Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ РЕСТОРАННОГО БІЗНЕСУ

Текстова частина до курсової роботи
за спеціальністю 121 „Інженерія програмного забезпечення”

Керівник курсової роботи
Яремко С. А.
(прізвище та ініціали)
_____
(підпис)
“____” _____ 2021 р.

Виконала студентка
Кузів В. Д.
(прізвище та ініціали)
“____” _____ 2021 р.

Київ 2021

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри інформатики,
доцент, кандидат фіз.-мат. наук
_____С. С. Гороховський
(підпис)
„____”_____2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студенту Кузів Василині
___3____ курсу
факультету інформатики
ТЕМА: «Розробка інформаційної системи для ресторанного бізнесу»
Вихідні дані:

Зміст ТЧ до курсової роботи:

Вступ
Розділ 1. Аналіз предметної області
1.1 Призначення системи
1.2 Збір фактів
1.3 Специфікація вимог
Розділ 2. Дизайн системи
2.1 Інфологічне та даталогічне проектування
2.2 Архітектура системи
2.3 Інтерфейс користувача
Розділ 3. Розробка продукту
3.1. Опис обраних технологій
3.2. Імплементація
Висновки
Список джерел
Додатки

Дата видачі „___” _____ 202_ р. Керівник __
(підпис)
Завдання отримано _____
(підпис)

Календарний план виконання роботи

Тема: «Розробка інформаційної системи для ресторанного бізнесу»

№ п/п	Назва етапу курсового проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	Листопад 2020р.	
2.	Вивчення предметної області.	Листопад 2020р.	
3.	Огляд літератури за темою роботи.	Грудень 2020р.	
3.	Оволодіння навичками роботи із React	Січень 2021р.	
4.	Проектування системи	Січень 2021р.	
5.	Програмування розробленої системи	Лютий – березень 2021р.	
6.	Написання пояснювальної роботи.	Квітень 2021р.	
7.	Створення слайдів для доповіді та написання доповіді.	Квітень 2021р.	
8.	Аналіз отриманих результатів з керівником, написання доповіді.	Квітень 2021р.	
9.	Остаточне оформлення пояснювальної роботи та слайдів.	Травень 2021р.	
10.	Захист курсової роботи (проекту)	24.05.2021р.	

Студент **Кузів В.Д.**

Керівник **Яремко С.А.**

“_____” _____

ЗМІСТ

АНОТАЦІЯ

У цій роботі виконується розробка інформаційної системи для ресторанного бізнесу з можливістю деталізованого бронювання місць. Спочатку виконується ґрунтовний аналіз предметної області: дослідження ринку, збір фактів, проведення опитування та інтерв’ю. На основі отриманих даних здійснюється моделювання персони та створення CJM. Після чого проводиться специфікація функціональних та нефункціональних вимог для майбутньої системи.

Наступним етапом є інфологічне та даталогічне проектування. Відповідно до вимог створюється архітектура системи та розробляється тестовий графічний інтерфейс користувача. Останній розділ містить опис обраних технологій та деталі реалізації основних функціональностей.

ВСТУП

Актуальність, наукове та практичне значення обраної теми

Дуже часто ми приходимо у заклад і бачимо, що всі місця зайняті, тому доводиться чекати або поспіхом обирати інший ресторан, кафе тощо. За допомогою системи, розробленій у цій курсовій роботі, ми зможемо заздалегідь бронювати місце, яку нам подобається, на зручний час.

Використання системи онлайн-бронювання ресторанів надає клієнтам свободу робити бронювання, коли завгодно, без необхідності телефонувати або приходити передчасно у заклад. За певний час світ звик до бронювання квитків у кінотеатри або купівлю квитків на залізничний або авіатранспорт, де можна обрати місця з вигідним розташуванням та максимально зручною датою і часом. Ми більше не спішимо у касу на декілька годин перед сеансом, щоб встигнути придбати хоча б якийсь квиток, а лише в декілька натискань у будь-який час доби бронюємо та купуємо все, що потрібно.

Цей застосунок для того, щоб максимально автоматизувати та спростити процес бронювання місць у закладах харчування. Крім того, він допомагає максимально деталізувати бронювання, шляхом можливості перегляду мапи ресторану та вибору столика відповідно до розташування у закладі.

На українському ринку наразі немає доступних аналогів такої системи, що є ключовим моментом актуальності та новизни цього дослідження.

Мета та завдання курсової роботи

Мета**:** Розробити інформаційну систему для ресторанного бізнесу з можливістю здійснення деталізованих бронювань.

Завдання**:** Спроектувати дизайн та архітектуру системи для автоматизованого бронювання місць у закладах харчування та реалізувати основні її функціональні можливості.

Об’єкт дослідження

Проектування та розробка веб-застосунку з використанням фреймворку React, технології Redux та NodeJS.

Практичне значення одержаних результатів

Розроблена система є повноцінним комерційним продуктом у сфері ресторанного бізнесу, готовим до експлуатації.

Джерела дослідження

Під час написання курсової роботи було досліджено наукові статті про сучасні патерни програмування різноманітних інформаційних систем. На етапі налізу було вивчено аналоги, різноманітні сервіси для бронювання та купівлі квитків для визначення найкращих підходів та обраних технологій для реалізації схожого функціоналу.

Структура роботи

Курсова робота складається з анотації, вступу, трьох основних розділів, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Призначення системи

У вітчизняному ресторанному бізнесі бронювання здійснюється зазвичай за допомогою телефонного дзвінку напряму у заклад або листування з адміністраторами закладу у соціальних мережах.

Дата система призначена для автоматизованого менеджменту бронювань місць у закладах харчування. Кінцеві користувачі мають змогу переглядати інформацію про заклади, наявність вільних місць на певну дату та час і бронювати певний столик відповідно до розташування у кафе, ресторані, кав'ярні тощо. Менеджери та керівники можуть оновлювати інформацію про заклади та переглядати історію здійснених операцій.

Зараз на українському ринку не представлено доступного аналогу такої системи. На світовому ринку можна виокремити такі застосунки:

a) Eat App [1];

Рисунок 0.1.1 – Інтерфейс застосунку «Eat App» на різних девайсах

Eat App - це глобальна хмарна крос-платформна система бронювання ресторанів. Вона включає потужне управління таблицями, базу даних гостей та функції CRM.

За даними компанії, на сьогодні система розмістила понад 6 500 000 гостей[3]. Eat була заснована Незаром Кадхемом та Девідом Фейяром у 2015 році і на сьогодні зібрала 7 200 000 доларів від 500 стартапів[4].

Ресторани можуть обрати одну з двох цінових позицій:

1. $ 119 / місяць: Повна система управління таблицями;
2. $ 209/місяць: Система управління таблицями з включеною інтеграцією POS;

b) Resy [5];

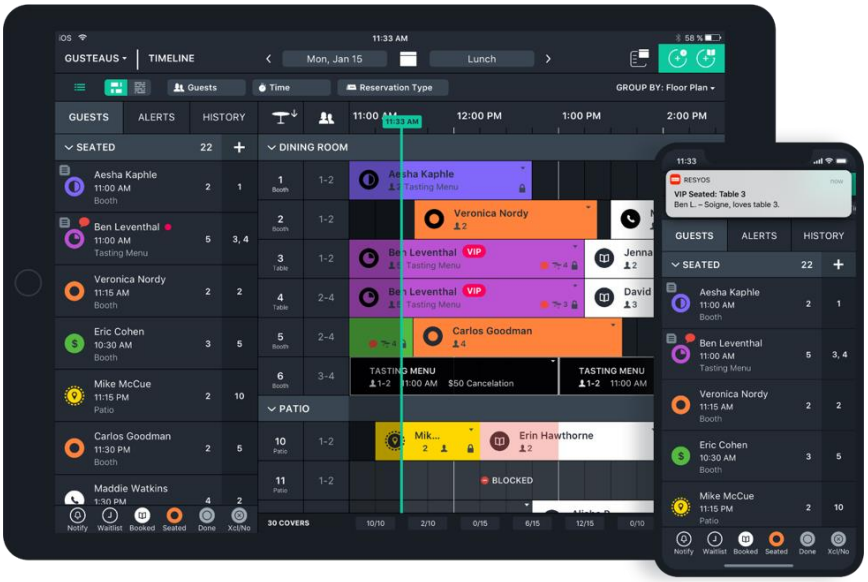


Рисунок 0.2.2 – Інтерфейс застосунку «Resy» на різних девайсах

Resy, як і інші повнофункціональні системи, дозволяє створити власний план закладу, включаючи барні крісла, їдальню та внутрішній дворик тощо. Гості можуть скасувати або підтвердити бронювання за допомогою месенджеру і навіть повідомити, що вони запізнюються.

За допомогою мобільного додатку Resy кожен гість зберігає власні дані, ділиться днями народження та вказує обмеження в їжі та наявність алергії.

Resy пропонує три рівні ціноутворення: Platform, Platform 360 та Full Stack. Починаючи з $ 189 на місяць рівень платформи пропонує необмежену кількість користувачів, пристроїв та покриттів. Рівень Platform 360 коштує 399 доларів на місяць і надає додаткові функції для брендингу та гостинності, включаючи настроювані гостьові повідомлення та пакет даних та аналітики. Повна версія стека починається з $ 899 на місяць, пропонуючи такі функції, як конфігурація мережі великого обсягу та доступ до API для ресторанів, які розширюють масштаби підприємств.

c) OpenTable [6];

У 1998 році веб-сайт розпочав роботу, пропонуючи бронювання в обмеженому виборі ресторанів у Сан-Франциско.

Після встановлення клієнти можуть бачити доступність закладів у режимі реального часу та бронювати столики через веб-сайт ресторану чи кафе.

Як і інші системи, OpenTable також зберігає корисні дані про клієнтів, такі як історія відвідувань та примітки про бронювання, щоб допомогти закладам надавати персоналізовані послуги.

Отже, мета усіх цих систем: автоматизувати процес бронювання, полегшивши таким чином роботу працівників закладу та зробити цей процес простим, зручним та приємним для клієнтів.

1.2 Збір фактів

1.2.1 Опитування та інтерв'ю

Це дослідження проводилось для того, щоб:

1. Дізнатися, як користувачі обирають заклад;
2. Проаналізувати досвід бронювання столика;
3. Дослідити, що є важливим для користувачів при бронюванні.

Опитування було проведено в онлайн-режимі за допомогою інструменту Google Forms. Питання були надіслані у телеграм-чати Києво-Могилянської академії, Українського католицького університету, МО "Молода Просвіта" та мережі колівінгів "Вільний"; у результаті отримано 146 відповідей[7].

Під час опитування було з'ясовано скільки опитуваних бронюють столик заздалегідь.

Рисунок 1.0.3 – Діаграма за результатами відповіді на питання «Чи бронюєте Ви столик заздалегідь?»

Лише 13% респондентів замовляють стіл завчасно; більшість робить це дуже рідко або майже ніколи.

Крім того, ми отримали дані про те, якому способу бронювання користувачі надають перевагу.

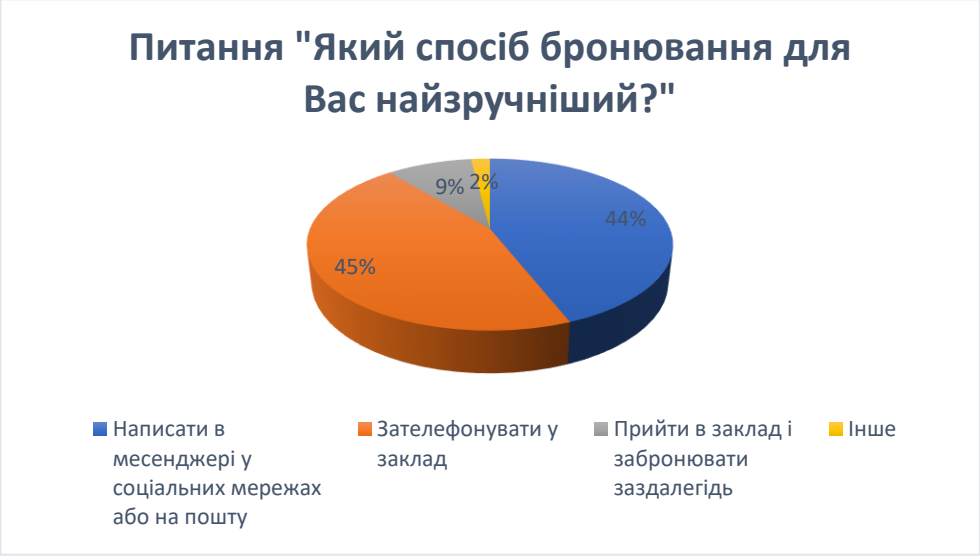


Рисунок 1.4 – Діаграма за результатами відповіді на питання «Який спосіб бронювання для Вас найзручніший?»

Отже, лише 9% людей здійснюють бронювання в офлайн-режимі. Інші обирають віддалене бронювання.

На питання, чому клієнти обирають саме такий спосіб, було отримані такі відповіді:

- «Максимально зручно - не потрібно чекати, поки піднімуть слухавку, пояснювати, виходити у тихе місце, щоб поговорити.»
- «Не хочу комунікувати напряму з людьми.»
- «Тому, що телефон завжди є під рукою, а можливість поїхати у заклад буває вкрай рідко.»
- «Це зручно і не потрібно витрачати власного часу на розмову. Коротко та швидко.»
- «Не люблю говорити по телефону з незнайомими людьми.»

Однією з найважливіших знахідок проведеного опитування є критерії при бронюванні. Респонденти виділили наступні пункти, як важливі для них:

- Точні дата та час
- Розташування столу та інтер’єр закладу
- Кількість місць

Тобто, під час опитування ми з’ясували, що менше 15% опитуваних регулярно бронюють столик у закладах харчування. Найзручнішими способами бронювання є : зателефонувати у заклад або написати у месенджері. Більшість респондентів хоче дуже стисло комунікувати з персоналом та скоротити спілкування до мінімуму.

Після цього було проведено інтерв’ю з 5-ма інтерв’юерами, два з яких працюють у сфері ресторанного бізнесу (додаток А). При проведенні інтерв’ю намагалися отримати відповіді на такі питання:

	Що виявляємо?	Для чого?
Потреби	Чи мають вони потребу завчасного бронювання місця у закладі?	Створення додатку для задоволення цієї потреби
Цінності	Що для них найважливіше при бронюванні?	Пріоритезація контенту та фіч нашого продукту
Мотивація	Чому вони хочуть бронювати стіл заздалегідь? Чому інколи цього не роблять?	Імідж бренду, врахування недоліків, інших способів, створення персони, щоб допомогти команді сфокусуватись
Мова	Мова, яку використовують користувачі	Дизайн і текст контенту Тон контенту
Розповідь	Досвід користувача, улюблені заклади	Створення карти користувацького досвіду, аналіз «найвищих» та «найнижчих» точок
Тип продукту	Чим зручніше користуватися додатком чи сайтом?	Визначення типу майбутнього продукту

Таблиця 1.1 – Цілі інтер'ю

Респонденти зазвичай бронюють столик, написавши у месенджер(соціальні мережі) ресторану, оскільки не хочуть спілкуватися з незнайомими людьми. При бронюванні для користувачів важливо вказати дату і час, кількість місць та розташування столу. Крім того, вони хотіли б мати можливість переглянути інтер'єр та меню закладу.

Висновок: клієнти прагнуть мінімізувати комунікацію з персоналом закладу і використовують для цього пошту або соціальні мережі. Потрібно врахувати побажання щодо бронювань при створенні функціоналу продукту.

1.2.2 Персона та CJM

Персона - це опис типового користувача певного продукту. Такий опис включає інформацію про вік, стать, місце проживання, основні цілі та проблеми умовного споживача. Використання персон дозволяє надати людям, які займаються розробкою нового продукту, детальний портрет кінцевого споживача[8].

Персона - це архетип користувача. Орієнтуючись на правильно складені персони, можна визначити, які функції повинні бути, яку навігацію очікує побачити користувач, якою має бути взаємодія і зовнішній вигляд продукту, щоб він був успішним. Робота з персонами допомагає оптимізувати розробку, скоротити Time-to-Market, а також уникнути непотрібних витрат часу і ресурсів.

На основі опитування, інтерв’ю та дослідження ринку була створена така персона кінцевого користувача: Максим – студент, який вивчає комп’ютерні науки та працює у IT-компанії неповний робочий день. Визначено його цілі, страхи й проблеми та складено CJM.

Customer Journey Map (CJM, в перекладі – карта користувацького шляху). CJM відображає весь шлях клієнта до досягнення своєї мети, повністю описує його взаємодію з продуктом або компанією у всіх точках контакту.

Було проаналізовано, як процес бронювання відбувається зараз: які дії виконує користувач на кожному кроці, з якими проблемами стикається і які покращення можна зробити (додаток В). Також створено модель TO-BE, тобто як цей шлях виглядатиме з нашим продуктом у розрізі: етап – цілі клієнта – дії клієнта (додаток С).

Визначено такі основні етапи процесу бронювання з відповідними цілями клієнта:

1. Вибір закладу – знайти бажаний заклад у додатку;
2. Перегляд сторінки закладу - дізнатися рейтинг закладу, переглянути інші фото, меню, інтер'єр та додаткові деталі;
3. Заповнення деталей бронювання – вказати дату, час початку та час закінчення бронювання, зазначити, чи потрібні дитячі стільчики або інші особливі умови, додати певні коментарі;
4. Вибір місця (столика) – переглянути наявні вільні місця, обрати столик, який найбільше сподобався;
5. Підтвердження бронювання – переглянути заповнену заявку та підтвердити її;
6. Похід у заклад – прийти у заклад у зазначений час та зайняти своє місце.

Крім кінцевих користувачів – клієнтів закладів – у нас є ще один тип користувачів системи – менеджери та адміністратори. Їхній користувацький шлях відрізняється. Основним кроком для них буде перегляд усіх деталей бронювань по датах для кожного закладу.

Також у даній системі існує окремий тип – адміністратор системи, функції якого – менеджмент усіх акаунтів закладів та операції з ними.

За допомогою створених CJM можна чітко окреслити функціонал майбутньої системи.

1.3 Специфікація вимог

1.3.1 Функціональні вимоги

Функціональні вимоги визначають поведінку системи, процес обчислення даних, обмін та керування даними та інших конкретних функцій, які система повинна робити[9].

Оскільки у системі передбачено декілька типів користувачів, то необхідно визначити функціональні вимоги для кожного окремо.

Функціональні вимоги для системи для типу користувача – клієнт:

1. Авторизація в системі відбувається за допомогою адреси електронної пошти та паролю.
2. Для реєстрації необхідно вказати прізвище, ім’я, унікальну пошту та номер телефону та підтвердити пароль.
3. Неавторизовані користувачі мають змогу переглядати, сортувати та фільтрувати усі заклади, а також переглядати інформацію про заклад на його сторінки, але вони не можуть переглядати вільні місця та здійснювати бронювання.
4. На сторінці пошуку картка закладу має включати: назву, головне фото, адресу, оцінку користувачів та вартість середнього прийому їжі. На сторінці закладу має відображатися його назва, опис, адреса, усі фото, години роботи, меню, оцінка та відгуки користувачів.
5. Система має забезпечувати сортування усіх закладів за середньою ціною (за спаданням і зростанням), рейтингом (на основі відгуків користувачів) та популярністю(на основі кількості бронювань), а

також фільтрацію закладів за категорією, середньою ціною, рейтингом та наявністю вільних місць на певну дату.

6. Має здійснюватися пошук закладу за назвою та перегляд закладів поблизу. Також має бути доступний перелік схожих закладів, до тих, які раніше відвідував або шукав користувач.

7. Авторизовані користувачі мають змогу дізнатися наявність вільних столиків на певну дату й час. Користувачі можуть здійснити бронювання, вказавши дату, час початку і час закінчення, та обравши один або декілька з вільних столиків у закладі. Крім того, клієнти можуть вказати, чи потрібне дитяче крісло та додати певний коментар. На цьому етапі статус бронювання визначається, як «Нове».

8. Користувачі можуть скасовувати або редагувати бронювання та переглядати усю історію бронювань.

9. При бронюванні можна замовити їжу з наявного меню на сторінці закладу.

10. Після здійснення походу у заклад, користувач має можливість оцінити заклад і залишити відгук, а статус бронювання змінюється на «Завершене».

Функціональні вимоги для адміністратора:

У обліковому запису адміністратора закладу можна створювати нові заклади та редагувати інформацію про них. Якщо заклад припиняє роботу з певних причин, його можна приховати.

Персонал закладу має можливість переглядати усю інформацію про бронювання для закладів, створених у їхньому акаунті.

Також функціонал для створення нових бронювань (якщо вони були здійснені по телефону або у інший спосіб) для обліку завантаженості.

Функціональною вимогою для адміністратора системи є створення нових облікових записів для менеджерів закладів харчування. Керівники закладів заповнюють заявку на створення такого акаунту, де заповнюють контактні дані і здійснюють оплату за користування послугами сервісу. Після чого адміністратор

системи створює новий акаунт за вказаними даними та надає персоналу доступ до системи.

1.3.2 Нефункціональні вимоги

Нефункціональні вимоги – це SRP-документ, який визначає якість атрибутів програмного забезпечення, такі, як надійність, зручність користування, стійкість до збоїв та швидкість реагування[10].

Функціональні вимоги описують, що система має робити, а нефункціональні вимоги вказують, як це має виконуватися[11]. Ці риси зазвичай розуміють як константи або критерії, які визначають рівень свободи розробників чи користувачів. Нефункціональні вимоги важкі для тестування; тому, вони, як правило, оцінюються суб’єктивно.

Можна специфікувати наступні нефункціональні вимоги. користувач може зареєструвати лише один профіль на одну адресу та номер телефону, тобто ці дані мають бути унікальними. Пароль не може містити менше, ніж 6 символів та зберігається у базі даних у зашифрованому вигляді. Якщо користувач вводить неправильні дані при реєстрації або авторизації, то отримує відповідне повідомлення. Користувач може переглядати заклади на головній сторінці або сторінці розширеного пошуку. На останній він може шукати заклад за іменем, сортувати та фільтрувати усі заклади. Якщо закладів з такими фільтрами немає, то не відображається жоден заклад.

Якщо неавторизований користувач спробує перейти на сторінку бронювання, то його буде перенаправлено на сторінку входу. Авторизовані користувачі можуть бачити лише свої бронювання.

Дата бронювання не можу бути меншою за сьогоднішній день, час закінчення бронювання повинен бути більшим за час початку. Після вибору дати та часу користувачу мають бути доступні для вибору лише вільні столики у обраному закладі. Перевірка наявності вільних місць здійснюється лише в межах конкретного ресторану.

Адміністратори мають мати можливість редагувати та приховувати лише ті заклади, які зареєстровані у їхньому профілі.

У цій курсовій роботі реалізовано не усі функціональні вимоги, а саме: пункти 1-4, 7 та частково 8 з «Функціональні вимоги для типу користувача – клієнт» та основний функціонал для бронювання. Інші пункти(5,6,8,9,10) залишаться, як подальші кроки та майбутні допрацювання.

РОЗДІЛ 2. ДИЗАЙН СИСТЕМИ

2.1 Інфологічне та даталогічне проектування

2.1.1 ER-модель

Початковий етап проектування бази даних, так званий етап інфологічного проектування, полягає у визначенні, яка інформація про предметну область повинна бути представлена в базі даних. Для задоволення інформаційних потреб певного кола користувачів розроблена конкретна інформаційна система з базою даних. Під предметною областю розуміють частину реального світу, інформація про яку цікавить цих користувачів, у нашому випадку – це сфера ресторанного бізнесу.

Таким чином, на цьому етапі постає питання побудови певної семантичної моделі домену, що відображає необхідну інформацію про вибраний предметний домен. Такою моделлю, яка широко використовується на етапі проектування інфологічних баз даних, є так звана модель сутності / відносин, скорочена як «ER-модель», запропонована в 1976 році Пітером Пін-Шань Ченом. Основними поняттями ER-моделі є сутність, зв'язок і атрибут. При інфологічному проектуванні бази даних довільний фрагмент предметної області може представляють як множину сутностей, між якими існує деяка множина зв'язків[12].

Моделювання предметної області базується на використанні графічних діаграм, що включають невелике число різнорідних компонентів. Модель "сутність-зв'язок" не визначає операції над даними й обмежується описом тільки їхньої логічної структури[13]. При проектуванні сутностей потрібно дотримуватися принципу відкритості/закритості (OCP), що був сформований Бертраном Меєром у 1988 році. У ньому стверджується: «Програмні сутності мають бути відкриті для розширення і закриті для зміни».

У системі, яка розглядається у цій роботі можна виділити такі сутності:

a) Адміністратор закладу;
b) Клієнт;

c) Заклад;

d) Столик;

e) Бронювання.

Сутність фактично задається множиною атрибутів, що описують властивості всіх членів даного набору сутностей і складають кортеж, що задає екземпляр сутності. П.Чен визначає атрибут як функцію, що відбиває набір сутностей у набір значень чи у декартів добуток наборів значень. Кожен атрибут забезпечується ім'ям, унікальним у межах сутності. Наприклад, атрибути сутності «Бронювання» можна описати так:

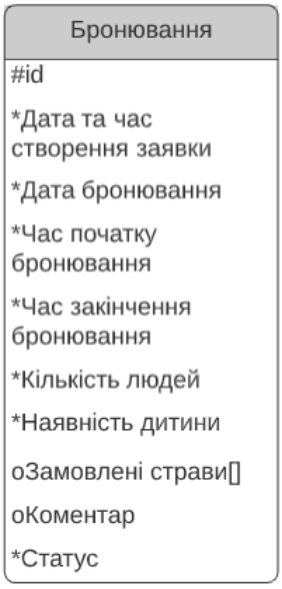


Рисунок 2.0.1 – Сутність «Бронювання» у ER-моделі

А про клієнта нам потрібно зберігати такі дані, як ПІБ, контакти та дані для авторизації.

Кожна сутність має свій унікальний ідентифікатор id, який однозначно визначає екземпляр класу. Після опису усіх необхідних атрибутів сутностей визначаємо зв’язки між ними. Розглянемо зв’язки між сутностями «Адміністратор закладу», «Заклад» та «Столик».

Рисунок 2.2 – Схема зв'язків між сутностями «Адміністратор закладу», «Заклад» та «Столик»

У одному закладі може працювати декілька адміністраторів та один працівник може займатися менеджментом декількох закладів у мережі, тому тут зв'язок багато-до-багатьох обов'язковий з обох сторін.

Столик обов'язково має бути розміщений у якому закладі, до того ж тільки одному, а заклад може мати багато столиків, але обов'язково хоча б один, тому між цими сутностями зв'язок один-до-багатьох обов'язковий з обох сторін.

У додатку D наведена ER-модель, яка задовільняє вимоги предметної області та дотримується наведених вище правил.

2.1.2 Реляційна модель

Наступний етап – даталогічне програмування, тобто перетворення ER-моделі на логічну модель даних, яка включає всю інформацію про предметну область. Реляційна модель – це структуроване на логічному рівні подання даних, яке має підпорядковуватися вимогам обраної СКБД(система керування базами даних)[14]. Кожна СКБД накладає ряд обмежень на проектування реляційної моделі, тому спочатку необхідно обрати СКБД та дослідити її особливості та специфіку.

Серед властивостей, які варто врахувати найважливішою є тип системи за моделлю організації даних. Розрізняють такі види: ієрархічні, мережні та реляційні бази даних[15]. У даній роботі використовується реляційна баз даних, що представлена у вигляді набору зв'язаних між собою таблиць. Іншим важливим чинником є кількість обмеження СКБД, а саме кількість можливих

атрибутів, записів у таблиці тощо. Потрібно обрати таку систему, вимоги якої не суперечать властивостям нашої предметної області.

При переході від ER-моделі до реляційної моделі потрібно дотримуватися наступних правил.

Для кожної сутності створюємо відповідну реляцію, де вказуємо атрибути, які необхідно зберігати в БД, тип даних, в якому виражається це поле та його обов'язковість. PRIMARY KEY нової реляції – це первинний ключ вихідної сутності[16]. Важливо розуміти, що не усі атрибути, які існували в моделі «сутність-зв'язок» потрібно зберігати в БД, деякі з них обраховуються у певні моменти або випливають з інших значень. Розглянемо перетворення сутності «Клієнт» у реляцію «User».

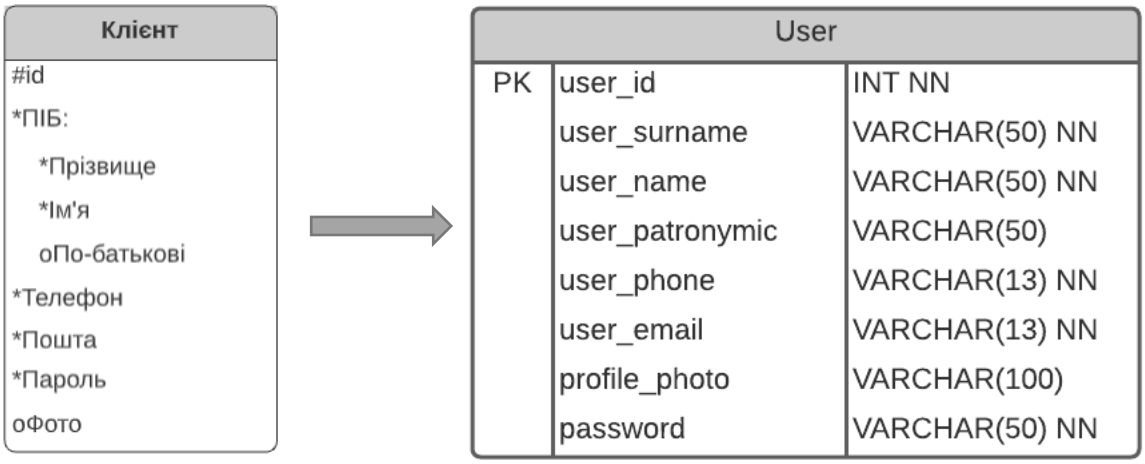


Рисунок 2.3 – Перетворення сутності «Клієнт» у реляцію «User»

У даному випадку нам необхідно зберігати усі атрибути, тому кожен атрибут сутності стає відповідним атрибутом реляції. Поле id є первинним ключем, тому стає PK відношення. Первинний ключ, прізвище, ім'я, телефон, пошта та пароль є обов'язковими, тому у реляції проставляємо NN(NOT NULL).

Якщо наявні багатозначні атрибути, то проводиться їх декомпозиція шляхом створення додаткових реляцій. Наприклад, у сутності «Відгук» можливий багатозначний атрибут «Фото». Тому при даталогічному проектуванні створюємо дві реляції «Review» та «Review_image», де зовнішнім ключем другого відношення буде первинний ключ першого.

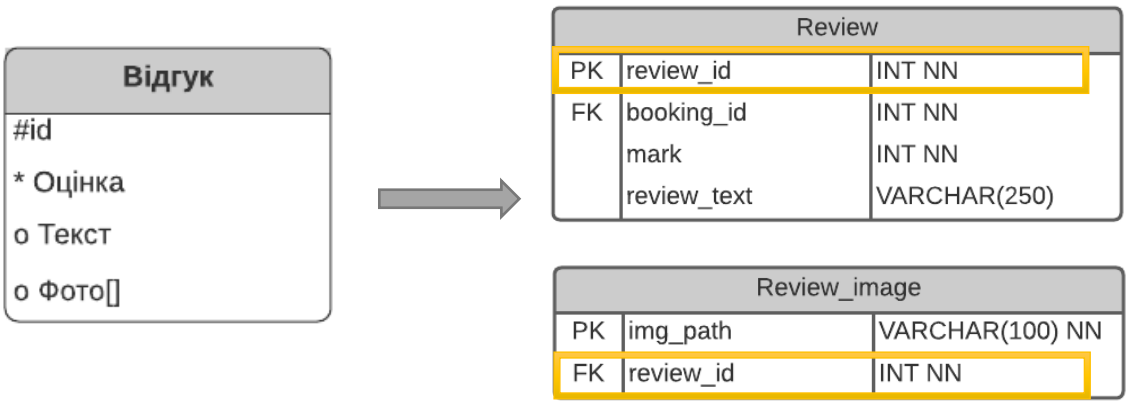


Рисунок 2.4 – Перетворення сутності «Відгук» у комбінацію відповідних реляцій

Аналогічним чином здійснюємо перетворення усіх сутностей ER-моделі у відповідні реляції логічної моделі (додаток Е).

2.2 Архітектура системи

«Мета архітектури програмного забезпечення – зменшити трудовитрати на створення і супровід системи»[17]. Для досягнення цієї мети програмне забезпечення має бути податливим – тобто має існувати можливість легко змінити його. Складність у такому випадку має бути пропорційна лише масштабам зміни, але не його формі.

Головне призначення архітектури – підтримка життєвого циклу системи. Гарна архітектура робить систему легкою в освоєнні, простою в розробленні, супроводі і розгортанні[18]. Кінцева її мета – мінімізувати витрати протягом терміну служби системи і максимізувати продуктивність програміста.

У даному проекті існує чіткий поділ на бекенд та фронтенд. Архітектура веб-застосунку матиме структуру, зображену на рисунку 2.5[19].

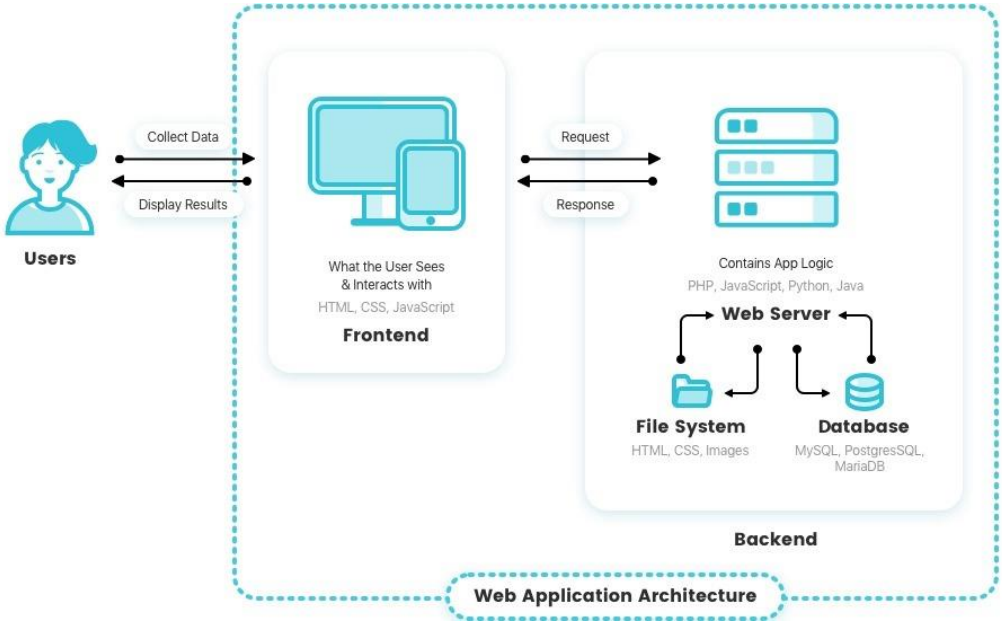


Рисунок 2.5 – Примітивна схема архітектури веб-застосунку
Джерело:https://reinvently.com/blog/fundamentals-web-application-architecture/

Back-end – програмно-апаратна частина сервіса, набір засобів, за допомогою яким здійснюється бізнес-логіка застосунку. Це серверна сторона, де отримуємо запити від клієнта та формуємо відповідь у вигляді інформації, отриманої з бази даних.

У файлі server.js визначено основні кінцеві точки API(endpoint API), а саме '/api/users', '/api/places' та '/api/bookings'. Для кожної кінцевої точки створено власний роутер, де здійснюється підключення до бази даних (додаток F).

Ці «роутери» здійснються SQL-запити, відповідно до запиту клієнта та надсилають відповідь у вигляді об’єкту або масиву об’єктів. Сортування, фільтрування та спеціалізована вибірка даних здійснюється засобами SQL, на відміну від методу, коли ми отримуємо всі наявні дані з таблиці, а тоді за допомогою функцій певної мови програмування виконувати маніпуляції. Наприклад, необхідно відобразити усі заклади, у яких тип – ресторан. Можна отримати всі дані з таблиці ‘places’ та для кожного закладу перевірити його тип. У даному застосунку це буде виконано таким чином – сервер отримує запит від клієнта, який включає набір можливих фільтрів, серед яких – фільтр по типу закладу. І після виконання запиту

```
query = "SELECT * FROM places WHERE type = '"+ type+"' ";
```

отримуємо лише ті заклади, які потрібно відобразити на сторінці. Аналогічно з процесом авторизації: ми отримуємо запит, який містить електронну пошту та пароль користувача. У системі не зберігається в момент звернення список усіх користувачів та їхніх даних, що є небезпечним з погляду безпеки, а здійснюється запит до бази даних для пошуку клієнта саме з такими даними, при успішному запиті отримуєму усю інформацію про цього користувача.

```
userRouter.post('/signIn', expressAsyncHandler(async (req, res) =>{
    connection.promise().query("SELECT *\n" +
        "FROM users AS U\n" +
        "WHERE U.user_email = '"+req.body.email + "';")
        .then( ([rows, fields]) =>{
            if (rows[0]){
                if ( bcrypt.compareSync(req.body.password,rows[0].password)){
                    let user = rows[0];
                    res.send({
                        _id: user["user_id"],
                        surname: user["user_surname"],
                        name: user["user_name"],
                        email: user["user_email"],
                        isAdmin: user["isAdmin"],
                        token: generateToken(user)
                    });
                    return;
                }
            }
            res.status(401).send({message:'Invalid email or password. Please try
again.'})
        })
        .catch(err =>{console.log(err);});
}));
```

Тобто клієнт завжди звертається на певну кінцеву точку, де здійснюється SQL-запит, який задовільняє саме той запит користувача. Тому можна сказати, що більша частина бізнес-логіки реалізовується функціями мови програмування структурованих засобів.

У проекті є два package.json: один для React та інший для nodejs API. Завжди найкращою практикою є наявність абсолютно різних модулів node_ для кожного з них. Таким чином, не виникне проблем зі злиттям або будь-яких інших проблем, пов'язаних зі зіткненням модулів веб-вузлів та серверів.

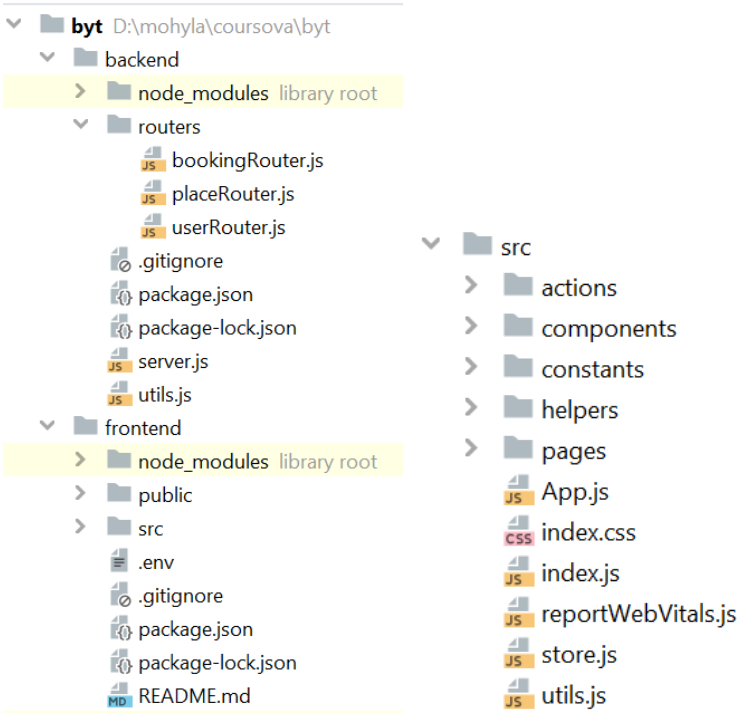


Рисунок 2.6 – Файлова структура проекту

/ src /…: усі ресурси, створені для клієнтської частини програми, повинні знаходитись у цьому каталозі.

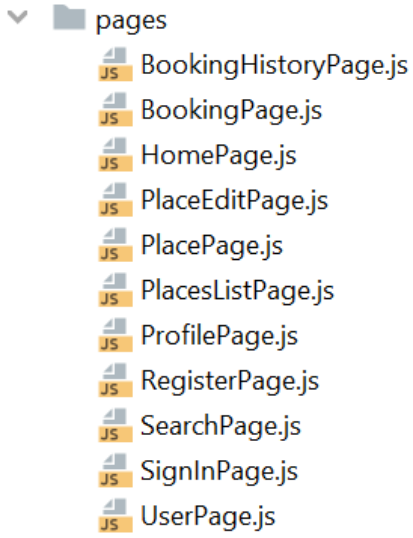


Рисунок 2.7 – Перелік об'єктів для відображення сторінок веб-застосунку

Файл "/src/App.js" є основним контейнером додатку і служить точкою входу (додаток G). У ньому визначаються кінцеві точки та адреси для усіх сторінок, створених у директорії "/src/pages". Сторінки створено відповідно до мапи сайту, створеної у частині 3 «Інтерфейс користувача» цього розділу. Тобто при авторизації користувач переходитиме за адресою ../SignIn, де відображатиметься файл SignInPage.js.

Деякі сторінки складаються з декількох частин, реалізованих у каталозі “/src/components”. Відповідно до функціональних вимог на головній сторінці потрібно відображати усі наявні заклади. Кожен заклад представлено у вигляді карточки з головним зображенням, назвою, адресою та рейтингом. Тому доцільно було виділити цей елемент, як окремий компонент Place.js. Також на головній сторінці (HomePage) використовуються інші компоненти: анімація завантаження(LoadingBox.js) та кастомне повідомлення (MessageBox.js).

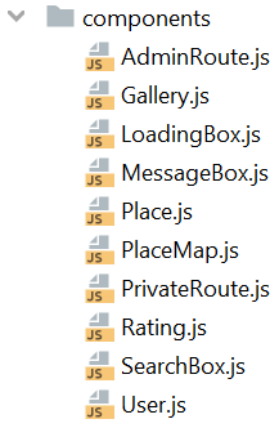


Рисунок 2.8 – Перелік додаткових компонентів

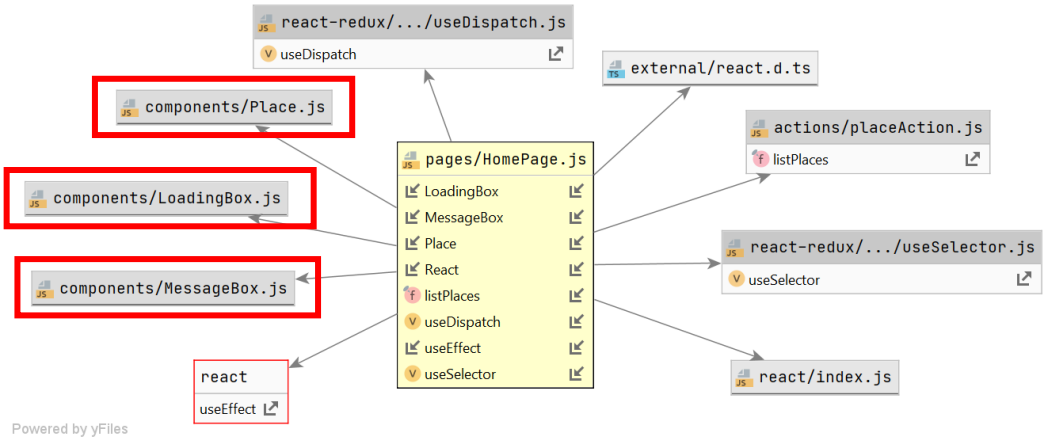


Рисунок 2.9 – Діаграма залежностей об’єкту HomePage

Однією з поширених практик побудови архітектурних додатків на React є використання Redux. Redux – це контейнер для управління стану застосунку та багато в чому нагадує Flux. Redux не прив'язується безпосередньо до React.js і може також використовуватися з іншими js-бібліотеками та фреймворками[20].

Ключові моменти Redux:

Сховище (store): зберігає стан застосунку.

Дії (actions): деякий набір інформації, який надходить із сайту до сховища та показує, що саме потрібно зробити. Для передачі цієї інформації в сховища використовується метод dispatch().

Творці дій(action creators): функції, які створюють дії.

Редюсер(reducer): функція (або кілька функцій), яка отримує дію і відповідно до цієї ситуації змінює стан сховища[21].

Спільну схему взаємодії елементів архітектури Redux можна виразити у наступній схемі:

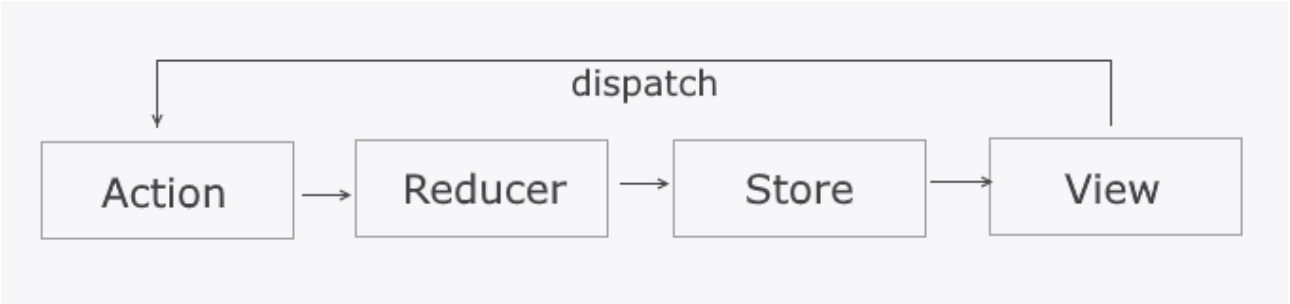


Рисунок 2.10 – Схема взаємодії елементів при використанні Redux

З View (тобто з компонентів React) надсилаємо дію, цю дію отримує функція reducer, яка відповідно оновлює стан сховища. Після чого компоненти React застосовують оновлений стан.

Для обробки дій користувача зі сторінок викликаються обробники дій з папки “/src/actions”. Саме там надсилаються GET та POST запити на сервер з певними параметрами. У даному проекті є три редюсери placeReducer, userReducer та bookingReducer, які оновлюють сховище store.js.

Отже, архітектура програмного забезпечення - це скелет системи та структура взаємодії усіх високорівневих компонентів. Загалом, за правильно розробленої допомогою архітектури програмного забезпечення можна контролювати продуктивність, масштабованість та надійність системи.

2.3 Інтерфейс користувача

Інтерфейс користувача – це засіб передачі інформації між користувачем та програмним забезпеченням. При розробці програмного забезпечення для кінцевих користувачів цей пункт має дуже велике значення, оскільки впливає на зручність та простоту використання, від чого прямо пропорційно залежить прихильність користувачів. За допомогою правильного UI користувач не потребує знань певних технологій, щоб маніпулювати апаратним забезпеченням та виконувати різні дії в застосунку[22].

Є декілька типів користувацького інтерфейсу, серед яких інтерфейс командного рядка(CLI) або текстовий, графічний(GUI), дизайн якого розроблятиметься на цьому етапі, тактильний, жестовий, голосовий та інші[23].

В основу розробки графічного користувацького інтерфейсу цієї системи лягли дослідження, описані у пункті 2 розділу 1(«Збір фактів»). Було створено графічний інтерфейс мобільного веб-застосунку за такими кроками:

a. Розробка інформаційної архітектури;
b. Проектування каркасу веб-сайту;
c. Розробка прототипу;
d. Перевірка ергономічності.

Інформаційна архітектура – це практика організації та структуризації контенту всередині продукту чи сервісу, управління змістом та інормаційна політика продукту. Це потрібно для зменшення когнітивного навантаження користувача, як результат - більш сфокусований шлях взаємодії з продуктом[24]. На даному етапі створено мапу сайту у додатку Н, що являє собою ієрархічну діаграму структури сайту і відображає пріоритизацію, лінкування та лейбелінг сторінок.

Для системи бронювання столиків виділено такі основні сторінки:

a. Старт пошуку/популярні;
b. Результати пошуку;
c. Заклад;
d. Бронювання;
e. Профіль користувача;
f. Вхід;
g. Реєстрація.

Деякі з них мають посилання на другорядні сторінки, наприклад, зі сторінки закладу можна перейти на сторінку з меню, відгуками або на форму реєстрації.

Після визначення компонентів продукту можна переходити до їх візуалізації, а саме створення wireframes – “креслення” майбутнього продукту,

що за допомогою простих фігур відображає його структуру, контент, шаблон та базові взаємодії[25]. Такий каркас майбутнього продукту потрібен для визначення візуальної ієрархії та контенту, а також для концентрації на структурі сторінки та планування подачі цього контенту. Для розробника він надає розуміння сценаріїв та взаємодій, допомагає при плануванні та оцінці зусиль з імплементації.

Ключовим моментом при розробці графічного інтерфейсу користувача є створення прототипу. Прототипування — симуляція, що дозволяє користувачу отримати досвід від вибраних аспектів продукту. Може мати різний вигляд в залежності від задачі, від текстового опису чи схеми до працюючого додатку з обмеженим функціоналом. Прототип створюється з метою перевірки придатності пропонованих для застосування концепцій, архітектурних і технологічних рішень, а також для представлення програми замовнику на ранніх стадіях процесу розробки.

На основі створеного макету розробляється деталізований план сторінок веб-застосунку. Для цього проекту використовувався інструмент Figma - онлайн-сервіс для розробки інтерфейсів і прототипування[26]. Створено інтерактивний прототип мобільної версії застосунку, що забезпечує можливість тестування основних функціональних вимог (Додаток І). Наступні зображення відображають процес авторизації при відсутності заповнення деяких полів.

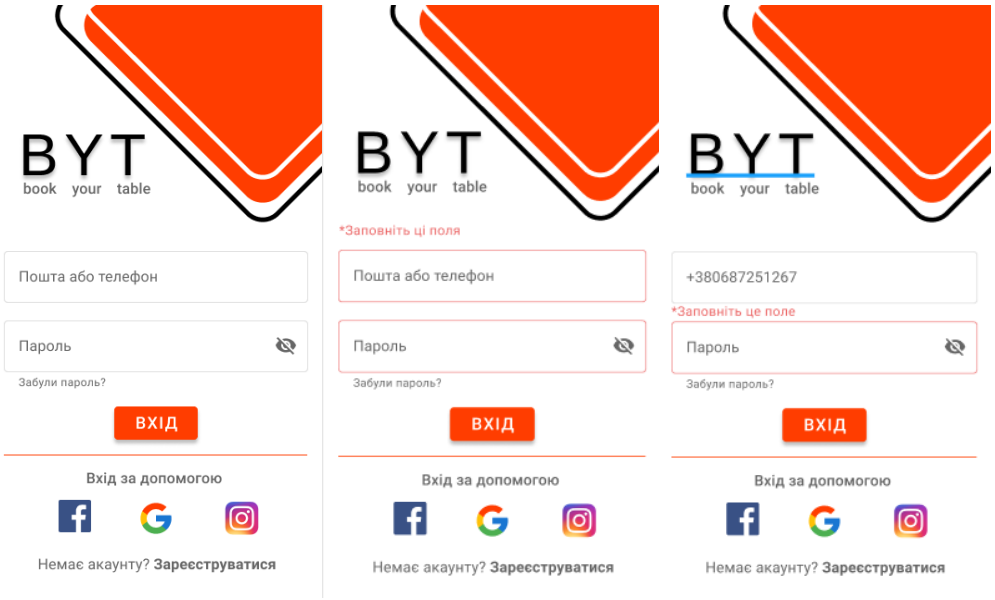


Рисунок 2.11 – Прототип графічного інтерфейсу користувача. Сторінка авторизації

Наступні макети дають увлення про саму структуру сайту та його контент. За результатами опитування було виявлено, що користувачам важливо мати можливість переглядти заклади поблизу, тому мобільну версію поділено на три основні сторінки: усі заклади, заклади поблизу та профіль користувача.

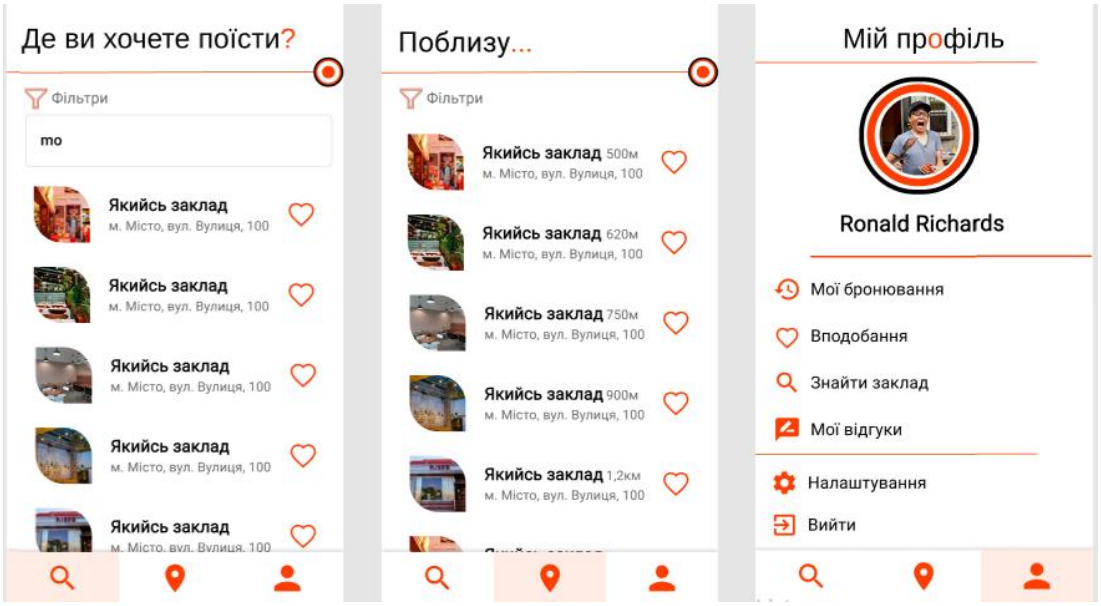


Рисунок 2.12 – Прототип графічного інтерфейсу користувача. Основні сторінки

Три натиску на назву або іконку закладу можна перейти на сторінку закладу, де відображається додаткова інформація та бнопка для переходу на форму бронювання. На сторінці бронювання можна обрати певну дату та час,

переглянути розташування місць, вільні столики, зафарбовані оранжевим кольором, та недоступні, зафарбовані сірим кольором, столики.

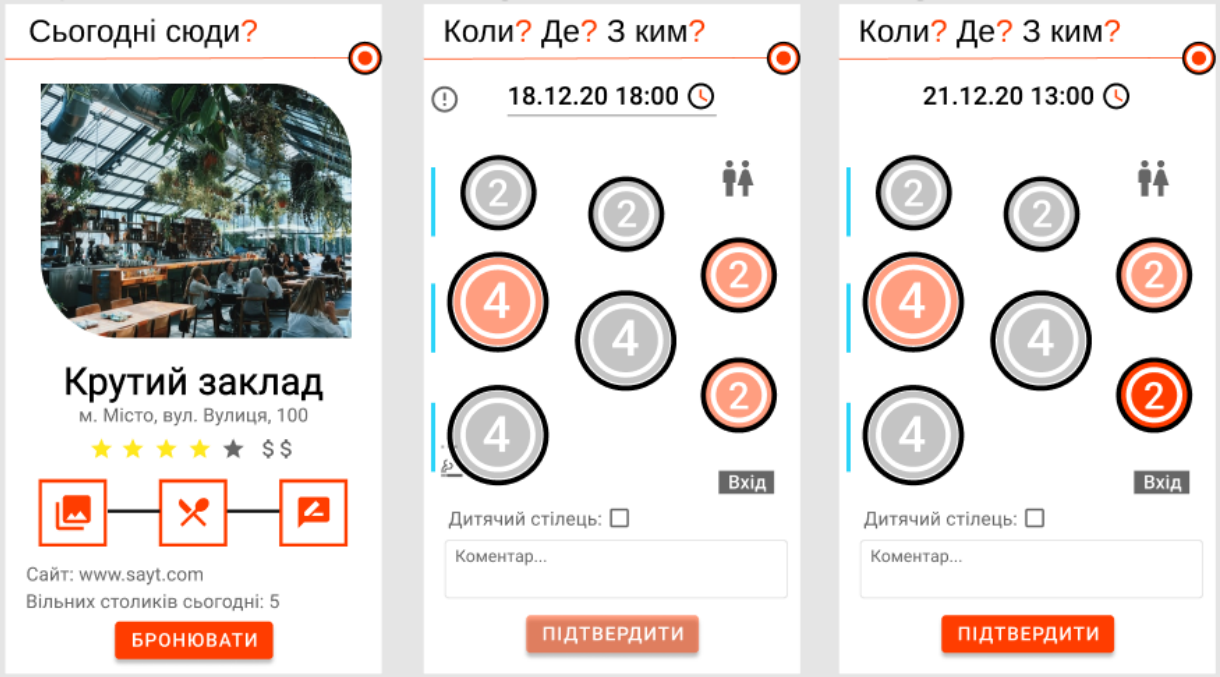


Рисунок 2.13 – Прототип графічного інтерфейсу користувача. Процес бронювання

Після вибору певного столика можна додати додаткову інформацію та підтвердити бронювання. Прототип дає змогу оцінити нашу реляційну модель, спроектовану на етапі даталогічного програмування та внести правки, якщо потрібно.

РОЗДІЛ 3. РОЗРОБКА ПРОДУКТУ

3.1. Опис обраних технологій

Вибір правильного стека технологій може вплинути на час розробки, вартість, якість програми, а також на масштабованість майбутнього продукту. Відповідно до архітектури, розробленої у попередньому розділі було обрано такі технології для кожного з рівнів.

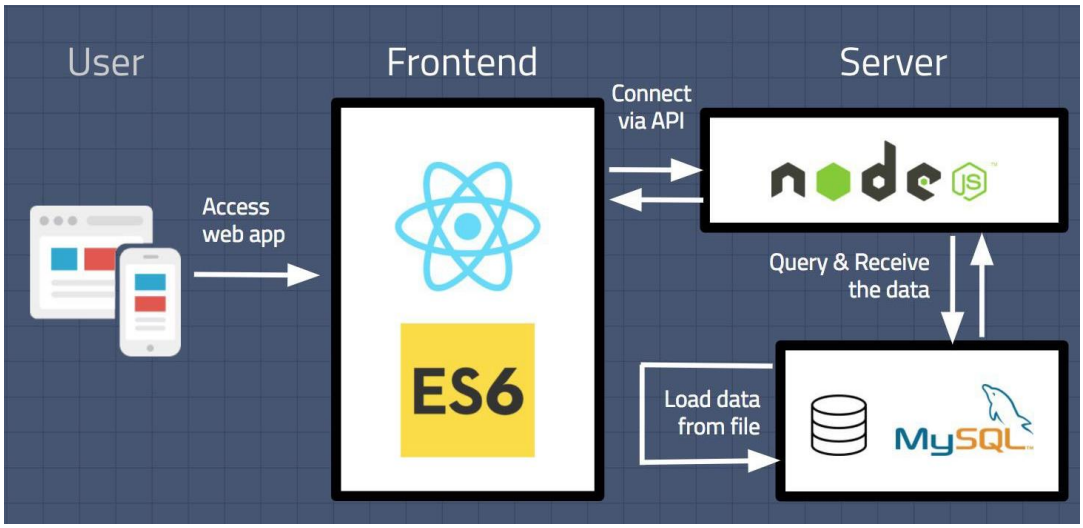


Рисунок 3.1 – Схема стеку обраних технологій, розділених по рівнях. Джерело: https://www.byperth.com/2018/04/19/guide-building-data-science-web-application-with-react-nodejs-and-mysql/stock-analyzer-project-architecture-2/

Користувач заходить на веб-сайт і бачить графічний інтерфейс, створений на React. Коли користувач запитує дані, інтерфейс з'єднується з кінцевими точками API, створеними NodeJS та запросить ці дані. NodeJS буде запитувати дані із бази даних MySQL, а потім відправить результат назад в веб-інтерфейс.

Можна зрозуміти доцільність JavaScript як мови веб-розробки з того факту, що згідно з останніми звітами, JavaScript наразі використовується у понад 94 відсотках всіх веб-сайтів[27].

JavaScript - це мова програмування на клієнтської сторони, яка допомагає розробнику веб-сайтів розробляти веб-додатки та створювати динамічні та інтерактивні веб-сторінки, реалізуючи власні сценарії на стороні клієнта. Розробники також можуть використовувати крос-платформні середовища виконання, такі як Node.js, для написання коду на серверній сторіні в JavaScript.

Є можливість створювати веб-сторінки, які добре працюють у різних браузерах, платформах та пристроях, поєднуючи JavaScript, HTML5 та CSS3.

На веб-ринку доступно багато платформ JavaScript, таких як AngularJS, ReactJS, NodeJS тощо. Використовуючи ці фреймворки, можна зменшити кількість часу та зусиль, необхідних для розробки сайтів та програм на основі JS. JavaScript дозволяє легко створювати масштабні веб-програми.

Для імплементації графічного інтерфейсу користувача було обрано JavaScript-фреймворк React. Сучасні фреймворки зовнішнього інтерфейсу можуть управляти всіма трьома елементами призначеного для користувацького інтерфейсу; у них є HTML-шаблони, стилі та інтерактивні функції. ReactJS - це не фреймворк; це UI-бібліотека, створена Facebook і підтримувана величезною спільнотою. ReactJS підходить для складних додатків і більше орієнтований на розширений користувацький інтерфейс і повторно використовувані компоненти на дуже складній логіці зовнішнього інтерфейсу.

ReactJS має багату екосистему, яка складається з самого ReactJS і бібліотеки React-DOM для управління об'єктами DOM. Крім того, існує React-Router, який відповідає за маршрутизацію. Фреймворк поставляється з JSX, який є розширенням синтаксису для Javascript для створення шаблонів в ReactJS. Також, оскільки майбутні кроки включають розробку мобільного додатку, ReactJS має структуру для мобільної розробки під назвою React Native, яка дуже схожа на сам ReactJS[28].

Особливості фреймворку:

a. компонентно-орієнтований;
b. декларативний;
c. продуктивний (завдяки React Virtual DOM);
d. серверний рендеринг;
e. наявність Redux;
f. підтримка PWA;
g. JSX.

React пропонує просте і функціональне створення компонентів, а також пропагує їх використання для підтримки елегантного коду API. Маючи великий вибір легкодоступних плагінів і розширень з відкритим кодом, можна розробити практично будь-який тип веб-сайту. З точки зору компанії, найбільш суттєвими перевагами ReactJS є відносно низька вартість розробки і короткі терміни розробки.

Node.js представляє середовище виконання коду на JavaScript, яке побудоване на основі движка JavaScript Chrome V8, який дозволяє транслювати виклики на мові JavaScript в машинний код[29]. Node.js перш за все призначений для створення серверних додатків на мові JavaScript. Движок V8 компілює JavaScript в машинний код, замість того, щоб інтерпретувати його або виконувати як байт-код, і це робить Node дійсно швидким. Полегшений Javascript забезпечує високу продуктивність з меншою кількістю рядків у порівнянні з Java або C.

В основі роботи Node.js лежить асинхронність[30]. В доповнення до неблокуючого вводу-виводу це дозволяє серверному додатку на Node.js обслуговувати набагато більше клієнтських запитів в одиницю часу, ніж аналогічним додатком, розробленим на більшості інших технологій серверної розробки.

Для даного проекту було обрано СКБД MySQL через ряд причин, наведених нижче[31].

1. Масштабованість та гнучкість

Сервер баз даних MySQL забезпечує максимальну масштабованість, підтримуючи здатність обробляти глибоко вбудовані додатки з розміром(application footprint) всього 1 МБ для запуску масивних сховищ даних, що містять терабайт інформації. Гнучкість платформи - це особливість MySQL з усіма можливостями Linux, UNIX та Windows. І, звичайно, природа MySQL з відкритим кодом дозволяє додати унікальні вимоги до сервера баз даних[32].

2. Висока продуктивність

Унікальна архітектура механізму зберігання дозволяє професіоналам конфігурувати сервер MySQL спеціально для певних додатків. Незалежно від того, чи майбутня програма - це високошвидкісна система обробки транзакцій або об'ємний веб-сайт, що обслуговує мільярд запитів на день, MySQL може задовольнити найвибагливіші очікування продуктивності будь-якої системи. Завдяки високошвидкісним утилітам для завантаження, відмінним кешам пам'яті, повнотекстовим індексам та іншим механізмам, що підвищують продуктивність, MySQL пропонує всі необхідні засоби для сучасних важливих бізнес-систем.

3. Висока доступність

Висока надійність та постійна доступність є головними ознаками MySQL, оскільки клієнти покладаються на MySQL, щоб гарантувати цілодобову безперебійну роботу. MySQL пропонує безліч варіантів високої доступності: від високошвидкісних конфігурацій реплікації до спеціалізованих кластерних серверів, що пропонують миттєвий збій та сторонніх постачальників, що пропонують унікальні високодоступні рішення для сервера баз даних MySQL.

4. Надійна підтримка транзакцій

MySQL пропонує один з найпотужніших на ринку механізм транзакційних баз даних. Особливості включають повну підтримку транзакцій ACID (атомарну, послідовну, ізольовану, довговічну), необмежене блокування на рівні рядків та підтримку транзакцій з декількома версіями, де читачі ніколи не блокують авторів і навпаки[31]. Повна цілісність даних також забезпечується завдяки посиленомому механізму посилальної цілісності(referential integrity), спеціалізованим рівням ізоляції транзакцій та миттєвому виявленню взаємного блокування.

5. Сильні сторони Web та сховища даних

MySQL є фактичним стандартом для веб-сайтів із високим трафіком завдяки високопродуктивній системі запитів, надзвичайно швидкій можливості вставки даних та потужній підтримці спеціалізованих веб-функцій, таких як швидкий пошук повного тексту. Ці ж сильні сторони також застосовуються до

середовищ зберігання даних, де MySQL масштабується до терабайтового діапазону або для окремих серверів, або для масштабованих архітектур. Інші функції, такі як таблиці основної пам'яті, B-дерева та хеш-індекси, а також стислі таблиці архівів, що зменшують вимоги до сховища на вісімдесят відсотків, роблять MySQL сильним як для веб-програм, так і для програм бізнес-аналітики.

6. Потужний захист даних

Оскільки захист даних компанії – це завдання номер один для професіоналів баз даних, MySQL пропонує виняткові функції безпеки, які забезпечують абсолютний захист даних. Що стосується автентифікації бази даних, MySQL забезпечує потужні механізми, що забезпечують доступ до сервера баз даних лише авторизованим користувачам, з можливістю блокування користувачів до рівня клієнтської машини. Також забезпечується підтримка SSH та SSL для забезпечення надійних та захищених з'єднань. Наявна гранульована структура об’єктів, завдяки якій користувачі бачать лише ті дані, до яких мають доступ, а потужні функції шифрування та дешифрування даних забезпечують захист конфіденційних даних від несанкціонованого перегляду. Крім того, утиліти для резервного копіювання та відновлення, що надаються через MySQL та сторонніх постачальників програмного забезпечення, дозволяють забезпечити повне логічне та фізичне резервне копіювання[33].

7. Комплексна розробка додатків

Однією з причин, чому MySQL є найпопулярнішою у світі базою даних з відкритим кодом, є те, що вона забезпечує всебічну підтримку для різних потреб в розробці додатків. У базі даних можна знайти підтримку для збереження процедур, тригерів, функцій, виглядів, курсорів тощо. Для вбудованих програм доступні бібліотеки плагінів для інтегрування бази даних MySQL майже в будь-яку програму. MySQL також пропонує роз'єми та драйвери (ODBC, JDBC тощо), які дозволяють програмам використовувати MySQL як бажаний сервер управління даними.

8. Легкість управління

MySQL пропонує виняткові можливості швидкого запуску, середній час від завантаження програмного забезпечення до завершення встановлення становить менше п’ятнадцяти хвилин. Це вимога забезпечується, незалежно від того, платформою є Microsoft Windows, Linux, Macintosh або UNIX. Після встановлення такі функції самокерування, як автоматичне розширення простору, автоматичний перезапуск та динамічні зміни конфігурації, виконують значну частину роботи адміністраторів баз даних. MySQL також пропонує повний набір графічних інструментів управління та міграції, які дозволяють керувати, усувати несправності та контролювати роботу багатьох серверів MySQL з однієї робочої станції.

Остання версія MySQL - одна з найпопулярніших баз даних у світі. Це відкритий код, надійний, сумісний з усіма основними хостинг-провайдерами, економічно ефективний і простий в управлінні. Саме тому MySQL було обрано, як СКБД для цього проекту.

3.2. Імплементація

Як зазначалося раніше у пункті 3 Розділу 1 «Специфікація вимог» у цій курсовій роботі реалізовано не всі функціональні вимоги. На цьому етапі розроблено графічний дизайн для десктопного веб-застосунку та

Для клієнта реалізовано авторизацію у системі за допомогою адреси електронної пошти та паролю.

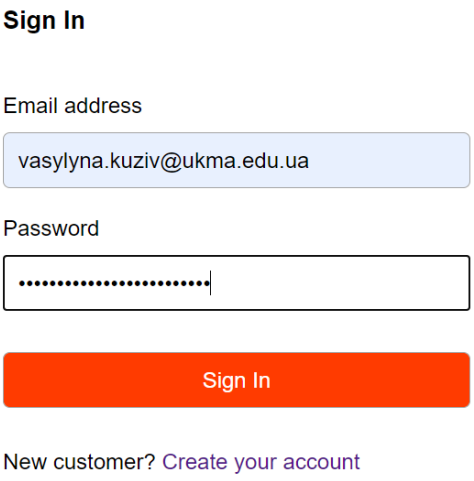


Рисунок 3.2 – Форма авторизації

При першому використанні сервісу можна зареєструватися для того, щоб мати можливість переглядати наявність вільних столиків та здійснювати бронювання. Щоб забезпечити обмежений доступ до певної інформації(перегляд деталей закладу, форма бронювання тощо) перевіряється, чи це авторизований користувач та чи це останній авторизований користувач на цьому пристрої. Це здійснюється за допомогою інформації станів Redux, збереженої у сховищі та унікального токена користувача.

```
export const signin = (email, password) => async (dispatch) => {
    dispatch({ type: USER_SIGNIN_REQUEST, payload: { email, password } });
    try {
        const { data } = await Axios.post( url: '/api/users/signIn',  data: { email, password })
        dispatch({ type: USER_SIGNIN_SUCCESS, payload: data });
        localStorage.setItem('userInfo', JSON.stringify(data));
    } catch (error) {
        dispatch({
            type: USER_SIGNIN_FAIL,
            payload:
                error.response && error.response.data.message
                    ? error.response.data.message
                    : error.message,
        });
    }
};
```

Рисунок 3.3 – Код для здійснення авторизації

Після авторизації користувач переходить на головну сторінку застосунку(HomePage), де відображаються усі заклади на платформі у вигляді карточок з зображенням, назвою, адресою, рейтингом та середньою ціною за один прийом їжу. Для цього надсилається запит на відповідну API адресу ‘api/places’ та оновлюється стан локального сховища. Після чого потрібна інформація відображається за допомогою React-компонента “Place”.

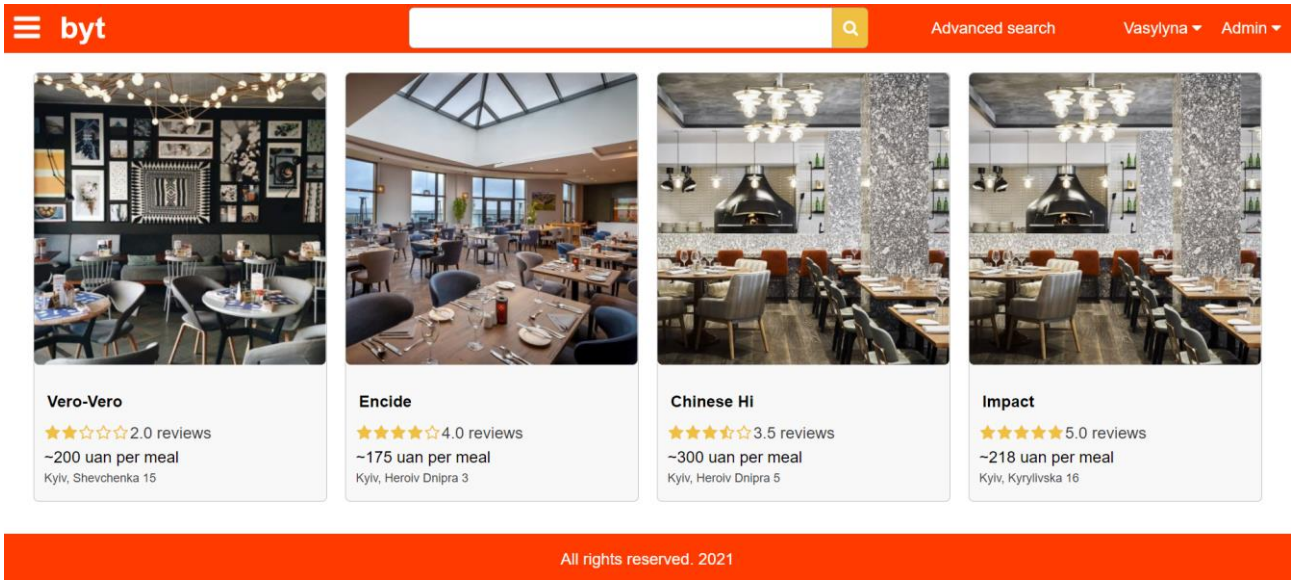


Рисунок 3.4 – Головна сторінка веб-застосунку

Аналогічним чином відбувається потік даних на сторінці кожного закладу та сторінці розширеного пошуку. Відмінність у тому, що для відображення використовуються різні сторінки та надсилаються запити на різні кінцеві точки, де на сервері здійснюються SQL-запити для отримання вибірки інформації з бази даних.

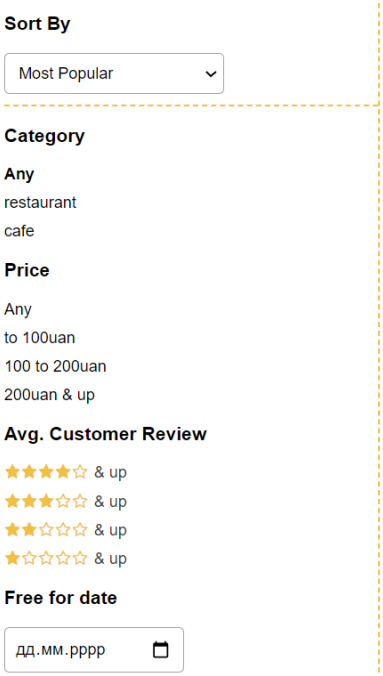


Рисунок 3.5 – Налаштування ільтрації та сортування на сторінці розширеного пошуку

Наприклад, щоб отримати дані про всі заклади на головній сторінці ми надсилаємо запит без параметрів, а щоб отримати заклади, відсортовані та відфільтровані за одним або декількома значеннями, виконується GET-запит такого вигляду:

`/api/places?name=**${name}**&category=**${category}**&min=**${min}**&max=**${max}**&rating=**${rating}**&order=**${order}**`.

Де усі параметри парсяться і використовуються у SQL-запиті.

Наразі користувач може здійснювати сортування за рейтингом, середньою ціною у порядку зростання або спадання та фільтрацію за типом закладу, середньою ціною за один прийом їжі, середнім рейтингом та доступністю на певну дату.

Користувачі мають змогу переглянути та змінити свою контактну інформацію на сторінці профілю. Для цього необхідно у полі, яке потрібно змінити, ввести нове значення та натиснути кнопку «Update». При зміні паролю необхідно ввести його попереднє значення, придумати нове та підтвердити його повторно.

Для того, щоб здійснити бронювання необхідно виконати наступну послідовність кроків.

Перейти на сторінку бажаного закладу та натиснути кнопку «Choose table», що розташована під описом закладу;

На відкритій сторінці бронювання у лівій частині необхідно обрати дату бронювання, час початку та час закінчення. На разі час доступний для бронювань не синхронізований з робочими годинами закладу.

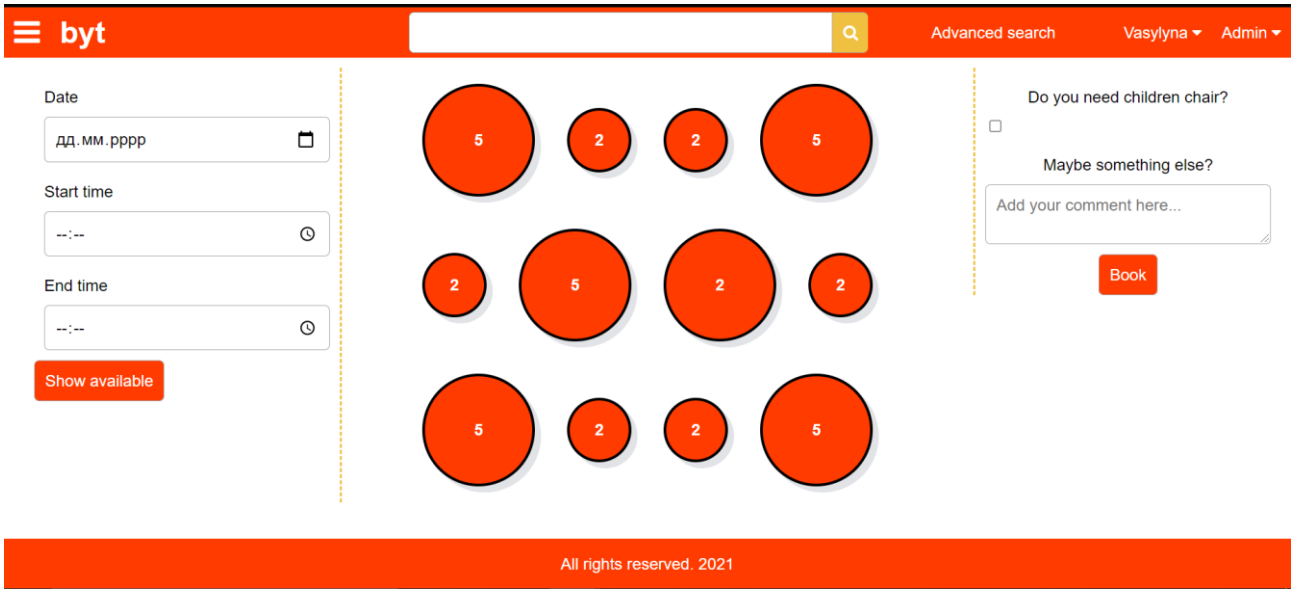


Рисунок 3.6 – Сторінка бронювання перед заповненням дати та часу

Натиснути кнопку «Show available», після цього середня частина екрану – мапа сайту зміниться у відповідності до завантаженості закладу на той період:

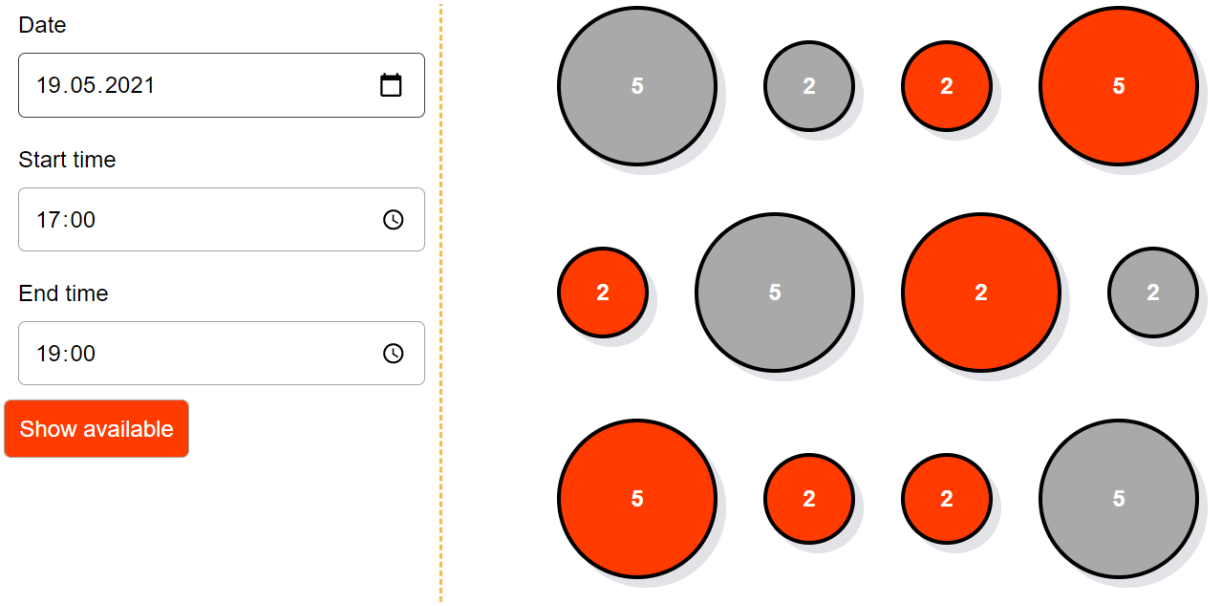


Рисунок 3.6 – Частина сторінки бронювання після вибору дати та часу

Мапа закладу створюється індивідуально для кожного закладу відповідно до розташування столиків та інших ключових елементів інтер'єру. Рівень деталізації може залежати від вподобань замовників та вартості їхньої підписки. У цій роботі створено макет одного уявного закладу для тестування функціоналу.

Після вибору вподобаного столику користувач може додати коментар та зазначити, чи потрібне дитяче крісло. Натиском на кнопку «Book» клієнт підтверджує бронювання та перенаправляється на сторінку історії бронювань, усі бронювання, включно з щойно зареєстрованим.

Для менеджера закладу реалізовано такий же функціонал з певними уточненнями: він може здійснювати та переглядати бронювання лише у межах своєї мережі закладів – для реєстрації резервацій, здійснених, наприклад, по телефону або вживу. Також адміністратор може змінювати інформацію про заклади та приховувати їх. У цій системі, якщо заклад через певні причини стає неактивним, то керівництво може не видалити його, а приховати для зберігання усієї документації та збереження цілісності даних.

ВИСНОВКИ

У цій курсовій роботі пройдено повний цикл створення програмного забезпечення: планування та аналіз, дизайн і розробка.

На першому етапі, описаному у Розділі 1, проведено дослідження предметної області, а саме здійснено огляд найкращих аналогів на світовому ринку, визначено їх переваги та недоліки для кінцевих користувачів. Для специфікації функціональних та нефункціональних вимог спочатку було проведено збір фактів, що включає у себе опитування та інтерв'ю. За результатами опитування ми визначили основні потреби 146-ти користувачів, їхні проблеми та вподобання. На основі цього було змодельовано персону та CJM, визначено основні типи користувачів. Це дало чітке уявлення про майбутній функціонал системи.

У Розділі 2 створено ER-модель, яка містить 5 основних сутностей: менеджер, клієнт, заклад, столик та бронювання. На етапі даталогічного проектування перетворено модель «сутність-зв'язок» у реляційну модель, відповідно до обмежень обраної СКБД. Для розробки архітектури застосунку було проаналізовано багато технічних праць та літературних джерел. В результаті чого, здійснено поділ на рівні та розподіл обов'язків з дотриманням основних правил «чистої» архітектури. На серверній частині працюють «роутери», які отримують запит від клієнтської сторони, звертаються до бази даних та формують відповідь у вигляді об'єкту або масиву об'єктів. Фронтенд працює таким чином: за допомогою графічного інтерфейсу, реалізованому на React, користувач здійснює певну дію, цю дію отримує функція reducer, яка оновлює стан сховища і оновлені дані відображаються на сторінці.

Дизайн графічного користувацького інтерфейсу створено за таким алгоритмом:

a. Розробка інформаційної архітектури;
b. Проектування макету застосунку у вигляді wireframe;
c. Розробка інтерактивного прототипу;

d. Перевірка ергономічності.

Опис, переваги та причини вибору технологій (Javascript, NodeJS, React, MySQL та інших) наведено у останньому розділі текстової частини. Такий стек технологій забезпечує легку масштабованість, високу надійність та простоту у реалізації веб-застосунків.

Отже, результатом курсової роботи є комплексна розробка інформаційної системи для ресторанного бізнесу та веб-застосунок з основним набором функціональностей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Eat App - офіційний сайт [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://eatapp.co/.
2. How to reach a million customers in one year [Електронний ресурс] // Wamda. – 2017. – Режим доступу до ресурсу: https://www.wamda.com/2016/03/how-to-reach-million-customers-in-one-year-eat#!.
3. Middle East based Restaurant Reservations Start-up seats its 600,000th diner [Електронний ресурс] // Prleap – Режим доступу до ресурсу: http://www.prleap.com/pr/238212/middle-east-based-restaurant-reservations-start.
4. Zaher A. Eat App Raises $5M In New Funding Ahead Of Saudi Expansion" [Електронний ресурс] / Amany Zaher // Forbes. – 2020. – Режим доступу до ресурсу: https://www.forbesmiddleeast.com/innovation/startups/eat-app-raises-%245m-in-new-funding-ahead-of-saudi-expansion/.
5. Resy - офіційний сайт [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://resy.com/?date=2021-05-08&seats=2.
6. OpenTable - офіційний сайт [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.opentable.com/.
7. Кузів В. Д. Опитування користувачів [Електронний ресурс] / Василина Дмитрівна Кузів. – 2020. – Режим доступу до ресурсу: https://docs.google.com/forms/d/e/1FAIpQLSeuy5dqC8biUIh0zYNkhnKL_Wf1g2XVVvNAif-KgvgK0mqBSQ/viewform.
8. Columbus L. "Personas Are The Future of Enterprise Apps: Lessons Learned From Scribe Software." [Електронний ресурс] / Louis Columbus // Enterprise Irregulars – Режим доступу до ресурсу: https://www.enterpriseirregulars.com/60471/personas-are-the-future-of-enterprise-apps-lessons-learned-from-scribe-software/.
9. Vandermolen F. Chapter 4: Requirements - Writing Requirements / Fulton Vandermolen // Airborne Electronic Hardware Design Assurance: A Practitioner's Guide to RTCA / Fulton Vandermolen., 2018. – (ISBN 9781351831420). – С. 89–93.
10. Loucopoulos P. Chapter 4: Requirements Engineering / Loucopoulos // Design Process Improvement: A Review of Current Practice / Loucopoulos., 2005. – С. 116–139.
11. Jump up to:a b Adams, K.M. "3.2 Definitions for Functional and Non-Functional Requirements". Non-functional Requirements in Systems Analysis and Design. Springer. С. 45–50.
12. Chen, Peter (March 1976). "The Entity-Relationship Model - Toward a Unified View of Data". ACM Transactions on Database Systems. 1 (1): 9–36.
13. A.P.G. Brown, "Modelling a Real-World System and Designing a Schema to Represent It", in Douque and Nijssen (eds.), Data Base Description, North-Holland, 1975.

14. Teorey, T.; Lightstone, S. and Nadeau, T.(2005) Database Modeling & Design: Logical Design, 4th edition, Morgan Kaufmann Press.
15. Томас Коннолли, Каролин Бегг. Базы данных. Проектирование, реализация и сопровождение. Теория и практика = Database Systems: A Practical Approach to Design, Implementation, and Management Third Edition. — 3-е изд. — М.: «Вильямс», 2003. — С. 1436.
16. Codd, E.F (1970). "A Relational Model of Data for Large Shared Data Banks". Communications of the ACM. Classics. 13 (6): 377–87.
17. Martin R. Clean code / Robert Martin // Clean code / Robert Martin., 2008. – С. 28–352.
18. Fowler M. Patterns of Enterprise Application Architecture / Martin Fowler., 2003.
19. Web Application Architecture: How the Web Works [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.altexsoft.com/blog/engineering/web-application-architecture-how-the-web-works/.
20. Redux - офіційний ресурс [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://redux.js.org/tutorials/fundamentals/part-3-state-actions-reducers
21. Інструкція з використання Redux [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.tutorialspoint.com/redux/redux_reducers.htm.
22. User Interface & User Experience Design | Oryzo | Small Business UI/UX [Електронний ресурс] // Oryzo. – 2019. – Режим доступу до ресурсу: https://oryzo.com/user-interface-design/.
23. Tidwell J. Designing Interfaces / Jenifer Tidwell., 2005.
24. Wodtke, Christina. Information Architecture - Blueprints for the Web. — 2nd. — New Riders, 2009.
25. Guilizzoni P. What Are Wireframes? [Електронний ресурс] / Peldi Guilizzoni. – 2021. – Режим доступу до ресурсу: https://balsamiq.com/learn/articles/what-are-wireframes/.
26. Кузів В. Клікабельний прототип [Електронний ресурс] / Василина Кузів. – 2021. – Режим доступу до ресурсу: https://www.figma.com/proto/YogCWBN3wOjL7kugvfk6i0/Project?scaling=scale-down&node-id=21%3A573.
27. Dats O. Why do Seed and Series A Startups Choose JavaScript for Frontend and Backend? [Електронний ресурс] / Oleg Dats. – 2021. – Режим доступу до ресурсу: https://blog.techmagic.co/why-do-seed-and-series-a-startups-choose-javascript-for-frontend-and-backend/.
28. ReactJS - офіційний ресурс [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://reactjs.org/
29. NodeJS - офіційний ресурс [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://nodejs.org/uk/.

30. Feoktistov I. Why and When to Use Node.js? [Електронний ресурс] / Ihor Feoktistov – Режим доступу до ресурсу: https://relevant.software/blog/why-and-when-to-use-node-js/.
31. MySQL - офіційний ресурс [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.mysql.com/.
32. Brosh A. How to Choose the Right Database [Електронний ресурс] / Anvy Brosh. – 2019. – Режим доступу до ресурсу: https://towardsdatascience.com/how-to-choose-the-right-database-afcf95541741.
33. Top Reasons to Use MySQL [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: http://www.databasequest.com/index.php/product-service/mysql-dbquest#:~:text=One%20of%20the%20reasons%20MySQL,for%20every%20application%20development%20need.&text=MySQL%20also%20provides%20connectors%20and,a%20preferred%20data%20management%20server .

ДОДАТКИ

Додаток А

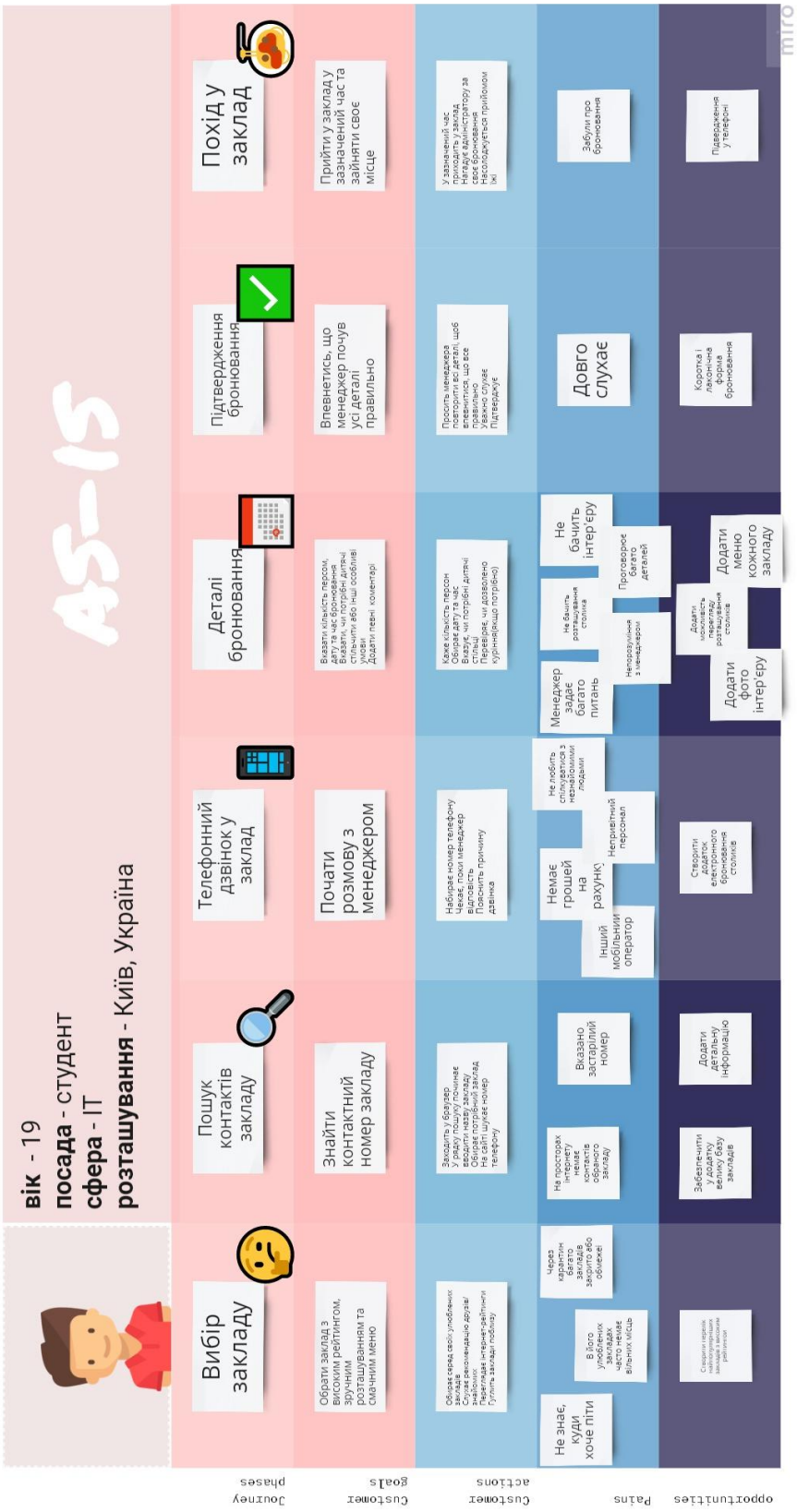
Скрипт інтерв’ю

Скрипт інтерв’ю

Вступ	Представити себе
	Пояснити мету інтерв'ю, як використовуватимуться дані та запевнити конфіденційність
Загальна інформація про відвідування закладів	Як часто ви відвідуєте заклади харчування?
	Яка середня вартість вашого чеку за прийом їжі(сніданок, обід, вечеря)?
	Як ви зазвичай обираєте заклад?
	Як часто бувало, що ви приходили заклад, а там не було місць? Що ви робите у такій ситуації?
	Чи користуєтесь ви додатками для відстежування рейтингів або перегляду меню закладів? Якими саме?
Досвід бронювання	Чи бронюєте(купуєте) ви квитки (кінотеатр, потяг тощо) завчасно?
	Розкажіть про свій останній досвід бронювання столика.
	Яким способом бронювання ви користуєтесь найчастіше? Чому?
	Чому ви не бронюєте столики частіше?
Визначення особливостей майбутнього продукту	Що для Вас є важливим при бронюванні столика?
	Чи вважаєте ви нормальним вносити певний завдаток при бронюванні, який потім вираховуватиметься з суми замовлення?
	Поділіться переліком ваших улюблених закладів.
	Які ви знаєте заклади, де майже завжди "повна посадка"?
Завершення	Можливо ви хочете поділитися з нами ще якоюсь історією, пов'язаною з закладами харчування та бронюванням столиків?

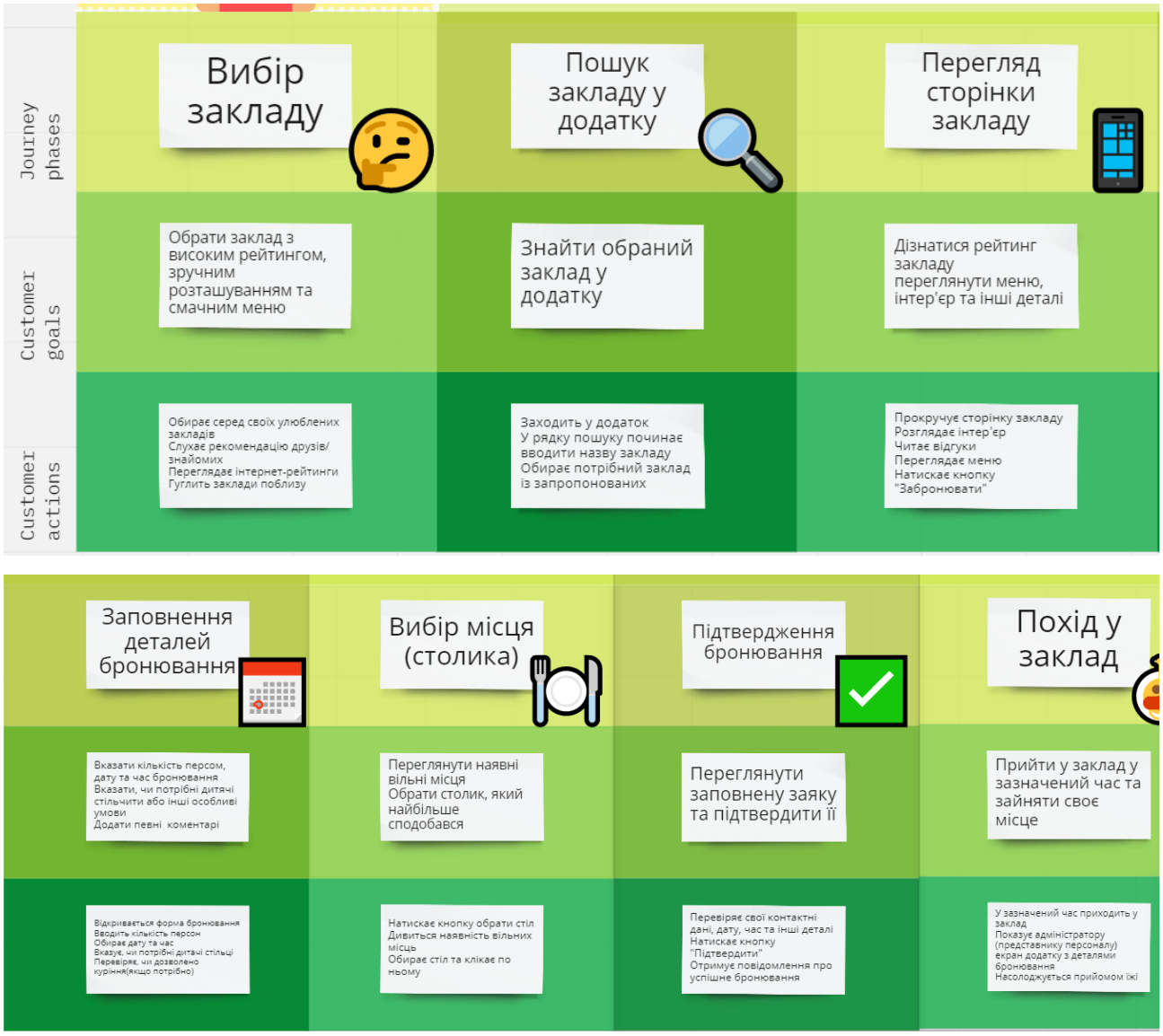
Додаток В

CJM – теперішній процес



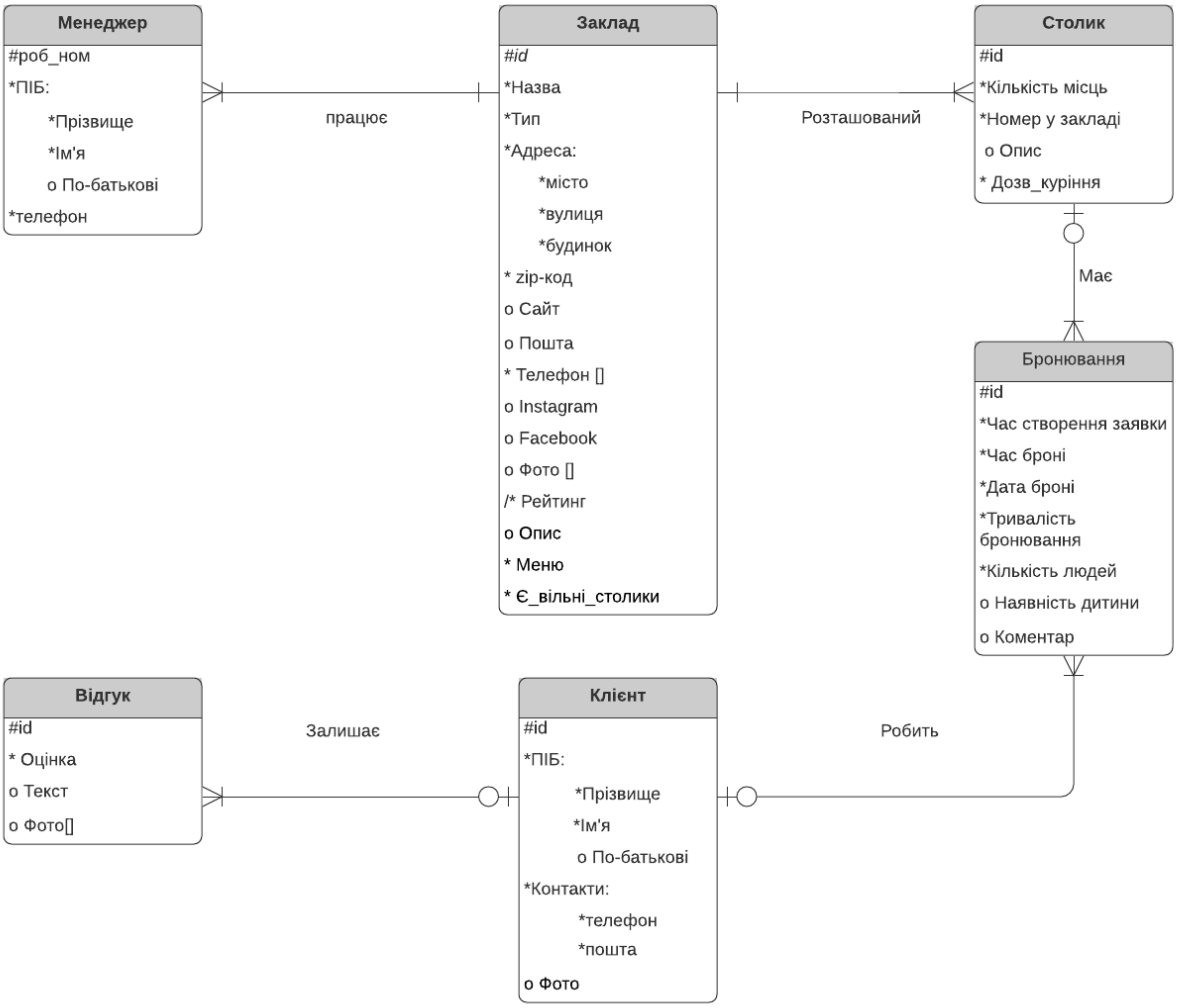
Додаток С

CJM – майбутній процес



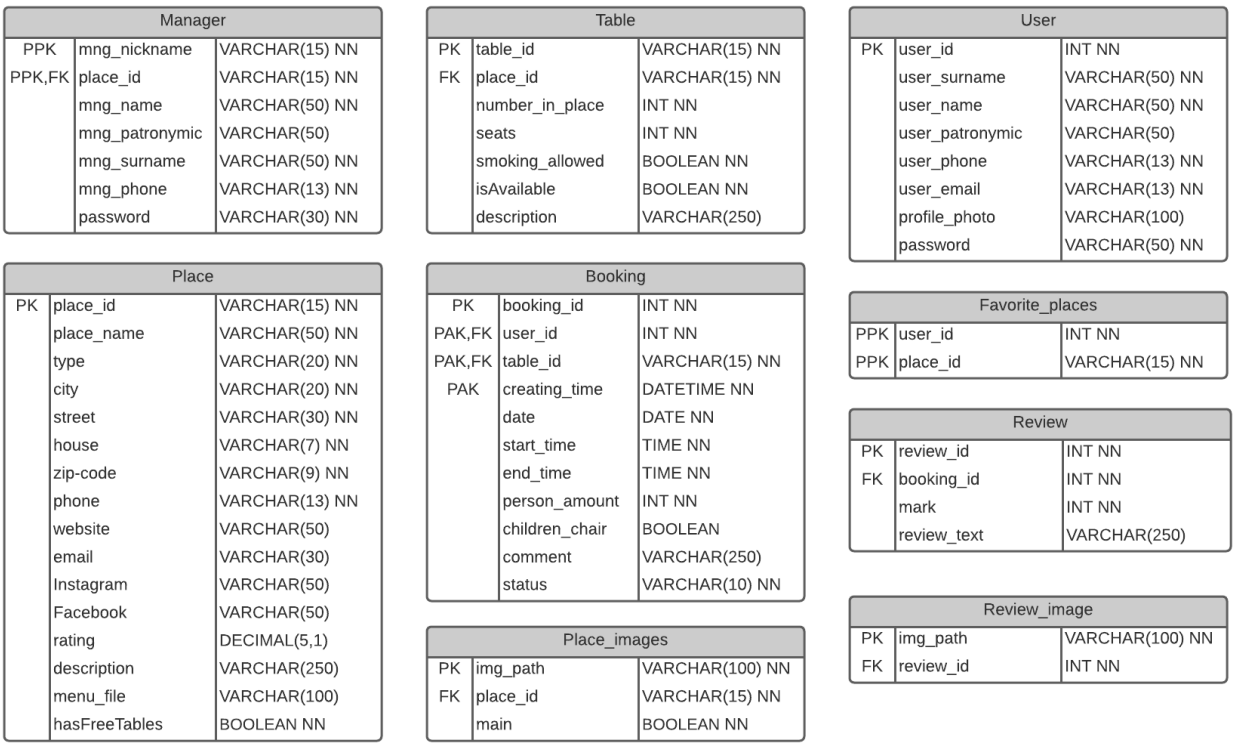
Додаток D

ER-модель



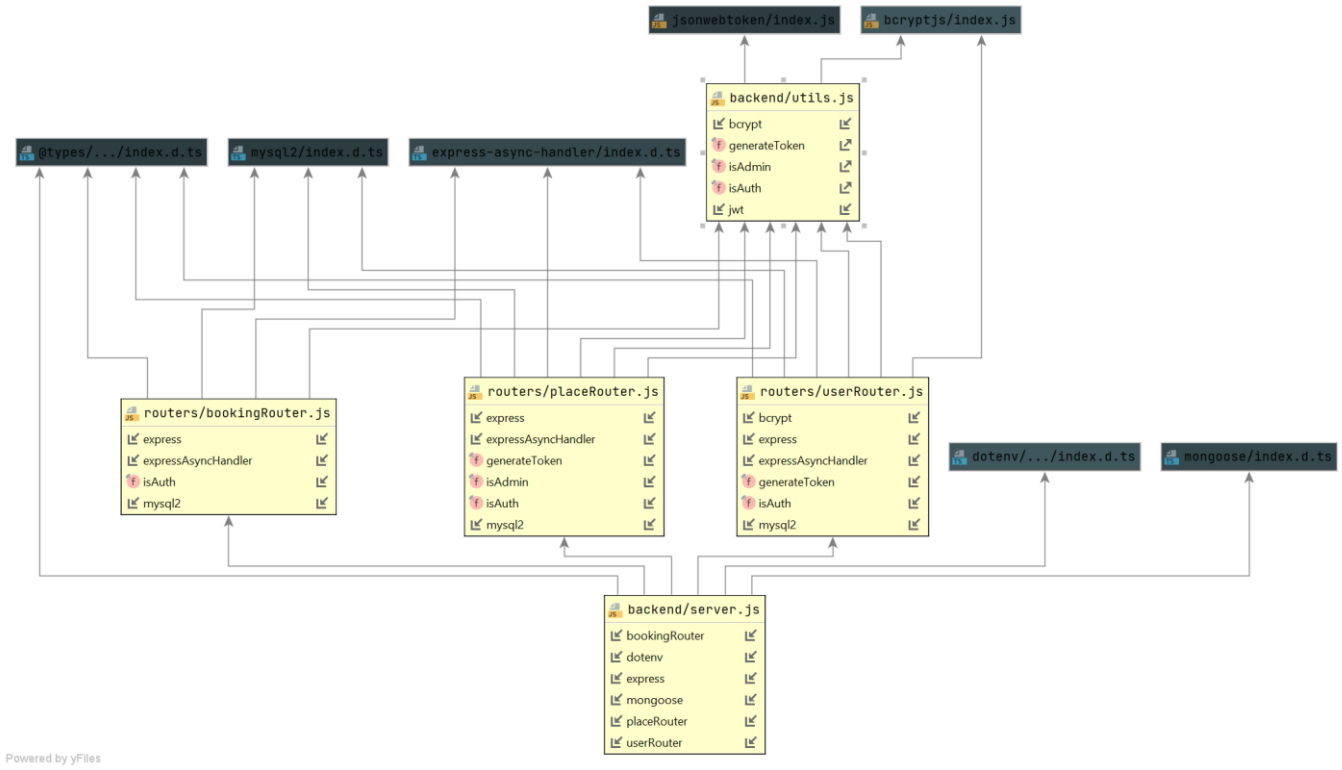
Додаток Е

Частина реляційної моделі для забезпечення основних функціональних вимог



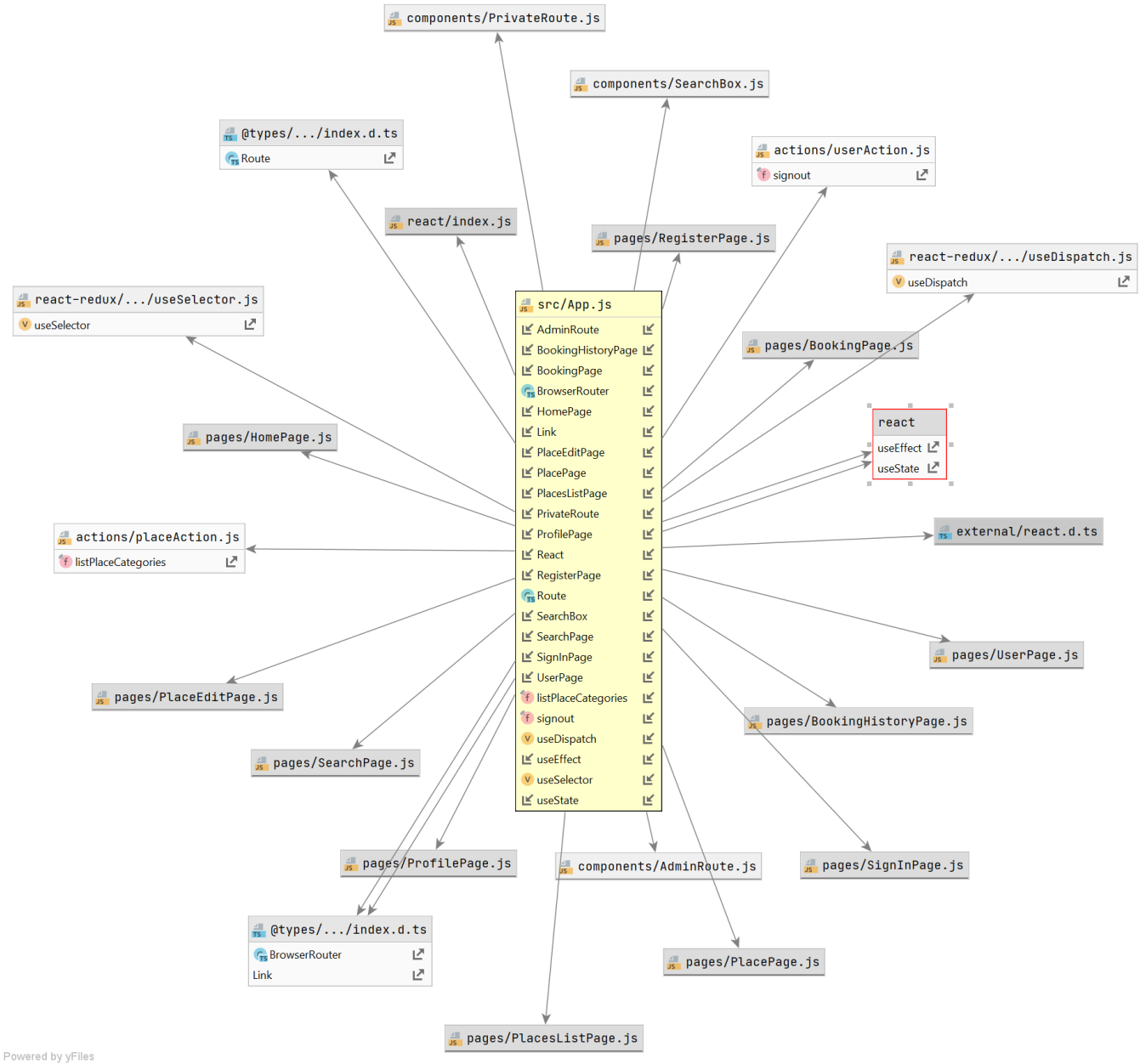
Додаток F

Архітектура серверної частини



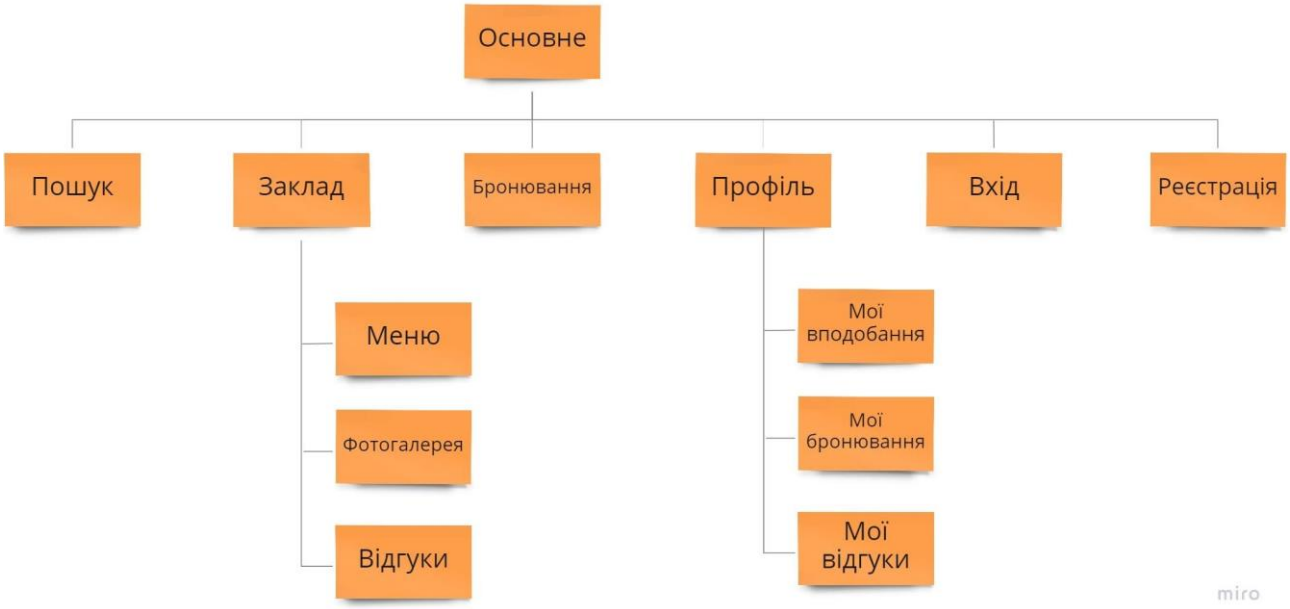
Додаток G

Діаграма зв’язків App.js та інших елементів



Додаток Н

Інформаційна архітектура. Мапа сайту



Додаток І

Основні сторінки прототипу мобільного застосунку