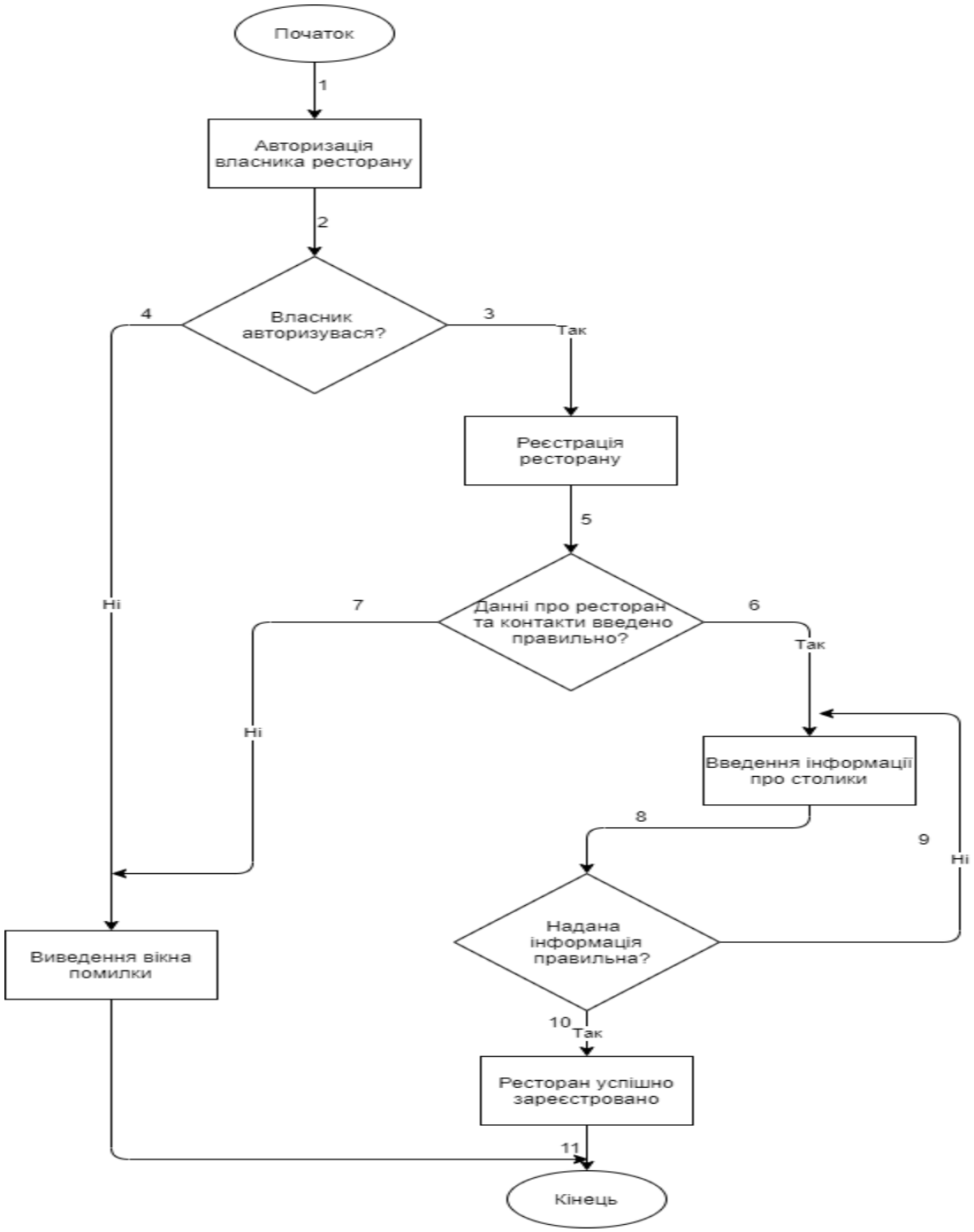

Рисунок 3.3 - Реєстрації ресторану

Алгоритм, що демонструє етапи взаємодії власника ресторану з онлайн-сервісом бронювання столиків у ресторані та реєстрацією його ресторану у систему зображено на рисунку 3.3.

3.3 Висновки

У третьому розділі було розроблено алгоритми основних модулів взаємодії користувачів та власників ресторану з онлайн-сервісом для бронювання столиків у ресторані. Показано структуру взаємодії користувача з графічним інтерфейсом та записом даних у БД. Основними етапами при бронюванні столика є реєстрація та авторизація користувача. Далі надається можливість вибору ресторанів серед списку методом сортування за заданими критеріями. Після цього виконується запит на наявність вільних столиків у ресторані і клієнт може забронювати їх при позитивному результатіт. Власнику ж ресторану потрібно пройти процес реєстрації ресторану до бази даних онлайн-сервісу. Даний процес вимагає підтвердження введених даних, тому подібні операції повинні проводитись в режимі онлайн з технічним персоналом онлайн-сервісу. В подальшому на онлайн-сервісі буде розміщена модель столиків ресторану та вказано усі критерії, які можуть зацікавити користувачів та наявні в списку критеріїв, та ресторану відповідно. Після даних дій, користувачі можуть бронювати столики у новому ресторані.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

4.1 Обгрунтування вибору мови програмування для реалізації онлайн-сервісу

Для проектування структури та реалізації клієнтської частини буде використано мову програмування JavaScript та таку бібліотеку до неї, React. Для маніпулювання даними на клієнтській частині буде використану бібліотеку Redux.

JavaScript — це обєктно–орієнтована мова скриптів, яка використовується для розробки вбудованих додатків, які виконуються як на стороні клієнта, так і на стороні сервера. Частіше за все вони виконуються в формі JavaScript в клієнтській частині і реалізуються в додатку у вигляді інтегруємого веб–браузером компонента, що дозволяє розробляти покращені інтерфейси і динамічні веб–сайти .

JavaScript використовує можливості середовища, в якому виконуються написані на ньому сценарії. JavaScript являється діалектом стандарта ECMAScript і охарактеризований як динамічна мова скриптів.

Мова JavaScript використовується для:

- написання сценаріїв веб-сторінок для надання їм інтерактивності;
- створення Single Page Application (односторінкові веб-додатки) (React, AngularJS, Vue.js);
- програмування на стороні сервера (Node.js);
- стаціонарних застосунків (Electron, NW.js);
- мобільних застосунків (React Native, Cordova);

сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter).

Таблиця 4.1 – Порівняння мов програмування

	JavaScript	C#
Швидкодія	+	-
Ручне управління пам’яттю	+	-
Перевантаження функцій	+	+
ООП	+	+
Шаблони	+	+
Значення параметрів за замовчуванням	+	+
Багатовимірні масиви	+	+
Динамічні масиви	+	+
Множинне наслідування	+	+

4.2 Обгрунтування вибору бібліотеки

Середовище веб-програмування на сьогодні розвивається дуже швидко. Існує багато різних бібліотек та фреймворків до даної мови. Було обрано бібліотеку React для полегшення розробки.

React – це бібліотека для мови JavaScript, яка дає змогу створювати SPA(Single Page Application) – односторінкові додатки. Переваги таких додатків над звичайними, це те, що браузер не буде перезавантажувати сторінку, коли користувач переходить з однієї сторінки на іншу. Такі додатки значно виграють у швидкодії, ніж звичайні веб-додатки.

Особливості React:

1. Односпрямована передача даних.

Властивості передаються від батьківських компонентів до дочірніх. Компоненти отримують властивості як безліч незмінних (англ. Immutable) значень, тому компонент не може безпосередньо змінювати властивості, але може викликати зміни через callback функції. Такий механізм називають «властивості вниз, події наверх».

2. Віртуальний DOM

React використовує віртуальний DOM (англ. Virtual DOM). React створює кеш структуру в пам'яті, що дозволяє обчислювати різницю між попереднім і поточним станами інтерфейсу для оптимального поновлення DOM браузера. Таким чином програміст може працювати зі сторінкою, вважаючи, що вона оновлюється вся, але бібліотека самостійно вирішує, які компоненти сторінки необхідно оновити.

3. JSX

JavaScript XML (JSX) - це розширення синтаксису JavaScript, яке дозволяє використовувати схожий на HTML синтаксис для опису структури інтерфейсу. Як правило, компоненти написані з використанням JSX, але також є можливість використання звичайного JavaScript. JSX нагадує іншу мову, створений в компанії Фейсбук для розширення PHP, XHP [3].

Для розробки дизайну буде використано HTML і CSS та бібліотеку Material UI, яка працює в парі з бібліотекою React та включає в собі вже готові компоненти, які можна використовувати.

Redux - це менеджер станів. Найчастіше його використовують з React, але його можливості не обмежуються однією цією бібліотекою. Хоча в React є власний метод управління станами він погано масштабується. Переміщення стану вгору по дереву працює для простих додатків, але в більш складних архітектур зміна стану проводиться через властивості (props). Ще краще робити це через зовнішнє глобальне сховище.

Redux - це спосіб управління станом додатку. Він заснований на декількох концепціях, вивчивши які, можна з легкістю вирішувати проблеми зі станом.

На рисунку 3.1 зображено логотип бібліотеки Redux

Рисунок 4.1 – Логотип бібліотеки Redux

Однією з характерних особливостей React, як було сказано вище є можливість використовувати JSX, мова програмування з близьким до HTML синтаксисом, який компілюється в JavaScript. Розробники можуть вимагати більшої продуктивності додатків за допомогою Virtual DOM. З React ви можете створювати ізоморфні додатки, які допоможуть вам позбавитися від неприємної ситуації, коли користувач з нетерпінням чекає, коли ж нарешті завершиться завантаження даних і на екрані його комп'ютера нарешті з'явиться щось крім анімації завантаження. Створені компоненти можуть бути з легкістю змінені і використані заново в нових проектах. Високий відсоток перевикористання коду підвищує покриваемості тестами, що, в свою чергу, призводить до більш високого рівня контролю якості. Використовуючи React Native мобільні додатки для Android і iOS, використовуючи досвід JavaScript і React розробки.

Ізоморфні додатки

Коли ми говоримо про ізоморфних додатках або про ізоморфних JavaScript, ми маємо на увазі, що ми можемо використовувати один і той же код як в серверній, так і в клієнтської частини програми. Коли користувач відкриває сайт в своєму браузері, вміст сторінки має бути завантажено з сервера. У випадку з

SPA-додатками (Single Page Application), це може зайняти деякий час. Під час завантаження користувачі бачать або порожню сторінку, або анімацію завантаження. З огляду на, що за сучасними стандартами очікування протягом більш ніж двох секунд може бути досить помітним незручністю для користувача, скорочення часу завантаження може виявитися вкрай важливим. А ось ще одна вагома проблема: пошукові машини не індексують такі сторінки так добре, як нам хотілося б. Виконання JavaScript коду на стороні сервера допомагає виправити подібні проблеми. Якщо ви створюєте ізоморфні додатки, ви можете отримати помітну вигоду, виробляючи рендеринг на стороні сервера. Після завантаження сторінки ви все ще можете продовжувати рендеринг компонентів. Така можливість рендеринга сторінок як на сервері, так і на клієнті приводить до помітних переваг, таким як можливість кращого індексування сторінок пошуковими машинами і поліпшення користувацького досвіду. Більш того, такий підхід дозволяє знизити час, що витрачається на розробку. При використанні деяких сучасних фреймворків, ви повинні створювати компоненти, які повинні рендери на стороні сервера, а також шаблони для клієнтської сторони додатки. React розробники можуть створювати компоненти, які працюють на обох сторонах.

Virtual DOM

Document Object Model, або DOM, - це спосіб представлення та взаємодії з об'єктами в HTML, XHTML і XML документах. Відповідно до цієї моделі, кожен такий документ являє собою ієрархічне дерево елементів, зване DOM-деревом. Використовуючи спеціальні методи, ми можемо отримає доступ до певних елементів нашого документа і змінювати їх так, як ми хочемо. Коли ми створюємо динамічну інтерактивну веб-сторінку, ми хочемо, щоб DOM оновлювався так швидко, як це можливо після зміни стану певного елемента. Для даної задачі деякі фреймворки використовують прийом, який називається «dirty checking» і полягає в регулярному опитуванні стану документа і перевірці змін в

структурі даних. Як ви можете здогадатися, подібне завдання може стати справжнім головним болем в разі високонавантажених додатків. Virtual DOM, в свою чергу, зберігається в пам'яті. Саме тому в момент, коли «справжній» DOM змінюється, React може змінювати Virtual DOM в одну мить. React «збирає» такі зміни порівнює їх зі станом DOM, а потім перемальовує змінилися компоненти.

При цьому підході ви не робите регулярне оновлення DOM. Саме тому може бути досягнута більш висока продуктивність React додатків. Другий наслідок випливає з изоморфной природи React: ви можете виробляти рендеринг на стороні сервера зовсім як на стороні клієнта.

Повторне використання коду

Мобільні додатки мають деякі переваги в порівнянні з сайтами. Їх можна використовувати без підключення до Інтернету. Вони мають доступ до таких можливостей пристрою, як спливаючі повідомлення. Також вони дозволяють бути в контакті з вашими користувачами в режимі 24/7. React Native - це фреймворк, який дозволяє вам створювати мобільні додатки, використовуючи React. Логіка додатка пишеться на JavaScript, таким чином, програмісту не потрібно відмовлятися від звичних прийомів веб-розробника. Все що потрібно - навчитися писати специфічний для пристрою код, який адаптує компоненти, раніше створені для веб-сайту до нового середовища проживання.

Якщо ми порівняємо витрати на розробку різних видів мобільних додатків, ми отримаємо приблизно такі результати:

У випадку з нативними додатками ви можете сподіватися на досить високу продуктивність, але вартість розробки буде досить високою;

Якщо ви віддасте перевагу фреймворкам, які дозволяють використовувати HTML5, CSS3 і JavaScript, наприклад PhoneGap, ви можете знизити вартість. Але в цьому випадку рівень продуктивності буде набагато нижче;

У разі React ви можете досягти рівня продуктивності, який можна порівняти з нативними додатками. При цьому вартість розробки порівнянна з

попереднім прикладом. Якщо ви плануєте створити корпоративний веб-додаток і не цілком впевнені, чи буде розробка мобільної версії цього ж додатка гарною ідеєю, ось що ви повинні пам'ятати. React Native дозволяє використовувати вже наявну логіку веб-додатку при створенні мобільного додатка. Це означає, що команда розробників може використовувати той же код, який був використаний в процесі створення сайту замість того, щоб починати з чистого аркуша.

Крім швидшої розробки, перевикористання коду дозволяє уникнути великої кількості помилок. Якщо ви створюєте добре спроектовані компоненти, які потім використовуєте знову, вам потрібно буде писати менше коду, коли ви вирішите створити з їхньою допомогою новий призначений для користувача інтерфейс. Чим менше нового коду вам потрібно, тим менше ймовірність виникнення нових помилок. До того ж, ви знаєте ваші компоненти. Ви вже використовували і тестували їх при роботі над реальним проектом, а значить при виникненні помилок зможете передбачити причину їх появи.

Заключення

Компонентно-орієнтований підхід, можливість з легкістю змінювати наявні компоненти і перевикористати код перетворюють React розробку в безперервний процес поліпшення. Компоненти, які були створені під час роботи над тим чи іншим проектом, не мають додаткових залежностей. Таким чином, ніщо не заважає використовувати їх знову і знову в проектах різного типу. Весь попередній досвід може бути з легкістю застосований при роботі над новим сайтом або навіть при створенні мобільного додатка. Використовуючи передові можливості, такі як Virtual DOM або ізоморфний JavaScript, React розробники можуть з високою швидкістю створювати високопродуктивні додатки, незважаючи на рівень їх складності. Можливість з легкістю заново використовувати вже наявний код підвищує швидкість розробки, спрощує процес тестування, і, як результат, знижує витрати. Той факт, що ця бібліотека розробляється і підтримується висококваліфікованими розробниками і набирає

все більшої популярності з кожним роком, дає підстави сподіватися, що тенденція до подальших поліпшень продовжиться.

4.3 Особливості середовища розробки та обґрунтування вибору

Visual Studio Code - редактор вихідного коду, розроблений Microsoft для Windows, Linux і macOS. Позиціонується як «легкий» редактор коду для кроссплатформенной розробки веб-і хмарних додатків. Включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense і засоби для рефакторинга. Має широкі можливості для кастомізації: призначені для користувача теми, поєднання клавіш і файли конфігурації. Розповсюджується безкоштовно, розробляється як програмне забезпечення з відкритим вихідним кодом, але готові збірки розповсюджуються під проприетарной ліцензією [10].

Visual Studio Code заснований на Electron - фреймворк, що дозволяє з використанням Node.js розробляти настільні додатки, які працюють на движку Blink. Незважаючи на те, що редактор заснований на Electron, він не використовує редактор Atom. Замість нього реалізується веб-редактор Monaco, розроблений для Visual Studio Online.

Visual Studio Code - це редактор вихідного коду. Він підтримує ряд мов програмування, підсвічування синтаксису, IntelliSense, рефакторинг, налагодження, навігацію по коду, підтримку Git та інші можливості. Багато можливості Visual Studio Code не доступні через графічний інтерфейс, найчастіше вони використовуються через палітру команд або JSON файли (наприклад, призначені для користувача настройки). Палітра команд представляє собою подобу командного рядка, яка викликається поєднанням клавіш.

Visual Studio також дозволяє замінювати кодову сторінку при збереженні документа, символи перекладу рядка і мову програмування поточного документа.

З 2018 року з'явилися розширення Python для Visual Studio Code з відкритим вихідним кодом. Воно надає розробникам широкі можливості для редагування, налагодження і тестування коду.

На березень 2019 року за допомогою вбудованого в продукт призначеного для користувача інтерфейсу можна завантажити і встановити кілька тисяч розширень тільки в категорії «programming languages» (мови програмування).

Приклад роботи програми Microsoft Visual Studio Code зображено на рисунку

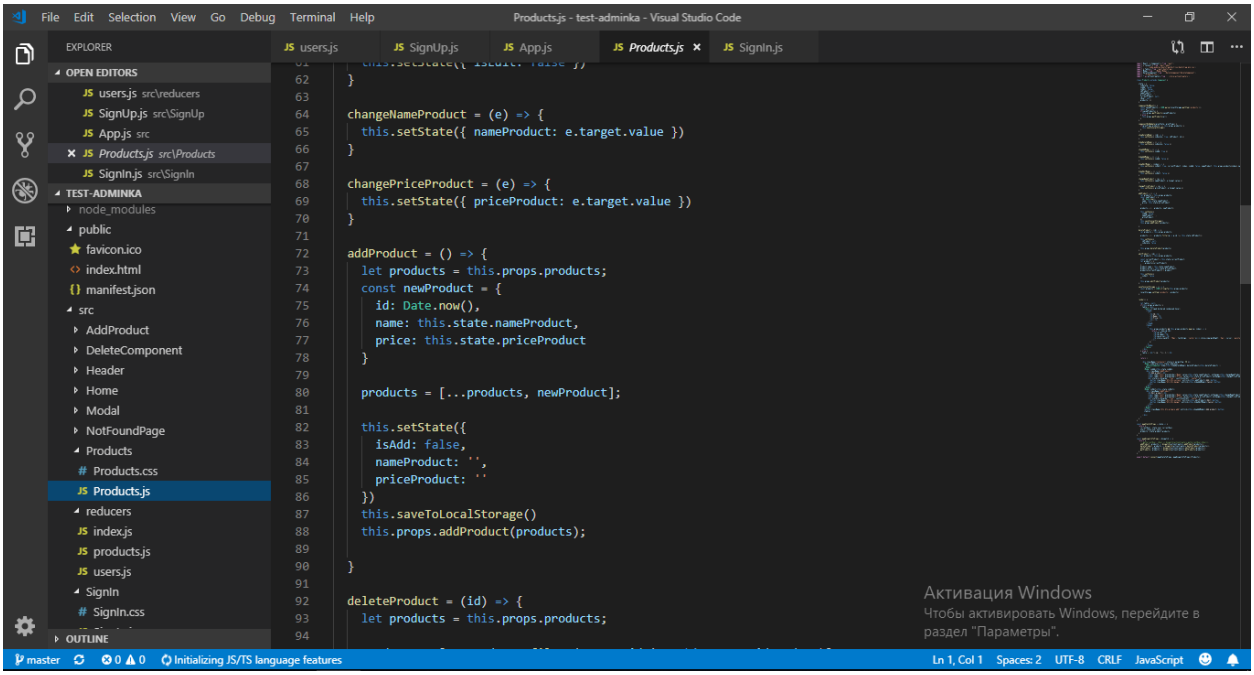

Рисунок 3.2 – Приклад роботи програми Microsoft Visual Studio Code

Brackets — текстовий редактор від компанії Adobe, призначений для редагування JavaScript, HTML і CSS. Сирцевий код Brackets написаний з використанням веб-технологій (JavaScript, HTML і CSS) і поширюється під ліцензією MIT. Редактор оформлений у вигляді відокремленого

настільного застосунка, для установки якого підготовлені deb-, dmg- і msi- пакети для Linux, OS X і Windows.

Brackets підтримує режим Live-розробки, при якому редагований контент (JavaScript, HTML і CSS) у міру зміни відразу відображається в синхронізованому з редактором вікні браузера — розробник може змінювати вміст і відразу спостерігати до яких наслідків приводять дані зміни. Налагодження також може виконуватися синхронно із браузером, розробник може встановити точку зупину або відкотитися на крок назад при перегляді результатів. Є вбудована підтримка препроцесорів LESS і SCSS. В інтерфейсі застосовується система контекстно-залежних інструментів, що з'являються в міру необхідності в основному вікні розробки. Для розширення можливостей редактора розвивається система доповнень. Серед доступних плагінів рекомендуємо наступні:

– Emmet - розширення для швидкого введення шаблонних команд;

– Beautifer - додаток для оформлення коду;

– FTP-Sync - плагін для синхронізації з віддаленим сервером;

– ColorHints - надбудова для зручного підбору кольорів.

– Брекет підійде для комп'ютерів на базі різних версій операційної системи - Віндовс 7-10, Vista.

Приклад роботи програми Brackets зображено на рисунку 3.3.

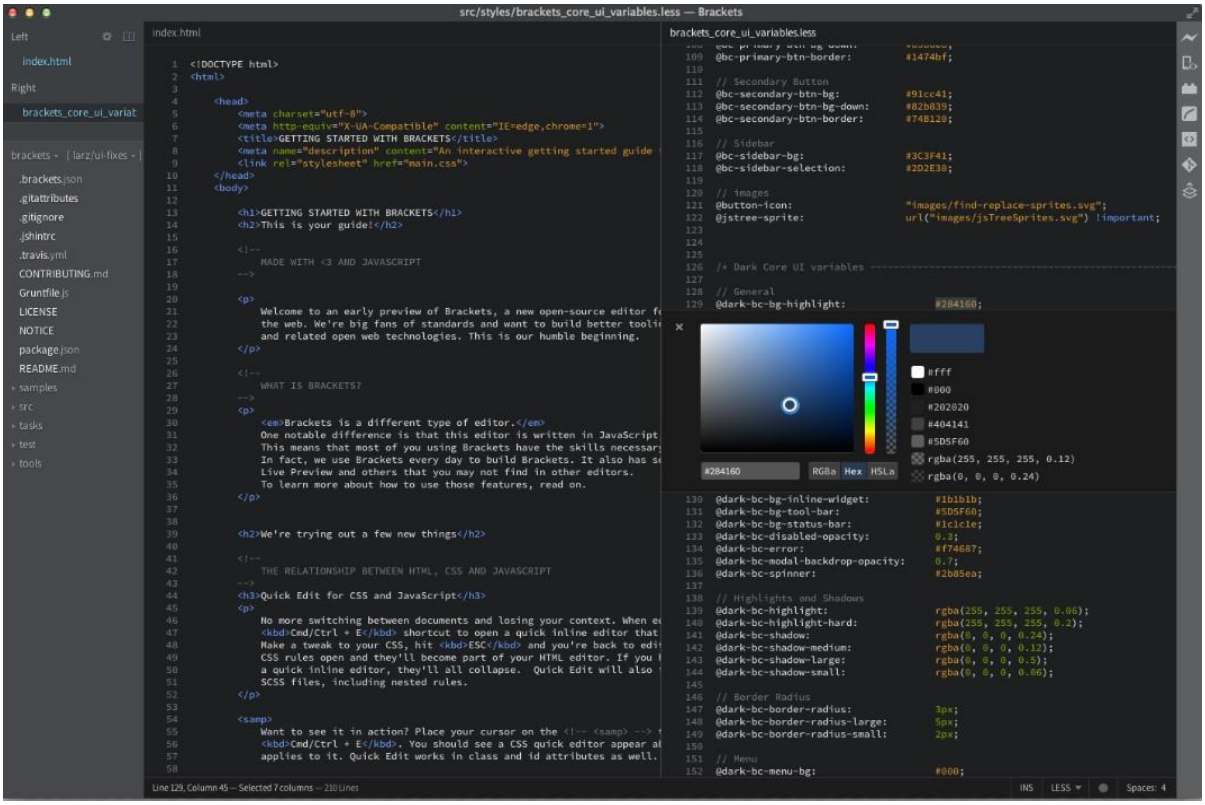

Рисунок 3.3 – Приклад роботи програми Brackets

Sublime Text — швидкий кросплатформенний текстовий редактор. Підтримує плагіни, розроблені за допомогою мови програмування Python.

Sublime Text не є вільним чи відкритим програмним забезпеченням, але деякі його плагіни розповсюджуються з вільною ліцензією, розробляються і підтримуються спільнотою розробників.

Редактор містить різні візуальні теми, з можливістю завантаження додаткових.

Користувачі бачать весь свій код в правій частині екрану у вигляді міні-карти, при кліці на яку можна здійснювати навігацію.

Є кілька режимів екрану. Один з них включає від 1 до 4 панелей, за допомогою яких можна показувати до чотирьох файлів одночасно. Повноцінний

(free modes) режим показує тільки один файл без будь-яких додаткових навколо нього меню.

Виділення стовпців цілком або розстановка кілька покажчиків по тексту, що робить можливим миттєву правку. Покажчики поводяться, ніби кожен з них - єдина в тексті. Команди типу: переміщення на знак, переміщення на рядок, вибірка тексту, переміщення на слово або його частини (CamelCase, розділений дефісом або підкресленням), перехід на початок або кінець рядка тощо, Впливає на всі покажчики незалежно і відразу, дозволяючи правити складноструктурований текст швидко, без використання макрокоманд або регулярних виразів.

Коли користувач набирає код, Sublime Text, в залежності від використовуваної мови, буде пропонувати різні варіанти для завершення запису. Редактор також автоматично завершує створені користувачем змінні.

Темний фон Sublime Text призначений для збільшення контрастності тексту. Основні елементи синтаксису виділені різними кольорами, які краще поєднуються з темним тлом, ніж зі світлим.

Sublime Text дозволяє користувачеві збирати програми і запускати їх без необхідності перемикатися на командний рядок. Користувач також може налаштувати свою систему збирання та включити автоматичну збірку програми кожного разу при збереженні коду.

Приклад роботи програми Sublime Text 3 зображено на рисунку 3.4.

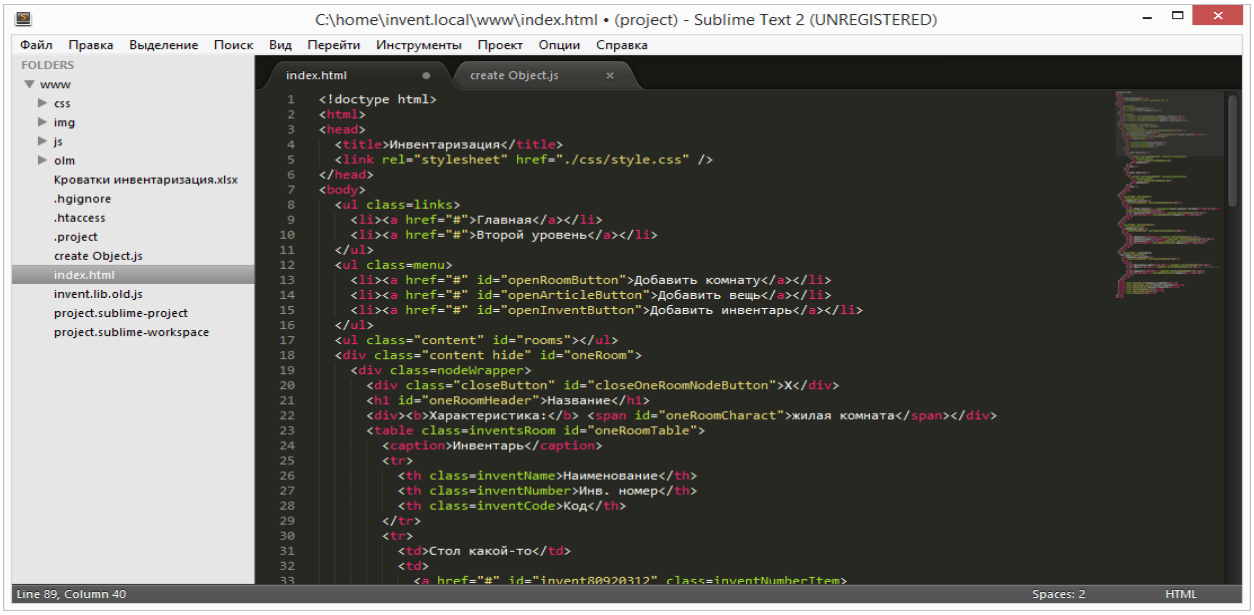

Рисунок 3.4 – Приклад роботи програми Sublime Text 3

Порівняння даних середовищ програмування наведено в Таблиці 3.2.

Таблиця 3.2 – Порівняння даних середовищ програмування

Функції	Середовище програмування		
	Brackets	Sublime Text 3	Microsoft Visual Studio
Режим відлагодження (0,9)	0.5	0.5	1
Кросплатформеність (0,6)	0	0	1
Функція авто заповнення (0,4)	0.5	0.5	1
Загальний коефіцієнт	0,65	0,5	1.9

4.4 Аналіз методів і засобів тестування

Тестування – це процес експериментування з продуктом за допомогою тестів з метою виявлення в ньому помилок (неточностей, допущених розробниками ПЗ). Основна ідея тестування – запустити ПЗ і спостерігати за його роботою та її наслідками. Якщо збій в роботі програмного забезпечення відбувся – аналізується звіт з метою виявлення місцезнаходження помилки, яка його викликала.

Тестування може відбуватися різними способами, однак не варто забувати про сам процес і стратегії тестування. Від нього залежить послідовність ваших дій. На сьогоднішній день, фахівці з тестування веб-сайтів застосовують такі види як:

1. Функціональне тестування. Один з важливих і незамінних видів тестування. Найголовніше правило функціонального тестування є правильні розрахунки функцій. Наприклад, візьмемо інтернет-магазин, у якого є не тільки знижки на товар, але і безліч статусів при покупці, n кількість товарів. Всі ці варіанти слід враховувати. Адже якщо функціонал проекту не працює в певному браузері, то він не буде працювати ніде.

2. Тестування зручності користування (юзабіліті). Тестування зручності користування (юзабіліті) – це вид тестування, який надає сайту зручність і практичність у використанні.

3. Тестування продуктивності. Тестування продуктивності – в основному це тестування навантаження. Тестування навантаження сайту перевіряється в більшості випадків автоматом, тобто спеціальними програмами. Це дає шанс перевірити, наскільки він буде працювати під певним навантаженням. Мета цього тестування, полягає в кількості віртуальних користувачів, які задають n кількість запитів, в один час (будь це секунди навіть). Тим самим результат дає зрозуміти, чи зміг наш проект витримати, наприклад,

100 користувачів, які одночасно купували товар або авторизувалися на сайті, відповідь показує, чи реально витримати сайту таке навантаження.

4. Тестування інтерфейсу користувача. UI testing – це тестування графічного інтерфейсу користувача, яке передбачає перевірку сайту на відповідність вимогам до графічного інтерфейсу.

5. Тестування безпеки. Це ключ до надійності веб-сайтів. Основні правила цього тестування – це перевірка на уразливість різних видів атак. Якщо це інтернет-магазин, то, швидше за все, слід перевіряти запити на Sql ін'єкцію (запити до бази даних). SQL-ін'єкції – це шкідливий код в запитах бази даних - найбільш небезпечний вид атак. Вона дає можливість впровадити довільний код, і атакувати комп'ютери користувачів, які переглядають заражені сторінки.

Важливими вимогами до сучасних сайтів є їх кросбраузерність та адаптивність. Під кросбраузерністю сайту розуміють коректне відображення елементів сторінки та їх робота в усіх браузерах.

Кросбраузерність розробленого сайту перевірялася в таких сучасних браузерах, як: Google Chrome, Firefox та Opera. При тестуванні не було виявлено жодних помилок, усі елементи сайту однаково відображаються в усіх цих браузерах.

4.5 Тестування основних модулів системи

В результаті розробки було створено онлайн-сервіс для бронювання місць в ресторані. Даний сервіс складається з таких модулів як:

- Модуль авторизації
- Модуль пошуку ресторана
- Модуль бронювання ресторану

Модуль авторизації. Для того щоб користувач міг користуватись сервісом повноцінно, спочатку йому потрібно авторизуватись. Перш за все потрібно

зареєструватись, ввівши наступні данні: ім’я, емейл, телефон, інтереси та пароль.

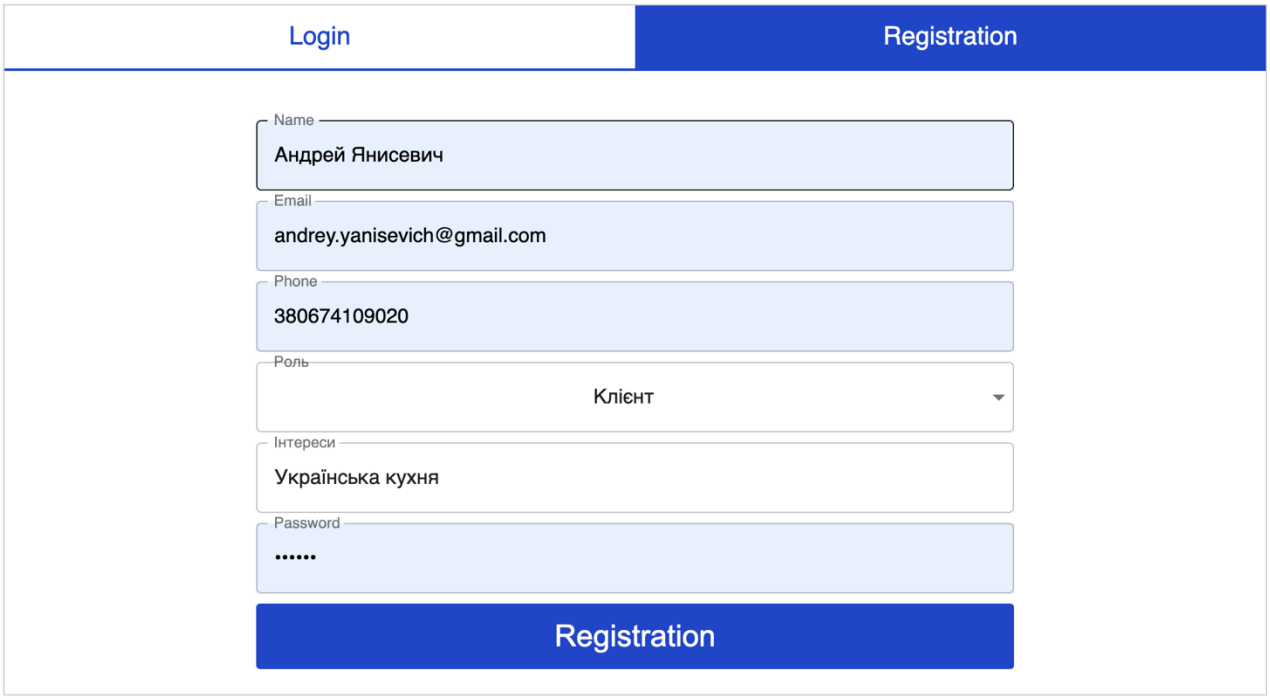

Рисунок 3.5 – Форма реєстрації

Після успішної реєстрації потрібно перейти на форму логіну і авторизуватись.

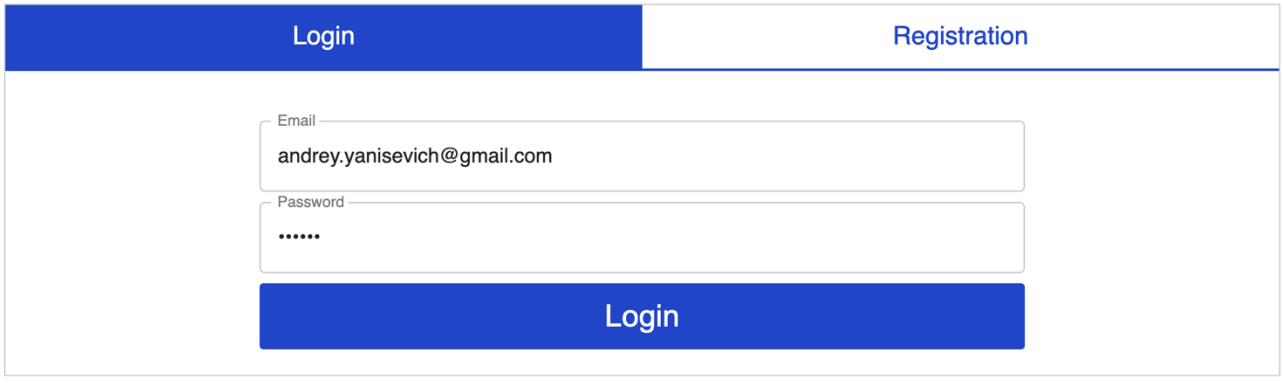

Рисунок 3.6 – Форма авторизації

Головний екран сервісу виглядає наступним чином (рис. 3.7) та включає в собі наступні компоненти: шапка сайту, форма пошуку ресторанів, список

ресторанів.

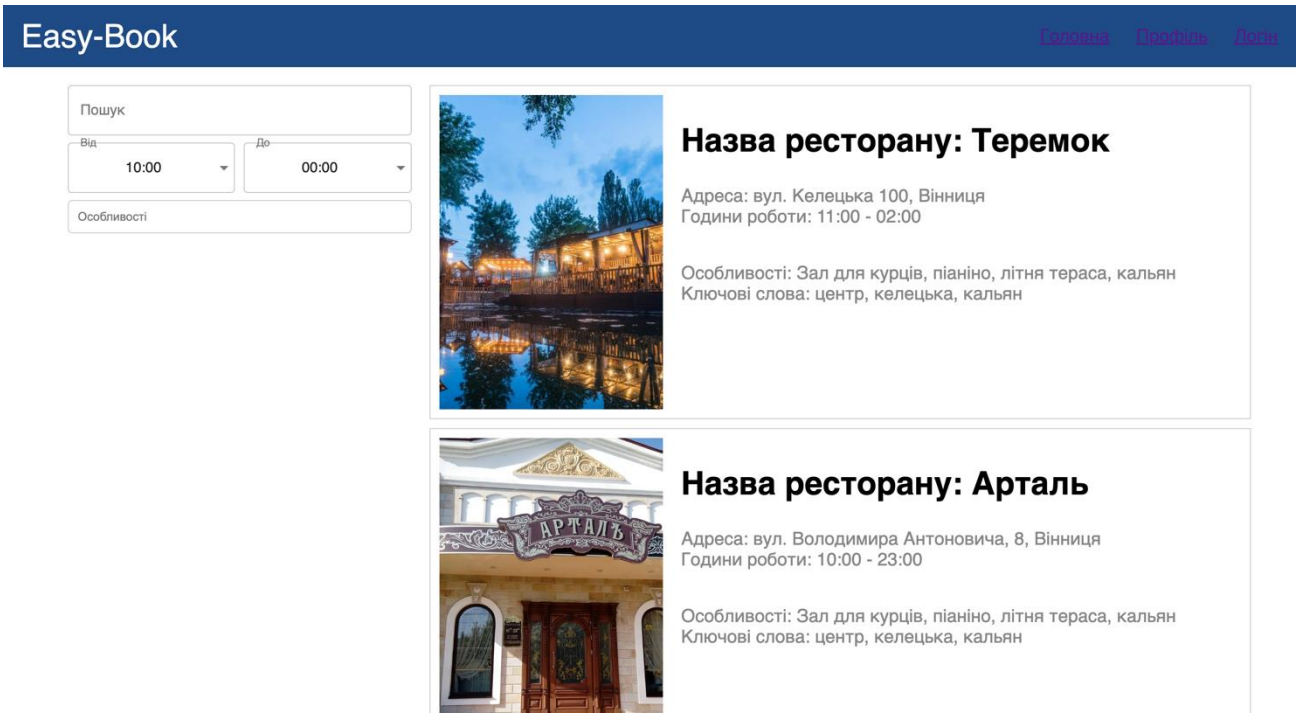

Рисунок 3.7 – Головний екран онлайн-сервісу

Модуль бронювання ресторану. Для того щоб забронювати ресторан спочатку необхідно визначитись з рестораном. Припустимо, що клієнт обрав ресторан **Aura**. Перейдемо на сторінку цього ресторану, просто клікнувши на блок з рестораном. Відкривається сторінка з детальною інформацією про цей ресторан (рис. 3.8)

Рисунок 3.8 – Сторінка з детальною інформацією про ресторан

На цій же сторінці присутня кнопка «Забронювати». Після натискання на неї з’являється модальне вікно, в якому користувач може ввести необхідні йому критерії для столика такі як: дата та час та необхідна кількість місць (рис. 3.9).

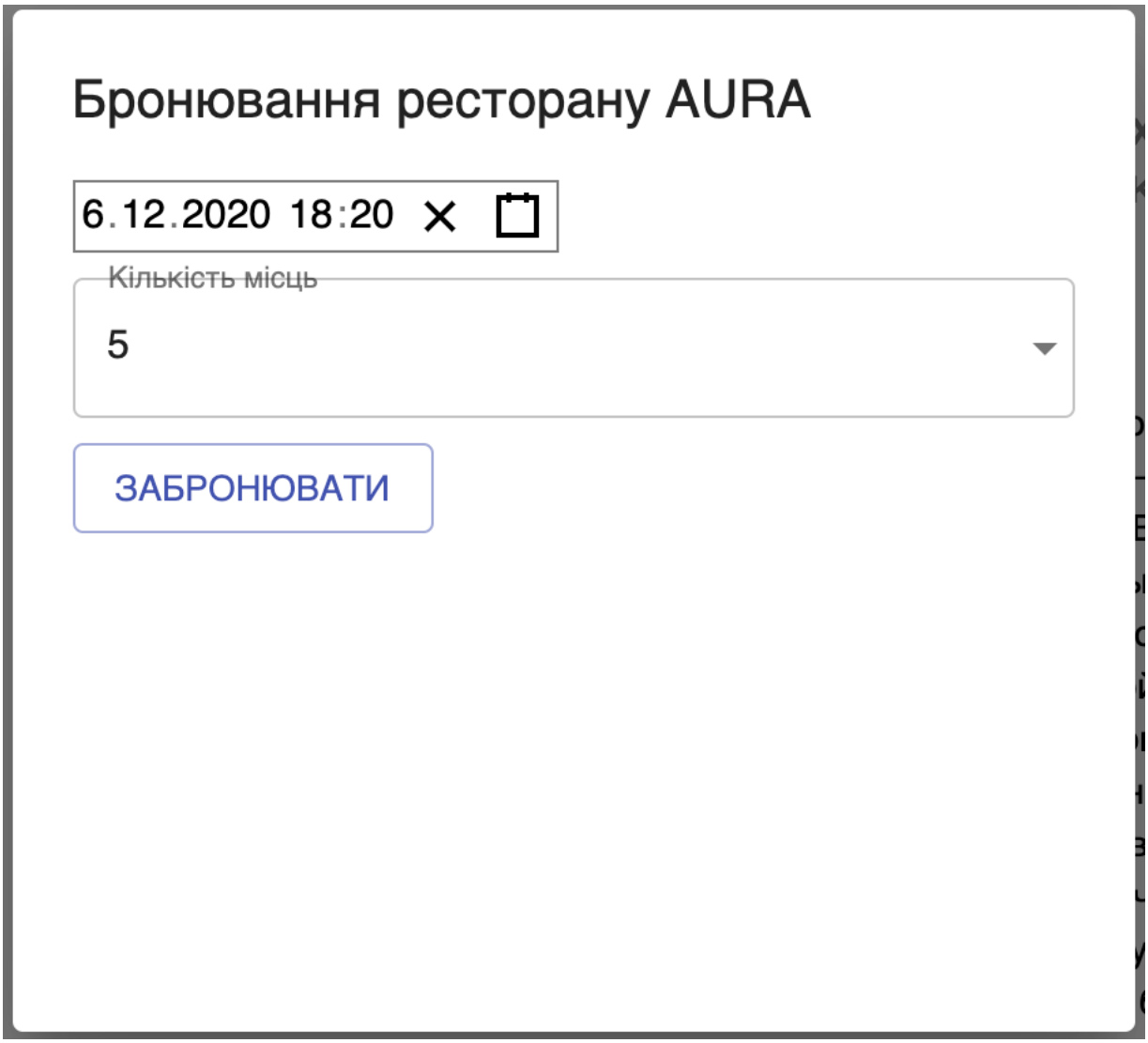

Рисунок 3.9 – Модальне вікно бронювання ресторану

Після введення користувачем цих даних залишається тільки натиснути кнопку «Забронювати».

Вся інформація про зроблені бронювання знаходиться в профілі користувача. Там користувач може побачити всі ресторани які він бронював (рис. 3.10).

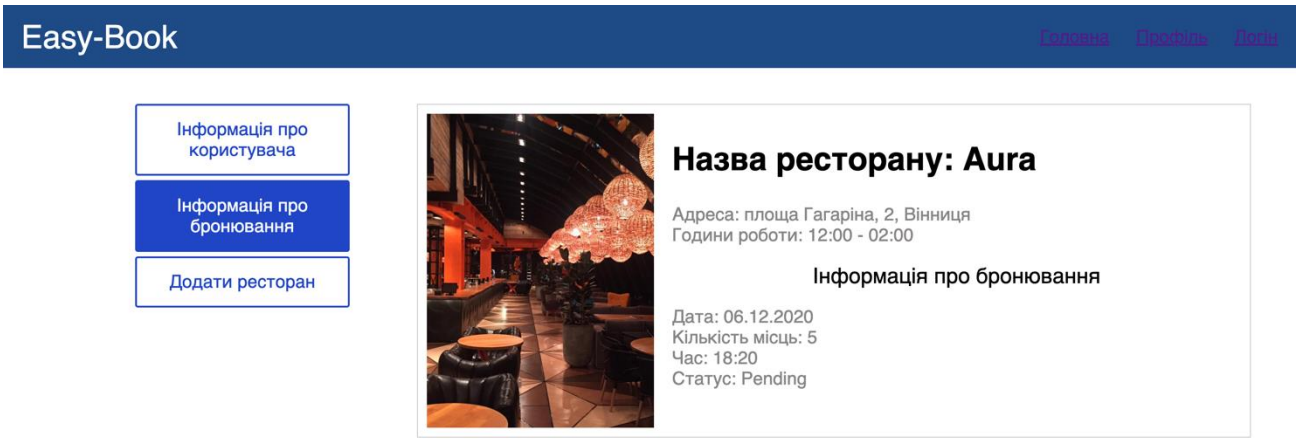

Рисунок 3.10 – Екран інформації про бронювання

4.6 Висновки

В четвертому розділі магістерської кваліфікаційної роботи виконано:

– обгрунтування та вибір мови програмування;
– обгрунтування та вибір середовища програмування;
– аналіз методів тестування;
– вибір методу тестування та саме тестування системи;
– проведено тестування.

В результаті аналізі методів тестування було обрано найбільш вигідний метод тестування системи, тестування за яким показало коректність роботи системи.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть Яровий А.А. та Петришин С.І.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.1.

Таблиця 5.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Яровий А.А.	2. Петришин С.І.
	Бали, виставлені експертами:	
1	4	4
2	4	3
3	4	4
4	4	4
5	3	4
6	3	4
7	3	3
8	4	4
9	3	3
10	3	4
11	4	3
12	3	3
Сума балів	$СБ_1 = 43$	$СБ_2 = 43$

Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 43$

Отже, з отриманих даних таблиці 4.1 видно, що нова розробка має високий рівень комерційного потенціалу.

5.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (5.1):

(5.1)

де М- місячний посадовий оклад конкретного розробника;

- кількість робочих днів у місяці, = 21 дні;

t - число днів роботи розробника, t = 60 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 5.2.

Таблиця 5.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	4000	190.48	10	1904.76
Інженер-програміст	8500	404.76	60	24285.71
Всього:				26190.47

Розрахуємо додаткову заробітну плату:

Нарахування на заробітну плату операторів НЗП розраховується як 37,5-40% від суми їхньої основної та додаткової заробітної плати:

5

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

(5.3)

де Ц – балансова вартість обладнання, грн;

– річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 5.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн
Персональний комп’ютер (ноутбук)	17000	25	3	1062,5
Всього:				1062,5

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

(5.4)

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 5.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Додатковий монітор	шт.	4000	1	4000
Комп'ютерна мишка	шт.	700	1	700
Клавіатура	шт.	300	1	300
Блокнот	шт.	80	1	80
Ручка	шт.	20	1	20
Всього з урахуванням транспортних витрат				5610

Витрати на силову електроенергію розраховуються за формулою:

; (5.5)

де В – вартість 1кВт-години електроенергії (В=1.7 грн/кВт);

Пк – установлена потужність комп'ютера (П=0,1кВт);

Пм – установлена потужність монітора (П=0,05кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=380 год.);

 – коефіцієнт використання потужності (< 1,).

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

(5.6)

Отже, розрахуємо інші витрати:

(грн).

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

В = ++1062,5+5610+63+ = 89562,87(грн.)

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

(5.7)

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

(5.8)

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

(грн.)

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

(5.9)

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки місць для реклами бронювання збільшиться на 75 процентів (що автоматично спричинить збільшення чистого прибутку підприємства на 50-75 процентів в залежності від вартості товару, далі будем використовувати середнє значення 62.5%), а кількість користувачів, які будуть користуватись торговим сервісом збільшиться: протягом першого року – на 350 користувачів, протягом другого року – на 200 користувачів, протягом третього року – 250 користувачів. Статистика онлайн-сервісу до впровадження інформаційної технології складала 1000 активних користувачів які в середньому приносили 300 гривень прибутку.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

Протягом другого року:

Протягом третього року:

Теперішню вартість інвестицій PV, що можуть бути вкладені в розроблену нами інтелектуальну систему, можна розрахувати за формулою:

$$PV = [(1…5) \times ЗВ] ,$$

де (1…5) – коефіцієнт, який враховує можливі додаткові витрати інвестора на можливе впровадження нашої розробки (оренда, підготовка персоналу, реклама тощо).

Для нашого випадку отримаємо:

$$PV = (1…5) \times ЗВ = 2.5 \times \ = 263420 \text{ (грн.)}.$$

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність вкладених інвестицій розраховується за формулою:

(5.10)

де ПП – приведена вартість всіх можливих чистих прибутків від реалізації розробки, грн;

PV – теперішня вартість інвестицій, PV = 263420 грн.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 5.1.

Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

(5.11)

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

(грн.)

Тоді розрахуємо

грн.

Оскільки , то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$ за формулою:

(5.12)

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

Т – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

або 63 %

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

,

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2020 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

Оскільки $E_в$ = 63% > $\tau_{мін}$ = 0,3 = 30%, то інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

5.5 Висновок

В даному розділі було здійснено оцінювання комерційного потенціалу розробки інформаційної технології з розробки та дослідження рекомендаційного сервісу для інтернет магазину з використанням технологій машинного навчання.

Проведено технологічний аудит з залученням двох експертів. Аналіз експертних даних показав, що рівень комерційного потенціалу розробки вище середнього. Дослідження комерційного потенціалу розробки підтвердило, що програмний продукт за своїми характеристиками випереджає аналогічні

програмні продукти і є перспективною розробкою. Він має кращі функціональні показники, а тому є конкурентоспроможним товаром на ринку.

Згідно із розрахунками всіх статей витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи загальна вартість витрат на розробку і впровадження складає грн.

Розрахована абсолютна ефективність вкладених інвестицій в сумі грн свідчить про отримання прибутку інвестором від впровадження програмного продукту у діяльність підприємства.

Щорічна ефективність вкладених в наукову розробку інвестицій складає 63%, що вище за мінімальну бар'єрну ставку дисконтування, яка складає 30%. Це означає потенційну зацікавленість інвесторів у фінансуванні розробки.

Термін окупності складає 1.59 року, що також свідчить про доцільність фінансування.

Усе це, узяте разом, забезпечує прийняття рішення про доцільність виготовлення нового продукту.

ВИСНОВКИ

Всі задачі, посавлені перед магістерською кваліфікаційною роботою виконано в повному об’ємі, а саме:

- обґрунтована доцільність створення онлайн-сервісу для бронювання столиків у ресторані;
- проаналізовані існуючі технології та методи створення клієнтської частини онлайн-сервісу;
- сформулювані вимоги до роботи технології та розроблене ТЗ;
- на основі існуючої моделі створення архітектури, розроблено архітектуру системи;
- розроблено алгоритми основних модулів системи та її структуру;
- проаналізовано основні технології для розробки клієнтської частини онлайн-сервісу та обрано найефективніший;
- проаналізовано основні технології тестування та проведено юзабіліті тестування системи;
- реалізовано та налаштовано роботу системи для бронювання столиків у ресторані;
- протестовано роботу системи
- виконано задачі економічного розділу.

Результати, одержані в процесі виконання магістерської кваліфікаційної роботи плануються до впровадження в розробки IT компанією «Delphi Software».

Метою роботи - розширення функціональних можливостей системи для бронювання місць в ресторані та покращення швидкості передачі даних між клієнтською частиною та серверною досягнуто шляхом розробки якісної клієнтської частини сервісу з використанням серверної частини, які найбільш краще взаємодіють між собою.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Що таке On-line сервіси? [Електронний ресурс] – Режим доступу: https://avada-media.ua/services/on-line-servisy/.
2. Онлайн сервіси і їх види [Електронний ресурс] – Режим доступу: http://book-science.ru/applied/it/onlajn-servisy-i-ih-vidy.html.
3. За і проти: Навіщо ресторанам сервіси бронювання столиків [Електронний ресурс] – Режим доступу: https://habr.com/ru/company/jowi/blog/369115/.
4. Клієнтська оптимізація і етапи розробки [Електронний ресурс] – режим доступа: http://tods-blog.com.ua/ezarabotok/klientskaya-chast/.
5. Купер А.Б. Про інтерфейс. Основи проектування взаємодії. – Лондон, 2012. – 365 с.
6. Раскин Д.М. Інтерфейс: нові напрямки в проектуванні комп'ютерних систем. – Париж, 2009. – 144 с.
7. Поради до розробки архітектури сайту [Електронний ресурс] – Режим доступу: https://semantica.in/blog/polnoe-rukovodstvo-po-planirovaniyu-arkhitektury-sajta-15-sovetov-dlya-maksimalnogo-seo.html#toc1Eclipse.
8. React [Електронний ресурс] – режим доступа: https://ru.wikipedia.org/wiki/React.
9. React Native [Електронний ресурс] – режим доступа: https://en.wikipedia.org/wiki/React_Native.
10. Переваги використання React Js [Електронний ресурс] – режим доступа: https://xbsoftware.ru/blog/pochemu-stoit-ispolzovat-react-js-razrabotke-prilozhenij/.