НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Вебзастосунок для підтримки діяльності сервісних центрів»

Виконала:
студентка IV курсу, групи КП-93
Коротюк Марія Ігорівна _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Заболотня Тетяна Миколаївна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри ММСА ІПСА, к.т.н., доцент,
Дідковська Марина Віталіївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___»_____2022 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Коротюк Марії Ігорівні

1. Тема проєкту «Вебзастосунок для підтримки діяльності сервісних центрів», керівник проєкту Заболотня Тетяна Миколаївна, к.т.н., доцент, затверджені наказом по університету від «31» травня 2023 р. № 2107-с

2. Термін подання студентом проєкту «16» червня 2023 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - огляд існуючих рішень;
   - вибір технологій для розробки вебзастосунку;
   - структурно-алгоритмічна організація застосунку;
   - особливості реалізації програмного забезпечення.

5. Перелік обов'язкового графічного матеріалу:
   - загальна структура застосунку (креслення);
   - схема бази даних(креслення);
   - діаграма варіантів використання (плакат);
   - діаграма потоку даних (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2022 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	13.11.2022	
2.	Розроблення та узгодження технічного завдання	25.11.2022	
3.	Розроблення структури вебзастосунку	15.12.2022	
4.	Підготовка матеріалів першого розділу дипломного проєкту	30.12.2022	
5.	Розроблення дизайну сторінок та графічних елементів	03.02.2023	
6.	Підготовка матеріалів другого розділу дипломного проєкту	19.02.2023	
7.	Програмна реалізація вебзастосунку	10.03.2023	
8.	Тестування вебзастосунку	17.03.2023	
9.	Підготовка матеріалів третього розділу дипломного проєкту	30.03.2023	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	12.04.2023	
11.	Підготовка графічної частини дипломного проєкту	21.04.2023	
12.	Оформлення документації дипломного проєкту	26.05.2023	

Студент Марія КОРОТЮК

Керівник проєкту Тетяна ЗАБОЛОТНЯ

АНОТАЦІЯ

Даний дипломний проєкт присвячений розробці вебзастосунку для підтримки діяльності сервісних центрів. Вебзастосунок має на меті допомогти клієнтам, адміністраторам та працівникам у організації процесу вибору потрібного сервісного центру, оформлення та контролю виконання замовлених послуг.

У проєкті було здійснено огляд наявних рішень для сервісних центрів, визначено переваги і недоліки кожного з них, обґрунтовано вибір засобів розробки програмного забезпечення, визначено функціональні та нефункціональні вимоги, структуру вебзастосунку та структури даних.

При виконанні даного проєкту було розроблено систему, що дозволяє клієнтам здійснювати пошук та вибір сервісного центру, надсилати заяви на отримання послуг, перевіряти етап виконання замовлення та залишати відгуки про якість обслуговування. Для адміністраторів вебзастосунок надає можливість додати сторінку власного сервісного центру, вести облік заяв клієнтів, створювати на їх основі замовлення. Для замовлень адміністратор може створювати завдання, які будуть призначенні для виконання певним працівником.

ABSTRACT

This thesis is devoted to the development of a web application for support of the service centers' activities. The web application has an aim to help either customers, administrators and employees in organizing process of selecting necessary service center, arrangement and monitoring ordered services execution.

In the thesis the available solutions for service centers were reviewed, the advantages and disadvantages of each of them were determined, the choice of software development tools was justified, functional and non-functional requirements and the structure of the web application and data structures were defined.

During the execution of this thesis, a system was developed that allows customers to search and select service center, send applications for services, check the order fulfillment stage and add response about service quality. For administrators, the web application provides an opportunity to add a page of their own service center, keep track of customers` applications, and create orders based on them. For orders, the administrator can create a tasks that will be issued to the certain employee.

ДП.045440-01-90 Вебзастосунок для підтримки діяльності сервісних центрів.
Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Вебзастосунок для	5	
	підтримки діяльності		
	сервісних центрів.		
	Технічне завдання		
ДП.045440-03-81	Вебзастосунок для	67	
	підтримки діяльності		
	сервісних центрів.		
	Пояснювальна записка		
ДП.045440-04-51	Вебзастосунок для	4	
	підтримки діяльності		
	сервісних центрів.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Вебзастосунок для	9	
	підтримки діяльності		
	сервісних центрів.		
	Керівництво користувача		
ДП.045440-06-99	Вебзастосунок для	1	
	підтримки діяльності		
	сервісних центрів.		
	Загальна структура		
	застосунку.		
	UML-діаграма		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-07-99	Вебзастосунок для	1	
	підтримки діяльності		
	сервісних центрів. Схема		
	бази даних. ER-діаграма		
ДП.045440-08-98	Вебзастосунок для	1	
	підтримки діяльності		
	сервісних центрів.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2022 р.

ВЕБЗАСТОСУНОК ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ СЕРВІСНИХ ЦЕНТРІВ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Марія КОРОТЮК

2022

ЗМІСТ

1. Найменування та галузь застосування.....3
2. Підстава для розроблення.....3
3. Призначення розробки.....3
4. Вимоги до програмного продукту.....3
5. Вимоги до проєктної документації.....4
6. Етапи проєктування.....5
7. Порядок тестування розробки.....5

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Вебзастосунок для підтримки діяльності сервісних центрів.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання клієнтами для вибору сервісних центрів та працівниками для організації роботи сервісних центрів.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Вебзастосунок повинен забезпечувати такі основні функції:

1) перегляд списку сервісних центрів та сторінки кожного з них;

2) реєстрація та вхід в обліковий запис;

4) перегляд та редагування профілю користувача;

5) створення, редагування та видалення сервісних центрів;

6) керування заявами, замовленнями, завданнями;

7) керування працівниками сервісних центрів.

Розробку виконати на платформі Node.js з використанням технології Express.js та React.

Додаткові вимоги:

1) фільтри та сортування для списків сервісних центрів, заяв, замовлень та завдань;
2) пошук заяв та замовлень за їх номером;
3) перегляд історії заяв та замовлень;
4) додавання відгуків для сервісного центру.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   - «Загальна структура застосунку. UML-діаграма»;
   - «Схема бази даних. ER-діаграма».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи.....16.11.2022

Розроблення та узгодження технічного завдання.....27.11.2022

Розроблення структури вебзастосунку.....05.12.2022

Розроблення дизайну сторінок та графічних елементів.....03.02.2023

Програмна реалізація вебзастосунку.....15.03.2023

Тестування вебзастосунку.....07.04.2023

Підготовка матеріалів текстової частини проєкту.....28.04.2023

Підготовка матеріалів графічної частини проєкту.....12.05.2023

Оформлення технічної документації проєкту.....25.05.2023

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Євгенія СУЛЕМА

“___” _____ 2023 р.

ВЕБЗАСТОСУНОК ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ СЕРВІСНИХ ЦЕНТРІВ

Пояснювальна записка

ДП.045440-03-81

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Марія КОРОТЮК

2023

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ.....3
ВСТУП.....4
1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ.....6
1.1. Огляд категорій вебзастосунів для сервісних центрів.....6
1.2. Аналіз існуючих рішень.....7
1.3. Розширена постановка завдання.....14
1.4. Висновки до першого розділу.....15
2. ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБЗАСТОСУНКУ.....17
2.1. Технології для розробки серверної частини.....17
2.2. Технології для розробки клієнтської частини.....19
2.3. База даних.....24
2.4. Висновки до другого розділу.....25
3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ ЗАСТОСУНКУ.....27
3.1. Опис вимог до розроблюваного застосунку.....27
3.2. Опис структурної організації застосунку.....36
3.3. Опис структур даних застосунку.....38
3.4. Алгоритмічне забезпечення застосунку.....45
3.5. Висновки до третього розділу.....46
4. ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....48
4.1. Реалізація серверної частини.....48
4.2. Реалізація клієнтської частини.....49
4.3. Тестування програмного забезпечення.....58
4.4. Напрямки подальшої реалізації.....62
4.5. Висновки до четвертого розділу.....63
ВИСНОВКИ.....64
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ.....65
ДОДАТКИ.....67

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

CSS (Cascading Style Sheets) – мова стилю вебсторінок, яка використовується для опису зовнішнього вигляду HTML елементів.

HTML (Hypertext Markup Language) – мова розмітки вебдокументів для відображення їх у браузері у вигляді вебсторінок.

HTTP (HyperText Transfer Protocol) – протокол прикладного рівня для передачі даних, а саме вебсторінок та інших типів файлів.

SQL (Structured Query Language) – декларативна мова програмування для збереження та обробки інформації в реляційних базах даних.

Авторизація (Authorization) – дозвіл на виконання певних дій користувачем в системі.

Запит (Request) – формулювання потреби отримання інформації з бази даних та інформаційних систем.

СУБД (Система управління базами даних) – набір даних та інструментів для управління ними.

ВСТУП

Кожен хоча б раз користувався послугами сервісних центрів, наприклад для ремонту комп'ютера, побутової техніки, автомобіля, заміни певної деталі чи перевірки та усунення несправності пристрою. При появі певних проблем з технікою чи обладнанням відразу виникає питання, до кого ж звертатися і як обрати сервісний центр, який найкраще і найшвидше допоможе їх вирішити.

Сервісний центр – це організація, призначена для надання послуг клієнтам з ремонту та обслуговування техніки та обладнання. Для того, щоб поширити інформацію про свій сервісний центр, а також за потреби оптимізувати організаційну роботу, власники замовляють розробку спеціалізованого програмного забезпечення, або використовують вже наявне, обираючи те, яке хоча б частково підходить під їх запити.

В інтернеті існує чимало вебсайтів, які пропонують перелік сервісних центрів та містять базову інформацію про них, як от розташування та опис послуг, які вони надають. Також багато сервісних центрів мають власне програмне забезпечення, яке дозволяє розмістити значно більше інформації та додати деякі функції, зокрема перевірку етапу ремонту та надсилання форми для отримання зворотнього зв'язку.

Проте в той же час жоден з вебзастосунків, не може допомогти вирішити одночасно завдання вибору сервісного центру для клієнтів з можливістю замовлення послуг онлайн, а також для власників сервісних центрів створення сторінки організації, керування замовленнями клієнтів та розподілом їх між працівниками, не витрачаючи додаткових ресурсів на створення власного вебсайту та його підтримки. Це є незручним для всіх груп користувачів, адже в наявних рішеннях неможливо зберігати усі потрібні для клієнтів та працівників дані, пов'язані з діяльністю сервісних центрів, в межах однієї системи, і саме тому розроблення такого вебзастосунка є актуальним завданням.

Вебзастосунок для підтримки діяльності сервісних центрів призначений допомогти клієнтам знайти та обрати потрібний сервісний центр, залишити відгук та переглянути відгуки інших користувачів. А також забезпечити взаємодію між клієнтами та працівниками сервісних центрів для того, щоб полегшити комунікацію та оптимізувати рутинні завдання як от отримання заяв та керування ними.

Отже, даний дипломний проєкт присвячено розробці програмного забезпечення, яке має на меті допомогти як працівникам, так і клієнтам сервісних центрів у здійсненні вибору та організації процесу надання послуг.

1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1. Огляд категорій вебзастосунів для сервісних центрів

Організація роботи сервісних центрів передбачає виконання багатьох процесів, які можуть бути автоматизованими, наприклад ведення обліку замовлених клієнтами послуг та контролю етапу їх готовності. У свою чергу для клієнтів важливо мати зручний інструмент для обрання потрібного центру та оформлення замовлень.

Було проведено огляд існуючих рішень і визначено, що ситуація характеризується таким чином: вебзастосунки, які на даний момент використовуються сервісними центрами, можна поділити на три категорії.

Перша – це каталоги закладів та організацій, які дозволяють здійснювати користувачу пошук, застосовуючи різні фільтри, та перегляд базової інформації про кожен з них. На таких платформах можна знайти дані як про магазини, студії танців чи фітнесу, гуртки, кафе, різноманітні курси, так і про сервісні центри. Часто пошук здійснюється відповідно до обраного міста.

До другої категорії належать сервіси для розміщення оголошень, як от купівлі-продажу та надання різноманітних послуг. Як і для першої категорії, до пошукових запитів можуть додаватись фільтри, що дозволяє швидше знайти потрібне оголошення. На таких ресурсах можна також знайти інформацію про послуги сервісних центрів.

До третьої категорії належить програмне забезпечення, створене безпосередньо на замовлення певного сервісного центру. Такі ресурси можуть містити інформацію і набір функцій, які стосуються саме цієї організації. Це може бути як просто вебсайт з переліком головної інформації про спектр послуг, який надається, вартості робіт та контактних даних, так і вебзастосунок з можливістю відстеження клієнтами етапу виконання замовлення, оформлення доставки, заповнення різних форм.

1.2. Аналіз існуючих рішень

Розглянемо переваги та недоліки існуючих рішень для підтримки діяльності сервісних центрів.

1.2.1. *list.in.ua*

Прикладом вебзастосунків, які належать до першої категорії є вебзастосунок list.in.ua, який є каталогом закладів у вибраному місті, тому належить до першої категорії розглянутого програмного забезпечення, за допомогою якого користувач може знайти потрібний заклад, а також переглянути сторінку з детальною інформацією про нього [1].

Щодо вибору сервісних центрів вебзастосунок дозволяє:

- шукати сервісні центри з урахуванням виду обладнання, яке потребує ремонту;
- переглядати списку знайдених сервісних центрів;
- переглядати карти з місцем розташування закладу;
- залишити відгук про якість обслуговування;
- створити та переглядати сторінки сервісного центру з описом, контактними даними та фотографіями.

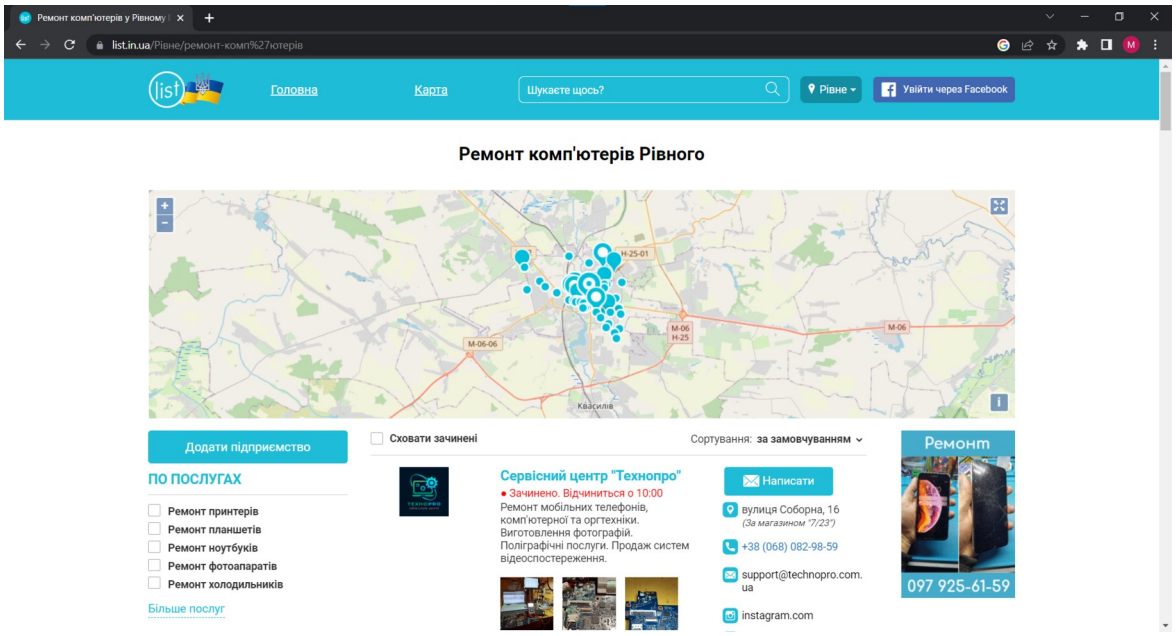


Рис. 1.1. Скріншот списку сервісних центрів у вебзастосунку list.in.ua

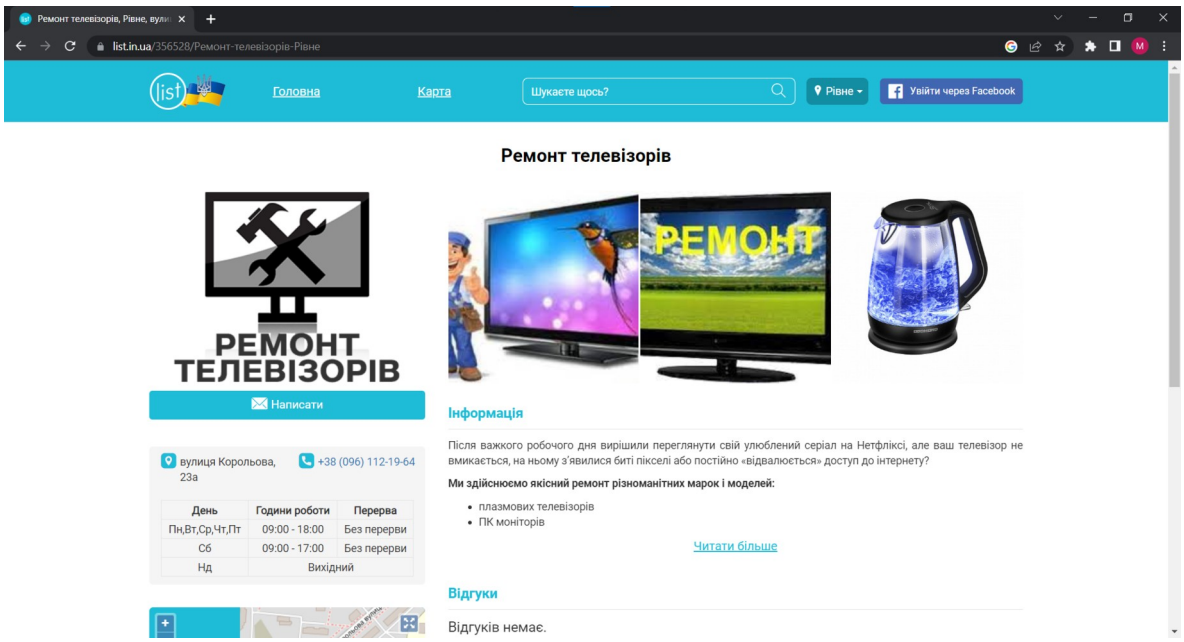


Рис. 1.2. Скріншот сторінки сервісного центру у вебзастосунку list.in.ua

Названі функції – це те, що робить вебзастосунок зручним у використанні. Проте зазначений вебзастосунок має ряд недоліків, якщо їх розглядати в контексті вибору сервісних центрів:

- велика кількість сфер закладів на платформі – немає чіткої спеціалізації;
- відсутність можливості перевіряти етап виконання замовлень.

1.2.2. *OLX.ua*

OLX.ua – один із найпопулярніших в Україні ресурсів для розміщення оголошень, що відповідно належить до другої категорії розглянутого програмного забезпечення. Його використовують для купівлі-продажу товарів та замовлення послуг. Зокрема на сайті можна знайти чимало оголошень про надання сервісними центрами послуг з ремонту та обслуговування [2].

Перевагами використання даного вебзастосунку є:

- сучасний та адаптивний дизайн;
- простота розміщення інформації про товар чи послугу;
- наявність великої кількості фільтрів та сортування отриманих

результатів, що значно пришвидшує та спрощує пошук потрібного оголошення;

- можливість оцінити взаємодію з автором оголошення;
- додавання оголошень в обране;
- можливість зв’язатись з автором повідомлення.

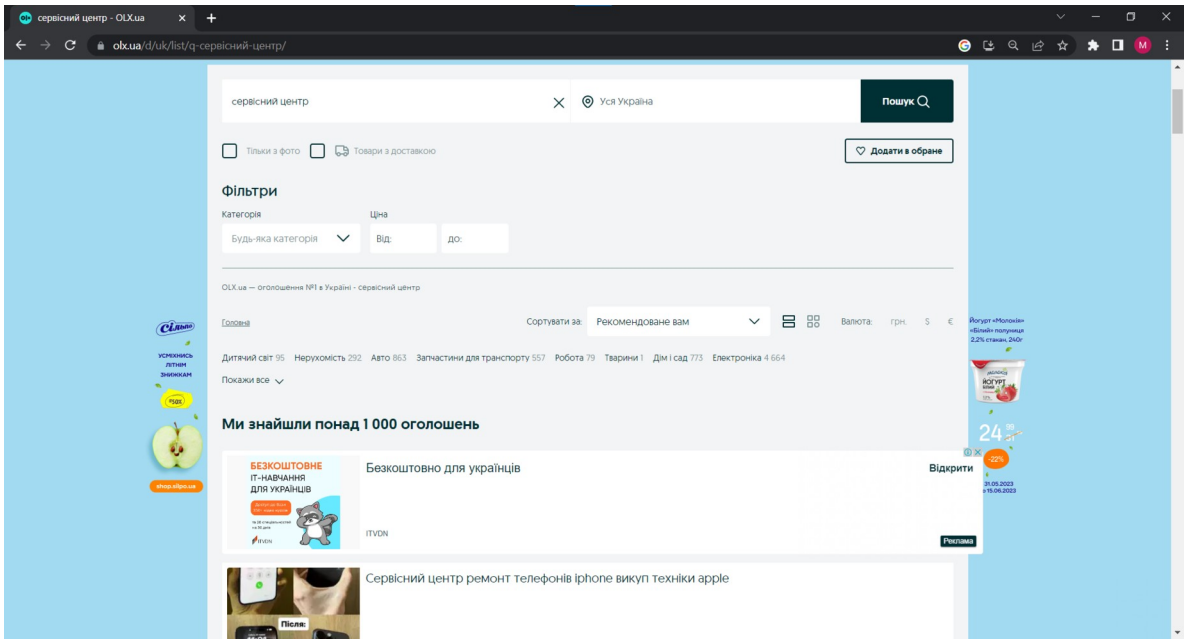


Рис. 1.3. Скріншот фільтрів у вебзастосунку OLX при пошуку сервісних центрів

Головним недоліком даного ресурсу, якщо розглядати його саме в контексті пошуку і вибору послуг сервісних центрів, є відсутність спеціалізації на певному напрямку послуг, оголошення про які розміщуються у вебзастосунку. Також можна назвати і такі недоліки:

- реклама на сайті;
- відсутність форм для замовлення послуг;
- неможливість керування залишеними клієнтами заявками.

1.2.3. *Вебсайт сервісного центру “Технарі”*

Також було розглянуто вебсайт сервісного центру “Технарі”, який належить до третьої категорії програмного забезпечення [3]. Серед

основних переваг можна виділити:

- наявність великої кількості корисної для клієнтів інформації щодо можливих послуг, яка зручно розміщена на вебсторінках;
- наявність форми для зв’язку з сервісним центром щодо замовлення ремонту та обслуговування техніки;
- можливість відслідковувати етап виконання замовлення;
- можливість залишити відгук;
- наявність карти із зазначеним місцем розташування відділень сервісного центру.

Рис. 1.4. Скріншот вебсайту сервісного центру “Технарі”

Проте цей вебсайт є програмним рішенням лише для одного сервісного центру, що унеможливлює порівняння користувачем декількох варіантів закладів в межах єдиного вебзастосунку.

1.2.4. *Сервісні Центри Bosch*

Сервісні Центри Bosch – вебзастосунок, призначений для оптимізації процесів обслуговування техніки виробника Bosch, а отже належить до

третьої категорії [4].

Перевагами використання вебзастосунку є:

- наявність карти для пошуку сервісного центру;
- перегляд детальної інформації про обраний сервісний центр;
- можливість перевірки гарантії приладу;
- можливість заповнення форми зворотнього зв’язку для того, щоб задати запитання.

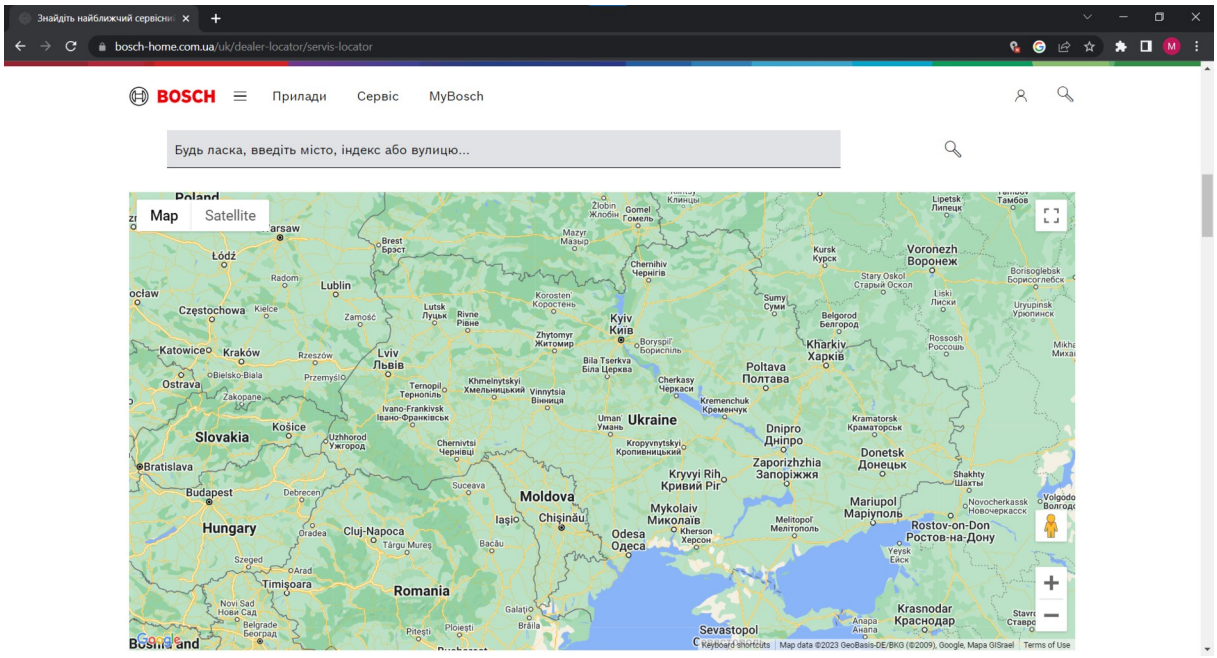


Рис. 1.5. Скріншот карти для пошуку сервісних центрів Bosch

Розглянутий вебзастосунок має і ряд недоліків:

- немає можливості залишити відгук;
- відсутня можливість перевірки стану ремонту;
- неможливо залишити заявку на замовлення послуг;
- можливо використувавти лише для сервісних центрів Bosch.

1.2.5. *Сервісні Центри Samsung*

Сервісні Центри Samsung – вебзастосунок, який є частиною платформи samsung.com, призначений для допомоги користувачам техніки

Samsung з вибором сервісного центру та отримання послуг. Зазначений вебзастосунок можна віднести до третьої категорії, адже він передбачає обслуговування лише техніки та електроніки Samsung [5].

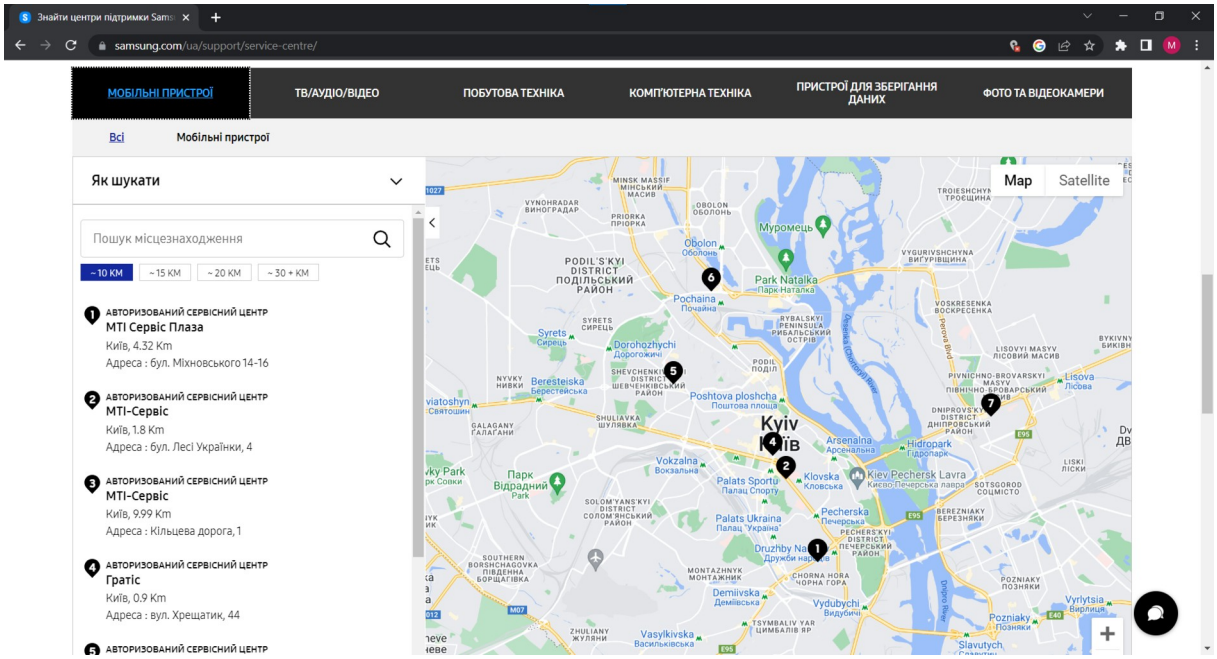


Рис. 1.6. Скріншот списку сервісних центрів з картою у вебзастосунку сервісних центрів Samsung

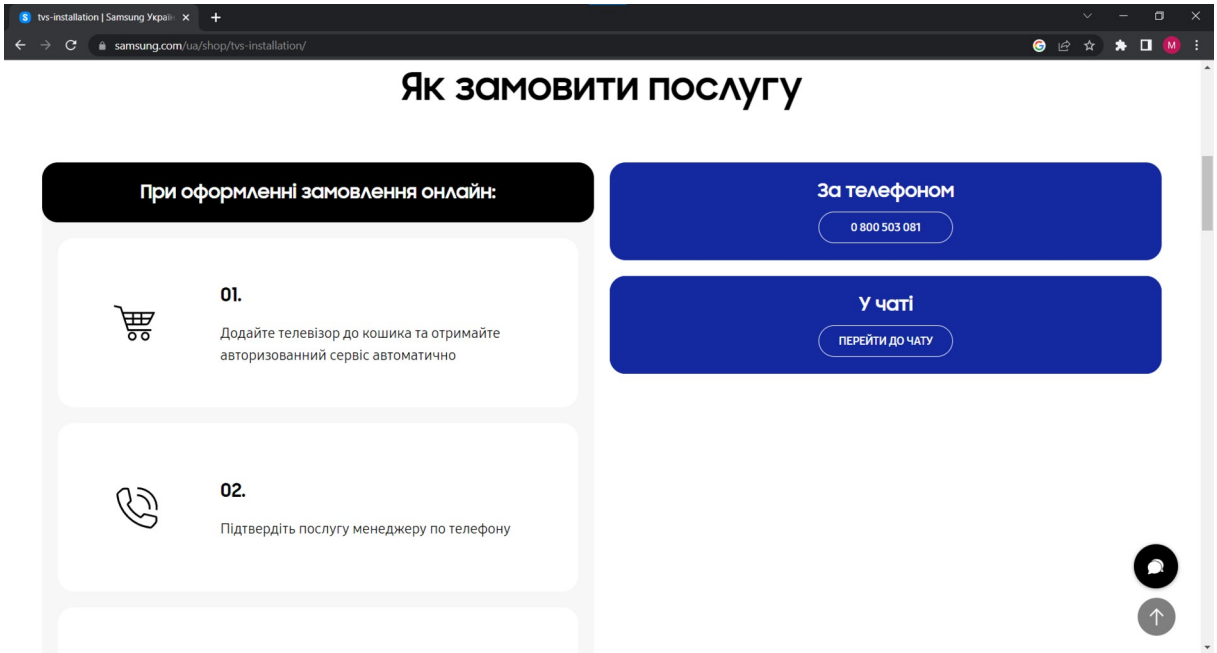


Рис. 1.7. Скріншот замовлення послуг у вебзастосунку сервісних центрів Samsung

Серед переваг вебзастосунку можна назвати:

- перегляд списку авторизованих сервісних центрів відповідно до обраної категорії та, за необхідності, потрібного типу пристрою;
- наявність карти з місцерозташуванням сервісних центрів;
- перегляд детальної інформації про обраний сервісний центр;
- можливість замовити послуги у чаті та за номером телефону;
- можливість заповнити форму для перевірки етапу ремонту;
- можливість забронювати візит до сервісного центру.

Головним недоліком даного рішення є неможливість використання власниками різних сервісних центрів.

1.2.6. *Вебзастосунок для сервісного центру "MUK Service"*

Вебзастосунок для сервісного центру "MUK Service", який належить до третьої категорії та пропонує ряд функцій, що оптимізує процеси діяльності даної організації [6].

Рис. 1.8. Скріншот форми для оформлення доставки у вебзастосунку для сервісного центру "MUK Service"

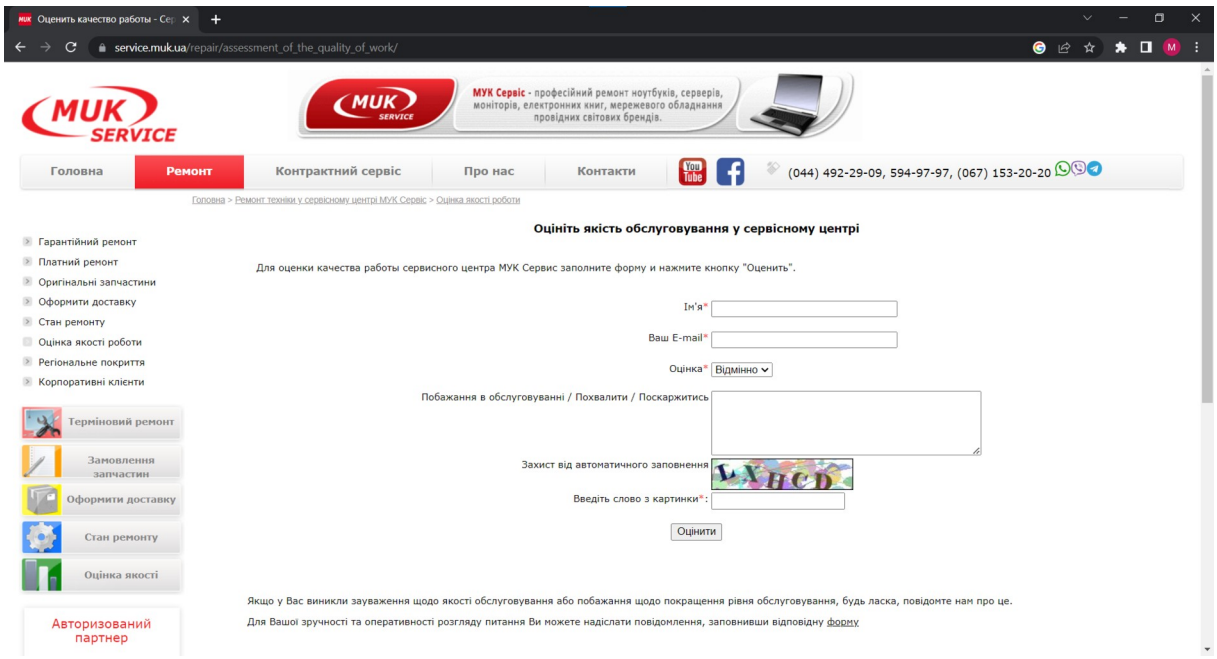


Рис. 1.9. Скріншот форми для оцінки якості обслуговування у вебзастосунку для сервісного центру “MUK Service”

Перевагами вебзастосунку є:

- можливість заповнення форми для оформлення доставки;
- можливість перевірки стану ремонту;
- можливість заповнення форми для замовлення запчастин;
- оцінка якості роботи.

Проте вебзастосунок має і деякі недоліки:

- можливість використання лише одним сервісним центром;
- відсутність можливості прочитати відгуки інших користувачів.

1.3. Розширена постановка завдання

Здійснивши аналіз існуючих рішень можна зробити висновок, що кожен розглянутий вебзастосунок має певні свої особливості, що вирізняє його серед інших та надає користувачам зручні інструменти для отримання послуг у сервісних центрах.

Можемо сформулювати вимоги до програмного забезпечення, які дозволять зробити пошук та вибір потрібного сервісного центру більш

зручним, швидким, а також оптимізувати процес комунікації та відстеження процесу виконання робіт.

Розроблений вебзастосунок допомагає у діяльності сервісних центрів як для клієнтів, так і працівників. Зокрема для клієнтів передбачені наступні можливості:

- реєстрація;
- перегляд списку сервісних центрів;
- перегляд інформації про обраний сервісний центр;
- створення заяви на отримання послуг;
- перевірка етапу виконання замовлення;
- перегляд історії власних заяв та замовлень;
- додавання відгуку.

Працівникам сервісного центру вебзастосунок дозволить використовувати такі функції:

- реєстрація;
- створення та редагування сторінки сервісного центру;
- перегляд заяв клієнтів;
- створення та редагування замовлення відповідно до заяви клієнта;
- створення завдань для замовлень;
- перегляд списку замовлень з можливістю сортування та застосування фільтрів.

1.4. Висновки до першого розділу

Було проаналізовано існуючі рішення для підтримки діяльності сервісних центрів. На основі проведених досліджень було систематизовано наявне програмне забезпечення та поділено його на три категорії.

До першої належать каталоги закладів, пошук яких зазвичай здійснюється відповідно до обраних категорії та міста (informisto.ua).

Друга категорія включає в себе сервіси для розміщення оголошень,

на яких також можна знайти послуги сервісних центрів (OLX.ua).

Третя категорія – це вебсайти сервісних центрів, які мають переважно інформаційну мету для детального ознайомлення клієнта із рядом послуг, які надаються, хоча вони також можуть містити і певні функції, наприклад для перевірки етапу ремонту (сервісні центри Samsung, Bosch, “Технарі” та “MUK Service”).

Для кожного розглянутого аналогу було виділено переваги та недоліки та після їх уважного вивчення сформульовано розширену постановку завдання.

2. ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБЗАСТОСУНКУ

2.1. Технології для розробки серверної частини

Для написання серверної частини було використано мову програмування JavaScript, а саме платформу Node.js. Для розробки API вебзастосунку було обрано фреймворк Express.js.

2.1.1. *JavaScript та C#*

Мови програмування JavaScript та C# досить часто використовуються розробниками для написання серверів вебзастосунків, але вони відрізняються одна від одної та призначенні для різних цілей.

JavaScript – прототипна, об’єктно-орієнтована мова програмування, яка має елементи функціонального програмування [7].

JavaScript виконується у браузері, адже вона є мовою скриптів, але при використанні платформи Node.js може виконуватись і на сервері. В основі JavaScript лежить реалізація стандарту ECMAScript. JavaScript є інтерпретованою мовою програмування, яка не має чіткої типізації.

Головним призначенням JavaScript було написання клієнтської частини вебзастосунків, а саме створення вебсторінок, з якими може взаємодіяти користувач, проте зараз розробники все частіше надають їй перевагу і при написанні серверів завдяки можливості використання Node.js.

Node.js – платформа з відкритим кодом, яка використовує рушій V8 для перетворення JavaScript коду у машинний код [8]. Node.js допомагає розробити високо-масштабовану серверну частину вебзастосунку на основі асинхронного та керованого подіями програмування.

Node.js використовується для генерування динамічного контенту вебсторінок, роботи з файлами на сервері та керувати даними у базі даних. За замовчуванням платформа використовує менеджер пакунків npm, що є найбільшим реєстром програмного забезпечення. Це дозволяє

використовувати вже готові рішення для пришвидшення та спрощення написання потрібної логіки програми. Кожен розробник може додавати власний код у реєстр, після чого ним зможуть користуватись усі охочі.

C# – об'єктно-орієнтована мова програмування, яка має статичну типізацію та використовується для написання надійних і масштабованих програм, а саме десктопних застосунків, мобільних додатків, вебзастосунків [9].

Ця мова програмування має багато доступних бібліотек і фреймворків, а також підтримку інструментів від Microsoft.

C# дозволяє використовувати фреймворк ASP.NET, який має велику кількість потужних інструментів та бібліотек для створення вебзастосунків. ASP.NET надає можливість розроблення програм за допомогою використання MVC (Model-View-Controller), Web API і SignalR.

Програми, що розробляються мовою C#, часто використовують платформою .NET, яка орієнтована на операційну систему Windows. Для того, щоб надати можливість кросплатформної розробки було створено фреймворк .NET Core. Новіші версії C# та .NET 5 також отримали можливості для підтримки кросплатформності. Це дозволило розробляти вебзастосунки, які працюватимуть не лише на Windows, а і на операційних системах Linux та macOS.

C# часто використовують для розробки великих та складних програм, зокрема і серверної частини вебзастосунків, JavaScript – для простіших, але у той же час гнучких та швидких рішень.

Отже для написання серверної частини дипломного проєкту було обрано мову програмування JavaScript, а саме платформу Node.js, тому що вона дозволяє швидко та легко розробляти потрібне API.

2.1.3. *Express.js та Nest.js*

Існує велика кількість Node.js фреймворків для напсання серверної

частини вебзастосунків. При виборі фреймворку для написання серверної частини дипломного проєкту розглядались два варіанти: Express.js та Nest.js. Обидва фреймворки є дуже популярними серед веброзробників.

Express.js є досить простим для вивчення та використання, але у той же час дозволяє створювати складні та швидкі вебзастосунки. Цей фреймворк використовує зручний інструмент маршрутизації з методами HTTP та URL адресами, а також передбачає можливість додавання проміжного програмного забезпечення, яке має доступ до запитів клієнта і може їх додатково опрацьовувати, наприклад аутентифікувати користувача у системі – це дозволяє уникнути надлишкового коду та повторюваної логіки [10].

Nest.js передбачає architecture-first підхід, який використовує три ключові типи компонентів: контролери, провайдери та модулі. Головною метою фреймворку є створення структури проєкту з використанням підходів об’єктно-орієнтованого та функціонального програмування. Завдяки модульній архітектурі код може легко бути перевикористаний велику кількість разів. Для Nest.js кожен компонент програми може бути визначений в межах певного модулю з налаштуваннями контролерів, залежностей та провайдерів [11].

Для дипломного проєкту з цих двох фреймворків було обрано Express.js через його високу продуктивність та зручність використання маршрутизації і проміжного програмного забезпечення.

2.2. Технології для розробки клієнтської частини

Для створення клієнтської частини було обрано бібліотеку React, яка дозволяє створити гнучкий та зручний вебзастосунок. Для розробки було використано також мову TypeScript та препроцесор SCSS з метою задання стилів компонентів.

2.2.1. *React, Angular та Vue*

Серед відомих та популярних фреймворків для розробки клієнтської частини вебзастосунків можна виокремити React, Angular та Vue. Усі вони призначені для розробки динамічних, гнучких та швидких вебзастосунків.

React – відкрита JavaScript бібліотека, яка може також використовуватися з мовою програмування TypeScript[12]. Для React було розроблено велику кількість бібліотек, які допомагають створювати гарні та адаптивні користувацькі інтерфейси.

React передбачає розробку клієнтської частини вебзастосунку з використанням компонентів, які є частинами користувацького інтерфейсу, що багаторазово використовуються.

Компоненти мають власні методи та стан, що дозволяє легше та ефективніше керувати ними. Складні програми можуть також використовувати додаткові бібліотеки, наприклад Redux та MobX, для керування глобальним станом застосунку.

React дозволяє використання синтаксису JSX (JavaScript XML) та TSX (TypeScript XML), в залежності від обраної мови програмування. Це дозволяє поєднувати JavaScript чи TypeScript з HTML в одному файлі, що робить розробку компонентів набагато зручнішою.

Потік даних у бібліотеці React є односпрямованим та передбачає передачу даних від батьківських до дочірніх компонентів. Чітко визначений потік даних спрощує розробку програми.

Маршрутизація у клієнтській частині вебзастосунку може бути додана за допомогою бібліотеки React Router, що дозволяє здійснювати навігацію між різними компонентами програми.

Angular – відкрита JavaScript платформа, розроблена компанією Google. Використовується зазвичай для складних вебзастосунків, які потребують масштабованості та продуктивності [13]. Angular використовує мову програмування TypeScript.

Платформа має багато різних інструментів, таких як Dependency

Injection, що дозволяє забезпечити ефективну роботу залежностей між компонентами, і Observable, що дозволяє відстежувати та реагувати на зміни в додатку. Основним підходом до розробки вебзастосунків з використанням Angular є шаблонізація, яка дозволяє розробникам створювати реактивні та динамічні інтерфейси користувача.

Angular дозволяє використання бібліотеки RxJS (Reactive Extensions for JavaScript), що надає можливість застосовувати оператори та утиліти для керування потоком даних у програмі та обробки асинхронних подій і функцій.

Як і React, Angular використовує компоненти для розробки графічного інтерфейсу. Компоненти в Angular складаються з шаблонів, стилів і логіки. Платформа підтримує двостороннє зв'язування даних, що забезпечує синхронізування даних між компонентом та інтерфейсом користувача, що полегшує керування станом програми.

Angular має багатофункціональний інтерфейс командного рядка (CLI), який включає різні команди для створення, тестування та розгортання застосунків, а також створення різних компонентів та модулів.

Angular також надає можливість тестування за допомогою інструментів Karma та Jasmine.

Vue – відкрита JavaScript бібліотека призначена для розробки графічних інтерфейсів вебзастосунків [14]. Ця бібліотека є легкою у вивченні, має хорошу документацію та дозволяє створювати динамічні та складні інтерфейси користувача. Vue може бути легко інтегрована з іншими технологіями, наприклад Vuex для управління станом додатку, або Vue Router для маршрутизації.

Як і React та Angular, Vue передбачає використання компонентів при розробці елементів користувацького інтерфейсу. При розробці клієнтської частини програми за допомогою Vue потрібно використовувати декларативний синтаксис шаблону, що надає можливість створювати шаблони HTML з додатковими директивами та синтаксисом прив'язки. Ці

директиви забезпечують прив’язування даних, обробку подій та перегляд списків.

Vue включає вбудовані директиви, за допомогою яких можна змінювати DOM на основі базових даних. Прикладом таких директив є v-bind, v-if, v-on і v-for, які призначені для зв’язування даних, умовного відображення елементів, обробки подій, відмальовування списків.

Директива v-model дозволяє використовувати двосторонню прив’язку даних, що забезпечує двонаправлену прив’язку між вхідними даними та властивостями даних, спрощуючи синхронізацію даних користувача із даними моделі.

Схоже до Angular, Vue дозволяє налаштовувати проєкт та особливості розробки за допомогою інструменту інтерфейсу командного рядка (CLI). Інтерфейс командного рядка можна використовувати для конфігурації збірки, створення проектних каркасів та управління плагінами.

Для розробки дипломного проєкту було обрано бібліотеку React через можливість розробки мовою програмування TypeScript, великою кількість додаткових бібліотек та зручність використання.

2.2.2. TypeScript та JavaScript

Для розробки клієнтської частини вебзастосунку було розглянуто мови програмування JavaScript та TypeScript. Вони схожі багато у чому:

- синтаксис TypeScript переважно ідентичний синтаксису JavaScript;
- основні можливості JavaScript підтримуються TypeScript, а саме змінні, функції, об’єкти, масиви та цикли;
- обидві мови дозволяють керувати елементами вебсторінок;
- бібліотеки, такі як React, Angular та Vue, можуть використовуватися як за допомогою мови JavaScript, так і TypeScript.

TypeScript – це мова програмування, яка була розроблена як розширення мови JavaScript [15]. Особливістю TypeScript є наявність власного компілятора, який конвертує код, написаний цією мовою у JavaScript, що дозволяє знаходити помилки перед запуском проєкту – під час компіляції. А також наявність типів даних часто є перевагою TypeScript, адже це значно покращує процес написання коду.

TypeScript дозволяє визначати інтерфейси компонентів та інших об’єктів, які з ними використовуються, наприклад зазначення властивостей та їх типів для об’єкту Props. Також TypeScript надає можливість розширення функціональних компонентів за допомогою власного інтерфейсу.

Важливою функцією, яку надає TypeScript при роботі з бібліотекою React є додавання перевірки типу даних у хуки, а також додавання перевірки для значення null, що дозволяє уникнути ряду помилок при розробці. Варто зазначити і про можливість використання спільних компонентів, які можна багаторазово застосовувати для різних типів даних.

Отже, TypeScript значно покращує процес розробки клієнтської частини, адже допомагає уникати помилок та надає зручні інструменти для створення компонентів і взаємодії з ними. Саме тому у дипломному проєкті разом із бібліотекою React використовується мова програмування TypeScript.

2.2.3. *CSS та SCSS*

Забезпечення гарного вигляду вебзастосунку є важливою частиною Frontend-розробки, тому для проєкту було використанно технологію SCSS.

CSS використовує простий і зрозумілий синтаксис, який включає селектори, властивості і значення. CSS надає базові функції для стилізації елементів, а саме вибір елементів за тегом, класом або ідентифікатором та задання для них кольорів, шрифтів, розташування тощо.

SCSS – це препроцесор, призначений для покращення і пришвидшення процесу написання стилей сторінки, адже є розширенням CSS [16]. Він дозволяє використання змінних, функцій наслідування та інших можливостей, що значно спрощує розробку клієнтської частини. Саме тому для розробки дипломного проєкту було обрано цю технологію, замість написання звичайних CSS стилів.

Для SCSS обов'язково потрібно застосовувати перетворення у CSS файли, адже браузери не розпізнають даний формат і тому не відображають стилі коректно.

Можливості, які надає SCSS:

- використання змінних, які можна використовувати у поєднанні з директивами, наприклад @for;
- вкладеність, яка дозволяє додавати CSS правила всередині інших CSS правил, що робить файл зручнішим для читання;
- директиви @for і @while, а також умова @if-else значно спрощують процес написання стилей;
- додавання функцій, що є зручним інструментом для уникнення повторів;
- міксини дозволяють багаторазово використовувати набір CSS стилів та поєднувати їх з іншими значеннями.

Зручність у використанні та ефективність можливостей препроцесора SCSS – причина вибору цієї технології для проєкту.

2.3. База даних

Для проєкту розглядались СУБД MongoDB та СУБД PostgreSQL. Вони відрізняються підходами до організації та зберігання даних. MongoDB належить до NoSQL баз даних, а PostgreSQL – до баз даних SQL.

Бази даних NoSQL – це нереляційні бази даних, які на відміну від реляційних, для зберігання інформації не використовують таблиці і схеми.

Бази даних NoSQL є досить простими у використанні. Їх існує чотири основних типи: ключ-значення, документо-орієнтовані, стовпчасті та графові.

MongoDB працює з документо-орієнтованою моделлю даних, що дозволяє значно підвищити продуктивність системи. Така модель є дуже гнучкою, тому що дозволяє зберігати неструктуровані дані [17]. Інформація, у вигляді ключ-значення, записується у документи, які об'єднуються у колекції.

Бази даних SQL – це реляційні бази даних, які використовують структурну мову запитів (SQL) та передбачають зберігання даних у вигляді таблиць, і схем за потреби встановленими між ними відношеннями. Для сутностей чітко визначаються поля та їх типи даних, чого немає у NoSQL.

PostgreSQL є потужною об'єктно-реляційною базою даних з відкритим кодом [18]. Вона дозволяє створювати складні системи та безпечно зберігати дані. PostgreSQL надає можливість використовувати:

- різні типи запитів;
- індекси для пришвидшення виконання пошукових запитів;
- тригери, які дозволяють виконувати набір інструкцій перед або після запиту;
- транзакції – одиниці роботи, які передбачають цілісність даних.

Для дипломного проєкту було обрано PostgreSQL, тому що вона є реляційною базою даних, яка забезпечує безпечне збереження даних.

2.4. Висновки до другого розділу

Для розробки серверної частин вебзастосунку для підтримки діяльності сервісних центрів було обрано мову програмування JavaScript. Зокрема було обрано Express.js – фреймворк на Node.js, адже він є простим, швидким та зручним у написанні і організації коду, а також взаємодії з базою даних.

Для клієнтської частини було обрано бібліотеку React, яка дозволяє

створювати гнучкі клієнтські рішення. Бібліотеку було вирішено використовувати з мовою програмування TypeScript, з метою покращення процесу розробки. Також було обрано препроцесор SCSS, який має зручні інструменти для написання стилів.

Як базу даних було обрано СУБД PostgreSQL, яка є об’єктно-реляційною і вважається надійною та потужною.

3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ ЗАСТОСУНКУ

3.1. Опис вимог до розроблюваного застосунку

Для того, щоб розроблюване програмне забезпечення виконувало усі заплановані завдання, було сформульовано функціональні вимоги, які були визначено з урахуванням ролей користувачів, а саме “Адміністратор”, “Клієнт”, “Працівник” та для неавторизованих користувачів. Визначені функціональні вимоги до застосунку наведено у табл. 3.1-3.7.

Таблиця 3.1

Функціональні вимоги до роботи застосунку з усіма користувачами

Код вимоги	Опис вимоги
F-1	Користувач повинен мати змогу переглядати список усіх сервісних центрів, в якому відображається назва, головне фото, рейтинг, короткий опис.
F-2	Має бути можливість застосовувати фільтри (місто, категорія) та сортування за назвою до списку сервісних центрів.
F-3	Користувач повинен мати можливість пошуку потрібного сервісного центру за назвою.
F-4	Користувач повинен мати змогу перейти на сторінку обраного зі списку сервісного центру.
F-5	Користувач повинен мати змогу переглянути детальну інформацію на сторінці обраного сервісного центру: • головне фото (за наявності); • назва; • галерея фото; • детальний опис; • категорії послуг, що надаються; • контакти; • карта з місцерозташуванням (за наявності); • відгуки клієнтів.

Продовження табл. 3.1

F-6	Користувач повинен мати можливість створити заяву на замовлення послуг ремонту або обслуговування на сторінці обраного сервісного центру. Для цього в модальному вікні заповнюється форма з наступними полями: • назва; • прізвище; • ім’я; • по батькові (не обов’язково); • email; • телефон; • об’єкт (те, що потребує ремонту чи обслуговування), • модель пристрою (за наявності); • опис (причина звернення). Після надсилання форми створення заяви, для неї автоматично додається дата створення та статус “Розглядається” та генерується номер, що містить 16 цифр у форматі “xxxx-xxxx-xxxx-xxxx” та надсилається на електронну пошту замовника, вказану в формі. При введені некоректних даних та / або незаповненні усіх обов’язкових полів у формі, користувач отримує повідомлення де зазначено, що потрібно вказати або змінити.
F-7	Користувач повинен мати можливість на сторінці заяв здійснювати перевірку етапу розгляду заяви. Для цього у відповідному пошуковому полі вводиться номер заяви. Якщо заява знайдена, користувач повинен мати можливість переглянути всю інформацію про неї. У випадку, якщо заяви з вказаним номером не існує або номер введено у не правильному форматі, користувач отримує повідомлення про це.
F-8	Користувач повинен мати можливість на сторінці замовлень здійснювати перевірку етапу виконання замовлення. Для цього у відповідному пошуковому полі вводиться номер замовлення. Якщо замовлення знайдене, користувач повинен мати можливість переглянути інформацію про неї. У випадку, якщо замовлення з вказаним номером не існує або номер введено у не правильному форматі, користувач отримує повідомлення про це.

Таблиця 3.2

Функціональні вимоги до роботи застосунку з неавторизованими користувачами

Код вимоги	Опис вимоги
F-9	Користувач повинен мати змогу зареєструватися у ролі адміністратора або клієнта. Для реєстрації користувач повинен перейти на сторінку реєстрації та в формі вказати: • фото (не обов'язково); • email; • пароль; • прізвище; • ім'я; • по батькові; • телефон; • дату народження. При введені некоректних даних та / або незаповненні усіх обов'язкових полів у формі, користувач отримує повідомлення де зазначено, що потрібно вказати або змінити.
F-10	Користувач повинен мати змогу увійти в обліковий запис за допомогою електронної пошти (email) та паролю. Для цього потрібно перейти на сторінку входу в обліковий запис та ввести відповідні дані у форму. Якщо введені некоректні дані, користувач отримує повідомлення про це.

Таблиця 3.3

Функціональні вимоги до роботи застосунку з усіма авторизованими користувачами

Код вимоги	Опис вимоги
F-11	Користувач повинен мати можливість переглядати сторінку власного профілю.
F-12	Користувач повинен мати можливість вийти з облікового запису.

Продовження табл. 3.3

F-13	При заповненні форми для створення заяви у сервісному центрі, автоматично заповнюватися поля з контактними даними користувача відповідно до його профілю.
F-14	Користувач повинен мати можливість змінити пароль на сторінці власного профілю. Для цього він повинен ввести старий пароль та новий пароль. Якщо старий пароль вказано неправильно, то пароль не змінюється на новий.

Таблиця 3.4

Функціональні вимоги до роботи застосунку з авторизованими користувачами у ролі адміністратора сервісних центрів

Код вимоги	Опис вимоги
F-15	Адміністратор повинен мати можливість створювати лише одну сторінку для власного сервісного центру, якшо вона ще не існує. Для цього у модальному вікні потрібно потрібно обрати потрібні категорії зі списку, обрати місто з випадаючого списку та ввести такі дані: • головне фото (не обов'язково); • назва; • короткий опис; • опис; • email; • телефон; • адреса; • широту та довготу для позначення координат сервісного центру на карті (не обов'язково). При введені некоректних даних та / або незаповненні усіх обов'язкових полів у формі, користувач отримує повідомлення де зазначено, що потрібно вказати або змінити.
F-16	Адміністратор повинен мати можливість редагувати усі дані сервісного центру, що були вказані при його створенні.
F-17	Адміністратор повинен мати можливість видаляти власний сервісний центр, при цьому видаляються облікові записи працівників сервісного центру, заяви, замовлення та завдання, що були створенні для цього сервісного центру.

Продовження табл. 3.4

F-18	Адміністратор повинен мати можливість реєструвати працівників власного сервісного центру. Для цього потрібно ввести такі дані: • фото (не обов'язково); • email; • пароль; • прізвище; • ім'я; • по батькові; • телефон; • дату народження; • посада; • дата початку співпраці. Після реєстрації працівник на електронну пошту, на яку створено обліковий запис, отримує згенерований пароль для доступу. При введені некоректних даних та / або незаповненні усіх обов'язкових полів у формі, користувач отримує повідомлення де зазначено, що потрібно вказати або змінити.
F-19	Адміністратор повинен мати можливість редагувати усі дані працівників власного сервісного центру, вказані при їх реєстрації, окрім поля "Email", а також вказувати та змінювати завершення співпраці. При введені некоректних даних та / або незаповненні усіх обов'язкових полів у формі, користувач отримує повідомлення де зазначено, що потрібно вказати або змінити.
F-20	Адміністратор повинен мати можливість видаляти працівника власного сервісного центру. При цьому видаляються усі завдання, які були створенні для цього працівника.
F-21	Адміністратор повинен мати можливість редагувати заяву, що була створена для його сервісного центру, а саме змінювати статус, вказуючи одне із значень "Розглядається", "Прийнято" або "Відхилено" та додавати коментар. При введені некоректних даних та / або незаповненні усіх обов'язкових полів у формі, користувач отримує повідомлення де зазначено, що потрібно вказати або змінити.

Продовження табл. 3.4

F-22	Адміністратор повинен мати можливість видаляти заяву, яка має статус “Відхилено”. Якщо для заяви існують замовлення, то адміністратор отримує повідомлення з номерами замовлень, та видалення не відбувається.
F-23	Адміністратор повинен мати можливість створювати замовлення. Для цього потрібно заповнити форму, яка містить ті ж дані, що і заява, а також поля “Ціна” та “Запланована дата завершення”. Для замовлення автоматично додається дата створення, статус “Створено” та генерується номер із 16 цифр у форматі xxxx-xxxx-xxxx-xxxx, який надсилається на електронну пошту замовника, вказану в формі. При введені некоректних даних та / або незаповненні усіх обов’язкових полів у формі, користувач отримує повідомлення де зазначено, що потрібно вказати або змінити.
F-24	Адміністратор повинен мати можливість створити замовлення на основі заяви, яка має статус “Прийнято”. У такому випадку відповідні поля форми автоматично заповнюються, але можуть бути змінені.
F-25	Адміністратор повинен мати можливість редагувати власні замовлення. Відредагованими можуть бути усі поля, що заповнювались адміністратором при створенні замовлення, а також можна вказувати та редагувати значення полів “дата завершення” та “Оплачено” та встановлювати полю “Статус” одне із значень: • створено; • виконується; • виконано; • проблема; • скасовано. При введені некоректних даних та / або незаповненні усіх обов’язкових полів у формі, користувач отримує повідомлення де зазначено, що потрібно вказати або змінити.
F-26	Адміністратор повинен мати можливість видаляти замовлення, яке має статус “Скасовано”. Якщо для замовлення існують завдання, то вони також видаляються.

Продовження табл. 3.4

F-27	Адміністратор повинен мати можливість створювати завдання на основі визначеного замовлення або не залежно. Для цього у формі в модальному вікні обирається відповідальний працівник, а також вказуються назва, опис, запланована дата завершення. Для замовлення автоматично додається дата створення та статус “Створено”. При введені некоректних даних та / або незаповненні усіх обов’язкових полів у формі, користувач отримує повідомлення де зазначено, що потрібно вказати або змінити.
F-28	Адміністратор повинен мати можливість редагувати власні завдання. Відредагованими можуть бути усі поля, що заповнювались адміністратором при створенні замовлення, а також можна вказувати та редагувати значення полів “дата завершення” та встановлювати полю “Статус” одне із значень: • створено; • виконується; • виконано; • проблема; • скасовано. При введені некоректних даних та / або незаповненні усіх обов’язкових полів у формі, користувач отримує повідомлення де зазначено, що потрібно вказати або змінити.
F-29	Адміністратор повинен мати можливість видаляти завдання, яке має статус “Скасовано”.

Таблиця 3.5

Функціональні вимоги до роботи застосунку з авторизованими користувачами у ролі клієнту сервісних центрів

Код вимоги	Опис вимоги
F-30	Клієнт повинен мати можливість переглядати на відповідних сторінках списки усіх своїх заяв та створених сервісними центрами на їх основі замовлень.

Продовження табл. 3.5

F-31	Клієнт повинен мати можливість переглядати у модальному вікні детальну інформацію про обрану зі списку власну заяву чи замовлення.
F-32	Клієнт повинен мати можливість залишити відгук про обслуговування на сторінці сервісного сервісу. Для цього він має заповнити форму у модальному вікні і вказати рейтинг від 1 (незадовільно) до 5 (добре). За бажанням клієнта також може бути написаний текст відгуку.

Таблиця 3.6

Функціональні вимоги до роботи застосунку з авторизованими користувачами у ролі працівника сервісних центрів

Код вимоги	Опис вимоги
F-33	Працівник повинен мати можливість змінювати статус завдання відповідно до етапу виконання. Для цього у формі у модальному вікні з випадаючого списку обирається потрібне значення “Виконується”, “Виконано” або “Проблема” та за потреби вказується коментар.

Таблиця 3.7

Функціональні вимоги до роботи застосунку з авторизованими користувачами у ролях працівника та адміністратора сервісного центру

Код вимоги	Опис вимоги
F-34	Користувачі повинні мати можливість переглядати дані працівників власного сервісного центру.
F-35	Користувачі повинні мати можливість переглядати список заяв клієнтів з можливістю фільтрів за статусом, сортуванням за датою створення або номером та пошуком за частиною номеру.

Продовження табл. 3.7

F-36	Користувачі повинні мати можливість переглядати список замовлень з можливістю фільтрів за статусом, сортуванням за датою створення, запланованою датою завершення, датою завершення або номером та пошуком за частиною номеру.
F-37	Користувачі повинні мати можливість переглядати список завдань з можливістю фільтрів за статусом, сортуванням за датою створення, запланованою датою завершення, датою завершення, номером або назвою та пошуком за частиною назви.
F-38	Користувачі повинні мати можливість переглядати список працівників сервісного центру.
F-39	Користувачі повинні мати можливість переглядати детальну інформацію про заяву у модальному вікні натискаючи відповідну кнопку в таблиці.
F-40	Користувачі повинні мати можливість переглядати детальну інформацію про замовлення у модальному вікні натискаючи відповідну кнопку в таблиці.
F-41	Користувачі повинні мати можливість переглядати детальну інформацію про завдання у модальному вікні натискаючи відповідну кнопку в таблиці.
F-42	Користувачі повинні мати можливість переглядати детальну інформацію про працівника у модальному вікні натискаючи кнопку “Деталі”.

Таблиця 3.8

Функціональні вимоги до роботи застосунку з авторизованими користувачами у ролях адміністратора та клієнта

Код вимоги	Опис вимоги
F-43	Користувачі повинні мати можливість редагувати усі свої дані вказані при реєстрації, окрім полів “Email” та “Пароль”.

До роботи застосунку було також визначено нефункціональні

вимоги, які необхідно виконати, щоб програмне забезпечення працювало правильно для різних користувачів. Нефункціональні вимоги наведено у табл. 3.9.

Таблиця 3.9

Нефункціональні вимоги до роботи застосунку

Код вимоги	Опис вимоги
NF-1	Інтерфейс користувача має коректно відображатись на телефонах, планшетах та комп'ютерах з екранами шириною більше 300 px.
NF-2	Застосунок має коректно працювати з наступними мінімальними вимогами до пристрою: • 1 ГБ оперативної пам'яті; • процесор з тактовою частотою 850 MHz.
NF-3	Сервер повинен працювати на ОС Windows 10.
NF-4	Застосунок має використовувати базу даних PostgreSQL.
NF-5	Клієнтська частина повинна підтримуватись браузерами Google Chrome, Microsoft Edge та Mozilla Firefox.

3.2. Опис структурної організації застосунку

Розроблюваний застосунок має клієнт-серверну структуру, а також використовує базу даних.

Клієнтська частина складається з

- модуля компонентів графічного інтерфейсу, що описує елементи, які бачить користувач, а також забезпечує можливість взаємодіяти з ними;
- модуля взаємодії з API, який передбачає надсилання запитів на серверну частину програми та отримання відповідей з неї, а також керування даними, які відображаються, отже взаємодіяти з даними, що зберігаються у базі даних через графічний інтерфейс.

Серверна частина включає в себе

- модуль маршрутизації, який визначає HTTP методи та URL адреси запитів, за потреби забезпечує виконання проміжного програмного забезпечення для аутентифікації користувача та викликає відповідний метод контролеру;
- проміжне програмне забезпечення, яке забезпечує перевірку прав користувача на доступ до ресурсів і при успішному виконанні передає отримані дані користувача методу контролера, а також повертає повідомлення про помилку для неуспішного запиту;
- контролери, які отримують запити з клієнта, опрацьовують їх, а саме здійснюють виклик валідації відповідно до визначених схем, викликають потрібні функції для роботи з базою даних та повертають на клієнт відповідь з потрібними даними та успішний статус або статус помилки та відповідне повідомлення;
- модуль валідації вхідних даних перевіряє коректність та наявність усіх потрібних вхідних даних отриманих із запиту, при негативному результаті перевірки повертається помилка з описом проблеми;
- модуль роботи з базою даних забезпечує доступ до інформації, що зберігається у базі даних та надає можливість отримувати дані з неї та керувати цими даними;
- моделі даних описують таблиці в базі даних, а саме визначають усі поля, їх типи даних, обов’язковість та унікальність, а також зв’язки між таблицями;
- допоміжні функції містять код, який багаторазово використовуються, тому винесення їх у окремий модуль дозволяє слідувати правилам написання чистого коду.

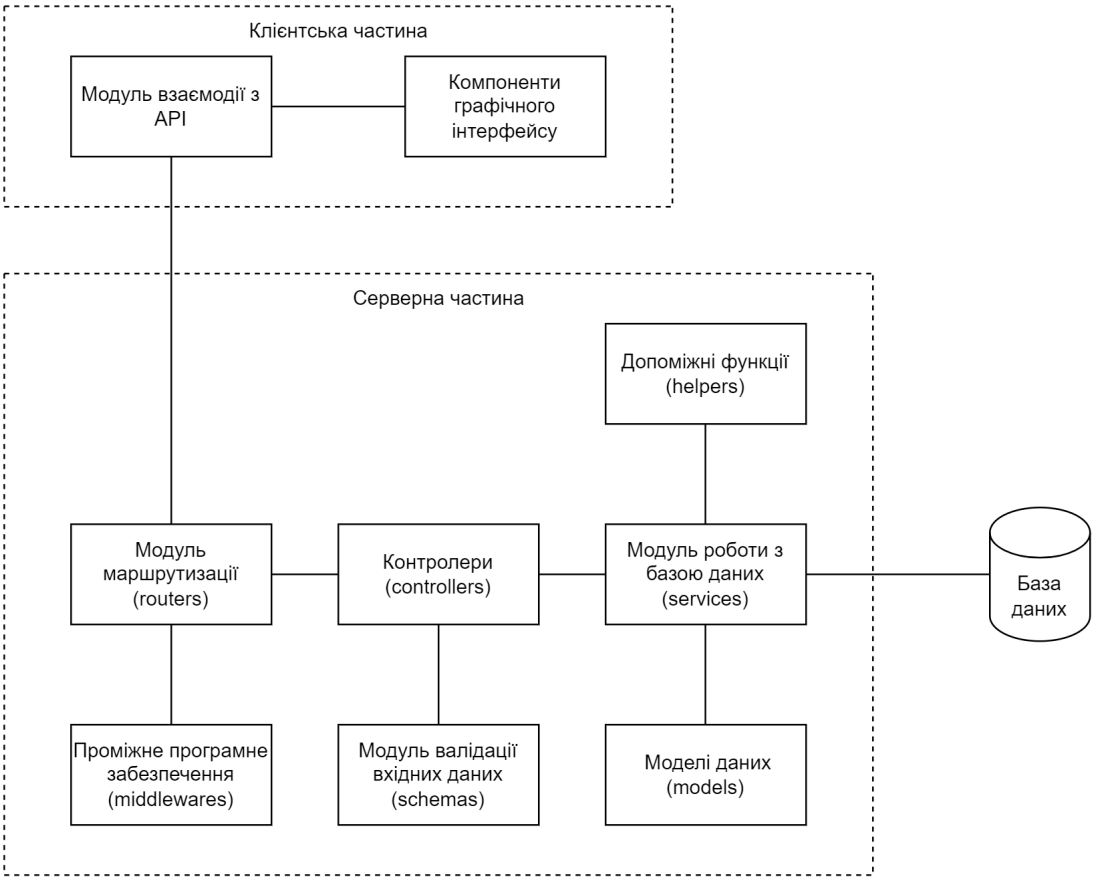


Рис. 3.1. Загальна структура застосунку. UML-діаграма

3.3. Опис структур даних застосунку

Для того, щоб створити базу даних застосунку потрібно спочатку визначити структури даних. Було визначено перелік сутностей, які потрібні для роботи програми, та описано всі їх поля.

Облікові дані (Credentials):

- електронна пошта (email) – електронна пошта, на яку зареєстровано обліковий запис користувача (унікальне значення);
- пароль (password) – пароль для доступу в обліковий запис.

Користувач (User):

- ідентифікатор (id) – ідентифікатор користувача;
- електронна пошта (email) – відповідно до поля електронна пошта (email) у таблиці Облікові дані (Credentials);
- роль (role) – роль зареєстрованого користувача. Приймає значення

адміністратор (administrator), клієнт (client) або працівник (employee);

- ім’я (firstName) – ім’я користувача;
- по батькові (middleName) – по батькові користувача (необов’язкове поле);
- прізвище (lastName) – прізвище користувача;
- дата народження (dateOfBirth) – дата народження користувача;
- телефон (phone) – телефон користувача;
- фото (photo) – фото користувача (необов’язкове поле).

Працівник (Employee)

- ідентифікатор (id) – ідентифікатор працівника;
- ідентифікатор користувача (userId) – ідентифікатор користувача;
- id сервісного центру (serviceCenterId) – сервісний центр, якому належить працівник;
- посада (position) – посада працівника;
- дата початку співпраці (cooperationStartDate) – дата початку співпраці з працівником;
- дата завершення співпраці (cooperationEndDate) – дата закінчення співпраці з працівником (необов’язкове поле).

Сервісний центр (ServiceCenter):

- ідентифікатор (id) – ідентифікатор сервісного центру;
- id адміністратора (administratorId) – користувач, якому належить сервісний центр;
- назва (name) – назва сервісного центру;
- електронна пошта (email) – електронна пошта сервісного центру;
- короткий опис (shortDescription) – відображається у списку сервісних центрів;
- опис (description) – відображається на сторінці цього сервісного центру;

- головне фото (mainPhoto) – відображається у списку сервісних центрів;
- id міста (cityId) – місто, у якому розташований сервісний центр;
- адреса (address) – адреса сервісного центру;
- телефон (phone) – телефон сервісного центру;
- широта (mapLatitude) – широта місця розташування сервісного центру для визначення координат на карті (необов'язкове поле);
- довгота (mapLongitude) – довгота місця розташування сервісного центру для визначення координат на карті (необов'язкове поле).

Фото (Photo):

- ідентифікатор (id) – ідентифікатор фото;
- id сервісного центру (serviceCenterId) – сервісний центр, якому належить фото;
- фото (photo) – фото сервісного центру.

Відгук (Response):

- ідентифікатор (id) – ідентифікатор відгуку;
- id клієнта (clientId) – користувач, який залишив відгук;
- id сервісного центру (serviceCenterId) – сервісний центр, для якого залишено відгук;
- дата (date) – дата створення відгуку. Приймає значення дати на момент створення запису у таблиці;
- рейтинг (rating) – оцінка сервісного центру користувачем. Приймає значення від 1 до 5;
- текст (text) – текст відгуку (необов'язкове поле).

Місто (City):

- ідентифікатор (id) – ідентифікатор міста;
- назва (name) – назва міста.

Категорія (Category):

- ідентифікатор (id) – ідентифікатор категорії;

- назва (name) – назва категорії.

Категорії сервісного центру (ServiceCenterCategory):

- ідентифікатор (id) – ідентифікатор категорії сервісного центру;
- id сервісного центру (serviceCenterId) – ідентифікатор сервісного центру;
- id категорії (categoryId) – ідентифікатор категорії.

Контакти клієнта (ClientContacts):

- ідентифікатор (id) – ідентифікатор контактів клієнта;
- id клієнта (clientId) – авторизований користувач, якому належить заява (необов'язкове поле);
- ім'я (firstName) – ім'я користувача;
- по батькові (middleName) – по батькові користувача (необов'язкове поле);
- прізвище (lastName) – прізвище користувача;
- електронна пошта (email) – електронна пошта користувача;
- телефон (phone) – телефон користувача;
- доставка в сервісний центр (deliveryToServiceCenter) – потреба у доставці в сервісний центр. Приймає значення true (так), false (ні);
- доставка з сервісного центру (deliveryFromServiceCenter) – потреба у доставці з сервісного центру. Приймає значення true (так), false (ні);
- адреса (address) – адреса для доставки (необов'язкове поле).

Дані об'єкта (ObjectData):

- ідентифікатор (id) – ідентифікатор даних об'єкта;
- тип об'єкт (objectType) – тип пристрою, який потребує ремонту чи обслуговування;
- модель (model) – модель пристрою, який потребує ремонту чи обслуговування (необов'язкове поле);
- опис (description) – детальний опис несправності та послуг, які

потрібні.

Заява (Application):

- ідентифікатор (id) – ідентифікатор заяви;
- номер (number) – рядок, який містить 16 цифр у форматі хххх-хххх-хххх-хххх. Генерується при створенні заяви;
- id контактів клієнта (clientContactsId) – ідентифікатор контактів клієнта;
- id даних об’єкта (objectDataId) – ідентифікатор даних об’єкта, який потребує ремонту чи обслуговування;
- id сервісного центру (serviceCenterId) – сервісний центр, для якого створено заяву;
- дата і час створення (dateTimeCreated) – дата і час створення заяви. Приймає значення дати і часу на момент створення запису у таблиці;
- статус (status) – статус заяви. Приймає значення розглядається (considered), прийнято (accepted) або відхилено (refused).
  При створенні запису у таблиці набуває значення considered;
- коментар (comment) – коментар адміністратора сервісного центру до заяви.

Замовлення (Order):

- ідентифікатор (id) – ідентифікатор замовлення;
- номер (number) – рядок, який містить 16 цифр у форматі хххх-хххх-хххх-хххх. Генерується при створенні замовлення;
- id контактів клієнта (clientContactsId) – ідентифікатор контактів клієнта;
- id даних об’єкта (objectDataId) – ідентифікатор даних об’єкта, який потребує ремонту чи обслуговування;
- id сервісного центру (serviceCenterId) – сервісний центр, для якого створено замовлення;

- id заяви (applicationId) – ідентифікатор заяви, для якої створено замовлення (необов’язкове поле);
- id статусу (statusId) – id етапу виконання замовлення;
- вартість (price) – вартість виконання замовлення;
- оплачено (payed) – оплата замовлення клієнтом. Приймає значення true (так), false (ні). При створенні запису в таблиці набуває значення false;

Завдання (Task):

- ідентифікатор (id) – ідентифікатор завдання;
- id сервісного центру (serviceCenterId) – сервісний центр, для якого створено завдання;
- id замовлення (orderId) – id замовлення, для якого створене завдання (необов’язкове поле);
- id працівника (employeeId) – id працівника, який виконує завдання;
- опис (description) – опис робіт, які мають бути виконані для завдання;
- id статусу (statusId) – id етапу виконання завдання.

Статус (status):

- ідентифікатор (id) – ідентифікатор статусу;
- назва (name) – назва етапу виконання. Приймає значення створено (created), виконується (in progress), виконано (completed), проблема (problem) або скасовано (canceled).
  При створенні запису у таблиці набуває значення created;
- коментар (comment) – коментар адміністратора (для замовлення) або працівника (для завдання) сервісного центру щодо етапу виконання (необов’язкове поле);
- дата і час створення (dateTimeCreated) – дата і час створення. Приймає значення дати і часу на момент створення запису у

таблиці;

- запланована дата завершення (plannedDateCompleted) – запланована дата завершення виконання (необов’язкове поле);
- дата завершення (dateCompleted) – дата завершення виконання (необов’язкове поле).

Також було побудовано ER-діаграму (рис. 3.2), яка зображує таблиці реляційної бази даних та їх поля на основі визначених сутностей, а також додано зв’язки між таблицями.

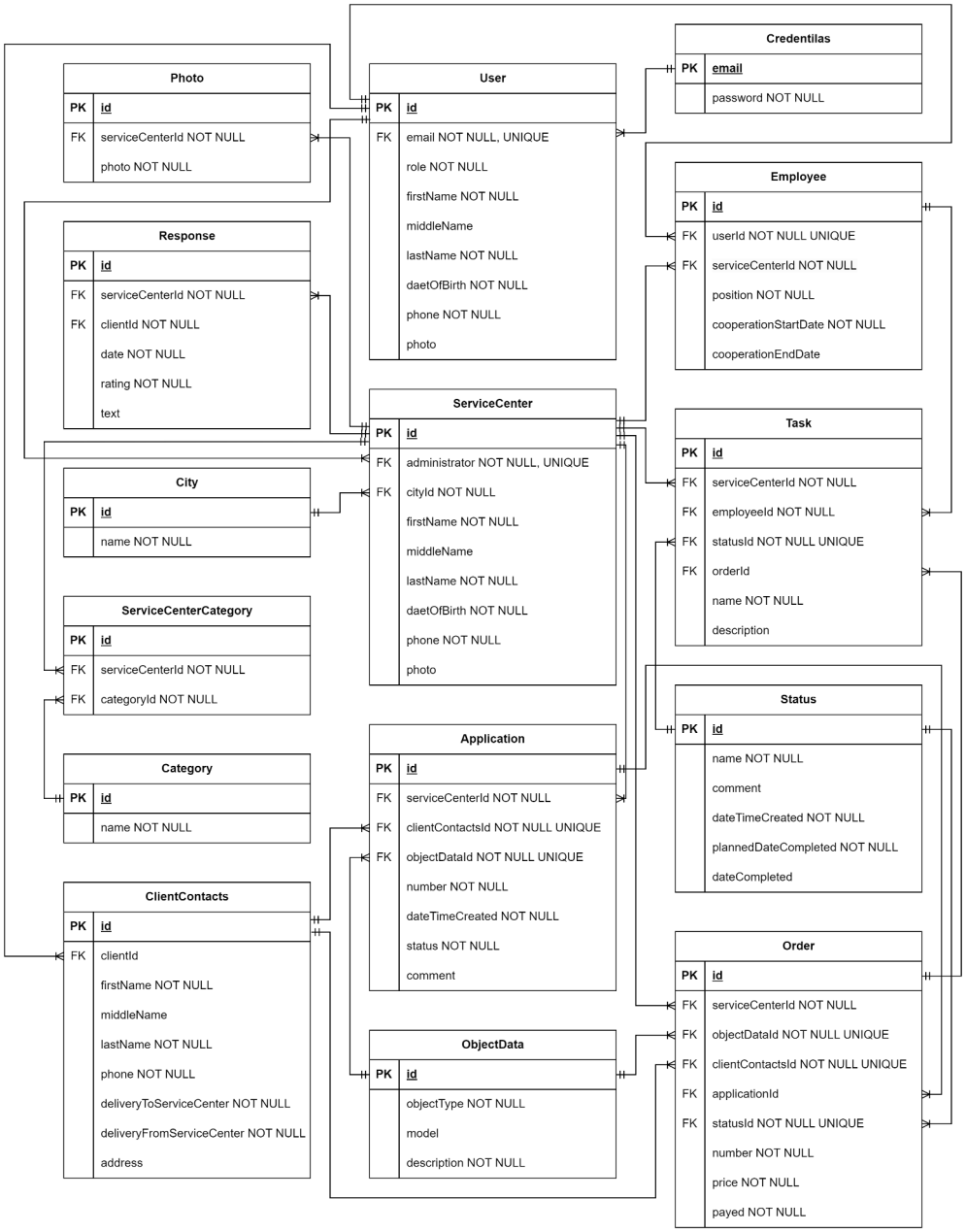


Рис. 3.2. Схема бази даних. ER-діаграма

3.4. Алгоритмічне забезпечення застосунку

Діаграма варіантів використання – це графічне зображення можливих функцій програми для користувачів. Таку діаграму було побудовано, щоб відобразити дії, що можуть бути виконаними різними користувачами вебзастосунку. А саме діаграма містить чотири актори: неавторизований користувач, адміністратор, клієнт, працівник. Також діаграма включає варіанти використання та відношення узагальнення між ними та акторами.

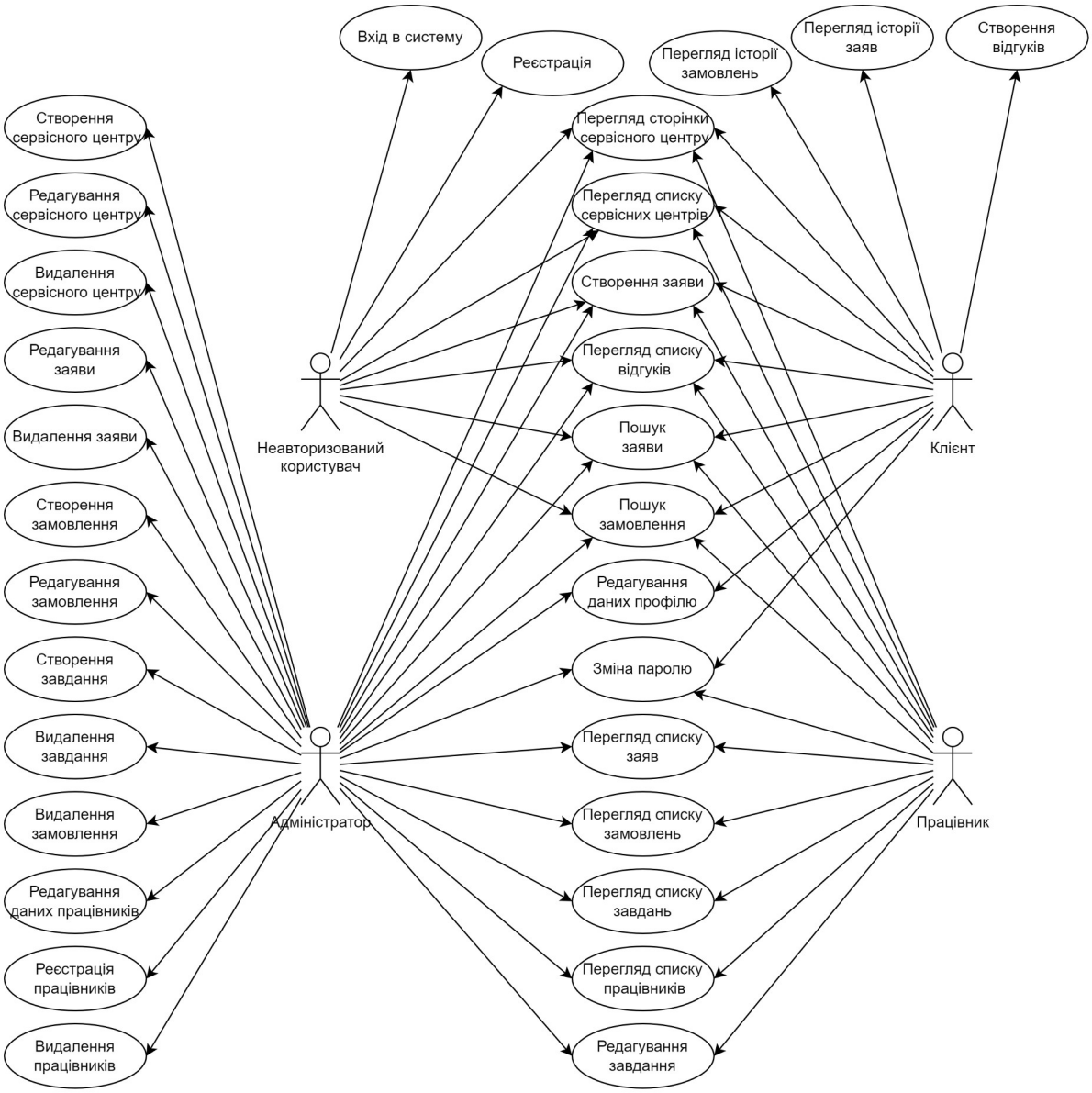


Рис. 3.3. Діаграма варіантів використання

Окрім того було побудовано діаграму потоку даних для процесу створення завдання на основі заяви клієнта. Діаграма потоку даних призначена для відображення особливостей передачі даних в інформаціних системах.

Розроблена діаграма містить такі процеси як створення заяви, замовлення і завдання, та використовує багато різної інформації: як з бази даних, так і даних введеними користувачами. Сховищами даних є бази даних сервісного центру, користувачів, працівників, заяв, замовлень та завдань. Зовнішніми сутностями є клієнт та адміністратор. Між усіма вказаними елементами було визначено потоки даних.

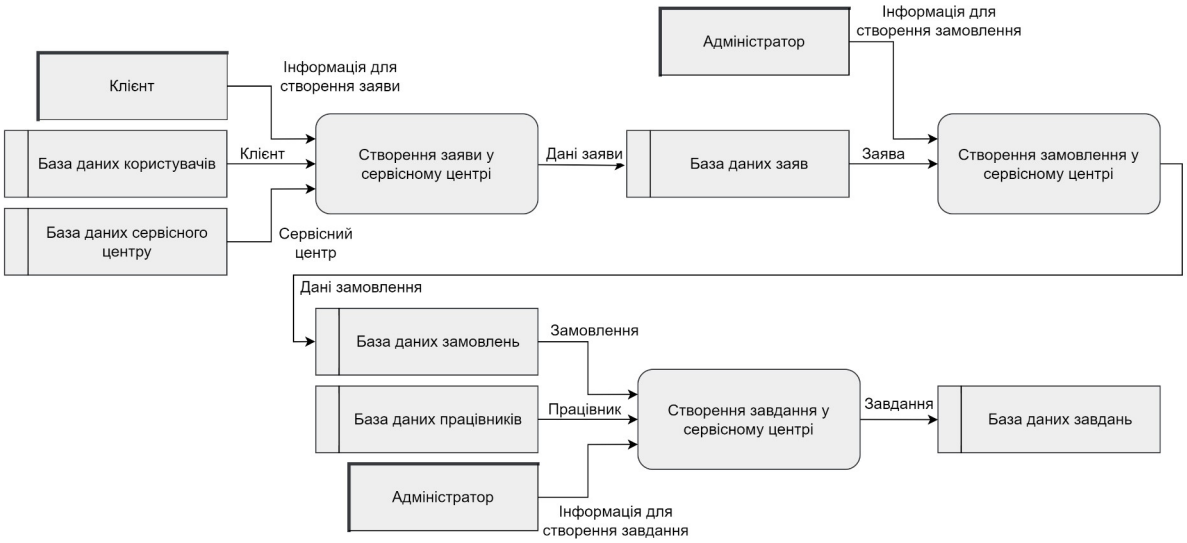


Рис. 3.4. Діаграма потоку даних. Процес створення завдання на основі заяви клієнта

3.5. Висновки до третього розділу

У третьому розділі було сформульовано функціональні та нефункціональні вимоги до розроблюваного програмного забезпечення. Вебзастосунок надає різні можливості для різних типів користувачів, що було детально описано. Програму можуть використовувати як неавторизовані користувачі, так і авторизовані, які мають одну з ролей “адміністратор”, “клієнт” або “працівник”.

Також було описано структурну організацію застосунку, а саме визначено які модулі матимуть клієнтська та серверна частини, як відбуватиметься взаємодія між ними, а також із базою даних.

Крім того було здійснено опис структур даних застосунку та визначено, що база даних матиме такі сутності: облікові дані, користувач, сервісний центр, фото, контакти, заява, замовлення, завдання, статус, відгук, місто, категорія. Було побудовано схему для бази даних із зазначенням усіх сутностей, їх полів, та зв’язків між таблицями.

Було реалізовано діаграму варіантів використання, що дозволило описати функції програми відносно ролей користувачів. Також було реалізовано діаграму потоку даних, що описує особливості керування даними для зазначеного процесу.

4. ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Реалізація серверної частини

Для серверної частини проєкту використовується фреймворк Express.js, який передбачає створення модулів маршрутизації, контролерів, проміжного програмного забезпечення та модуля роботи із базою даних.

Для роботи із базою даних було використано бібліотеку “sequelize”, яка є ORM (об’єктно-реляційне відображення) інструментом, призначеним для керування даними у базі, який дозволяє зменшити використання SQL-запитів за допомогою спеціальних методів визначених бібліотекою. До таких методів належать findOne, findMany, update, create тощо. Для того, щоб використовувати можливості бібліотеки було описано моделі для кожної таблиці бази даних. Модель – це об’єкт, у якому ключами є назви полів таблиці, а значеннями – їх опис, такі як тип даних, унікальність, обов’язковість. Ці моделі забезпечують доступ та можливість отримання та зміни даних у відповідних таблицях.

Важливим етапом написання серверної частини було реалізація реєстрації користувача та можливості входу в обліковий запис, а також аутентифікації. Це дозволило використання ролей для користувачів, кожна з яких передбачає різні можливості керування даними.

При реєстрації введений користувачем пароль хешується за допомогою бібліотеки “bcrypt” і тільки після цього зберігається у базі даних. Це зроблено для того, щоб захистити дані користувача, адже у випадку викрадення даних з бази, зловмисник не зможе дізнатися пароль. При вході в обліковий запис введений користувачем пароль хешується та порівнюється з тим, що міститься у базі даних. У випадку, якщо вони однакові, створюється та підписується JWT токен (JSON Web Token), який призначений для аутентифікації користувача. У розробленому проєкті токен зберігає ідентифікатор, роль та email користувача у вигляді рядка, який надсилається авторизованим користувачем разом із запитами, що

дозволяє аутентифікувати його, тобто перевірити його права на виконання цього запиту. Аутентифікація виконується у проміжному програмному забезпеченні.

Також при написанні серверної частини було реалізовано можливість програмно надсилати листи електронною поштою. Для цього було використано бібліотеку “nodemailer” та створено обліковий запис Google для сервісного центру. При створені заяв та замовлень на вказані у формі електронні пошту надсилаються листи зі створеної пошти сервісного центру з їх згенерованими номерами для можливості перевіряти етап їх виконання. А також при реєстрації працівників на вказані електронні адреси надсилаються листи з паролями доступу до облікового запису у вебзастосунку.

4.2. Реалізація клієнтської частини

Інтерфейс користувача розроблювався за допомогою бібліотеки React, що дозволило створити швидкий та гнучкий вебзастосунок. Для реалізації клієнтської частини було визначено маршрутизацію, створено компоненти графічного інтерфейсу для відображення даних на сторінках та можливості керування ними, виконано стилізацію цих компонентів та передбачено їх адаптивність для коректної роботи на пристроях з різними розмірами екрану.

4.2.1. *Маршрутизація*

Важливим завданням при розробці клієнтської частини є маршрутизація, яка забезпечує навігацію між сторінками вебзастосунку. У дипломному проєкті її було реалізовано за допомогою інструменту React Router. Для цього спочатку було визначено перелік вебсторінок, шлях до них та компоненти, які будуть рендеритись при переході користувача за відповідним посиланням.

Таблиця 4.1

Маршрутизація у клієнтській частині

Вебсторінка	**Шлях**	**Назва компоненту**
Список сервісних центрів	/	ServiceCenterList
Сервісний центр	/service-center/:id	ServiceCenter
Реєстрація	/registration	Registration
Логін	/login	Login
Профіль користувача	/profile	Profile
Керування сервісним центром	/service-center-management	ServiceCenterManagement
Заяви	/application-control	ApplicationControl
Замовлення	/order-control	OrderControl

4.2.2. *Компоненти графічного інтерфейсу*

Оскільки React є частим вибором розробників для реалізації клієнтської частини вебзастосунків, то існує велика кількість допоміжних інструментів, які можуть використовуватись разом із цією бібліотекою, щоб спростити і пришвидшити процес написання коду.

Прикладом є бібліотека MUI (Material User Interface), яка була використана при реалізації дипломного проєкту. Вона містить багато компонентів, які часто потрібні при розробці клієнтської частини, наприклад кнопки, поля для вводу даних (текст, число, дата), випадаючі списки, компоненти для відображення даних на сторінці та навігації. Варто зазначити, що ця бібліотека є дуже популярною серед розробників через свою простоту у використанні, багатофункціональність та наявність хорошої документації, яка чітко та зрозуміло описує можливості використання кожного компоненту.

Використання готових компонентів бібліотеки MUI при створенні власних компонентів дозволило створити гарний інтерфейс у єдиному стилі.

Також було використано бібліотеку Leaflet для додавання карти з місцерозташуванням сервісного центру на його сторінці в окремій вкладці “Карта”. Координати, що задаються на карті для додавання маркера, використовуються з даних, які адміністратор вводить при створенні та редагуванні сервісного центру. У випадку, якщо координати не задані, вкладка не відображається.

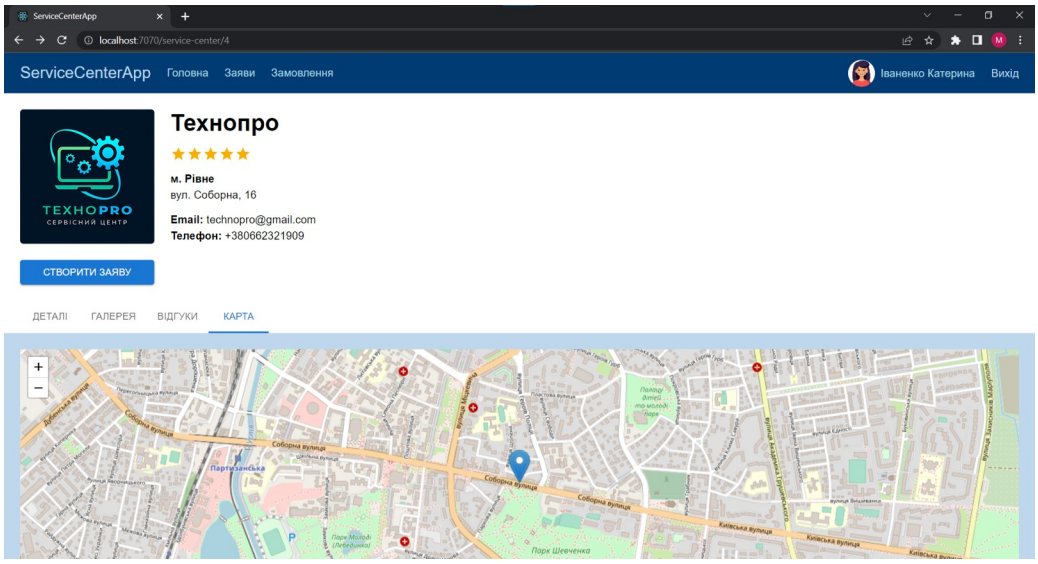


Рис. 4.1. Сторінка сервісного центру. Вкладка “Карта”

4.2.3. *Стилізація та адаптивність*

Для того, щоб надати компонентам бажаний вигляд та розташувати у потрібному місці на сторінці було використано препроцесор стилів SCSS. Для кожного блоку компонентів було створено відповідний файл styles.scss, де за допомогою вкладених CSS правил, змінних і функцій було визначено необхідні стилі для HTML елементів та класів елементів.

Інтерфейс користувача було розроблено адаптивним за допомогою використання стилів (media-запитів CSS, інструментів flexbox та grid) та властивостей використаних компонентів з бібліотеки MUI. Це дозволяє

використовувати вебзастосунок на різних пристроях, таких як комп’ютери, ноутбуки, планшети, телефони, які мають різні розміри екрану та співвідношення ширини та висоти. На рисунках 4.2.1 та 4.2.2 наведений приклад відображення форми реєстрації на комп’ютері (екран розміром 1536 × 722) та телефоні (екран розміром 412 × 914) відповідно.

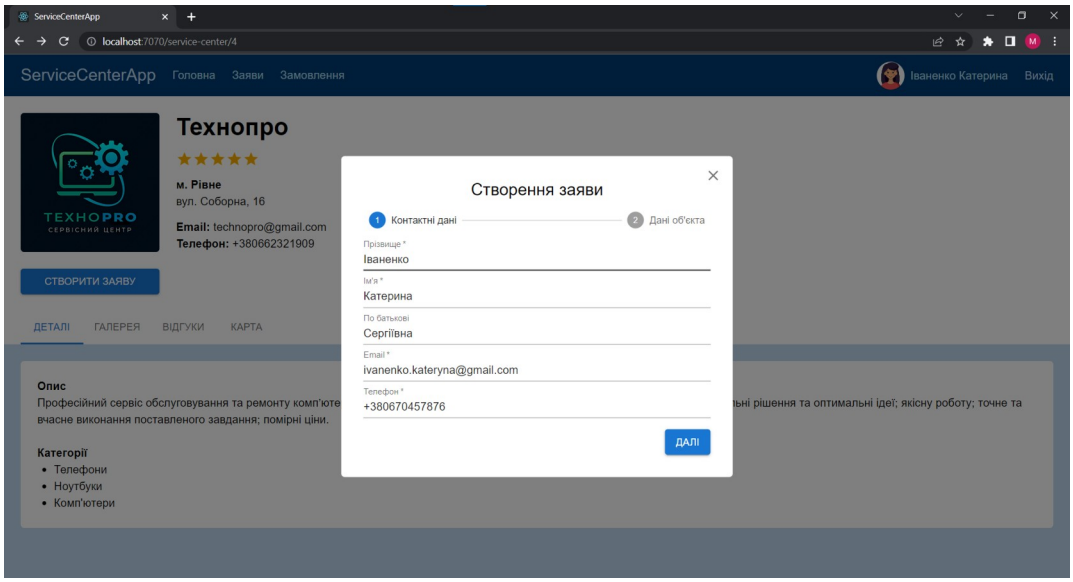


Рис. 4.2. Форма створення заяви на комп’ютері

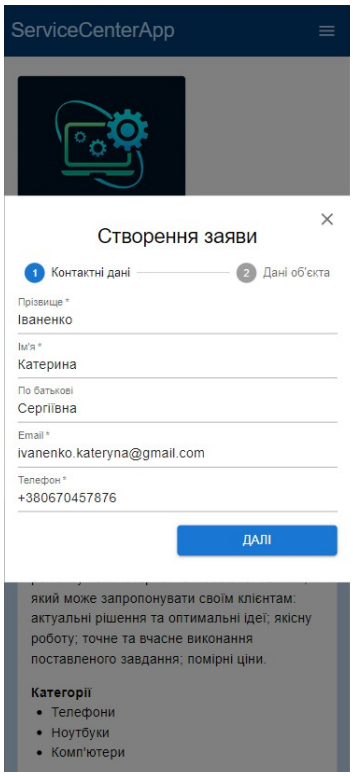


Рис. 4.3. Форма створення заяви на телефоні

4.2.4. *Клієнт-серверна взаємодія*

Важливим етапом при розробці клієнтської частини вебзастосунку є забезпечення можливості взаємодії з сервером, що дозволяє відображати на сторінці отримані з сервера дані, які зберігаються у базі даних. Також аутентифікація – перевірка прав користувача на перегляд певних сторінок та даних, забезпечується через надсилання токена на сервер. Заповнення різноманітних форм у вебзастосунку також передбачає надсилання даних на обробку сервером та у разі успіху додавання їх у базу даних для можливості подальшого їх перегляду та керування ними за допомогою користувацького інтерфейсу програми.

Для реалізації взаємодії з сервером у проєкті було створено модуль, який містить перелік методів для надсилання асинхронних HTTP запитів на сервер та отримання відповідей з нього за допомогою бібліотеки “axios”. Кожен з цих методів приймає об’єкт, який містить перелік вхідних даних, що мають бути переданими у запиті.

Для надсилання запиту вказується потрібна адреса (одна з тих, що були визначені маршрутизацією на стороні сервера), відповідний метод запиту, наприклад post або get, а також заголовки запиту. До заголовків, що використовувались у даній роботі, належать “Authorization”, який приймає як значення токен, та “Content-type”, призначений для визначення типу переданих даних (“application/json” – для даних у форматі JSON, “multipart/form-data” дозволяє завантажувати файли).

Після опрацювання запиту на сервері, клієнт отримує відповідь із потрібними даними, які потім відображаються на вебсторінках, або повідомлення про помилку та її причину.

Окрім виконання запитів на сервер, клієнт-серверна взаємодія передбачає побудову усього react-проєкту, під час якої в кореневій папці проєкту створюється папка “build”, яка містить у собі згенеровані файли на основі написаного коду, а також статичні файли проєкту. Файли з цієї папки передаються браузеру сервером при рендерингу вебсторінок.

4.2.5. *Опис вебсторінок застосунку*

Опис графічного інтерфейсу розроблених вебсторінок:

- Вебсторінка “Список сервісних центрів” є головною і відображає список усіх сервісних центрів, що були створені у застосунку, а також перелік фільтрів, які можуть бути для них застосовані. Вона доступна для перегляду всім користувачам.

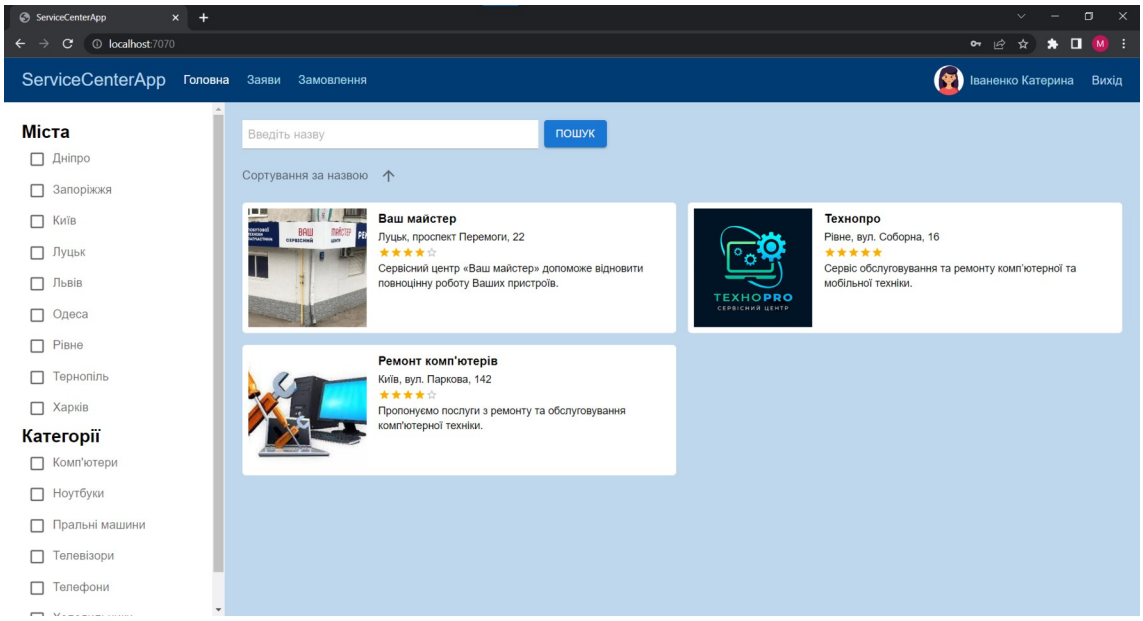


Рис. 4.4. Вебсторінка списку сервісних центрів

- Вебсторінка “Профіль” містить дані користувача, є доступною для всіх авторизованих користувачів. Усі авторизовані користувачі окрім працівників мають можливість редагувати дані заповнивши форму.
- Вебсторінка сервісного центру відображає інформацію про сервісний центр відповідно до ідентифікатора. Вона містить вкладки “Деталі”, “Галерея”, “Відгуки” та “Карта”. Сторінка доступна до перегляду всім користувачам застосунку.

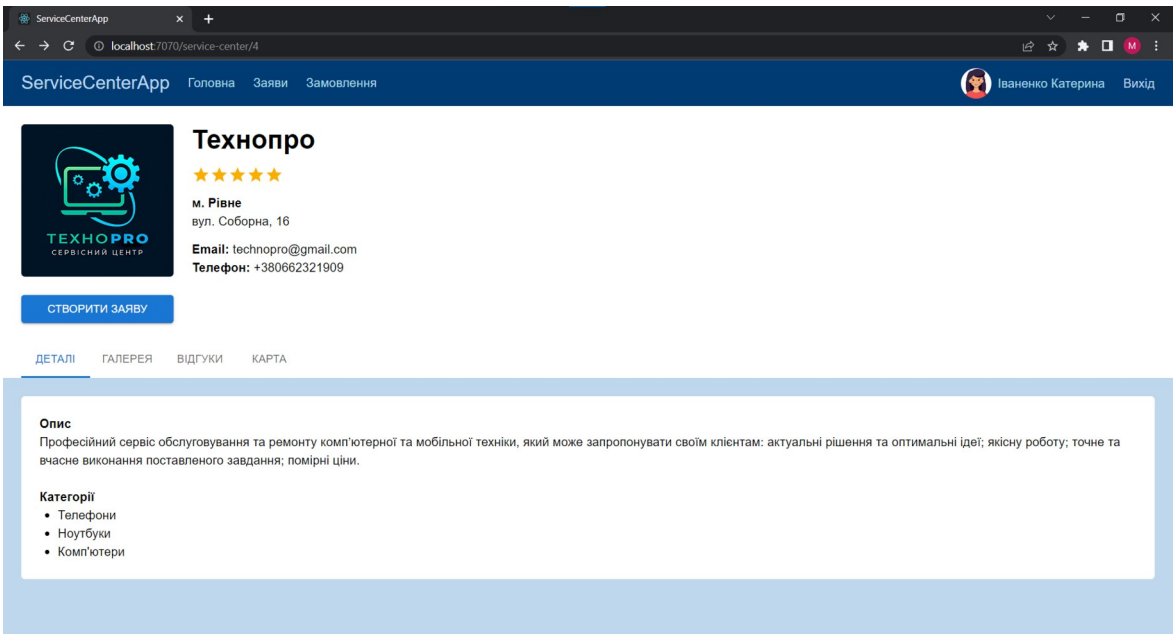


Рис. 4.5. Вебсторінка сервісного центру. Вкладка “Деталі”

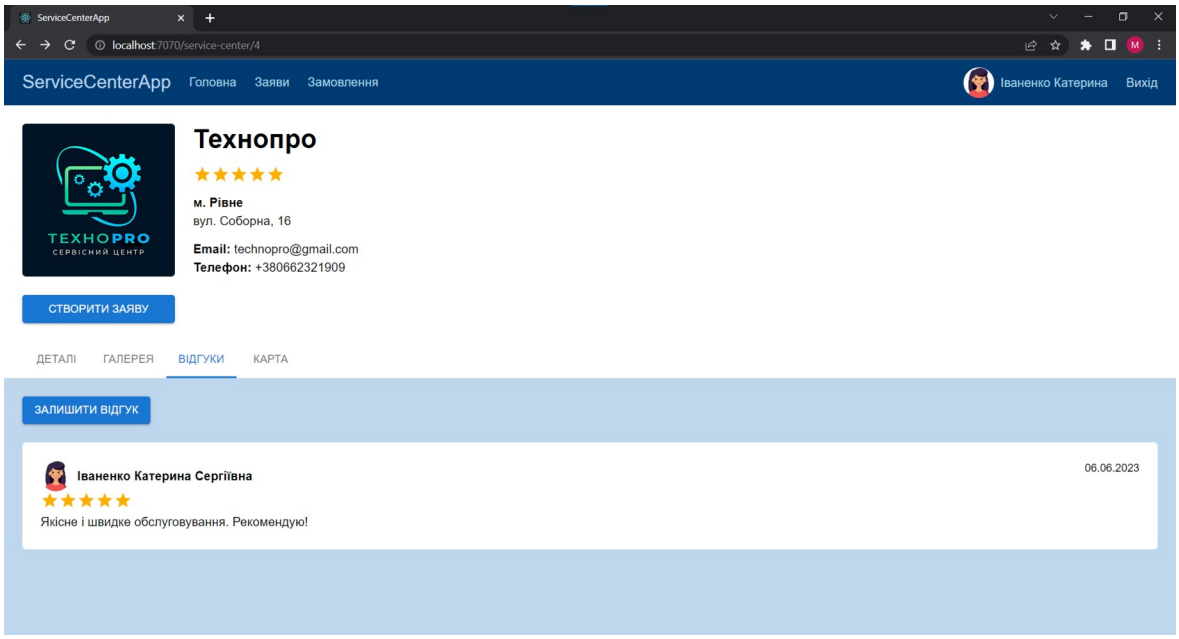


Рис. 4.6. Вебсторінка сервісного центру. Вкладка “Відгуки”

- Вебсторінка “Керування сервісним центром” відображає компоненти для керування заявами, замовленнями, завданнями та даними працівників сервісного центру (рис. 4.7). Вона доступна для адміністратора та працівників сервісного центру. Якщо сервісного центру ще не існує , то адміністратор може його створити заповнивши відповідну форму (рис. 4.8).

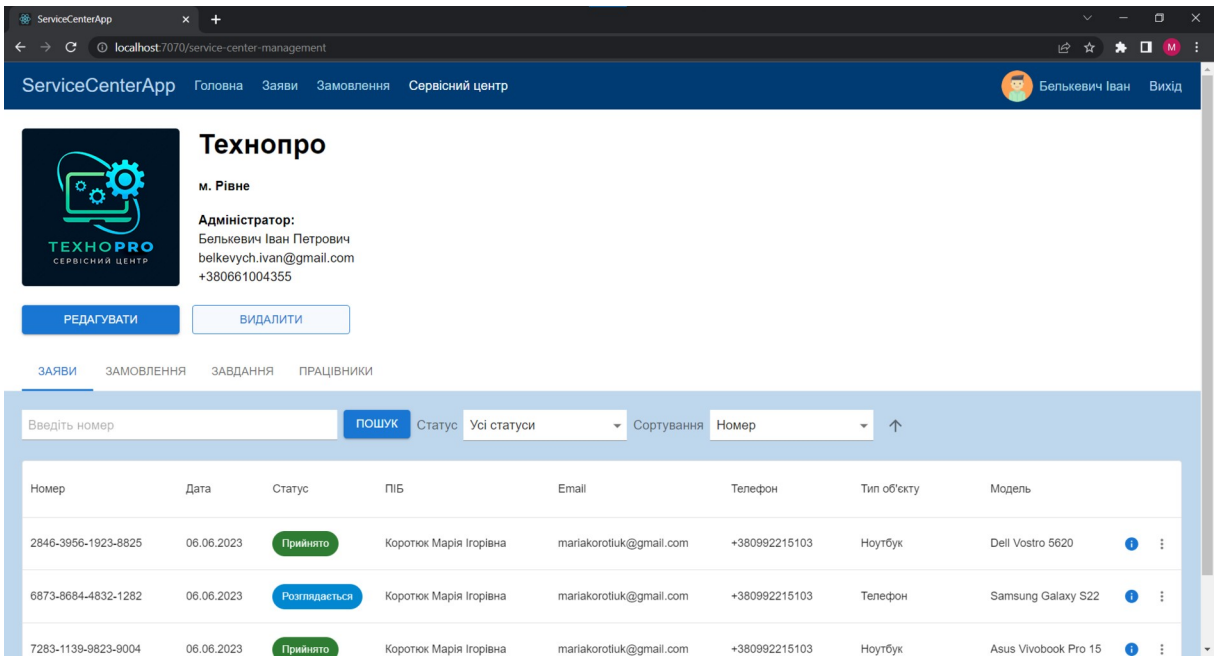


Рис 4.7. Вебсторінка “Керування сервісним центром”. Вкладка “Заяви

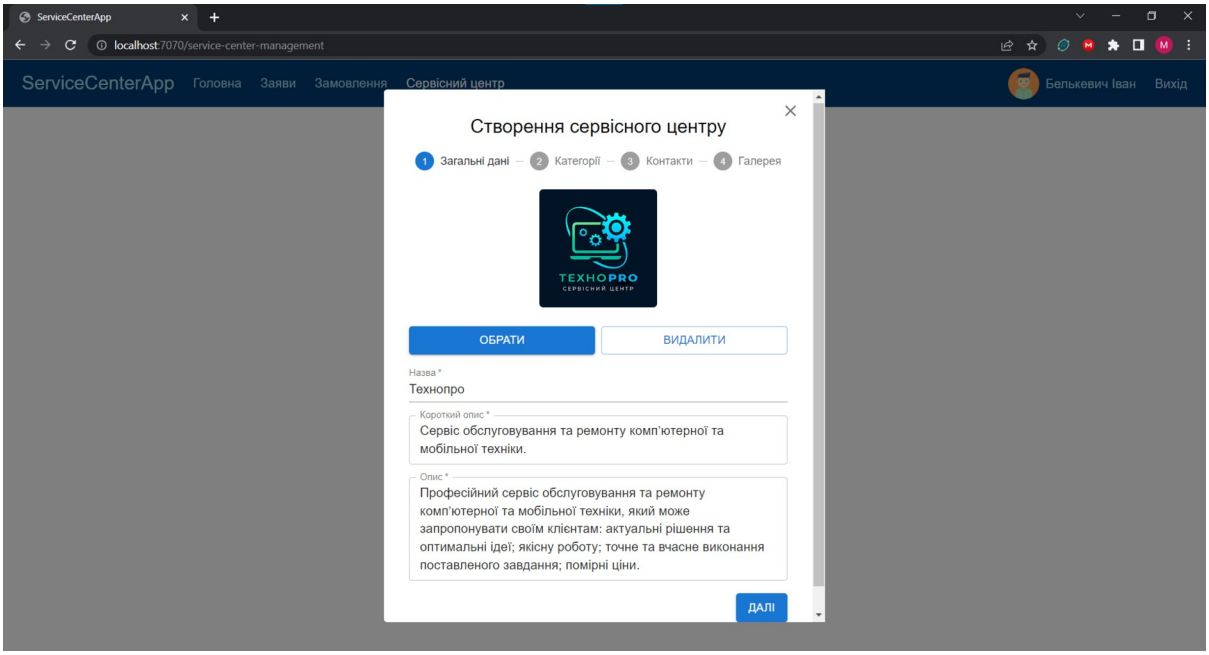


Рис. 4.8. Форма створення сервісного центру

- Вебсторінка “Реєстрація” містить форму для реєстрації користувача у ролі адміністратора або клієнта сервісного центру і є доступною для усіх користувачів застосунку.

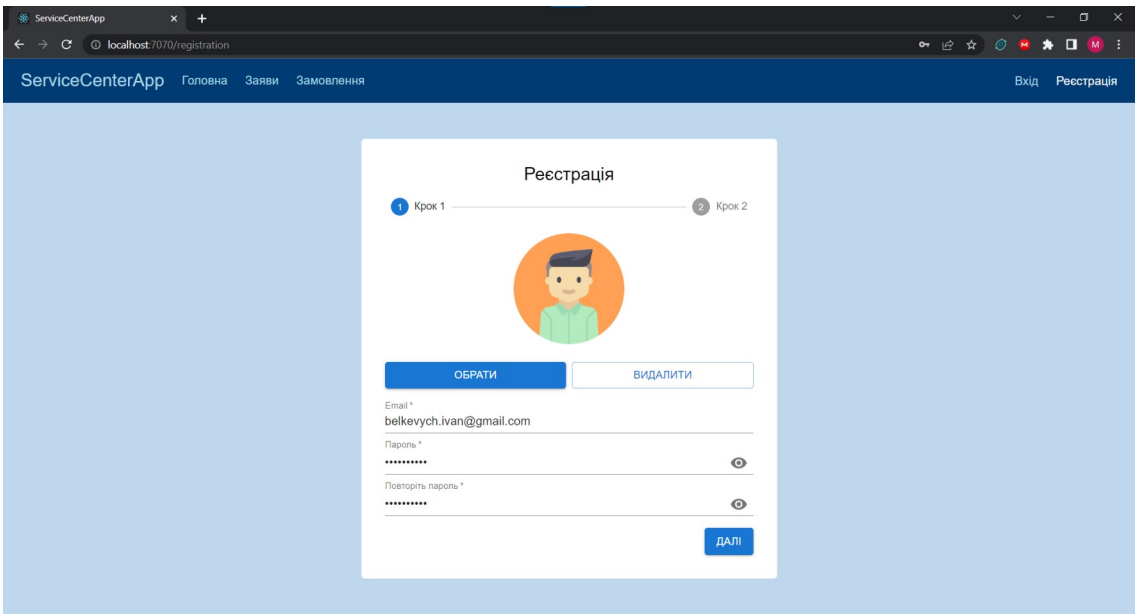


Рис. 4.9. Вебсторінка реєстрації

- Вебсторінка “Логін” містить форму для входу користувача в обліковий запис і є доступною для усіх користувачів застосунку.

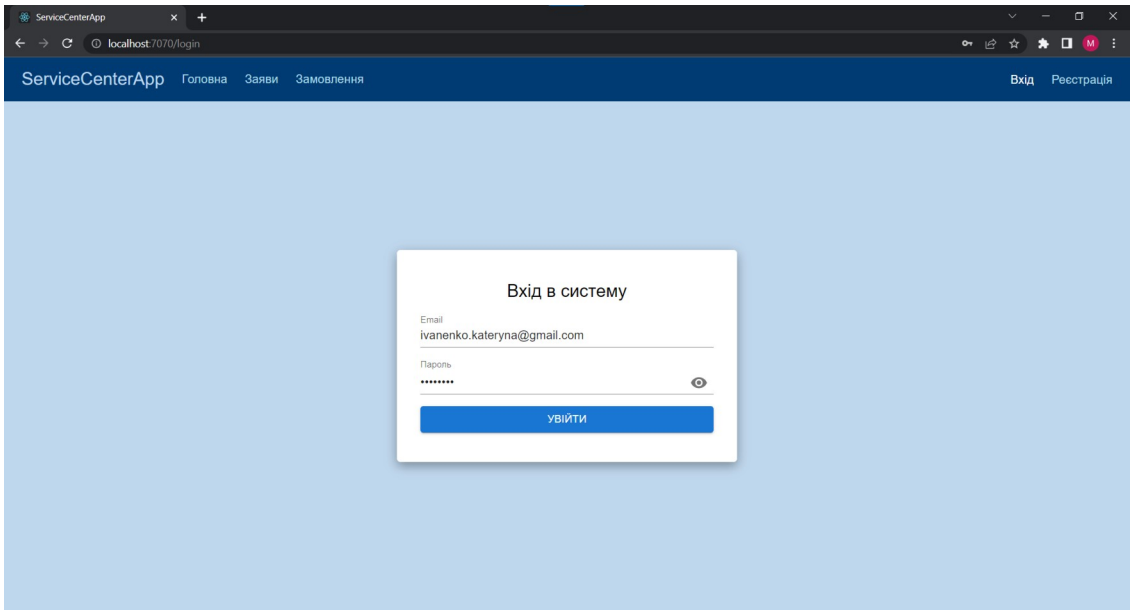


Рис. 4.10. Вебсторінка входу в обліковий запис

- Вебсторінка “Замовлення” містить пошукове для перевірки замовлення за номером, є доступною для всіх користувачів. Клієнти можуть також бачити історію своїх замовлень.
- Вебсторінка “Заяви” містить пошукове поле для перевірки заяви

за номером, є доступною для всіх користувачів. Клієнти можуть також бачити історію своїх заяв.

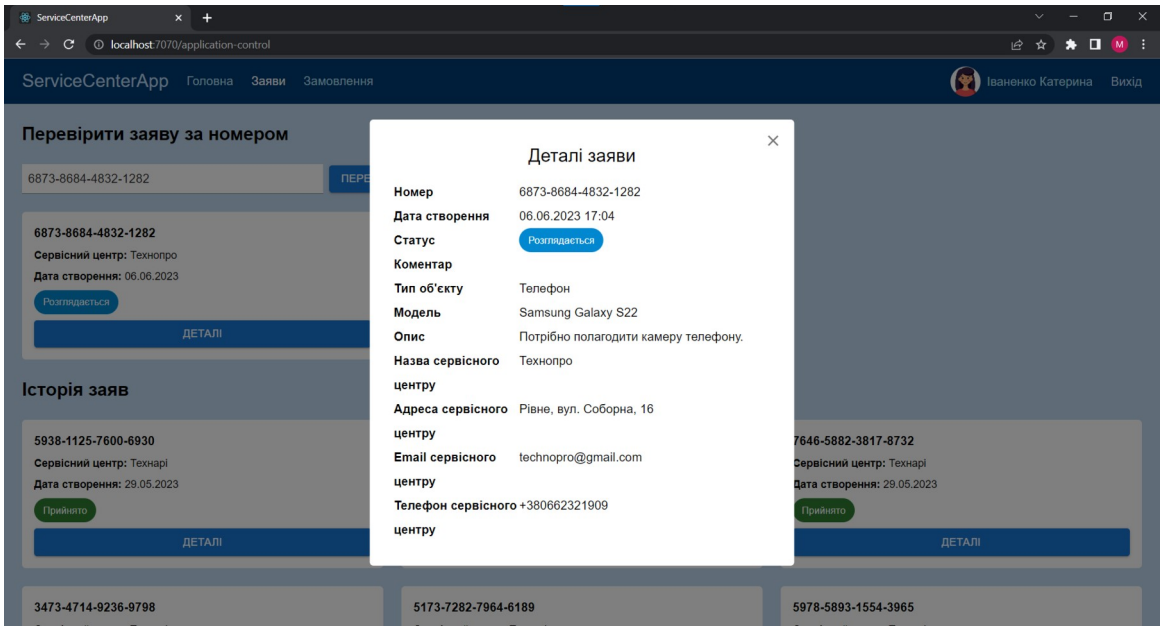


Рис 4.11. Вебсторінка “Заяви”

4.3. Тестування програмного забезпечення

Було розроблено тест-кейси, які знаходиться в табл. 4.2, для перевірки правильності роботи основних функцій вебзастосунку методом димового тестування.

Таблиця 4.2

Тест-кейси для тестування вебзастосунку

№	Код вимоги	Дії	Очікуваний результат
1	F-1	Відкрити вебзастосунок у браузері за посиланням.	Відкривається головна сторінка зі списком сервісних центрів.
2	F-4	1. Відкрити головну сторінку зі списком сервісних центрів. 2. Натиснути на сервісний центр у списку.	Відкриється сторінка обраного сервісного центру.

Продовження табл. 4.2

3	F-6	1. Відкрити головну сторінку зі списком сервісних центрів. 2. Натиснути на сервісний центр у списку. 3. На сторінці сервісного центру натиснути кнопку “Створити заяву”. 4. Заповнити усі обов’язкові поля форми коректними даними. 5. У формі натиснути кнопку “Створити”.	Заяву створено, на вказану в формі електронну пошту надсилається повідомлення з номером заяви.
4	F-7	1. Відкрити сторінку “Заяви”. 2. У пошуковому полі ввести номер існуючої заяви у правильному форматі. 3. Натиснути кнопку “Пошук”.	З’являється результат пошуку, який містить дані заяви, для якої здійснювався пошук.
5	F-8	1. Відкрити сторінку “Замовлення”. 2. У пошуковому полі ввести номер існуючого замовлення у правильному форматі. 3. Натиснути кнопку “Пошук”.	З’являється результат пошуку, який містить дані замовлення, для якого здійснювався пошук.
6	F-9	1. Відкрити сторінку реєстрації. 2. Заповнити усі обов’язкові поля форми коректними даними. 3. У формі натиснути кнопку “Зареєструватися”.	Користувача зареєстровано у застосунку відповідно до введених даних у формі та перенаправлено на сторінку входу.
7	F-10	1. Відкрити сторінку входу в обліковий запис. 2. Ввести коректні email та пароль. 3. У формі натиснути кнопку “Увійти”.	Користувача авторизовано у застосунку відповідно до введених даних.
8	F-12	1. Авторизуватися у застосунку. 2. Натиснути в меню “Вихід”.	Користувач вийшов з облікового запису.

Продовження табл. 4.2

9	F-15	1. Авторизуватися у застосунку у ролі адміністратора, для якого ще не створений сервісний центр. 2. Перейти на сторінку керування сервісним центром (в меню кнопка “Сервісний центр”). 3. Заповнити усі обов’язкові поля форми коректними даними. 4. Натиснути кнопку “Створити”.	Сервісний центр створено і дані сервісного центру відображаються на сторінці керування.
10	F-18	1. Авторизуватися у застосунку у ролі адміністратора, для якого створено сервісний центр. 2. Перейти на сторінку керування сервісним центром (в меню кнопка “Сервісний центр”). 3. Перейти на вкладку “Працівники”. 4. Натиснути кнопку “Зареєструвати працівника”. 5. Заповнити усі обов’язкові поля форми коректними даними. 6. Натиснути кнопку “Зареєструвати”.	Працівника зареєстровано у застосунку, на вказану в формі електронну пошту надсилається лист з вказаним згенерованим паролем доступу в обліковий запис.
11	F-21	1. Авторизуватися у застосунку у ролі адміністратора, для якого створено сервісний центр. 2. Перейти на сторінку керування сервісним центром (в меню кнопка “Сервісний центр”). 3. Перейти на вкладку “Заяви”. 4. Для потрібної заяви натиснути кнопку “Редагувати”. 5. Заповнити усі обов’язкові поля форми коректними даними. 6. Натиснути кнопку “Зберегти”.	Заяву редаговано, оновлені дані відображаються на сторінці у списку заяв.

Продовження табл. 4.2

12	F-23	1. Авторизуватися у застосунку у ролі адміністратора, для якого створено сервісний центр. 2. Перейти на сторінку керування сервісним центром (в меню кнопка “Сервісний центр”). 3. Перейти на вкладку “Замовлення”. 4. Натиснути кнопку “Створити замовлення”. 5. Заповнити усі обов’язкові поля форми коректними даними. 6. Натиснути кнопку “Створити”.	Замовлення створено, на вказану в формі електронну пошту надсилається повідомлення з номером замовлення.
13	F-24	1. Авторизуватися у застосунку у ролі адміністратора, для якого створено сервісний центр. 2. Перейти на сторінку керування сервісним центром (в меню кнопка “Сервісний центр”). 3. Перейти на вкладку “Замовлення”. 4. Для потрібної заяви, яка має статус “Прийнято”, натиснути кнопку “Створити замовлення”. 5. Заповнити усі обов’язкові поля форми коректними даними. 6. Натиснути кнопку “Створити”.	Замовлення створено, на вказану в формі електронну пошту надсилається повідомлення з номером замовлення.
14	F-27	1. Авторизуватися у застосунку у ролі адміністратора, для якого створено сервісний центр. 2. Перейти на сторінку керування сервісним центром (в меню кнопка “Сервісний центр”). 3. Перейти на вкладку “Завдання”. 4. Натиснути кнопку “Створити завдання” на сторінці або на потрібному замовленні. 5. Заповнити усі обов’язкові поля форми коректними даними. 6. Натиснути кнопку “Створити”.	Завдання створено.

Продовження табл. 4.2

15	F-33	1. Авторизуватися у застосунку у ролі працівника. 2. Перейти на сторінку керування сервісним центром (в меню кнопка “Сервісний центр”). 3. Перейти на вкладку “Завдання”. 4. Натиснути кнопку “Редагувати завдання” на потрібному завданні. 5. Заповнити усі обов’язкові поля форми коректними даними. 6. Натиснути кнопку “Зберегти”.	Завдання редаговано, оновлені дані відображаються на сторінці у списку завдань.

4.4. Напрямки подальшої реалізації

Багато сервісних центрів мають не одне відділення, а декілька, тому буде корисно створити у вебзастосунку можливість для додавання декількох адрес одного сервісного центру і відображати їх усі на одній карті. Таким чином клієнти зможуть швидше знайти сервісний центр з найбільш зручним місцерозташуванням.

Також важливо буде реалізувати додавання розташування сервісного центру на карті лише за його адресою – без задавання точних координат. Це буде значно зручніше для адміністраторів, тому що їм достатньо буде знати адресу і не потрібно буде додатково дізнаватись координати організації.

Вебзастосунок було розроблено адаптивним, тому він коректно відображається на пристроях з різним розміром екрану, зокрема і на телефонах. Проте багато користувачів надають перевагу мобільним застосункам, тому необхідно розробити і такий варіант програми. Окрім того користувачі зможуть отримувати сповіщення відразу у мобільному застосунку, наприклад про створення замовлення, і їм не потрібно буде перевіряти електронну пошту.

Також корисною функцією для подальшої реалізації є онлайн-чат з

працівниками сервісного центру, що дозволить користувачам поставити запитання без необхідності телефонувати чи очікування на результат розгляду заяви. Адміністратори та працівники також зможуть писати повідомлення клієнтам після того, як вони надіслали заяву в їх сервісний центр. Усі листування будуть збережені в облікових записах користувачів.

4.5. Висновки до четвертого розділу

У четвертому розділі було охарактеризовано особливості розробки програмного забезпечення. Було описано реалізацію серверної частини вебзастосунку із зазначенням використаних бібліотек.

Також було пояснено, які дії виконувались при створенні клієнтської частини вебзастосунку, а саме визначення маршрутизації на стороні клієнта, створення компонентів графічного інтерфейсу, їх стилізація та забезпечення адаптивності для коректного відображення на різних типах пристроїв. Крім того було описано графічний інтерфейс розроблених вебсторінок та реалізацію взаємодії клієнта і сервера, що забезпечую цілісну роботу застосунку.

Крім того було складено та виконано план тестування розробленого програмного забезпечення, а саме було використано спосіб димового тестування.

Також було запропоновано напрямки подальшої реалізації вебзастосунку.

ВИСНОВКИ

При виконанні дипломного проєкту було розроблено вебзастосунок для підтримки діяльності сервісних центрів, що передбачає вибір сервісного центру та автоматизацію процесів отримання послуг та контролю етапів їх виконання.

Функціональні та нефункціональні вимоги до роботи застосунку, які дозволять вирішити поставлені задачі, було визначено після аналізу переваг та недоліків вже існуючого програмного забезпечення. Було визначено структуру програмного забезпечення, а саме клієнт-серверну, та обрано технології, які найкраще підходять для розробки програми такого типу. Також чітко визначивши, яка інформація має зберігатися у застосунку, було розроблено структуру бази даних, а саме спроектовано таблиці та зв’язки між ними. Після цього було розроблено програмне забезпечення згідно з визначеними вимогами. Окрім того було запропоновано напрямки подальшої реалізації, що дозволить зробити застосунок зручнішим для користувачів.

Розроблений вебзастосунок відповідає вимогам, описаним у Технічному завданні, та правильність його роботи було перевірено за допомогою тестування програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

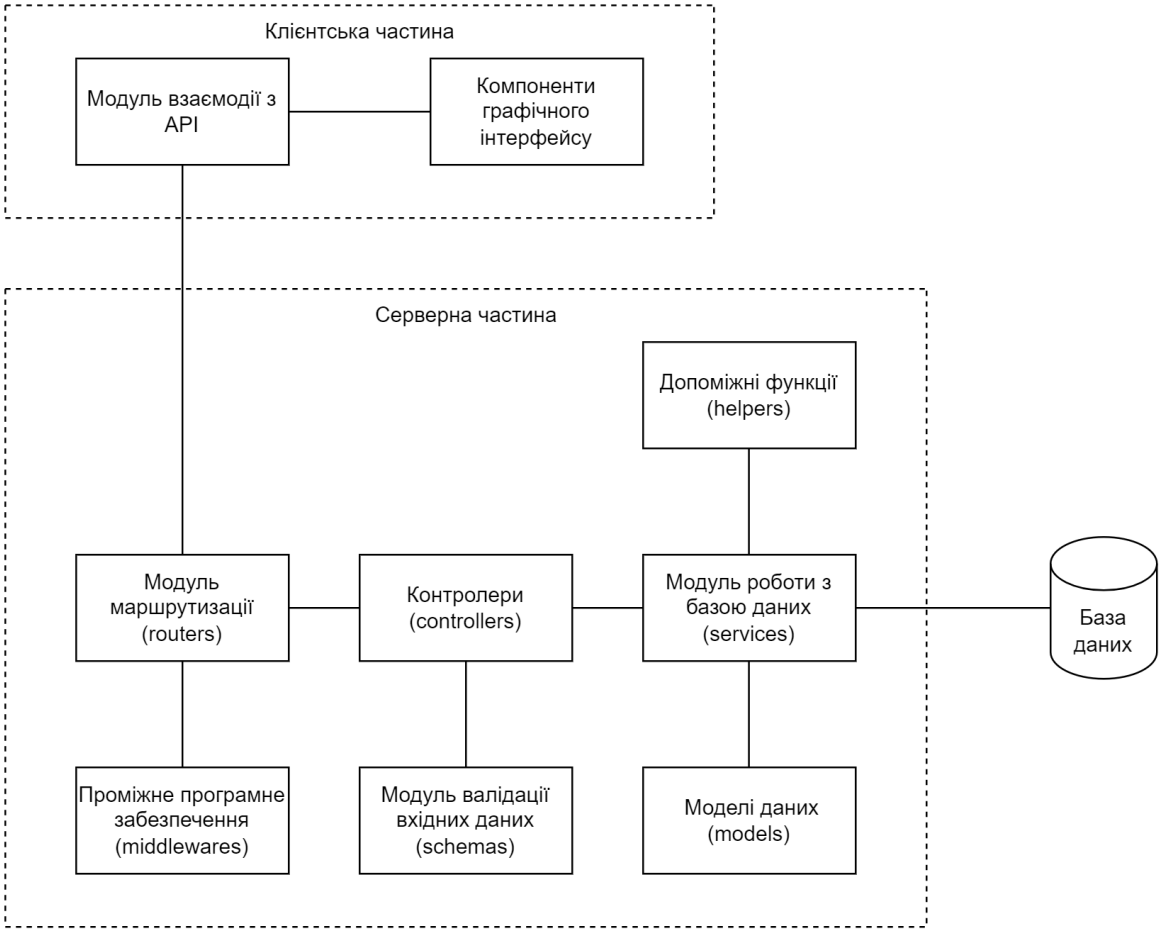
1. List.in.ua [Електронний ресурс] — Режим доступу до ресурсу: https://list.in.ua/.
2. OLX [Електронний ресурс] — Режим доступу до ресурсу: https://www.olx.ua/.
3. Мережа сервісних центрів Технарі [Електронний ресурс] — Режим доступу до ресурсу: https://technari.com.ua/ua/.
4. Офіційні сервісні центри Bosch [Електронний ресурс] — Режим доступу до ресурсу: https://www.bosch-home.com.ua/uk/servis/.
5. Офіційний сервіс Samsung [Електронний ресурс] — Режим доступу до ресурсу: https://www.samsung.com/ua/support/contact/.
6. Мережа сервісних центрів Технарі [Електронний ресурс] — Режим доступу до ресурсу: https://service.muk.ua/.
7. JavaScript [Електронний ресурс] — Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript.
8. C# documentation [Електронний ресурс] — Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/csharp/.
9. About Node.js [Електронний ресурс] — Режим доступу до ресурсу: https://nodejs.org/en/about.
10. Express [Електронний ресурс] — Режим доступу до ресурсу: https://expressjs.com/.
11. Nest [Електронний ресурс] — Режим доступу до ресурсу: https://docs.nestjs.com/.
12. React [Електронний ресурс] — Режим доступу до ресурсу: https://react.dev/.
13. Angular features [Електронний ресурс] — Режим доступу до ресурсу: https://angular.io/features.
14. Vue.js [Електронний ресурс] — Режим доступу до ресурсу: https://vuejs.org/.

15. TypeScript [Електронний ресурс] — Режим доступу до ресурсу: https://www.typescriptlang.org/.
16. Sass Basics [Електронний ресурс] — Режим доступу до ресурсу: https://sass-lang.com/guide.
17. MongoDB [Електронний ресурс] — Режим доступу до ресурсу: https://www.mongodb.com/.
18. PostgreSQL [Електронний ресурс] — Режим доступу до ресурсу: https://www.postgresql.org/.

ДОДАТКИ

Додаток 1

Копії графічних матеріалів

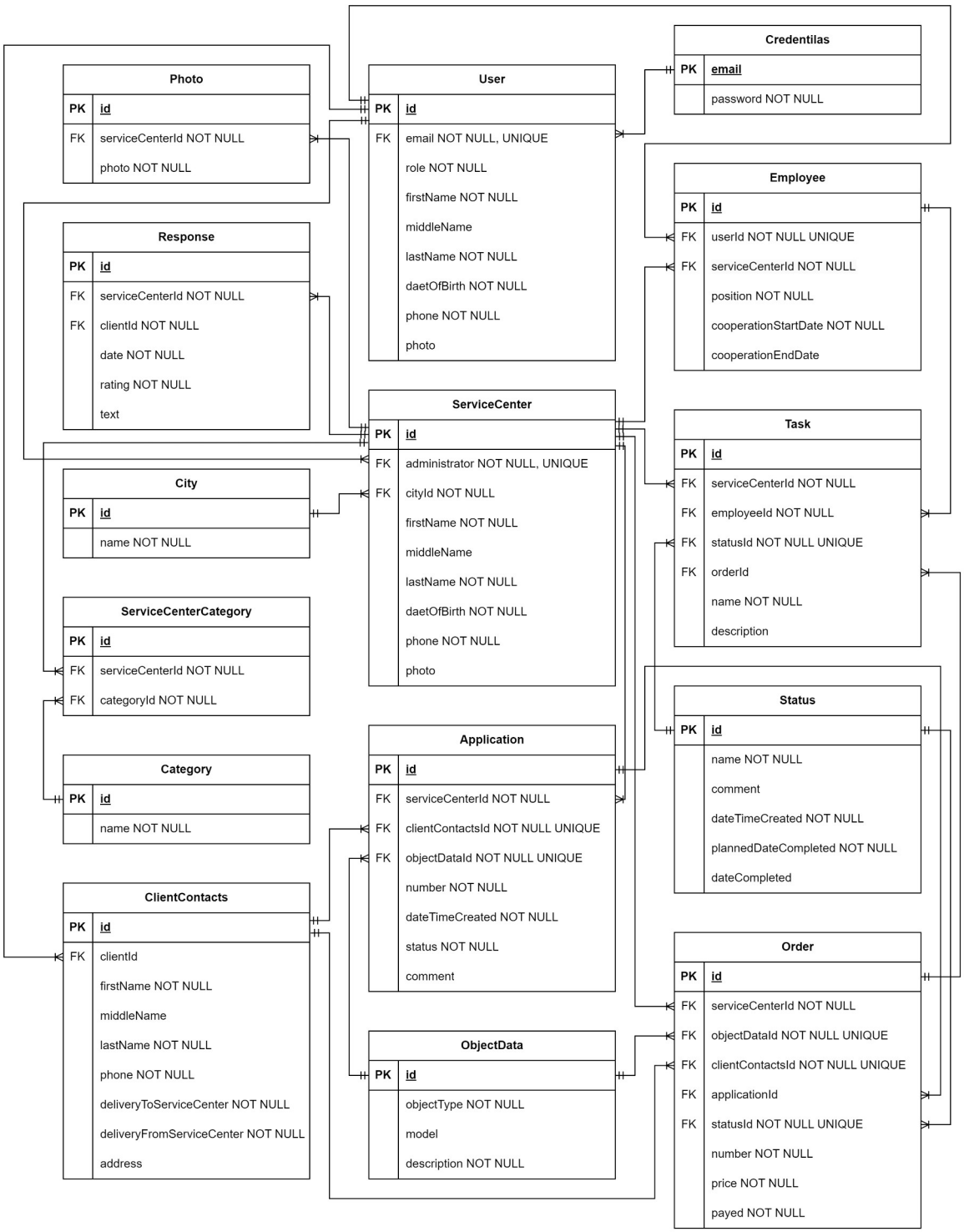


ДП.045440-06-99

Вебзастосунок для підтримки діяльності сервісних центрів.

Загальна структура застосунку.

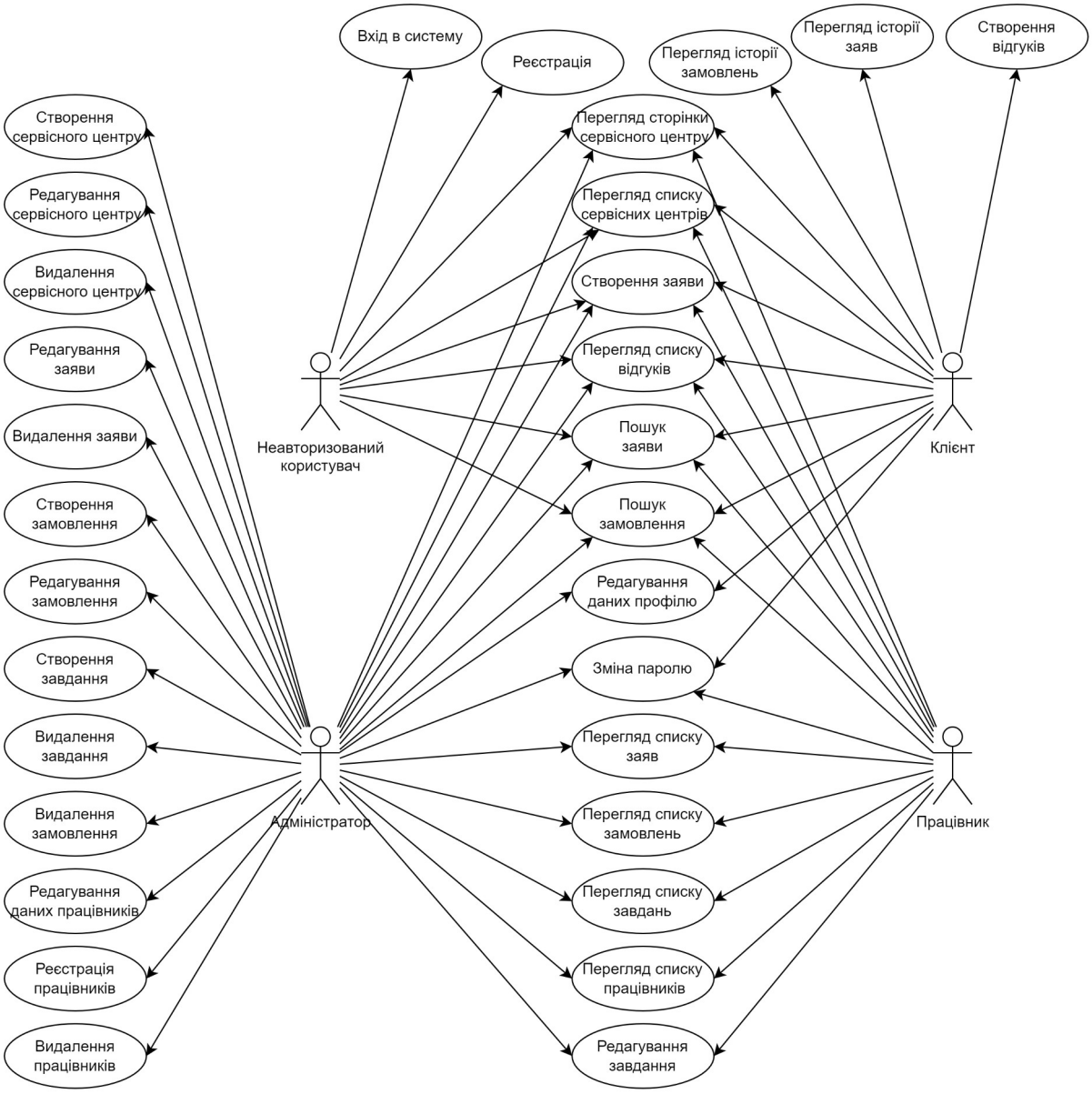
UML-діаграма



ДП.045440-07-99

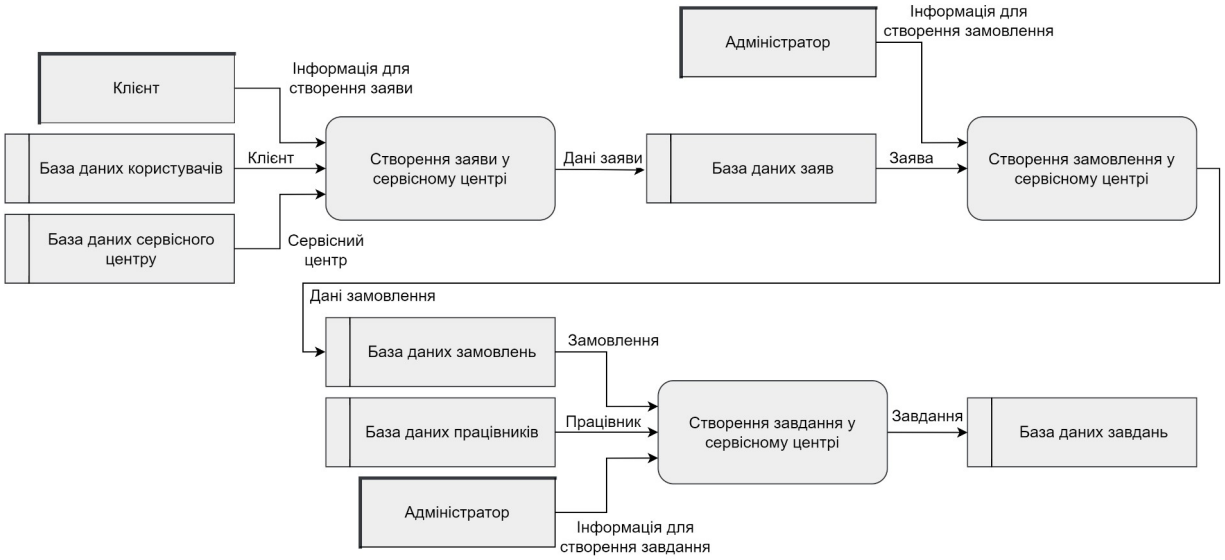
Вебзастосунок для підтримки діяльності сервісних центрів.

Схема бази даних. ER-діаграма



Діаграма варіантів використання

Коротюк М.І., група КП-93



Діаграма потоку даних. Процес створення завдання на основі заяви клієнта

Коротюк М.І., група КП-93

Додаток 2

Лістинг програми

Модуль маршрутизації

```
const router = require('express').Router();
const userRouter = require('./user-router');
const cityRouter = require('./city-router');
const taskRouter = require('./task-router');
const photoRouter = require('./photo-router');
const orderRouter = require('./order-router');
const categoryRouter = require('./category-router');
const responseRouter = require('./response-router');
const employeeRouter = require('./employee-router');
const applicationRouter = require('./application-router');
const serviceCenterRouter = require('./service-center-router');
const authenticationRouter = require('./authentication-router');

router.use('/user', userRouter);
router.use('/city', cityRouter);
router.use('/task', taskRouter);
router.use('/photo', photoRouter);
router.use('/order', orderRouter);
router.use('/category', categoryRouter);
router.use('/response', responseRouter);
router.use('/employee', employeeRouter);
router.use('/application', applicationRouter);
router.use('/serviceCenter', serviceCenterRouter);
router.use('/authentication', authenticationRouter);

module.exports = router;
```

Модуль маршрутизації працівників

```
const router = require('express').Router();
const employeeController = require('./../controllers/employee-controller');
const { administratorMiddleware } = require('../middelwares/administrator-
middleware');
const { administratorEmployeeMiddleware } = require('../middelwares/administrator-
employee-middleware');

router.get('/list', administratorEmployeeMiddleware,
employeeController.listEmployees);
router.post('/create', administratorMiddleware, employeeController.createEmployee);
router.put('/update/:id', administratorMiddleware, employeeController.updateEmployee);
router.delete('/delete/:id', administratorMiddleware,
employeeController.deleteEmployee);

module.exports = router;
```

Модель заяви

```
const { Sequelize, DataTypes } = require("sequelize");
const ObjectData = require('./object-data-model');
const ServiceCenter = require('./service-center-model');
const ClientContacts = require('./client-contacts-model');
const DbHelper = require('../helpers/db-helper');

const sequelize = DbHelper.getSequalize();

const Application = sequelize.define("application", {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true
  },
  number: {
    type: DataTypes.TEXT,
    allowNull: false
  },
  dateTimeCreated: {
    type: 'TIMESTAMP',
    defaultValue: Sequelize.literal('CURRENT_TIMESTAMP'),
    allowNull: false
  },
  status: {
    type: DataTypes.ENUM('considered', 'accepted', 'refused'),
    defaultValue: "considered",
    allowNull: false
  },
  comment: {
    type: DataTypes.TEXT,
    allowNull: true
  },
});

ObjectData.hasOne(Application, {
  foreignKey: {
    name: "objectDataId",
    allowNull: false,
    unique: true
  }
});

ServiceCenter.hasOne(Application, {
  foreignKey: {
    name: "serviceCenterId",
    allowNull: false,
  }
});

ClientContacts.hasOne(Application, {
  foreignKey: {
    name: "clientContactsId",
    allowNull: false,
    unique: true
  }
});

(async () => {
  await sequelize.sync({ alter: true });
})();

module.exports = Application;
```

Контролер. Керування замовленнями

```
const orderService = require('./../services/order-service');
const orderSchemas = require('./../schemas/order-schema');

const createOrder = async (req, res) => {
  try {
    const data = req.body;
    await orderSchemas.createOrderSchema.validateAsync(data);
    const order = await orderService.createOrder(req.user, data);
    res.status(200).send(order);
  } catch (error) {
    res.status(400).send({ "message": error.message });
  }
}
const updateOrder = async (req, res) => {
  try {
    const data = req.body;
    await orderSchemas.updateOrderSchema.validateAsync(data);
    const order = await orderService.updateOrder(req.user, req.params.id, data);
    res.status(200).send(order);
  } catch (error) {
    res.status(400).send({ "message": error.message });
  }
}
const listOrders = async (req, res) => {
  try {
    const orders = await orderService.listOrders(req.user);
    res.status(200).send(orders);
  } catch (error) {
    res.status(400).send({
      "message": error.message
    });
  }
}
const listClientOrders = async (req, res) => {
  try {
    const applications = await orderService.listClientOrders(req.user);
    res.status(200).send(applications);
  } catch (error) {
    res.status(400).send({ "message": error.message });
  }
}
const getOrderByNumber = async (req, res) => {
  try {
    const orders = await orderService.getOrderByNumber(req.params.number);
    res.status(200).send(orders);
  } catch (error) {
    res.status(400).send({ "message": error.message });
  }
}
const deleteOrder = async (req, res) => {
  try {
    await orderService.deleteOrder(req.user, req.params.id);
    res.sendStatus(200);
  } catch (error) {
    res.status(400).send({ "message": error.message });
  }
}

module.exports = {
  listOrders,
  listClientOrders,
  getOrderByNumber,
  createOrder,
  updateOrder,
  deleteOrder
}
```

Модуль взаємодії з базою даних. Керування користувачами

```
const bcrypt = require('bcrypt');
const bcrypt = require('bcrypt');

const User = require('../models/user-model');

const DbHelper = require('../helpers/db-helper');

const getUser = async (data) => {
  await DbHelper.createConnection();

  const user = await User.findOne({ where: { id: data.id } });
  if (!user) {
    await DbHelper.closeConnection();
    throw new Error('Користувача не знайдено');
  }

  await DbHelper.closeConnection();

  if (user.photo) {
    user.photo = user.photo.toString("base64")
  }

  return user;
}

const updateUser = async (user, data, photo) => {
  await DbHelper.createConnection();

  let updatedUser = await User.findOne({ id: user.id });

  if (!updatedUser) {
    await DbHelper.closeConnection();
    throw new Error('Користувача не знайдено');
  }

  if (updatedUser.role === "employee") {
    await DbHelper.closeConnection();
    throw new Error('Користувача з роллю "Працівник" не можна редагувати');
  }

  try {
    updatedUser = await User.update({ ...data, photo: photo?.data || null }, { where:
{ id: user.id } });
  } catch (e) {
    await DbHelper.closeConnection();
    throw new Error(e.message);
  }

  updatedUser = await User.findOne({ where: { id: user.id } });

  await DbHelper.closeConnection();

  if (updatedUser.photo) {
    updatedUser.photo = updatedUser.photo.toString("base64")
  }

  return updatedUser;
}

module.exports = {
  getUser,
  updateUser
};
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ВЕБЗАСТОСУНОК ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ СЕРВІСНИХ ЦЕНТРІВ

Виконала: Коротюк Марія Ігорівна

Керівник: Доцент, к.т.н., доц. Заболотня Т.М.

Київ – 2023

АКТУАЛЬНІСТЬ

1. Кожен хоча б раз користувався послугами сервісних центрів.
2. Проблема вибору сервісного центру клієнтами.
3. Потреба у поширенні інформації про сервісний центр та автоматизації процесів організації його роботи.
4. Потреба у зберіганні даних для клієнтів та працівників сервісних центрів в межах однієї системи.
5. Наявні рішення не мають усіх функцій, що потрібні користувачам.

ПОСТАНОВКА ЗАДАЧІ

Мета проєкту: вирішити проблему вибору сервісного центру клієнтами та керування процесом надання послуг сервісними центрами шляхом розроблення вебзастосунку.

Завдання:

1. Проаналізувати існуючі рішення.
2. Обрати засоби реалізації вебзастосунку.
3. Визначити вимоги до програмного забезпечення.
4. Розробити вебзастосунок згідно з вимогами.
5. Протестувати розроблене програмного забезпечення.
6. Визначити напрямки подальшої реалізації.

АНАЛОГИ

Категорії існуючих рішень:

1. Каталоги закладів та організацій (list.in.ua).
2. Сервіси для розміщення оголошень (OLX).
3. Власні вебсайти сервісних центрів (сервісні центри Samsung, Bosch, “Технарі” та “MUK Service”).

ФУНКЦІОНАЛЬНІ ВИМОГИ ДО РОЗРОБЛЮВАНОГО ЗАСТОСУНКУ

Вебзастосунок надає можливості відповідно до ролі користувача:

1. Реєструватися.
2. Створювати, редагувати та видаляти сторінку сервісного центру.
3. Переглядати список сервісних центрів та сторінку кожного з них.
4. Створювати заяви на отримання послуг.
5. Додавати відгуки.
6. Переглядати історію власних завдань та замовлень.
7. Керувати заявами, замовленнями та завданнями у сервісному центрі.
8. Реєструвати, редагувати та видаляти дані працівників.

СТРУКТУРА РОЗРОБЛЕНОГО ЗАСТОСУНКУ

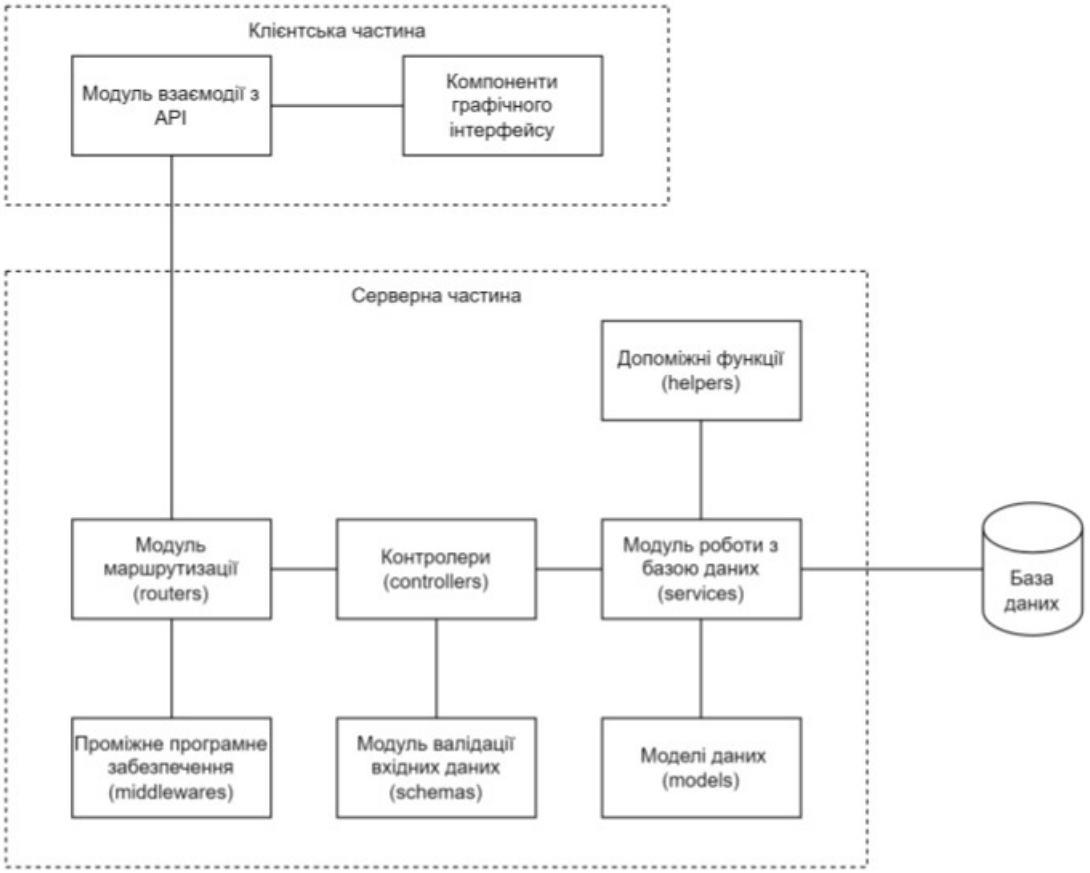
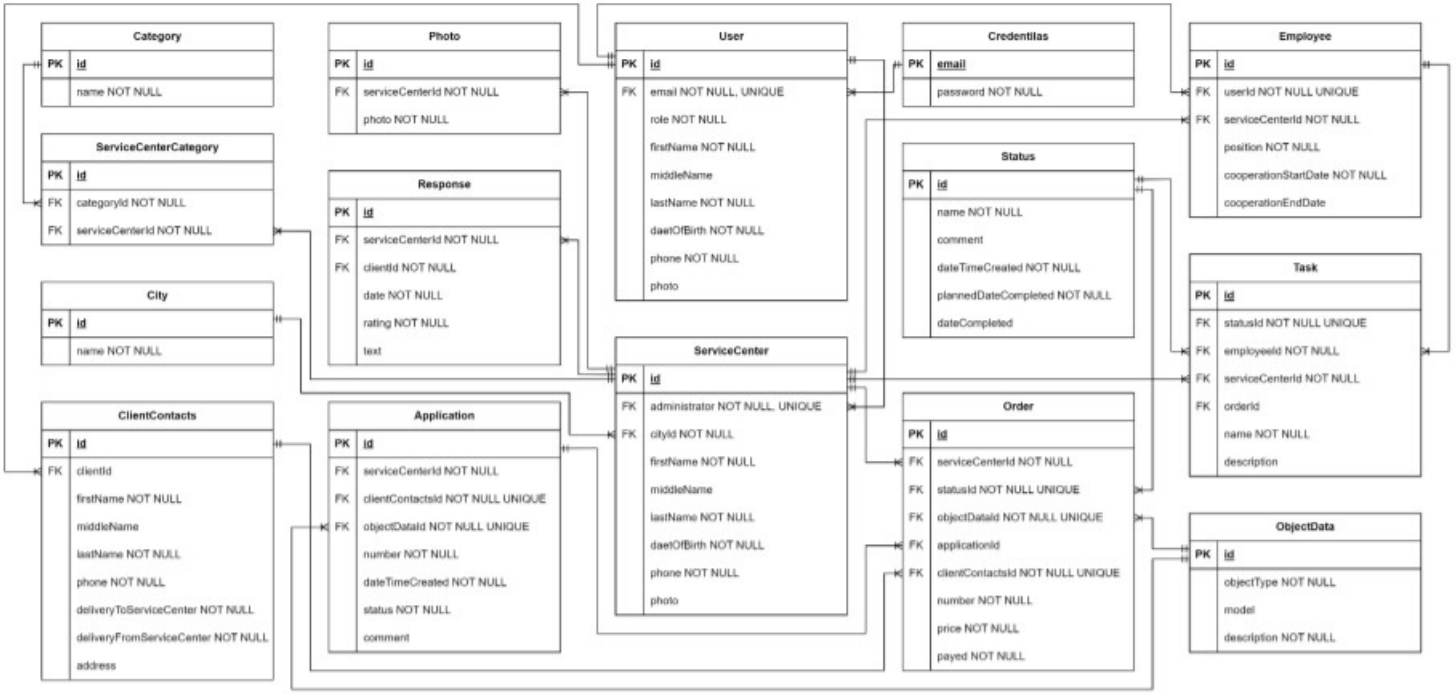
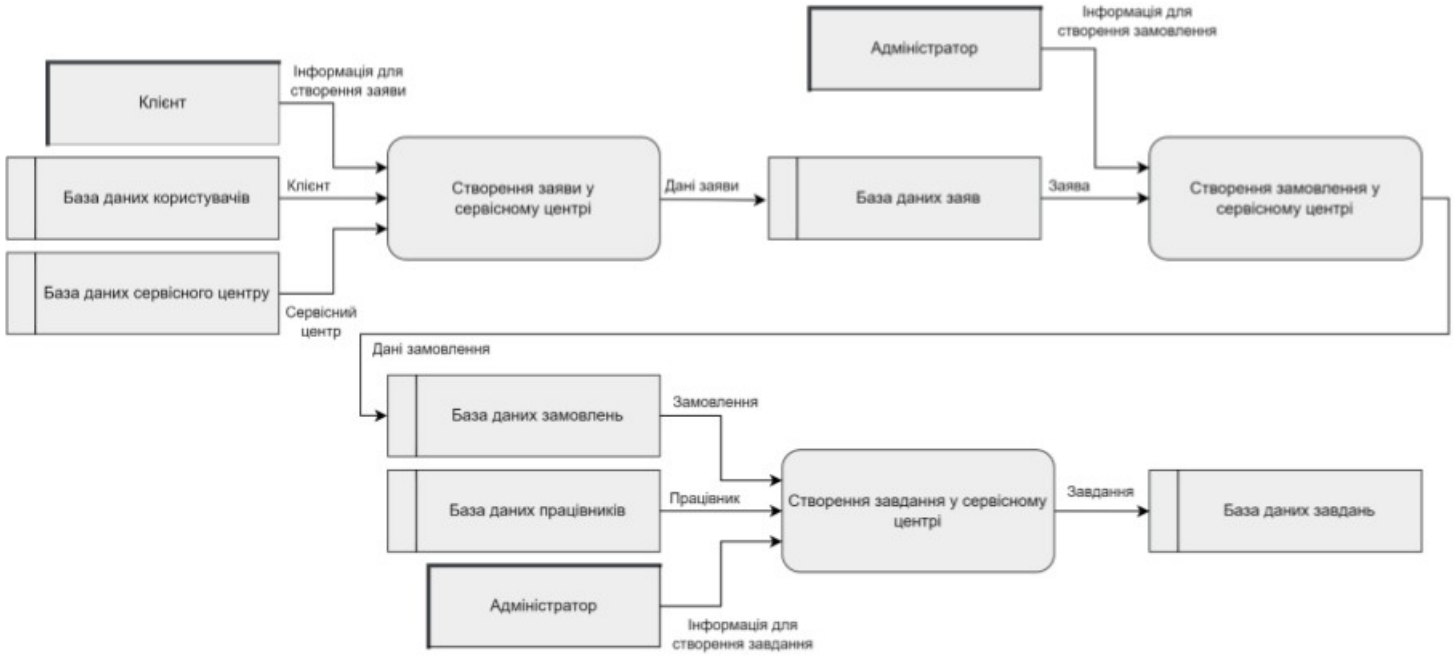


СХЕМА БАЗИ ДАНИХ



ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

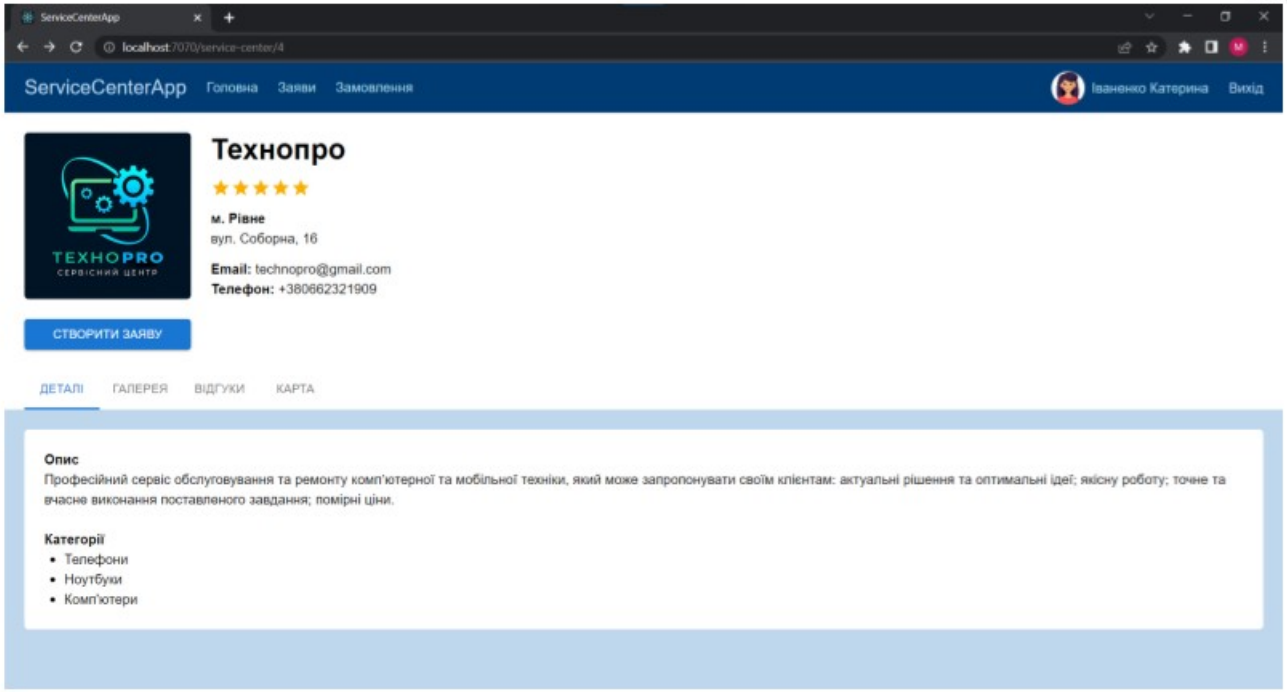
ДІАГРАМА ПОТОКУ ДАНИХ ДЛЯ ПРОЦЕСУ СТВОРЕННЯ ЗАВДАННЯ НА ОСНОВІ ЗАЯВИ КЛІЄНТА



ЗАСОБИ РЕАЛІЗАЦІЇ

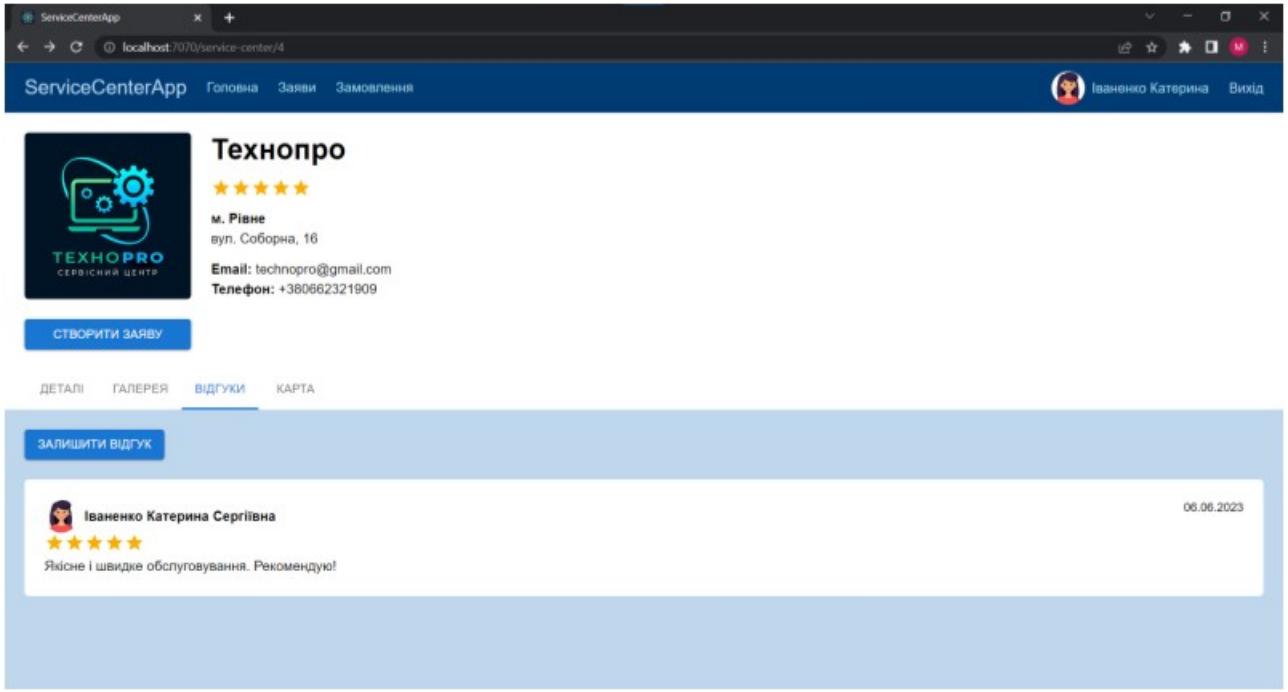
ЗАСОБИ РЕАЛІЗАЦІЇ. БІБЛІОТЕКИ

ІНТЕРФЕЙС КОРИСТУВАЧА



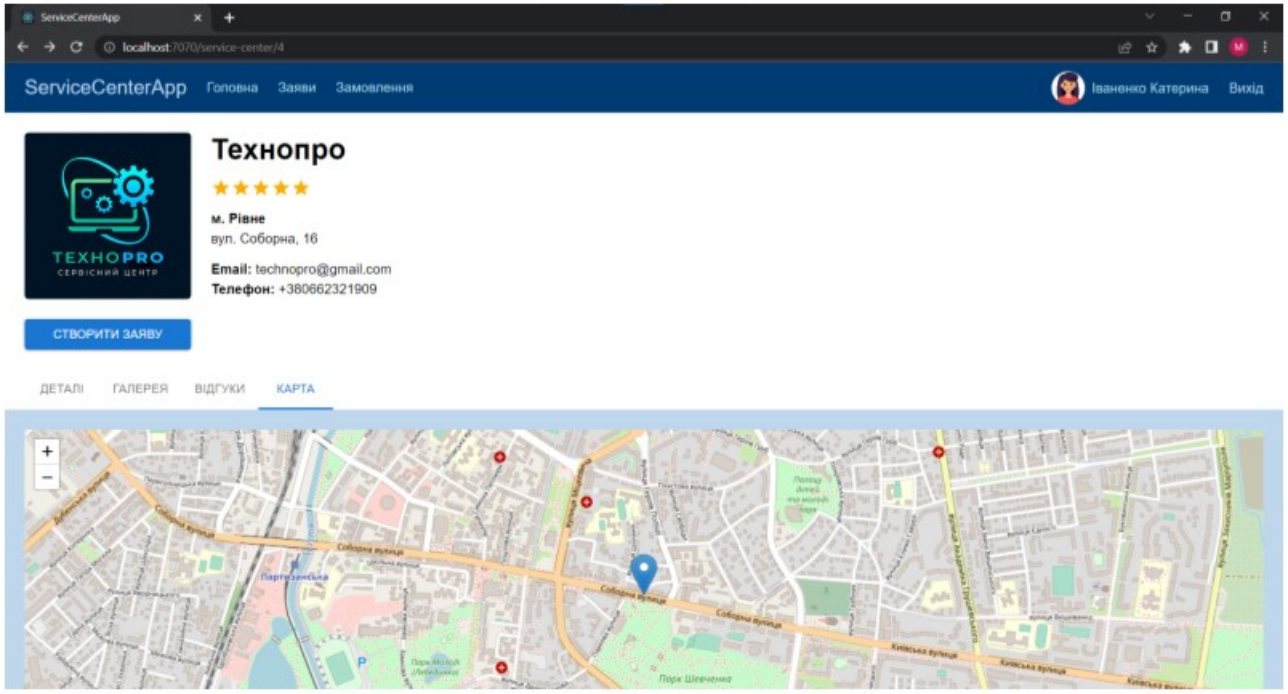
Сторінка сервісного центру. Вкладка “Деталі”

ІНТЕРФЕЙС КОРИСТУВАЧА



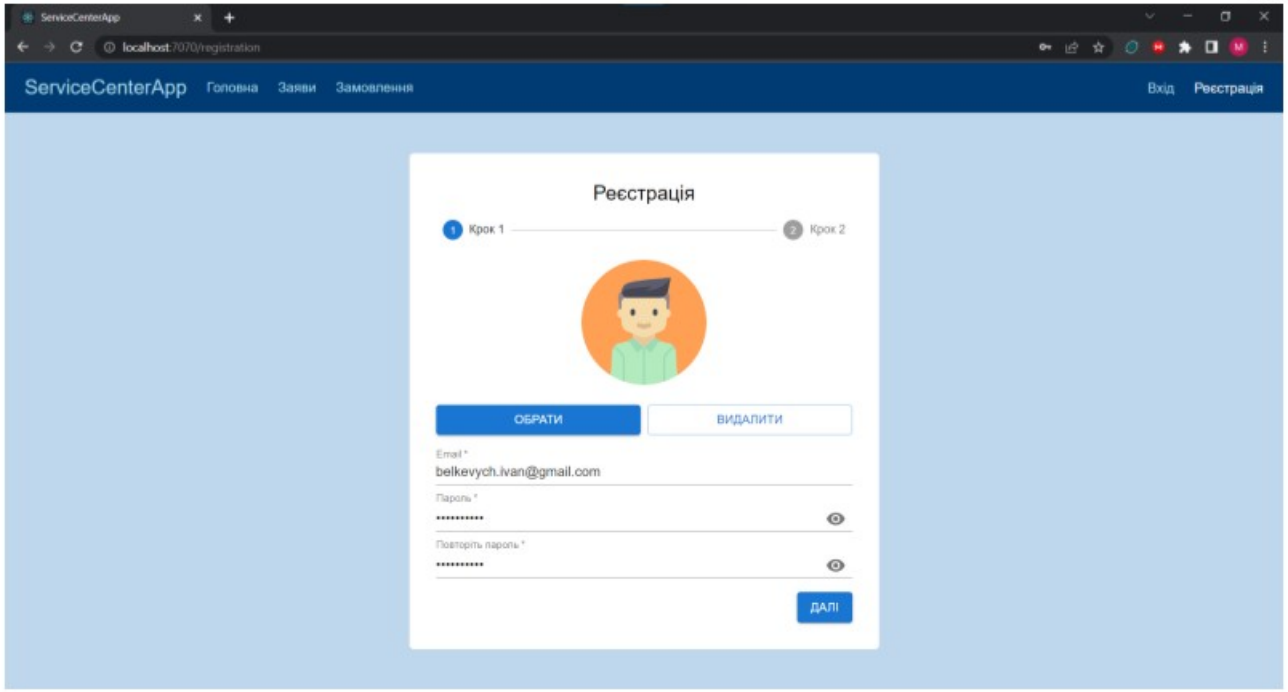
Сторінка сервісного центру. Вкладка “Відгуки”

ІНТЕРФЕЙС КОРИСТУВАЧА



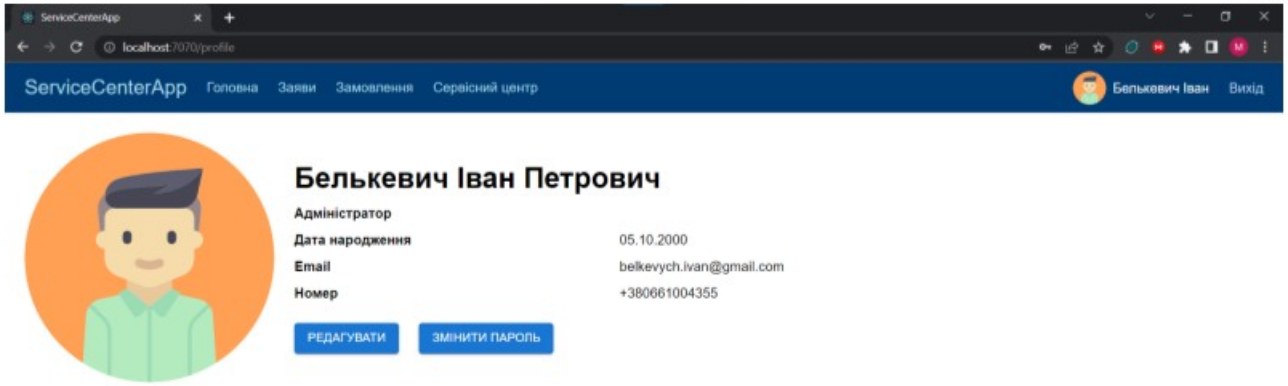
Сторінка сервісного центру. Вкладка “Карта”

ІНТЕРФЕЙС КОРИСТУВАЧА



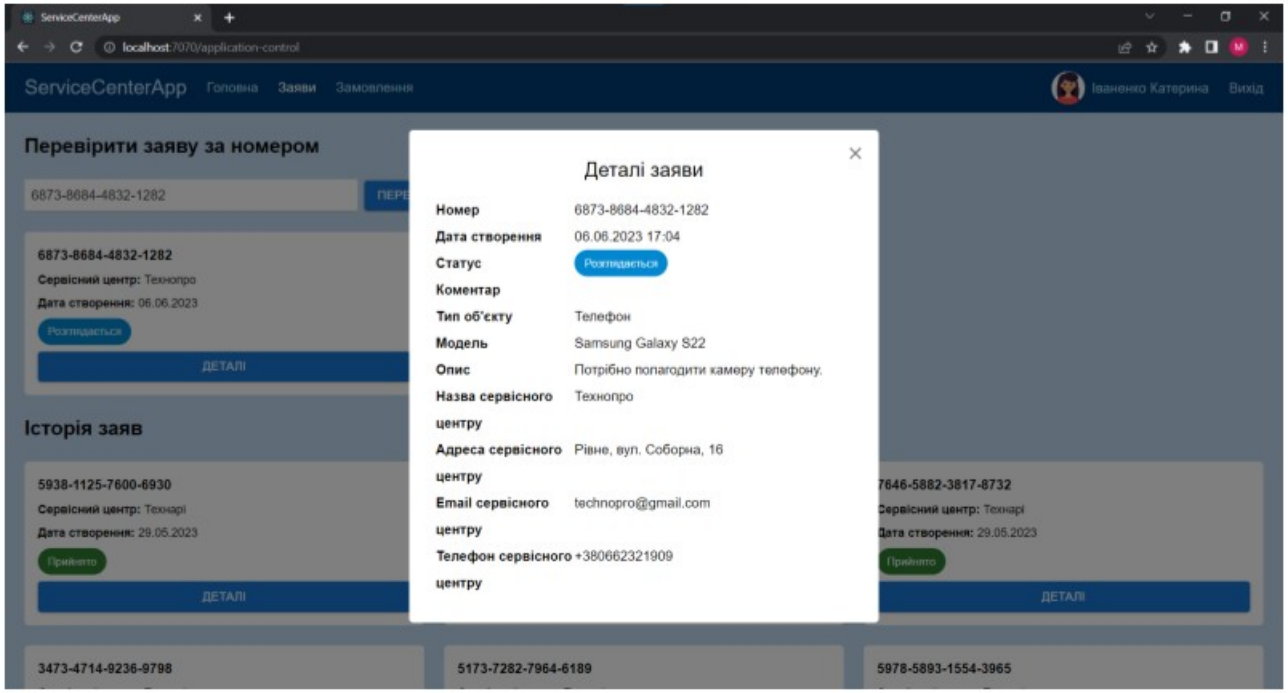
Сторінка реєстрації

ІНТЕРФЕЙС КОРИСТУВАЧА



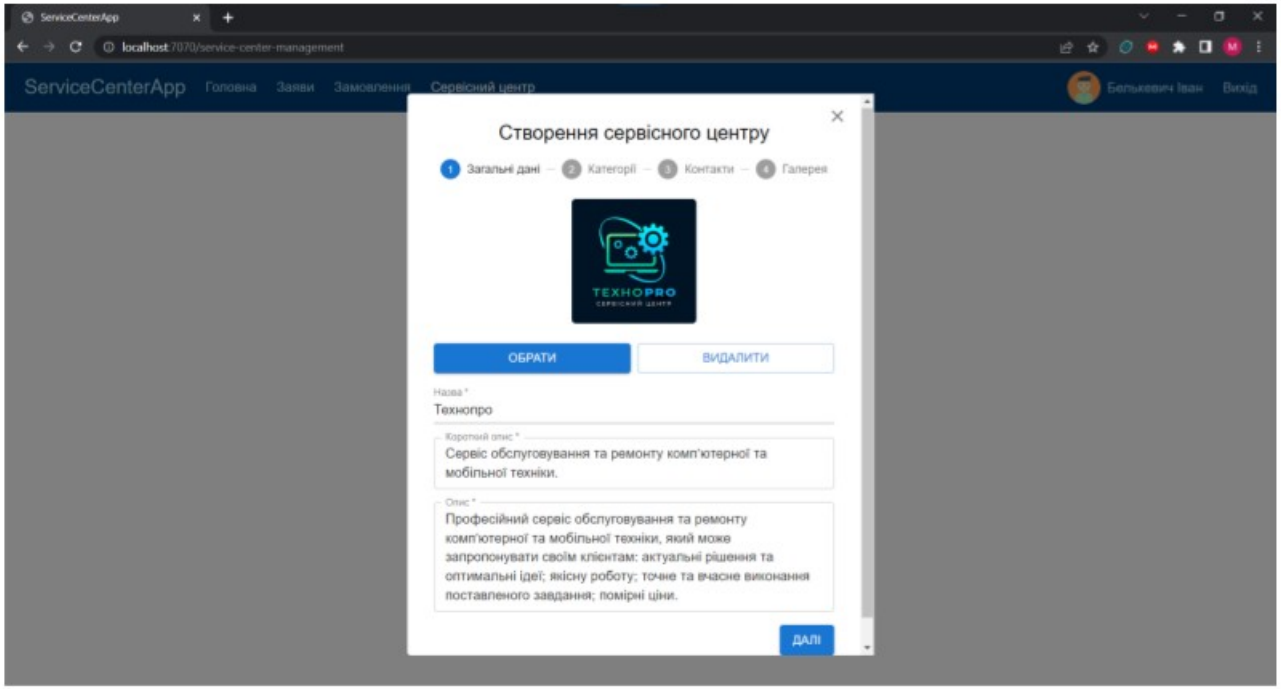
Сторінка профілю користувача

ІНТЕРФЕЙС КОРИСТУВАЧА



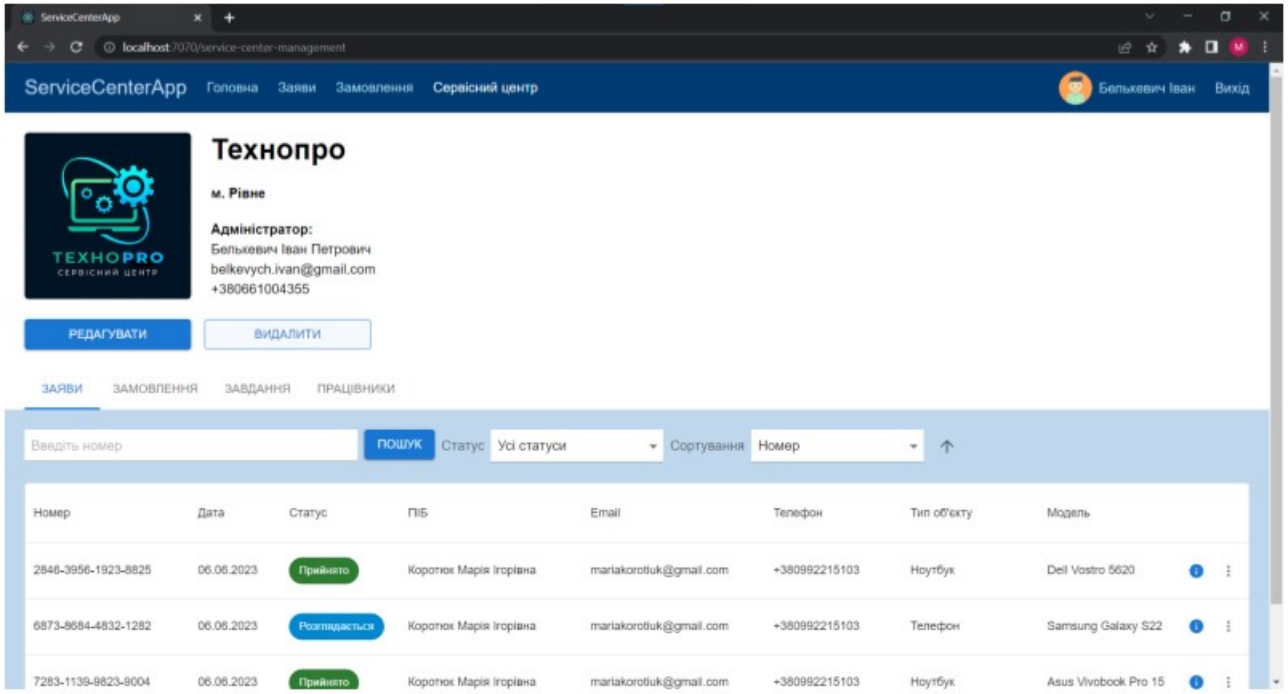
Сторінка заяв. Перевірка заяви за номером

ІНТЕРФЕЙС КОРИСТУВАЧА



Форма створення сервісного центру

ІНТЕРФЕЙС КОРИСТУВАЧА



Сторінка керування сервісним центром. Вкладка “Заяви”

ТЕСТУВАННЯ РОЗРОБЛЕНОГО ЗАСТОСУНКУ

1. Було розроблено тест-кейси для перевірки правильності роботи основних функцій вебзастосунку методом димового тестування.
2. Було виправлено знайдені помилки у роботі програмного забезпечення.

НАПРЯМКИ ПОДАЛЬШОЇ РЕАЛІЗАЦІЇ

1. Реалізувати можливість додавання декількох адрес для одного сервісного центру.
2. Додати можливість визначення координат сервісного центру на карті за його адресою.
3. Розробити версію мобільного застосунку.
4. Додати онлайн-чат для комунікації з працівниками сервісних центрів.

РЕЗУЛЬТАТИ ПЕРЕВІРКИ НА ПЛАГІАТ

0.73%
Совпадения

Наибольшее совпадение: **0.12%** с Интернет-источником (https://ela.kpi.ua/bitstream/123456789/35643/1/Ilin_bakalavr.p

0.31% Источники из Интернета	4	Страница 51
0.73% Источники из Библиотеки	14	Страница 51

0% Цитат

Не найдено ни одной цитаты

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников

ВИСНОВКИ

1. Проаналізовано існуючі рішення для сервісних центрів.
2. Визначено вимоги до програмного забезпечення.
3. Обрано засоби реалізації програмного забезпечення.
4. Розроблено структуру бази даних.
5. Розроблено вебзастосунок для підтримки діяльності сервісних центрів згідно з вимогами.
6. Протестовано розроблене програмне забезпечення на відповідність вимогам.
7. Запропоновано напрямки подальшої реалізації.

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2022 р.

ВЕБЗАСТОСУНОК ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ СЕРВІСНИХ ЦЕНТРІВ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Марія КОРОТЮК

2022

ЗМІСТ

1. Об’єкт випробувань.....3
2. Мета тестування.....3
3. Методи тестування.....3
4. Засоби та порядок тестування.....4

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Вебзастосунок для підтримки діяльності сервісних центрів, який являє собою вебдодаток, створений на платформі Node.js з використанням технології Express.js та React.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) правильна робота елементів сторінок вебзастосунку;
2) наявність доступу до бази сервісних центрів та користувачів;
3) взаємодія серверної та клієнтської частин вебзастосунку;
4) адаптивність усіх елементів інтерфейсу на пристроях різних типів;
5) зручність роботи з вебзастосунком;
6) відповідність вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Smoke Testing, що передбачає швидку оцінку основних функцій системи, а саме перевірку, чи критичні та найважливіші функції програмного забезпечення працюють належним чином.

Використовуються наступні методи:

1) тестування основних функцій для перевірки ключових сценаріїв використання вебзастосунку;
2) тестування інтеграції для перевірки правильної взаємодії між модулями вебзастосунку;
3) тестування інтерфейсу, зокрема перевірки правильного відображення усіх необхідних елементів на сторінці для пристроїв різних типів.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність вебзастосунку перевіряється шляхом:

1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) тестування вебзастосунку в різних браузерах;
4) тестування стабільності роботи при різних умовах;
5) тестування зручності використання;
6) тестування інтерфейсу;
7) тестування адаптивності інтерфейсу для різних типів пристроїв.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2023 р.

ВЕБЗАСТОСУНОК ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ СЕРВІСНИХ ЦЕНТРІВ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Марія КОРОТЮК

2023

ЗМІСТ

1. Опис структури вебзастосунку.....3
2. Опис вмісту статичних вебсторінок.....3
3. Процедура авторизації користувача.....9

1. Опис структури вебзастосунку

Вебзастосунок для підтримки діяльності сервісних центрів складається з вебсторінок, вміст яких формується динамічно.

Вебсторінки, які належать вебзастосунку:

- Головна;
- Сторінка сервісного центру;
- Заяви;
- Замовлення;
- Профіль користувача;
- Керування сервісним центром;
- Реєстрація;
- Вхід в обліковий запис.

Кожна вебсторінка містить меню, яке дозволяє здійснювати перехід між зазначеними сторінками.

2. Опис вмісту статичних вебсторінок

Сторінка “Головна” (рис. 1) містить список сервісних центрів з основною інформацією про них. Для списку можуть бути застосовані фільтри за категоріями та містами, а також сортування та пошук за назвою.

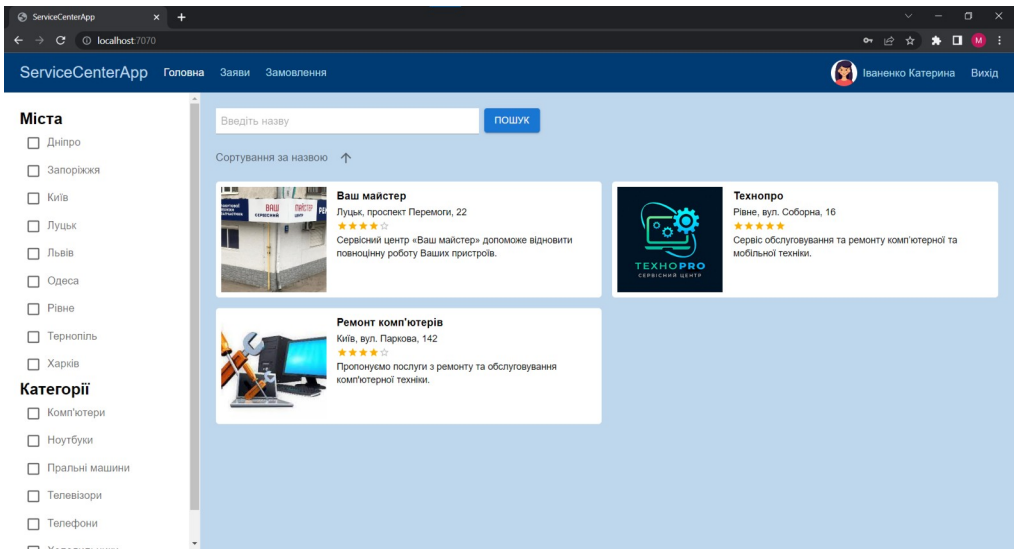


Рис. 1. Сторінка “Головна”

Сторінка сервісного центру (рис. 2) містить інформацію про сервісний центр, його фото, а також вкладки “Деталі” (містить опис та категорії), “Галерея” (містить список фото), “Відгуки” (містить список відгуків та кнопку “Додати відгук”, яка видима лише для користувача у ролі клієнта), “Карта” (містить карту з розташуванням сервісного центру, вкладка видима лише якщо для сервісного центру було визначено координати).

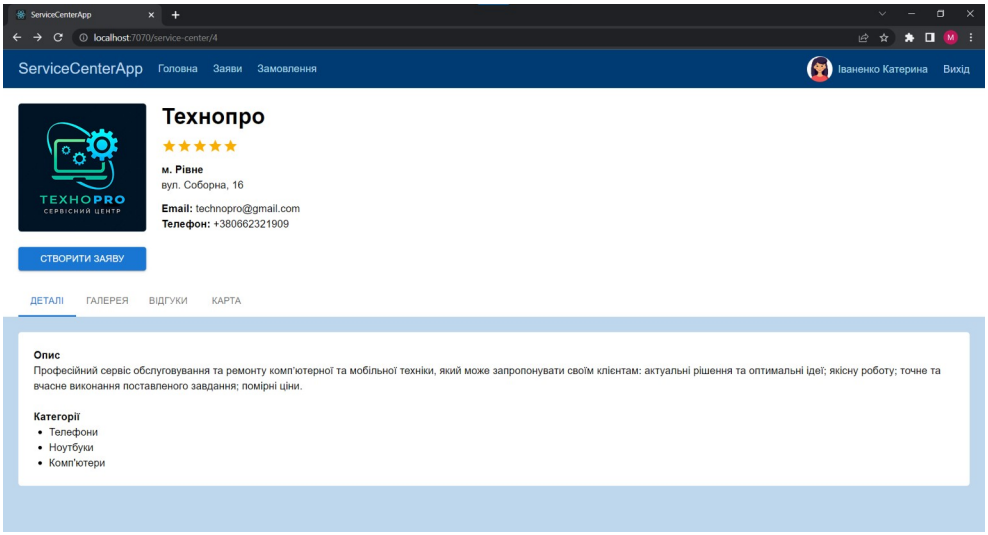


Рис. 2. Сторінка сервісного центру

Сторінка “Замовлення” містить пошукове поле для знаходження замовлення за його номером. Після знаходження замовлення, коротка інформація про нього з’являється під пошуковим полем. Для користувачів у ролі клієнтів сторінка також містить історію їх замовлень. Для знайденого замовлення та замовлень з історії можна переглянути деталі у модальному вікні після натискання кнопки “Деталі”.

Сторінка “Заяви” (рис. 3) містить пошукове поле для знаходження заяви за її номером. Після знаходження заяви, коротка інформація про неї з’являється під пошуковим полем. Для користувачів у ролі клієнтів сторінка також містить історію їх заяв. Для знайденої заяви та заяв з історії можна переглянути деталі у модальному вікні після натискання кнопки “Деталі”.

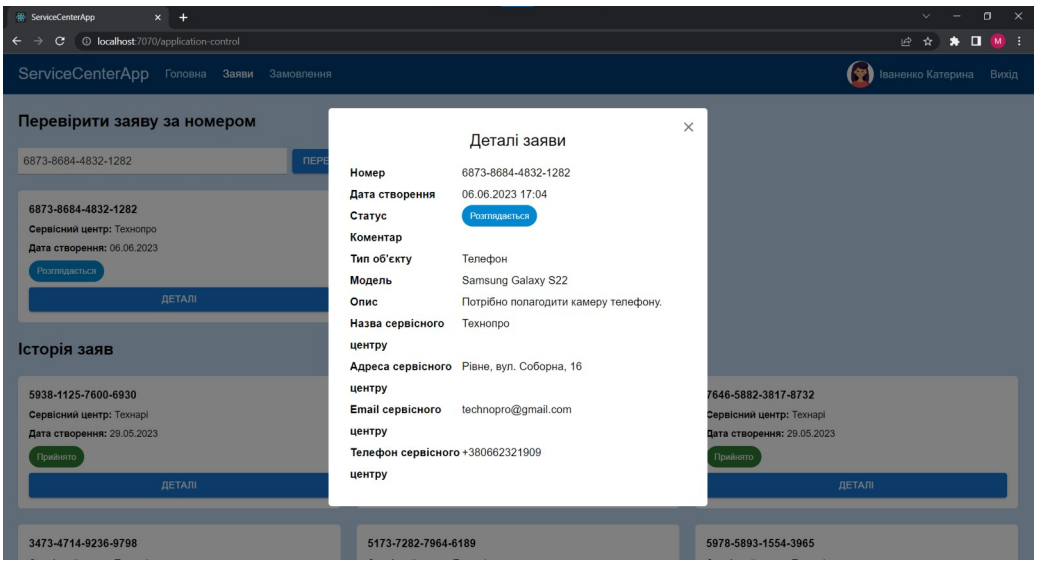


Рис. 3. Сторінка “Заяви”

Сторінка “Профіль користувача” (рис. 4) містить інформацію про користувача, який авторизований у застосунку. Для неавторизованих користувачів сторінка є недоступною. Також сторінка містить кнопку “Редагувати”, яка є доступною лише для користувачів у ролі адміністратора або клієнта. При натисканні на цю кнопку з’являється модальне вікно з формою для редагування даних користувача.

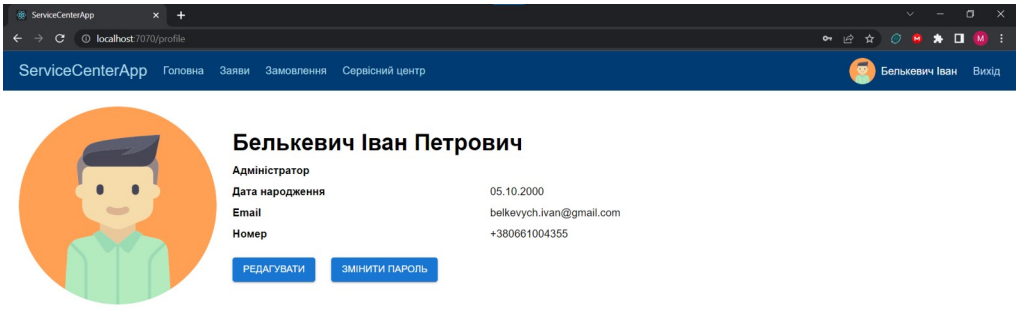


Рис. 4. Сторінка “Профіль користувача”

Сторінка “Керування сервісним центром” (рис. 5) містить назву сервісного центру, його фото, інформацію про адміністратора сервісного

центру, а також вкладки “Заяви”, “Замовлення”, “Завдання” та “Працівники”. Для адміністратора також відображаються кнопки “Редагувати” та “Видалити”, при натисканні на які з'являється відповідне модальне вікно з формою для редагування або видалення.

Вкладка “Заяви” містить таблицю з інформацією про отримані сервісним центром заяви працівників. Про кожну з них можна переглянути детальну інформацію натиснувши відповідну кнопку в потрібному рядку таблиці. Також вкладка містить поле для пошуку заяви за її номером, випадаючий список для додавання фільтрів за статусом заяви, а також випадаючий список для вибору поля, за яким відбувається сортування, та кнопка для визначення сортування за зростанням чи за спаданням.

Адміністратор може редагувати, видаляти кожну заяву та створювати для неї замовлення після натискання відповідної кнопки в таблиці та заповнення даних у формах, але кнопка видалення доступна лише для заяв зі статусом “Відхилено”, а створення замовлення для заяв зі статусом “Прийнято”.

Вкладка “Замовлення” містить таблицю з інформацією про замовлення у сервісному центрі. Про кожне з них можна переглянути детальну інформацію натиснувши відповідну кнопку в потрібному рядку таблиці. Також вкладка містить кнопку для створення замовлення окремо від надісланих клієнтами заяв, поле для пошуку замовлення за його номером, випадаючий список для додавання фільтрів за статусом замовлення, а також випадаючий список для вибору поля, за яким відбувається сортування, та кнопка для визначення сортування за зростанням чи за спаданням.

Адміністратор може редагувати, видаляти кожне замовлення та створювати для нього завдання після натискання відповідної кнопки в таблиці та заповнення даних у формах, але кнопка видалення доступна лише для замовлень зі статусом “Скасовано”, а створення завдання для замовлень зі статусом “Створено”, “Виконується” та “Проблема”.

Вкладка “Завдання” містить таблицю з інформацією про завдання у сервісному центрі. Про кожне з них можна переглянути детальну інформацію натиснувши відповідну кнопку в потрібному рядку таблиці. Також вкладка містить кнопку для створення завдання окремо від замовлень, поле для пошуку завдання за його номером, випадаючий список для додавання фільтрів за статусом завдання, а також випадаючий список для вибору поля, за яким відбувається сортування, та кнопка для визначення сортування за зростанням чи за спаданням.

Адміністратор може редагувати, видаляти кожне завдання, але кнопка видалення доступна лише для замовлень зі статусом “Скасовано”, а створення завдання для замовлень зі статусом “Створено”, “Виконується” та “Проблема”.

Вкладка “Працівники” містить список працівників сервісного центру, про кожного працівника можна переглянути детальну інформацію у модальному вікні натиснувши відповідну кнопку. Для адміністратора також доступні форми реєстрації працівника, редагування та видалення.

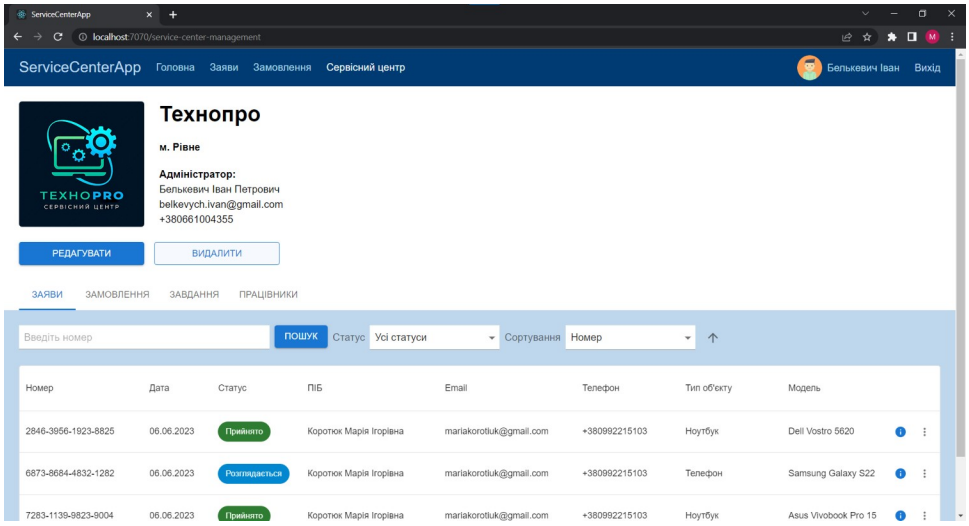


Рис. 5. Сторінка “Керування сервісним центром”

Сторінка “Керування сервісним центром” є доступною лише для адміністратора та працівника цього сервісного центру. Якщо сервісний центр ще не було створено, то сторінка доступна лише для адміністратора

та містить кнопку “Створити сервісний центр”. При натисканні на неї з'являється модальне вікно з формою, яку необхідно заповнити та натиснути кнопку “Створити”.

Сторінка “Вхід в обліковий запис” (рис. 6) містить форму для входу в обліковий запис, яка має два обов’язкових поля “Email” та “Пароль”.

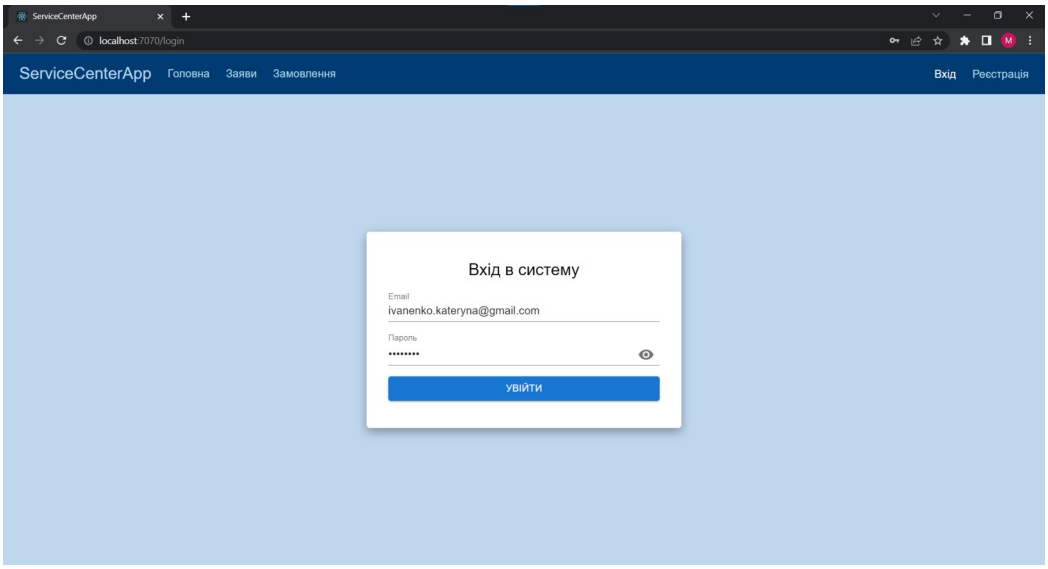


Рис. 6. Сторінка “Вхід в обліковий запис”

Сторінка «Реєстрація» (рис. 7) містить форму, яку необхідно заповнити для реєстрації у застосунку. Після заповнення форми користувач має натиснути кнопку “Зареєструватися”.

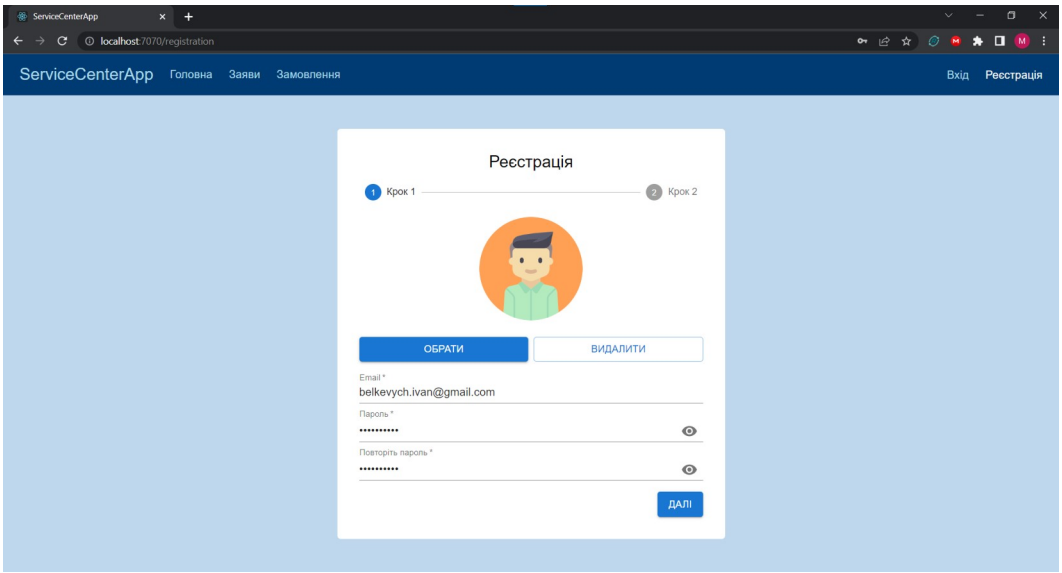


Рис. 7. Сторінка “Реєстрація”

3. Процедура авторизації користувача

Авторизація користувача відбувається на сторінці входу в обліковий запис. Користувач має ввести свої логін (електронну пошту) і пароль та натиснути кнопку “Увійти”, після чого відбувається перехід до головної сторінки вебзастосунку, яка містить список сервісних центрів. При введені неправильного логіну та / або паролю користувач отримує повідомлення “Ведено неправильний логін або пароль”.