

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Веб-сайт кол-центру»

Виконала студентка групи ЗІПЗ-20-1

СУРІКОВА Ольга Андріївна

Керівник роботи:

ЄФРЕМОВ Юрій Миколайович

Рецензент:

ТОЛСТОЙ Ігор Анатолійович

Житомир – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **СУРІКОВА Ольга Андріївна**

Керівник роботи: **ЄФРЕМОВ Юрій Миколайович**

Тема роботи: **«Веб-сайт кол-центру»**,

затверджена Наказом закладу вищої освіти від **«23» лютого 2024 р., №75с**

Вихідні дані для роботи: розробка веб-сайту, виправлення недоліків, які присутні у конкурентів

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Єфремов Ю. М.	27.02.2024р.	27.02.2024р.
2	Єфремов Ю. М.	27.02.2024р.	27.02.2024р.
3	Єфремов Ю. М.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Об'єктом дослідження є товариство з обмеженою відповідальністю аутсорсинговий колл-центр «ЛЕТСКОЛЛ».

Предметом дослідження є діяльність працівників та клієнтів. Пояснювальна записка містить текстову частину викладено на 59 сторінках друкованого тексту. Список використаних джерел містить 15 найменувань та займає 2 сторінки. В роботі наведено 39 рисунки. Загальний обсяг роботи 78 сторінок.

Метою випускної кваліфікаційної роботи бакалавра є розробка та аналіз концепції веб-сайту колл-центру, що відповідатиме сучасним потребам та вимогам, здатного забезпечити ефективне обслуговування клієнтів та покращити роботу колл-центру в цілому.

У роботі був проведений аналіз внутрішнього середовища компанії. Проведене концептуальне проєктування, в результаті якого були визначені сутності бази даних. Також проведено логічне та фізичне проєктування.

Були розроблені автоматизовані робочі місця для власника, співробітників та клієнтів, що мають зручний та зрозумілий інтерфейс.

КЛЮЧОВІ СЛОВА: ВЕБ-САЙТ, КОЛЛ-ЦЕНТР, РОЗРОБКА, СТРУКТУРА ВЕБ-САЙТУ, БАЗА ДАНИХ

					ІПЗ.КР.Б – 121 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.	Сурикова О.А.				Веб-сайт кол-центру	Літ.	Арк.	Аркуші
Керівник	Єфремов Ю.М.						3	
Рецензент	Толстой І. А.							
Зав. каф.	Вакалюк Т.А.					ФІКТ Гр. ЗІПЗ-20-1		

ABSTRACT

The object of the study is the outsourcing call center “LET’S CALL” limited liability company.

The subject of the study is the activities of employees and customers. The explanatory note contains part of the text laid out on 59 pages of printed text. The list of references includes 15 titles and occupies 2 pages. The paper contains 39 figures. The total volume of the work is 56 pages.

The purpose of the bachelor's thesis is to create a website for a call center. It is intended to introduce potential customers to the company, its activities and the range of services provided.

The work included an analysis of the company's internal environment. Conceptual design was carried out, as a result of which the entities of the database were identified. We also carried out logical and physical design.

Automated workstations for the owner, employees, and customers were developed, which have a convenient and intuitive interface.

KEYWORDS: WEBSITE, CALL CENTRE, DEVELOPMENT, WEBSITE STRUCTURE, DATABASE

					ІПЗ.КР.Б – 121 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.	Сурікова О.А.				Веб-сайт кол-центру	Літ.	Арк.	Аркуші
Керівник	Єфремов Ю.М.						4	
Рецензент	Толстой І. А.							
Зав. каф.	Вакалюк Т.А.					ФІКТ Гр. ЗІПЗ-20-1		

ЗМІСТ

ВСТУП.....	6
РОЗДІЛ 1. СУЧАСНИЙ СТАН ТА АКТУАЛЬНІСТЬ ВПРОВАДЖЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ У КОЛЛ-ЦЕНТРИ.....	9
1.1. Опис проєкту та актуальність впровадження.....	9
1.2. Аналіз існуючих розробок.....	11
1.3. Завдання та цілі кваліфікаційної роботи	16
Висновки до першого розділу	18
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ САЙТУ	19
2.1. Розробка структурної схеми роботи сайту	19
2.2. Розробка функціональної схеми сайту.....	25
2.3. Проєктування та реалізація алгоритмів роботи системи	31
2.4. Проєктування бази даних	35
2.5. Вимоги для розробки та розробка дизайну веб-сайту	42
Висновки до другого розділу	48
РОЗДІЛ 3. АНАЛІЗ СЕРЕДОВИЩ РОЗРОБКИ ТА РОЗРОБКА ВЕБ САЙТУ	50
3.1. Аналіз обраної середовища програмування.....	50
3.2. Обґрунтування вибору технологій для розробки	52
3.3. Реалізація веб-сайту.....	56
Висновки до третього розділу	63
ВИСНОВКИ	64
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	65
ДОДАТКИ.....	67

ВСТУП

У сучасному світі, де онлайн-комунікація стає все більш поширеною, а клієнтські очікування щодо зручності та швидкості обслуговування зростають, веб-сайти колл-центрів набувають неабиякої актуальності. Однак традиційні колл-центри мають свої обмеження, такі як високі витрати на утримання, обмежений час роботи, складність у веденні статистики та аналізу даних. Саме тому сучасні технології, зокрема веб-розробка, надають можливість створити інноваційний інструмент - веб-сайт колл-центру.

Однією з найбільших проблем є погана автоматизація процесів. До робочого процесу необхідно відносити не лише безпосередній факт надання та виконання послуг, але й ведення звітності.

На даний час більшість операцій по запису послуг та клієнтів проводяться в таблиці Excel, що ускладнює роботу власника та адміністраторів. Для прикладу, під час запису клієнта доводиться враховувати наступні фактори:

- графік надання послуги;
- наявність вільного часу для виконання;
- послуга, яку хоче отримати клієнт.

Пошук цієї інформації займає значний час, що ускладнює роботу адміністраторів, особливо, якщо інші записи знаходяться у самого власника. Всі ці проблеми в сукупності призводять до уповільнення роботи, також немає можливості контролю виконаних послуг. У підсумку це знижує ефективність роботи підприємства в цілому.

Найбільш прогресуючим напрямком є розробка і впровадження інформаційної системи з використанням сучасних засобів. Інформаційна система це комунікаційна система, що забезпечує збирання, пошук оброблення даних та пересилання інформації.

Взаємодію з веб-сайтом повинні знати не лише адміністратори, але і директор, адже, у будь-якому випадку, знати софт повинні більше ніж дві людини у компанії. Це необхідно з ряду причин:

- при звільненні когось із персоналу навчити нового можна всередині самої компанії, часто не вдаючись до послуг розробників;
- при збої в софті і при відсутності директора на місці тільки співробітник, що розбирається в програмі, зможе грамотно сформулювати питання менеджеру;
- при відсутності директора навчений співробітник зможе коректно заносити інформацію в програму, що не дасть отримувати недостовірну звітність в подальшому.

Все можна спростити з впровадженням у процеси роботи веб-сайту. Більшість дій можливо вирішити завдяки автоматизації.

Актуальність теми полягає в тому, що веб-сайт колл-центру, будучи сучасним інструментом, може значно підвищити ефективність роботи колл-центру, забезпечити зручність для клієнтів та підвищити рівень задоволеності.

Метою дипломної роботи є розробка та аналіз концепції веб-сайту колл-центру, що відповідатиме сучасним потребам та вимогам, здатного забезпечити ефективне обслуговування клієнтів та покращити роботу колл-центру в цілому.

Предметом дослідження є процес створення веб-сайту для аутсорсингового колл-центру «ЛІЕТСКОЛЛ», включаючи його архітектуру, функціональність, інтерфейс, технології та методи розробки.

Об'єктом дослідження є покращення автоматизації управління процесами планування часу.

Головними задачами, що стоять при розробці сайту є:

- ознайомлення клієнтів з компанією та послугами;
- запис клієнтів;
- скорочення часу обробки та отримання даних;
- зберігання інформації про клієнтів, працівників, видів послуг та тарифів, бронювання послуги;

- скорочення часу обробки та отримання даних;
- проектування структурних складових та алгоритмів роботи системи.

Тобто покращення автоматизації основних функцій у колл-центрі це не гонитва за новинками. Це необхідність для збереження та розвитку бізнесу. Результати дослідження дозволять сформуванати уявлення про можливості використання веб-сайту колл-центру для підвищення ефективності роботи колл-центру та покращення взаємодії з клієнтами.

Випускна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із 15 найменувань (65 стор.), 2 додатків (67 стор.). Повний обсяг роботи становить 78 сторінок, з них 59 сторінок основного тексту.

РОЗДІЛ 1. СУЧАСНИЙ СТАН ТА АКТУАЛЬНІСТЬ ВПРОВАДЖЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ У КОЛЛ-ЦЕНТРИ

1.1. Опис проєкту та актуальність впровадження

У сучасному динамічному світі, де інформація доступна миттєво, а споживач очікує швидких та ефективних рішень, роль колл-центрів значно зросла. Вони стали невід'ємною частиною багатьох компаній, забезпечуючи зручний та доступний канал взаємодії з клієнтами. Однак традиційні колл-центри мають свої обмеження, такі як високі витрати на утримання, обмежений час роботи, складність у веденні статистики та аналізу даних. Саме тому сучасні технології, зокрема веб-розробка, надають можливість створити інноваційний інструмент - веб-сайт колл-центру.

Колл-центр це компанія, яка займається обробкою телефонних дзвінків, електронної пошти та інших каналів комунікації з клієнтами. Це свого роду «центр комунікації» між певною компанією та клієнтом. Ця діяльність є різновидом економічної діяльності.

Збільшення кількості клієнтів, які надають перевагу онлайн-комунікації, робить веб-сайт колл-центру незамінним інструментом для сучасного бізнесу. Він дозволяє підприємствам ефективно задовольняти потреби клієнтів у будь-який час та з будь-якого місця.

Веб-сайт колл-центру дозволяє ефективно управляти потоком дзвінків, оптимізуючи роботу операторів та покращуючи загальну продуктивність. За допомогою інтеграції з телефонними системами, веб-сайт може забезпечити автоматичне розподілення дзвінків, відстеження статусу дзвінків, зберігання історії спілкування та багато іншого. Це допомагає підвищити швидкість обслуговування клієнтів та знизити витрати на телефонні дзвінки.

Сучасний веб-сайт колл-центру може пропонувати широкий спектр функціональних можливостей для покращення взаємодії з клієнтами. Наприклад, клієнти можуть:

1. Отримати інформацію про товари та послуги, ознайомитися з часто задаваними питаннями (FAQ) та отримати відповіді на свої питання безпосередньо на веб-сайті.
2. Залишити заявку на дзвінок, вказавши зручний час та тему розмови.
3. Відстежувати статус своїх запитів та звернень.
4. Оцінити якість обслуговування та залишити відгук.

Веб-сайт колл-центру може автоматизувати багато рутинних завдань, звільняючи операторів від зайвої роботи та дозволяючи їм зосередитися на вирішенні складніших питань. Це підвищує продуктивність праці та загальну ефективність колл-центру.

Завдяки автоматизації та оптимізації процесів, веб-сайт колл-центру може значно знизити витрати на обслуговування клієнтів. Це стосується як витрат на телефонні дзвінки, так і витрат на утримання персоналу.

Сучасний веб-сайт колл-центру може позитивно вплинути на імідж компанії, демонструючи її клієнтоорієнтованість, сучасність та інноваційність.

Багато власників не замислюються про покращення автоматизації, адже звикли до стандартних методів. У більшості таких закладів існує стандартний журнал записів клієнтів у адміністратора компанії в файл Excel. Комп'ютер, більшу частину часу, використовується адміністратором у власних цілях – для спілкування у соціальних мережах або проведення часу в іграх.

На прикладі реального колл-центру «ЛЕТСКОЛЛ», завдяки якому було досліджено весь робочий процес, стало зрозуміло, що є багато незручностей які можливо вирішити завдяки покращення автоматизації процесів.

В офісі працює адміністратор, який організовує всі робочі процеси. Основним завданням є прийом клієнтів, прийом дзвінків від клієнтів, уточнення інформації з приводу послуги та надання повної інформації клієнту, або передача інформації про бажану послугу відповідному менеджеру для

детальнішого ознайомлення. Також, він контролює графік роботи усіх співробітників, забезпечує чистоту у приміщенні, веде правильний підрахунок заробітної плати та оплати послуг клієнтам.

Протягом робочого часу працівники колл-центру повинні пам'ятати багато важливої інформації, оброблювати велику кількість дзвінків, вирішувати конфліктні ситуації із клієнтами. Зараз, загальна інформація від кожного співробітника фіксується у спеціальній програмі від кожної компанії, із якою співпрацює даний колл-центр, а адміністратор – контролює та збирає загальну інформацію про виконання роботи, робочий час та проблеми, які виникали та відстежує показники через файли Excel. Отже, можна дійти до висновку, що це призводить до незручностей та втрати часу при роботі із клієнтами та співробітниками, що тим самим призводить до їх невдоволення.

Всі ці процеси уповільнюють процес роботи як адміністратора, так і власника, відсутня можливість контролю виконаної роботи та дублювання дій. Це призводить до того, що знижується ефективність роботи. Також, як відомо, більшість людей шукають усю необхідну інформацію на просторах інтернету. Зазвичай, організації створюють веб-сайт, який призначений щоб розповісти користувачам про організацію, послуги, вартість та надати контактні дані. Колл-центр «ЛЕТСКОЛЛ» має слабо інформативний сайт. Це є однією із слабких сторін.

Дослідивши дану сферу можна зрозуміти, що автоматизування колл-центрів – найбільша перспектива на даному етапі. Це допоможе підняти процеси управління та систематизації на новий, більш якісний рівень, а також допоможе вдосконалити систему комунікації з клієнтами. Це підтверджує актуальність даної теми.

1.2. Аналіз існуючих розробок

В епоху онлайн-комунікації, веб-сайт колл-центру стає незамінним. Він дозволяє задовольнити потреби клієнтів 24/7, незалежно від їхнього місця

розташування. Завдяки автоматизації рутинних завдань, веб-сайт звільняє операторів для вирішення складніших питань, підвищуючи продуктивність та ефективність роботи колл-центру. Крім того, це допомагає знизити витрати на обслуговування клієнтів.

Проаналізувавши декілька готових програмних продуктів, можна стверджувати що вони мають потужну базу. Функції, що були реалізовані в даних системах, роблять велику частину завдань з управління, організації та обліку.

Середня вартість впровадження таких систем обійдеться дорого. Постійна підтримка та оновлення, багато функцій, які не будуть використовуватись, також, такі системи мають складний інтерфейс. Це призводить до зниження ефективності роботи. А ще важливо примітити, що не кожен співробітник добре знається на принципах роботи в складних системах.

Для прикладу, розглянемо один із найпопулярніших сайтів аусторсингового колл-центру – grace[8]. Даний сайт справляє позитивне враження, пропонуючи сучасний дизайн та чітку структуру. Візуально привабливий та інформативний, він добре структурований для зручного перегляду. Дизайн доволі сучасний та адаптивний, виглядає добре на всіх пристроях. Доволі функціональний – має і відео-матеріали, і форма зворотного зв'язку, і брошури, які можна завантажити, проте перевірити можливість роботи його внутрішнього функціоналу неможливо(рис.1.1). Проте є певні недоліки: немає чату для онлайн-спілкування (це пришвидшить комунікацію із клієнтом), відсутній розділ із запитаннями та можна розширити розділ «Відгуки».

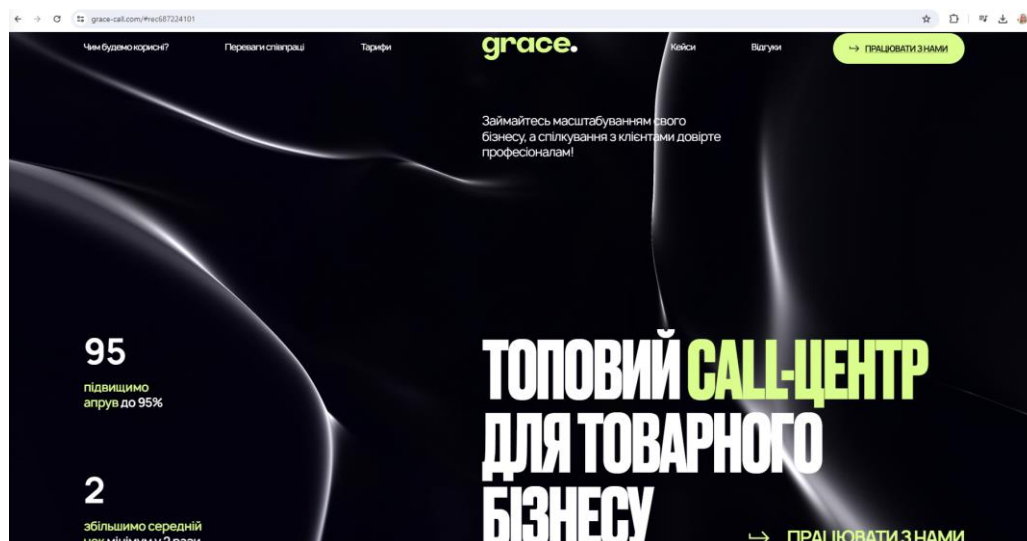


Рисунок 1.1. Приклад сайту колл-центру з надання аутсорсингових послуг

Іншим прикладом, розглянемо сайт contactis [15]. Сайт не дуже інформативний, доволі довго завантажується (рис. 1.2). Погано реалізована функція зворотного зв'язку з компанією. Із плюсів – сайт дуже простий, тобто відсутня складна анімація, яка буде погано впливати на його роботу. Також, сайт адаптовано для усіх розмірів і навіть із телефону його переглядати зручно.

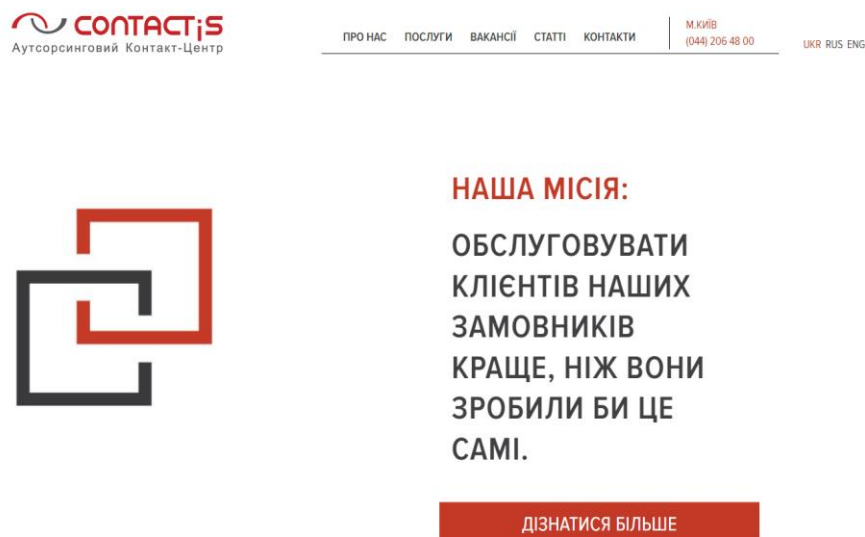


Рисунок 1.2. Приклад сайту contactis

Основними потребами, які потрібно врахувати при розробці веб-сайту є наступні пункти:

1. Управління дзвінками.

Веб-сайт має давати доступ до інформації про всі дзвінки, в тому числі до часу дзвінків, тривалості, ідентифікатора дзвінків, статусу (прийнято/неприйнято), записи дзвінків (якщо потрібно).

Можливість створювати різні черги для різних груп клієнтів, налаштування пріоритетів, управління очікуванням.

Гнучкий механізм переадресації дзвінків до інших операторів або відділів, можливість встановлення правил переадресації.

Здатність веб-сайту ідентифікувати дзвінків (наприклад, за номером телефону) та відображати інформацію про клієнта під час дзвінка.

Зв'язок з CRM, ERP та іншими системами для обміну інформацією про клієнтів та їхні запити.

Можливість автоматичного оброблення простих запитів, надання інформації про компанію та її послуги.

2. Ведення статистики та звітність.

Основними потребами мають бути збирання даних про дзвінки, тобто реєстрація часу дзвінків, їх тривалість, статус, тема розмови, категорія. Здатність генерувати різні звіти про ефективність роботи колл-центру (наприклад, кількість дзвінків, тривалість очікування, задоволеність клієнтів), звітів про роботу операторів (кількість дзвінків, час обробки, продуктивність).

Здатність аналізувати дані та виявляти тенденції, що дозволить покращити роботу. Можливість представляти статистику в зрозумілій та зручній формі для аналізу.

3. Взаємодія із клієнтами.

Однією із можливостей взаємодії є онлайн-чат. Він надає здатність вести чат з клієнтами для вирішення їх запитів або отримання зворотного зв'язку. Ще одним видом комунікації є форма зворотного зв'язку – зручний спосіб для клієнтів надавати зворотній зв'язок про якість обслуговування. Також, є можливість підключатись до соціальних мереж для спілкування із клієнтами та отримання їхньої думки.

Також, можливо отримувати інформацію із самого сайту – надається інформація про компанію, її послуги та продукти. Також надається доступ клієнтам до бази знань, яка містить відповіді на найбільш поширені запитання.

4. Додаткові потреби.

До додаткових потреб необхідно віднести безпеку першою чергою, адже клієнти мають бути впевнені, що їх дані будуть надійно захищені і інформація надалі нікуди не передається. Також, має бути захищена інформація про роботу самого колл-центру.

В другу чергу, має бути надійний доступ до веб-сайту з різних пристроїв (комп'ютер, планшет, мобільний телефон). Має бути простий та інтуїтивно зрозумілий інтерфейс для користувачів.

При розробці веб-сайту для колл-центру важливо враховувати всі перераховані потреби. Це дозволить створити ефективний інструмент для управління дзвінками, ведення статистики та взаємодії з клієнтами. Важливо пам'ятати, що веб-сайт для колл-центру - це не просто візитка, а інструмент, який дозволяє покращити роботу колл-центру та підвищити задоволеність клієнтів.

Для покращення автоматизації роботи із співробітниками та клієнтами компанії необхідно ввести спеціальну програму для планування. Було запропоновано наступне програмне забезпечення – ShiftON – автоматизований сервіс планування [9]. Багатофункціональна програма, інтуїтивна, створена для зручності з точки зору адміністраторів та власника

(рис. 1.3). Програма пропонує автоматизувати усі бізнес-процеси, що, в свою чергу, спростить роботу адміністраторів, власника, співробітників, скоротить витрати фінансові та часові, а також надати повноцінну картину роботи бізнесу.

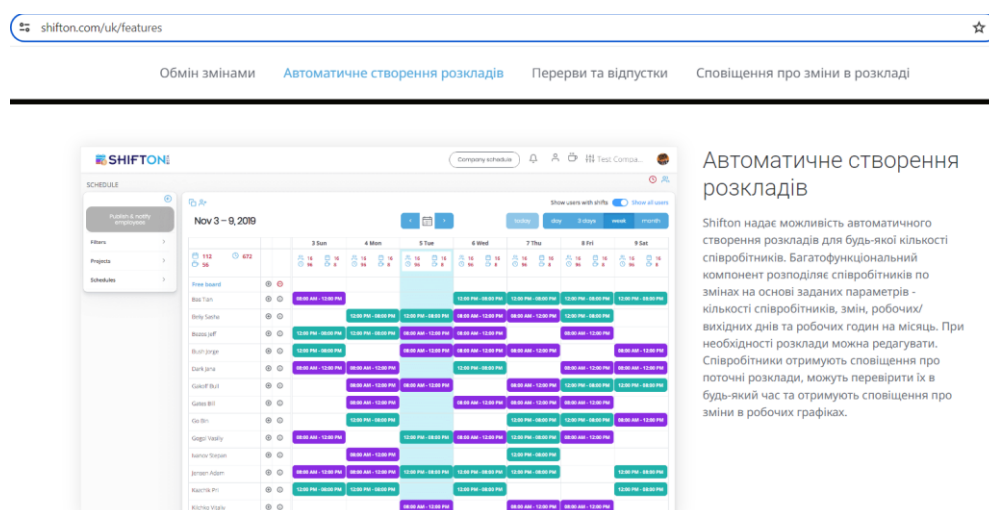


Рисунок 1.3. Приклад роботи системи ShiftON

Висока вартість застосування та обслуговування, висока складність програмного продукту впливає на рішення керівників про автоматизацію діяльності. В результаті аналізу, побачивши всі переваги та недоліки можна зрозуміти, які цілі та задачі стоять в перспективі. Також, можна зрозуміти що автоматизація закладів є перспективним розвитком технологій, що сприятиме підвищенню рівня якості надання послуг.

1.3. Завдання та цілі кваліфікаційної роботи

Метою роботи є розробка та аналіз концепції веб-сайту колл-центру, що відповідатиме сучасним потребам та вимогам, здатного забезпечити ефективне обслуговування клієнтів та покращити роботу колл-центру в цілому. Автоматизація багатьох процесів забезпечить зручність для співробітників та клієнтів. Побудувавши весь процес управління правильно, система допоможе забезпечити якісне управління, систематизацію та ведення документації.

Створення сайту допоможе забезпечити ефективну організацію надання послуг, консультування клієнтів та ведення обліку всіх процесів. Створення сайту покаже, що підприємство долучається до сучасних тенденцій.

Сайт – найперше, що бачить клієнт шукаючи необхідну послугу. Якісний сайт – можливість для організації отримати довіру клієнта, навіть якщо він не заходить до офісу. Це може підштовхнути його замовити послугу саме в цій організації.

Основними властивостями сайту є:

- надання загальної інформації;
- наявність видів послуг та цін, місцерозташування, контактних даних;
- своєчасне оновлення актуальної інформації.

Завданням на кваліфікаційну роботу є:

- визначення функціональних можливостей;
- дизайн та розробка;
- тестування та впровадження.

Основним завданням сайту є надання інформації клієнту інформацію про компанію, зацікавити послугами, забезпечити клієнта можливістю запам'ятати дану організацію.

Для співробітників сайт забезпечить своєчасне внесення запитів клієнтів, розподіл завдань, контроль продуктивності, зручне ведення оплати клієнтами послуг. Також забезпечить кращий контроль за робочими процесами.

Метою кваліфікаційної роботи буде:

- ознайомлення клієнтів з компанією та послугами;
- систематизоване зберігання інформації;
- пришвидшений пошук та отримання даних.

Додатково буде розроблена автоматизація записів про клієнтів, контроль та завантаження даних. Найважливіше – вся інформація буде зберігатись в одному місці та у зручному форматі.

Отже, під час виконання даної кваліфікаційної роботи буде зроблено наступне:

1. Розроблено структуру сайту.
2. Розроблено схеми взаємодії із сайтом різними користувачами.
3. Спроектовано базу даних.
4. Проаналізовано сучасні методи розробки.
5. Створено веб-сайт.

Висновки до першого розділу

У першому розділі було розглянуто, наскільки актуальною є тема кваліфікаційної роботи, також було коротко описано проєкт.

Як підсумок, можна зазначити наступні причини актуальності:

1. Зростання онлайн-взаємодії.
2. Покращення обслуговування клієнтів.
3. Зменшення витрат.
4. Підвищення ефективності.
5. Підвищення конкурентної переваги.

Також, у розділі розглянуто та проаналізовано існуючі засоби за тематикою кваліфікаційної роботи. Після чого було визначено мету та завдання на кваліфікаційну роботу.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ САЙТУ

2.1. Розробка структурної схеми роботи сайту

Структурна схема роботи сайту - це графічне представлення взаємодії між різними елементами веб-проекту. Вона показує, як інформація передається між користувачем, сервером, базою даних та іншими компонентами сайту.

Основними елементами структурної схеми є:

- користувач – особа, яка взаємодіє із веб-сайтом;
- веб-браузер – програма, що відображає веб-сторінки;
- веб-сервер – комп'ютер, що зберігає веб-файли та обробляє запити від користувачів;
- база даних – система зберігання та управління даними, які використовуються на сайті (наприклад, інформація про користувачів, товари, статті) [2];
- веб-сторінки – файли, що складаються з HTML, CSS та JavaScript, що відображаються у браузері [1];
- скрипти – програми, що виконуються на сервері або в браузері для обробки даних, взаємодії з базою даних та створення динамічного контенту;
- інтерфейси програмування – набори правил, що дозволяють різним програмам взаємодіяти між собою.

Також, вони розрізняються за наступними типами:

1. Функціональна – показує взаємодію між різними функціями сайту.
2. Технологічна – відображає взаємодію між різними технологіями, що використовуються на сайті.
3. Дані – показує потоки даних між різними частинами сайту.
4. Архітектурна – відображає загальну структуру сайту, включаючи сервери, мережі та пристрої.

Наприклад, може бути наступна взаємодія на структурній схемі:

1. Користувач вводить адресу сайту в браузері.
2. Браузер надсилає запит на сервер.
3. Сервер обробляє запит, отримує необхідні дані з бази даних, обробляє їх за допомогою скриптів та формує відповідь у вигляді HTML-коду.
4. Браузер отримує HTML-код і відображає веб-сторінку користувачеві.

Для того, щоб розробити систему необхідно спроектувати можливі варіанти використання системи. Розробка схеми веб-сайту відбуватиметься за допомогою мови UML.

UML - це візуальна мова моделювання, що використовується для створення діаграм, які описують структуру та поведінку програмних систем. Вона дозволяє розробникам та іншим зацікавленим сторонам спілкуватися та документувати свої ідеї на зрозумілій мові.

UML використовує графічні нотації для представлення різних аспектів системи. Діаграми UML складаються з елементів, таких як класи, об'єкти, атрибути, методи, асоціації, залежності, агрегації та композиції. Ці елементи з'єднуються за допомогою різних типів ліній та символів, що дозволяє візуалізувати складні взаємодії між різними частинами системи.

Основними перевагами використання є зрозумілість – її легко розуміти як розробнику, так і користувачу; UML забезпечує спільну мову для спілкування між розробниками, аналітиками та менеджерами проєктів; можуть служити документацією для проєкту, що спрощує його підтримку та розвиток; моделювання UML допомагає виявити потенційні проблеми в дизайні системи на ранньому етапі розробки.

Це відобразить склад та взаємодію частин програмного забезпечення (рис.2.1).

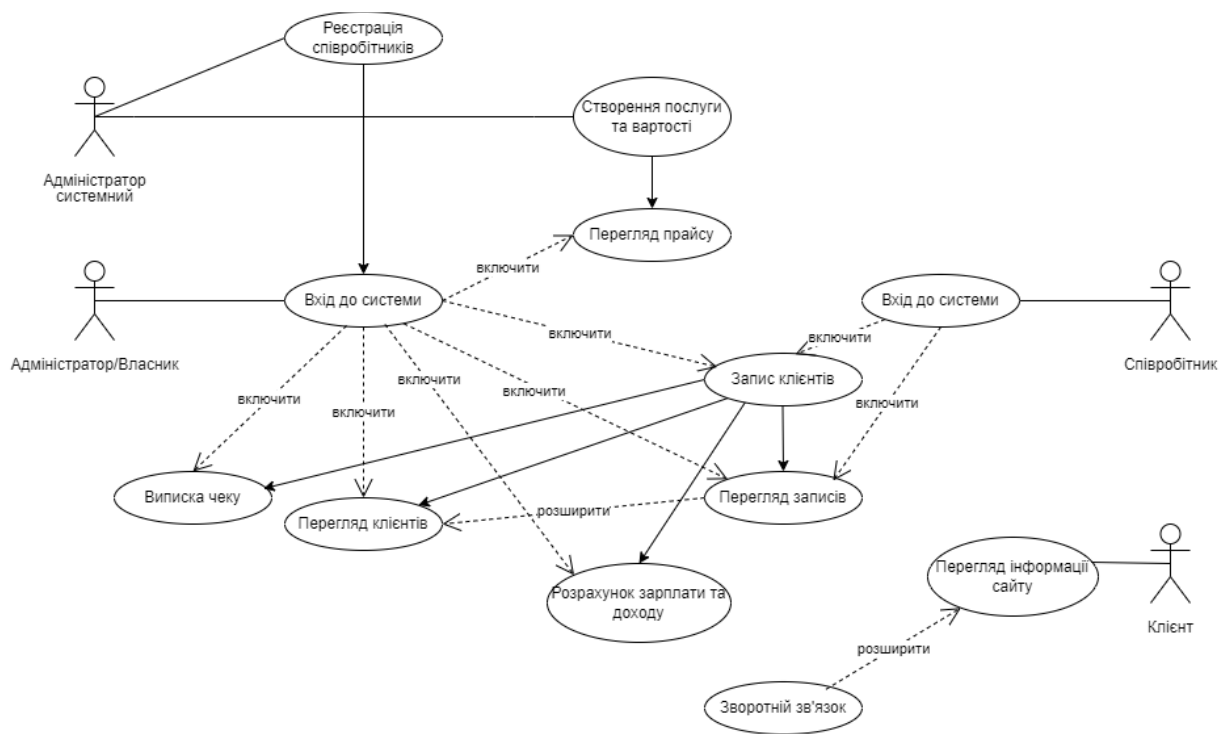


Рисунок 2.1. Діаграма варіантів використання системи

Дана діаграма відображає можливі зв'язки з системою. На ній вказані актори, що будуть враховані при проєктуванні та розробці сайту. Акторами тут є «Адміністратор системний», «Адміністратор/Власник», «Співробітник» та «Клієнт». При удосконаленні та при розширенні даних, дана діаграма може бути розширена та доповнена необхідними користувачами, функціями та зв'язками.

Опис діаграми варіантів використання наведено в таблицях 2.1 – 2.4.

Таблиця 2.1

Опис використання «Реєстрація»

Назва варіанту використання	Реєстрація
Основні актори	Системний адміністратор
Короткий опис	Користувач відповідатиме за реєстрацію працівників, створення послуг та їх вартості, додавання прайсу до послуг

Таблиця 2.2

Опис використання «Керування»

Назва варіанту використання	Керування
Основні актори	Адміністратор, власник
Короткий опис	Користувачі мають можливість переглядати прайс, записувати клієнтів, переглядати інформацію про клієнтів, переглядати інформацію про запити клієнтів, виписувати чек за обрану послугу, розраховувати заробітну плату та дохід

Таблиця 2.3

Опис використання «Взаємодія»

Назва варіанту використання	Керування
Основні актори	Співробітник
Короткий опис	Користувач має змогу при вході до системи записувати клієнтів, переглядати інформацію про попередні записи та про запити клієнта, надавати інформацію

Таблиця 2.4

Опис використання «Використання»

Назва варіанту використання	Використання
Основні актори	Клієнт
Короткий опис	Користувач має змогу переглядати інформацію на сайті та надавати зворотній зв'язок

Також розроблено структурну схему роботи сайту (рис. 2.2).



Рисунок 2.2. Структурна схема роботи сайту

Елементами даної схеми є:

1. Користувач – особа, яка переглядає сайт.
2. Веб-браузер - програма, що відображає веб-сторінки.
3. Веб-сервер - комп'ютер, що зберігає веб-файли і обробляє запити.
4. База даних - зберігає інформацію про послуги, кейси, команду, новини та інші дані сайту [1].
5. Скрипти - програми, що виконуються на сервері для обробки даних, взаємодії з базою даних та створення динамічного контенту.
6. HTML, CSS, JavaScript - мови програмування, що використовуються для побудови веб-сторінок [3].
7. Форми - інтерфейси для збору інформації від користувачів (наприклад, контактна форма, запит на послуги).
8. Система аналітики: Збирає статистику про відвідувачів сайту.

Структурна схема відображає наступну взаємодію:

1. Користувач вводить адресу сайту в браузері.

2. Браузер надсилає запит на веб-сервер.
3. Веб-сервер отримує запит, визначає потрібну сторінку (Послуги, Кейси, Про Нас...) та звертається до бази даних за відповідними даними.
4. Скрипти обробляють отримані дані та формують HTML-код для сторінки.
5. Веб-сервер надсилає HTML-код браузеру.
6. Браузер відображає сторінку користувачеві.
7. Форма зворотного зв'язку збирає відгуки від користувачів.
8. Інтеграція з CRM-системою збирає дані про взаємодію з клієнтами та допомагає в обробці запитів.

Дана схема показує основні принципи роботи. У подальшому можна використовувати більш складні технології та мати додаткові функції.

Також, для подальшої розробки, створено схему сайту, на якій будуть визначені всі сторінки сайту (рис. 2.3).

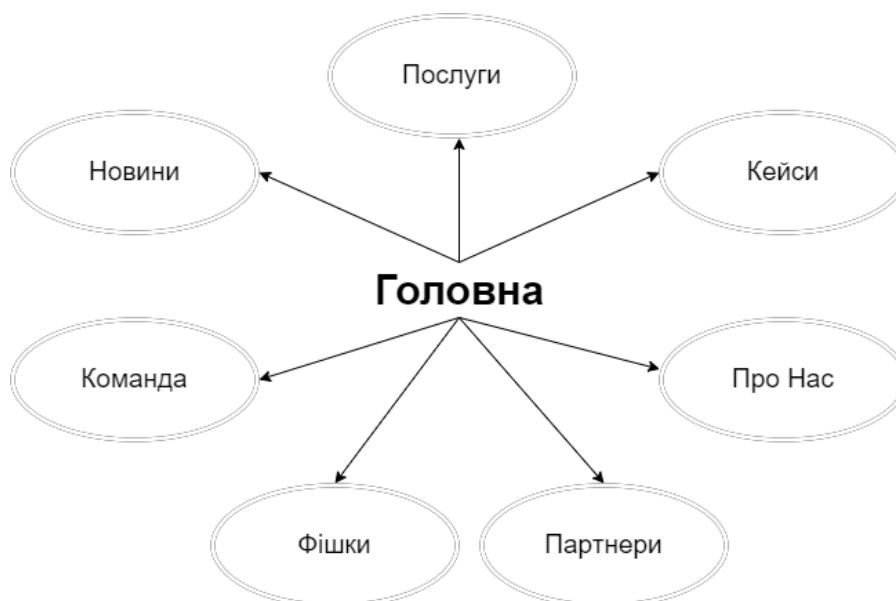


Рисунок 2.3. Схема сторінок сайту

2.2. Розробка функціональної схеми сайту

Функціональна схема сайту - це графічне представлення функцій та можливостей, які доступні на веб-сайті. Вона описує, як різні елементи сайту взаємодіють між собою та як користувач може використовувати сайт для досягнення своїх цілей.

Функціональна схема сайту може включати такі елементи:

1. Функції та можливості: це перелік функцій та можливостей, які доступні на сайті. Наприклад, це можуть бути функції пошуку, реєстрації користувача, додавання товару до кошика, оплати замовлення та інші.
2. Взаємодія користувача з сайтом: це опис того, як користувач може взаємодіяти з сайтом, виконуючи певні дії. Наприклад, користувач може шукати інформацію на сайті, додавати товари до кошика, оформляти замовлення тощо.
3. Процеси на сайті: це опис процесів, які відбуваються на сайті. Наприклад, це можуть бути процеси оплати замовлення, обробки запиту користувача, відправки повідомлень тощо.
4. Інтеграція з іншими сервісами: це опис інтеграції сайту з іншими сервісами, такими як соціальні мережі, електронна пошта, платіжні системи тощо.
5. Архітектура сайту: це опис структури сайту та його компонентів. Наприклад, це можуть бути база даних, сервер, програмне забезпечення та інші компоненти.

Функціональна схема сайту допомагає розробникам та дизайнерам краще зрозуміти функції та можливості сайту, а також спрощує процес розробки та впровадження нових функцій на сайт.

Система для колл-центру призначається як для клієнта, так і для співробітників. На наступному рисунку зображена можливість клієнта користуватись сайтом для отримання зворотного зв'язку (рис.2.4).

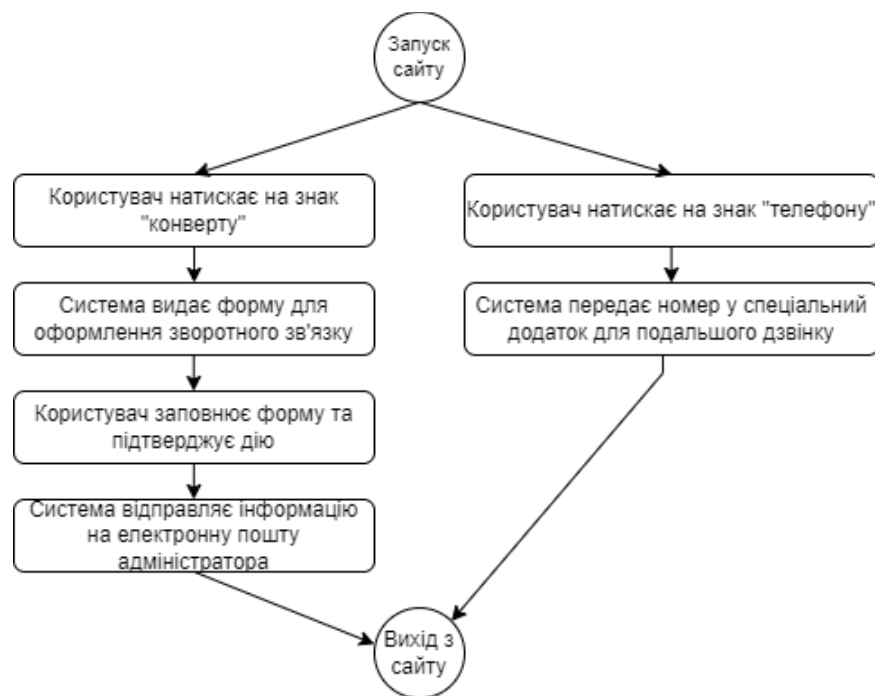


Рисунок 2.4. Функціональна схема заповнення заявки клієнтом для отримання зворотного зв'язку

При перегляді сайту клієнт може ознайомитись з інформацією про послуги, які надаються. Також є можливість отримати більш детальну інформацію з приводу послуги, яка зацікавила. Перший це натиснути на «конверт», після чого необхідно буде заповнити форму для зворотного зв'язку. Потім інформацію отримає адміністратор та зв'яжеться з клієнтом для консультування за наявними послугами. Другий – натиснути на знак «телефону», тоді система передає номер телефону у додаток для дзвінку і клієнт отримує дзвінок від оператора для отримання повноцінної інформації. Також оператор зможе проконсультувати, якщо клієнт матиме додаткові запитання, відповіді на які не отримав при перегляді інформації на сайті.

Також, створено схему для співробітника (рис. 2.5).

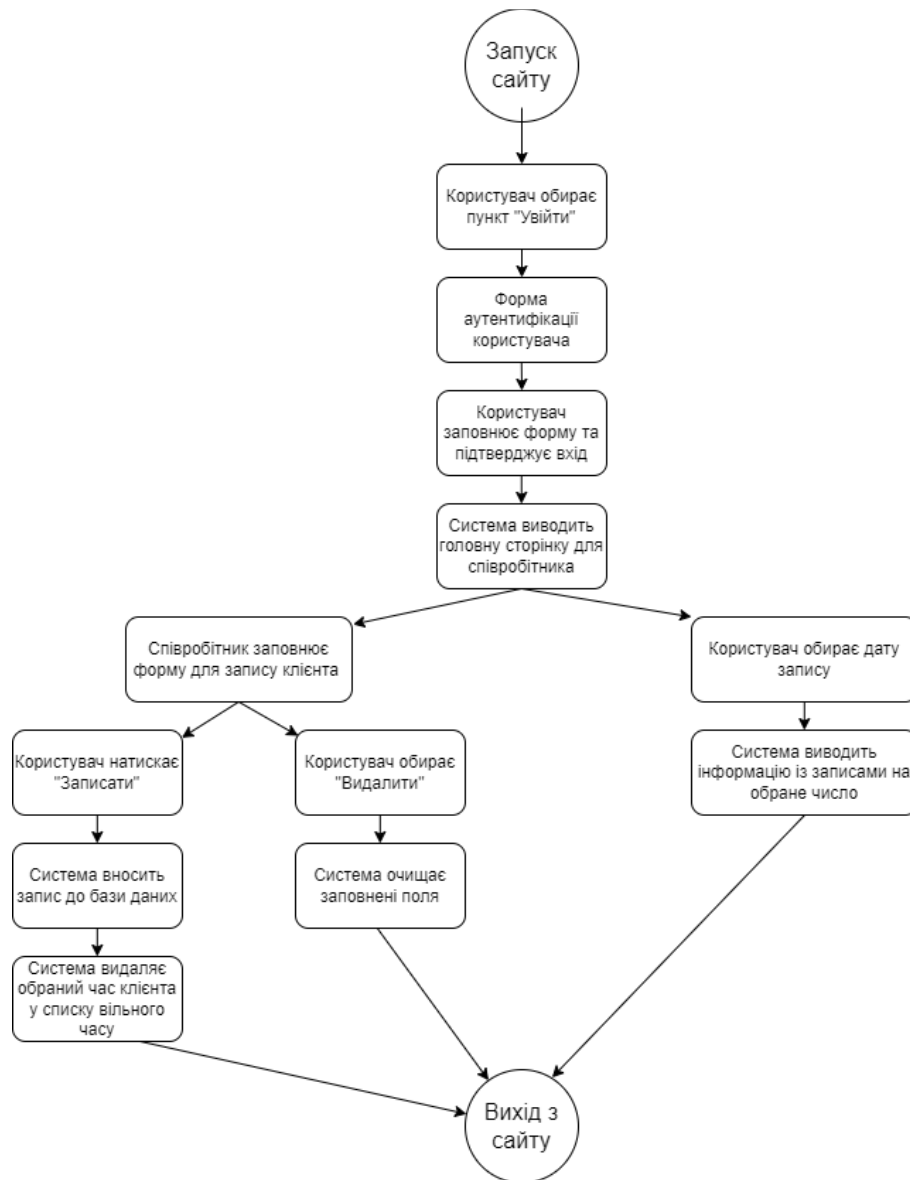


Рисунок 2.5. Функціональна схема користування співробітниками

Співробітник, після авторизації на сайті може зробити багато дій, для прикладу – робити запис клієнта на певну послугу. Це він може робити одразу через форму – натискає записати, вносить всю необхідну інформацію для запису та натискає зберегти. Інформація заноситься до бази даних. Або, можна зробити запис

В офісі також працюють адміністратори, які можуть зареєструвати співробітників та надавати їм доступ до редагування інформації на частині сайту. Як можна побачити на схемі (рис. 2.3), після авторизації в системі перед користувачем відображається сторінка для запису клієнтів. Заповнюючи

форму, інформація потрапляє до бази даних, після чого вона записується та співробітники можуть переглядати свої записи та записи своїх колег, та редагувати власні записи.

Велику кількість функцій виконує адміністратор. На рисунку 2.6 відображено функціональну схему використання сайту адміністратором/власником для роботи з інформацією, що пов'язана з клієнтурою. В першу чергу, він повинен увійти на сайт, пройшовши аутентифікацію. Після чого сайт переводить користувача на головну сторінку, де знаходяться функції запису клієнта та перегляд цих записів, які є такими ж як і у співробітника (рис. 2.6). Інша функція варіанта використання – це «Клієнти». Система видає список усіх клієнтів та форму його редагування. Доступні такі функції редагування, як «Видалити», «Додати», «Оновити», для їх використання потрібно заповнити форму та натиснути на потрібну кнопку [6].

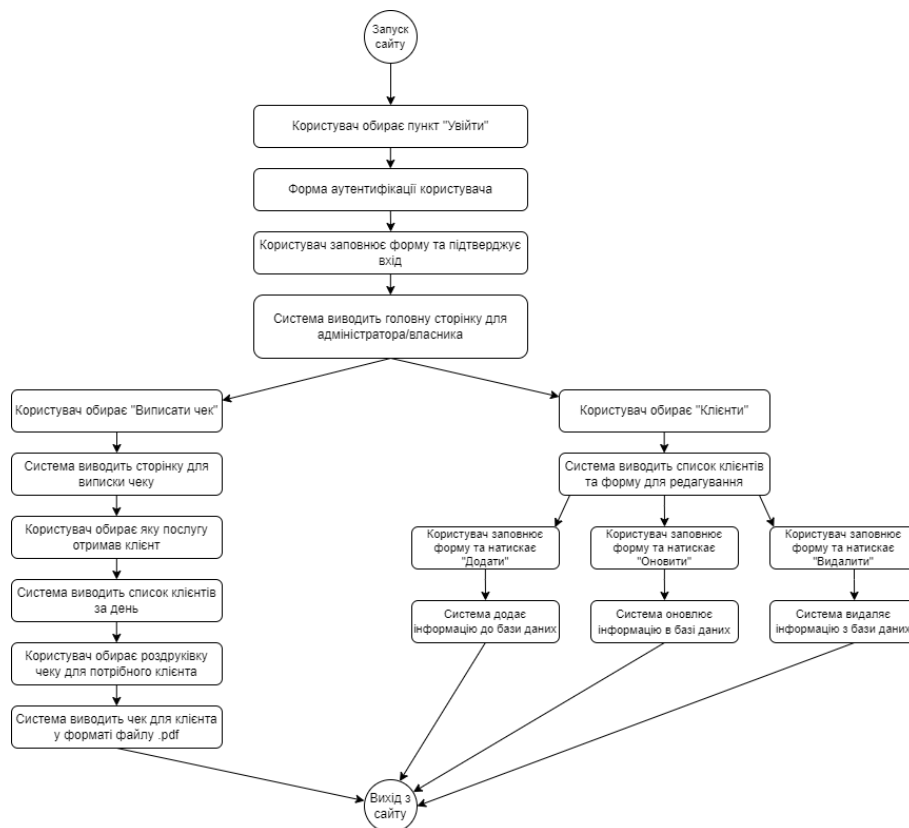


Рисунок 2.6. Функціональна схема користування адміністратором/власником

Після того як користувач вибере функцію варіанта використання – «Виписати чек», перед ним з’явиться спеціальна сторінка, де потрібно обрати послугу, яка надавалась клієнту. Після чого можна побачити повний список клієнтів, що були протягом дня записані. Адміністратор повинен обрати клієнта, для якого потрібно роздрукувати чек и перед ним з’явиться документ у форматі pdf.

При користуванні сайтом основними користувачами будуть працівники та клієнти колл-центру «ЛЕТСКОЛЛ». Сайт виконуватиме функції користувачів при взаємодії з інтерфейсом. Виконуючи функції, сайт виводить результат, що дозволяє знайти необхідну інформацію, пришвидшити діяльність управління та якісно та швидко формувати звіти.

Архітектурна схема веб-сайту для колл-центру – це візуальне представлення того, як різні компоненти веб-сайту взаємодіють один з одним. Вона показує структуру, взаємозв'язки та функції різних частин системи, дозволяючи зрозуміти, як веб-сайт працює як ціле.

Архітектурна схема складається із шести основних елементів:

- 1) клієнтська частина – це частина, з якою взаємодіють користувачі. Вона включає в себе веб-браузер, мобільний додаток або інше програмне забезпечення, що дозволяє користувачам взаємодіяти з веб-сайтом;
- 2) серверна частина - це сервер, який обробляє запити від клієнтів, зберігає дані та забезпечує функціональність сайту;
- 3) база даних - місце, де зберігаються всі дані, що пов'язані з веб-сайтом, включаючи інформацію про користувачів, прецеденти, квитки, статистики тощо.
- 4) Інтерфейси - способи взаємодії між різними частинами системи. наприклад, API (Application Programming Interface) – це набір правил, що дозволяють різним частинам системи спілкуватися між собою;

- 5) системи безпеки - захищають веб-сайт від несанкціонованого доступу, зловмисних атак та інших загроз;
- 6) системи моніторингу - відстежують продуктивність сайту, виявляють помилки та забезпечують безперебійну роботу.

Архітектурні схеми поділяються на два типи. Перший: трирівнева архітектура – класична модель, що розділяє систему на три рівні: рівень представлення – веб-браузер або мобільний додаток; рівень бізнес-логіки – сервер, який обробляє запити, виконує логіку та управляє даними; рівень доступу до даних – база даних, яка зберігає дані. Другий: мікросервісна архітектура – розділяє систему на дрібні незалежні сервіси, які можуть працювати окремо. Це дозволяє легко масштабувати систему, додавати нові функції та швидко реагувати на зміни.

Архітектурна схема будь-якого сайту складається із наступних компонентів:

1. Фронтенд – React.js, Next.js (або інший фреймворк).
2. Бекенд – Сервер (AWS, GCP, Azure).
3. База даних – MySQL [5].
4. API - RESTful API для взаємодії між фронтендом та бекендом.
5. Сервіси: Інші служби, як CRM, ERP, телефонні системи.

Наприклад, опишемо наступну взаємодію:

1. Користувач: Відвідує веб-сайт колл-центру та взаємодіє з різними сторінками (Головна, Послуги, Контакти).
2. Фронтенд: Отримує запити від користувача та відправляє їх на сервер через API.
3. API: Отримує запити від фронтенду, перевіряє їх та передає на сервер.
4. Сервер: Обробляє запити, взаємодіє з базою даних MySQL[6], а також може взаємодіяти з іншими службами (CRM, ERP, телефонні системи).

5. База даних: Зберігає інформацію про послуги, контактні дані та іншу інформацію веб-сайту.
6. Сервер: Повертає відповідь на API з даними, отриманими з бази даних або інших служб.
7. API: Передає відповідь на фронтенд.
8. Фронтенд: Отримавши відповідь, відображає інформацію на сторінці користувача.

Реалізація даного опису зображено на наступній схемі (рис. 2.7).

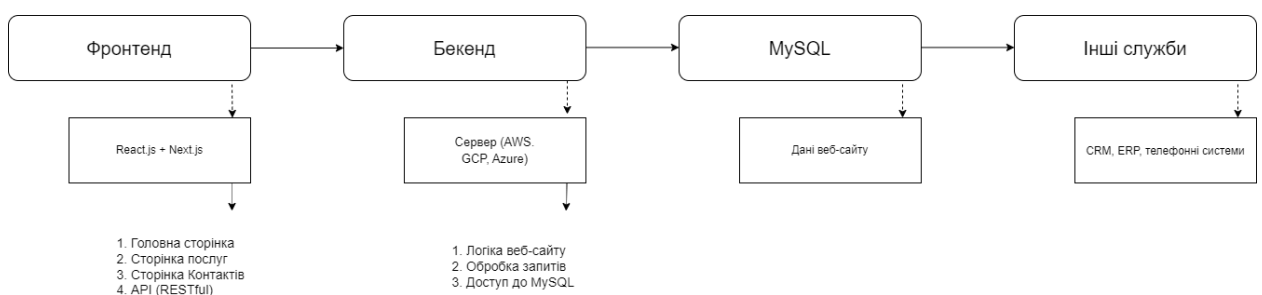


Рисунок 2.7. Архітектурна схема сайту

2.3. Проектування та реалізація алгоритмів роботи системи

Проектування алгоритмів роботи системи веб-сайту колл-центру - це процес розробки чітких інструкцій та логіки для кожної функції сайту, які забезпечать правильну обробку запитів користувачів та ефективну взаємодію з різними компонентами системи.

Кожна програма має свій порядок роботи, це називають алгоритмом.

Алгоритм – набір інструкцій, які описують порядок дій виконавця, щоб досягти результату розв’язання задачі за скінченну кількість дій. Для візуалізації алгоритмів часто використовують блок-схеми.

Можемо визначити наступні етапи проектування алгоритмів:

- 1) визначення потреб та цілей;
- 2) аналіз функціональності;
- 3) розробка алгоритмів для кожного модуля;

4) тестування та оптимізація.

Система напряду пов'язана з базою даних, тому наведемо декілька прикладів алгоритмів базових програм, які використовуються при роботі.

Команда SELECT використовується для отримання певних даних з певної таблиці (рис. 2.8).

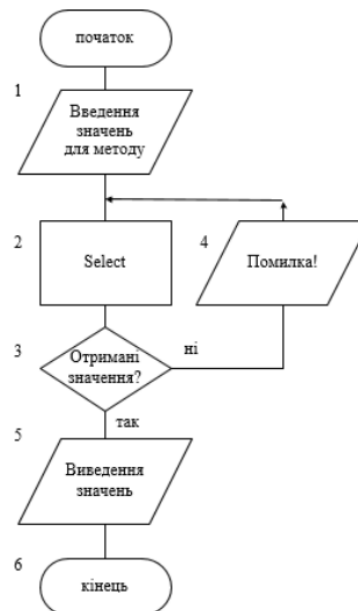


Рисунок 2.8. Алгоритм команди «Select»

Команда DELETE використовують для видалення записів з таблиці (рис. 2.9).



Рисунок 2.9. Алгоритм команди «Delete»

Без підключення до бази даних використання системи буде марним, адже це головний алгоритм при роботі (рис. 2.10).



Рисунок 2.10. Приклад алгоритму підключення до бази даних

Проектування алгоритмів роботи системи веб-сайту колл-центру - це складний процес, який вимагає глибокого розуміння функціональності сайту, потреб користувачів та принципів розробки веб-додатків. Правильно розроблені алгоритми забезпечують ефективну роботу сайту, покращують досвід користувачів та допомагають досягти цілей бізнесу.

Реалізація алгоритмів роботи системи веб-сайту колл-центру передбачає розробку логіки та послідовності дій для обробки запитів користувачів та взаємодії з різними компонентами системи.

Основними алгоритмами є:

1. Обробка запиту користувача:

1.1. Прийом запиту – веб-сервер отримує запит від браузера користувача.

1.2. Визначення типу запиту – сервер аналізує запит, визначаючи яка сторінка або функція сайту була запрошена.

1.3. Виконання запиту – сервер знаходить відповідний файл HTML на сервері та відправляє його браузеру або сервер виконує скрипти, які отримують дані з бази даних, обробляють їх та формують відповідь для користувача [11].

1.4. Відправка відповіді – веб-сервер відправляє результат обробки запиту браузеру користувача.

2. Взаємодія із CRM-системою:

2.1. Інтеграція – веб-сайт може бути інтегрований з CRM-системою, що дозволяє збирати дані взаємодії з клієнтами, оброблювати запити та відстежувати статус звернення.

2.2. Обмін даними – сайт може передавати дані про користувачів та їх дії в CRM-системі.

2.3. Відстеження – сайт може отримувати дані з CRM-системі.

3. Взаємодія із базою даних:

3.1. Отримання даних – скрипти веб-сайту звертаються до бази даних для отримання необхідної інформації.

3.2. Обробка даних – скрипти обробляють отримані дані, формуючи відповідь для користувача або зберігаючи нові дані в базі даних.

3.3. Оновлення бази даних – скрипти можуть оновлювати дані в базі даних.

Реалізація алгоритмів складається із різних систем та фреймворків. Для прикладу – мова програмування може бути PHP, Python, JavaScript; фреймворки - Laravel, Django, Node.js; база даних - MySQL, PostgreSQL, Oracle [6].

Реалізація алгоритмів роботи веб-сайту колл-центру вимагає глибокого розуміння функціональності системи, взаємодії між компонентами, а також досвіду в розробці веб-додатків.

2.4. Проектування бази даних

Логічне проектування бази даних - це процес створення моделі бази даних, яка описує структуру та взаємозв'язки між даними, що зберігаються в базі даних. Цей процес включає в себе розробку схеми бази даних, визначення таблиць, полів та зв'язків між ними.

На етапі логічного проектування бази даних створюється концептуальна модель, яка описує сутності та їх взаємозв'язки. Для цього можуть бути використані різні методи, такі як ER-модель або UML-діаграми.

Після створення концептуальної моделі, проводиться процес нормалізації, який допомагає зменшити дублювання даних та забезпечити цілісність бази даних. Нормалізація полягає в розбитті таблиць на менші, щоб уникнути повторення даних та забезпечити їх коректну обробку.

Після проведення нормалізації створюється логічна схема бази даних, яка описує структуру бази даних з точки зору таблиць, полів та зв'язків між ними. Ця схема може бути створена за допомогою спеціальних інструментів проектування баз даних, таких як ER-діаграми або UML-діаграми.

ER-модель – це концептуальна модель даних, що використовується для представлення сутностей (об'єктів) та взаємозв'язків між ними в інформаційній системі. Вона дозволяє візуалізувати структуру бази даних та зрозуміти логічні зв'язки між даними.

Основними елементами є сутності, атрибути та зв'язки. Типи зв'язків поділяються на:

- 1) один-до-одного – одна сутність пов'язана з однією іншою;
- 2) один-до-багатьох – одна сутність пов'язана з кількома іншими;
- 3) багато-до-багатьох – кілька сутностей пов'язані з кількома іншими.

Логічне проектування бази даних є важливим етапом у розробці будь-якої бази даних, оскільки воно допомагає забезпечити коректну та ефективну роботу з даними, зменшити кількість помилок та покращити продуктивність системи.

На підставі ER–моделі бази даних, база даних що проектується, має містити наступні таблиці:

- customers (клієнти);
- employees (співробітники) ;
- orders (замовлення);
- services (послуги);
- orderstatuschanges (зміна статусу замовлення);
- callrecordings (записи дзвінків).

Параметри атрибутів таблиць, які описано вище, вказані на рисунках 2.11 - 2.16.

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	id 🗑️	int			Hi	Немає		AUTO_INCREMENT
2	name	varchar(255)	utf8mb4_0900_ai_ci		Hi	Немає		
3	phone	varchar(20)	utf8mb4_0900_ai_ci		Hi	Немає		
4	email	varchar(255)	utf8mb4_0900_ai_ci		Hi	Немає		
5	status	enum('Active', 'Inactive')	utf8mb4_0900_ai_ci		Hi	Active		
6	created_at	timestamp			Hi	CURRENT_TIMESTAMP		DEFAULT_GENERATED
7	updated_at	timestamp			Hi	CURRENT_TIMESTAMP		DEFAULT_GENERATED

Рисунок 2.11. Список атрибутів сутності Клієнти

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	id 🗑️	int			Hi	Немає		AUTO_INCREMENT
2	name	varchar(255)	utf8mb4_0900_ai_ci		Hi	Немає		
3	position	varchar(255)	utf8mb4_0900_ai_ci		Hi	Немає		
4	phone	varchar(20)	utf8mb4_0900_ai_ci		Hi	Немає		
5	email	varchar(255)	utf8mb4_0900_ai_ci		Hi	Немає		
6	status	enum('Active', 'Inactive')	utf8mb4_0900_ai_ci		Hi	Active		
7	created_at	timestamp			Hi	CURRENT_TIMESTAMP		DEFAULT_GENERATED
8	updated_at	timestamp			Hi	CURRENT_TIMESTAMP		DEFAULT_GENERATED

Рисунок 2.12. Список атрибутів сутності Співробітники

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	id 🗑️	int			Hi	Немає		AUTO_INCREMENT
2	customers_id 🗑️🗑️	int			Hi	Немає		
3	services_id 🗑️🗑️	int			Hi	Немає		
4	order_date	date			Hi	Немає		
5	status	enum('New', 'In progress', 'Completed', 'Cancelled...')	utf8mb4_0900_ai_ci		Hi	New		
6	created_at	timestamp			Hi	CURRENT_TIMESTAMP		DEFAULT_GENERATED
7	updated_at	timestamp			Hi	CURRENT_TIMESTAMP		DEFAULT_GENERATED

Рисунок 2.13. Список атрибутів сутності Замовлення

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	id 🗑️	int			Hi	Немає		AUTO_INCREMENT
2	name	varchar(255)	utf8mb4_0900_ai_ci		Hi	Немає		
3	description	text	utf8mb4_0900_ai_ci		Hi	Немає		
4	price	decimal(10,2)			Hi	Немає		
5	created_at	timestamp			Hi	CURRENT_TIMESTAMP		DEFAULT_GENERATED
6	updated_at	timestamp			Hi	CURRENT_TIMESTAMP		DEFAULT_GENERATED

Рисунок 2.14. Список атрибутів сутності Послуги

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	id 🗑️	int			Hi	Немає		AUTO_INCREMENT
2	orders_id 🗑️🔒	int			Hi	Немає		
3	employees_id 🗑️🔒	int			Hi	Немає		
4	status	enum('New', 'In progress', 'Completed', 'Cancelled')	utf8mb4_0900_ai_ci		Hi	Немає		
5	changed_at	datetime			Hi	Немає		
6	created_at	timestamp			Hi	CURRENT_TIMESTAMP		DEFAULT_GENERATED
7	updated_at	timestamp			Hi	CURRENT_TIMESTAMP		DEFAULT_GENERATED

Рисунок 2.15 Список атрибутів сутності Зміна статусу замовлення

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	id 🗑️	int			Hi	Немає		AUTO_INCREMENT
2	customers_id 🗑️🔒	int			Hi	Немає		
3	employees_id 🗑️🔒	int			Hi	Немає		
4	call_date	datetime			Hi	Немає		
5	call_duration	int			Hi	Немає		
6	call_notes	text	utf8mb4_0900_ai_ci		Hi	Немає		
7	created_at	timestamp			Hi	CURRENT_TIMESTAMP		DEFAULT_GENERATED
8	updated_at	timestamp			Hi	CURRENT_TIMESTAMP		DEFAULT_GENERATED

Рисунок 2.16 Список атрибутів сутності Записи дзвінків

В результаті проєктування бази даних створено логічну модель, що відповідає реляційній моделі даних. Всі відношення знаходяться у третій нормальній формі, всі атрибути перебувають у повній функціональній залежності від ключа.

Концептуальне проєктування бази даних - це процес створення моделі бази даних, яка описує загальну структуру та взаємозв'язки між сутностями, що зберігаються в базі даних. Цей процес передуює фізичному проєктуванню бази даних і зосереджується на визначенні концепційних аспектів бази даних без прив'язки до конкретного засобу зберігання.

Під час концептуального проєктування бази даних створюється концептуальна модель, яка описує сутності (або об'єкти) та їх атрибути, а також взаємозв'язки між ними. Ця модель може бути створена за допомогою різних методів, таких як ER-модель (Entity-Relationship) або UML-діаграми (Unified Modeling Language).

Основна мета концептуального проектування полягає в тому, щоб зрозуміти потреби користувачів та бізнес-логіку системи, і на основі цього розробити модель бази даних, яка відображає ці потреби. Концептуальна модель не залежить від конкретного засобу зберігання даних, тому вона є незалежною від платформи або системи управління базами даних.

Концептуальне проектування бази даних включає в себе такі етапи, як збір інформації, ідентифікація сутностей та їх атрибутів, визначення взаємозв'язків між сутностями, аналіз потреб користувачів та бізнес-правил, і розробку концептуальної моделі на основі цих даних.

Концептуальне проектування бази даних є важливим етапом у розробці системи, оскільки воно допомагає зрозуміти та визначити основну структуру та взаємозв'язки даних, що зберігаються в базі даних. Це створює основу для подальшого фізичного проектування бази даних та реалізації системи управління базами даних.

Інструментами концептуального проектування є:

1. Діаграми ER: Найпоширеніший інструмент для візуалізації концептуальної моделі даних.
2. Мови моделювання даних: SQL, UML, XML, RDF.
3. Програми для моделювання даних: PowerDesigner, ERwin, MySQL Workbench.

До початку створення сайту, необхідно спроектувати базу даних. Її суть у створенні концептуальної моделі даних для частини підприємства, що буде аналізуватись. При розробці спочатку необхідно визначити сутності та встановити зв'язки між ними.

Після аналізу предметної області, визначаємо наступні сутності:

- клієнти
- співробітники
- замовлення
- послуги
- зміна статусу замовлення

- записи дзвінків

Таким чином маємо ER-модель бази даних (рис. 2.17)

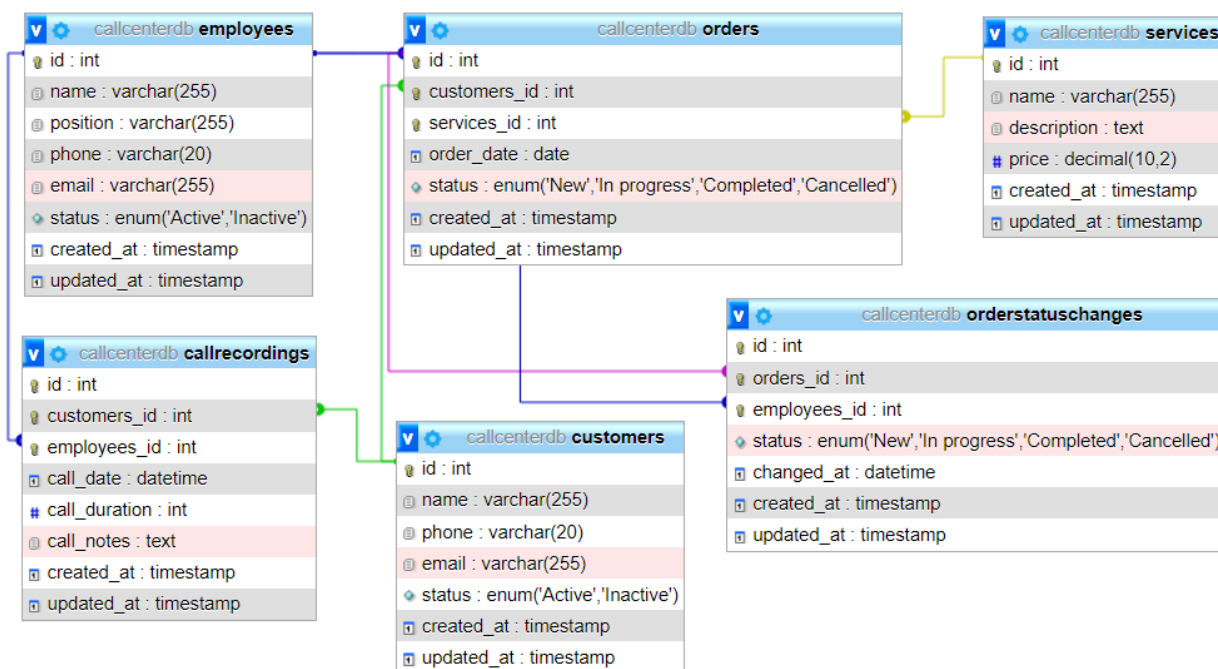


Рисунок 2.17. ER-модель бази даних

Детальніше переглянемо кожен зв'язок сутностей.

Customers (клієнти) і Orders (замовлення) - Один клієнт може мати декілька замовлень.

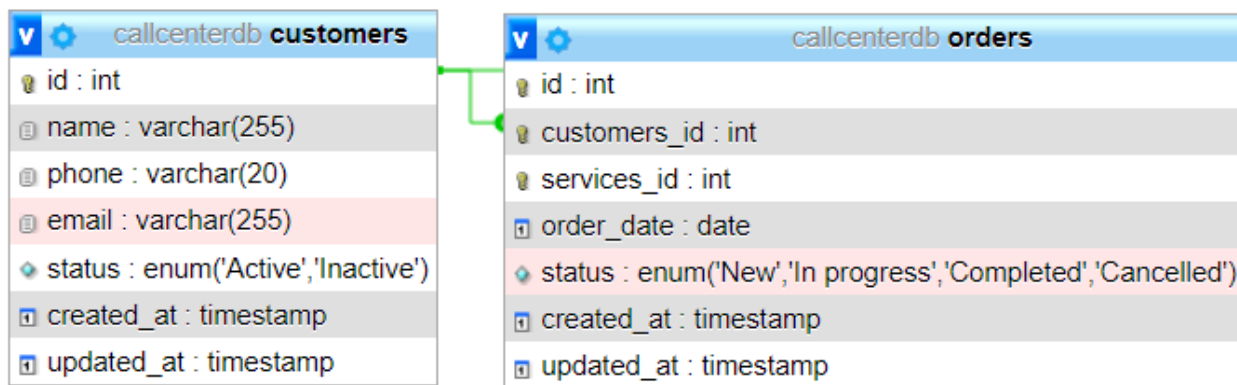


Рисунок 2.18. Зв'язок сутностей Клієнти - Замовлення

Між сутностями Services (послуги) і Orders (замовлення) - Одна послуга може бути включена в декілька замовлень.

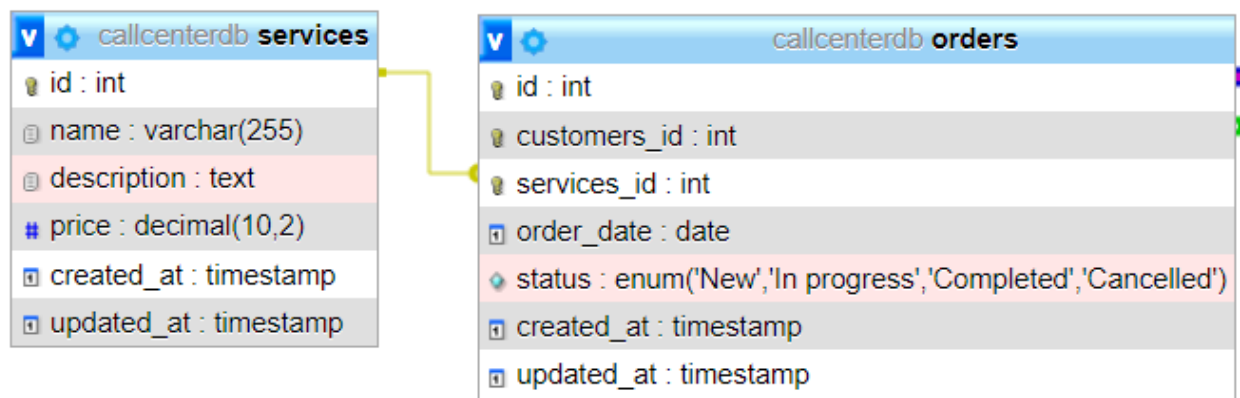


Рисунок 2.19. Зв'язок сутностей Послуги - Замовлення

Між сутностями Employees (співробітники) і Orders (замовлення) утворюється зв'язок, адже один співробітник може обробляти декілька замовлень.

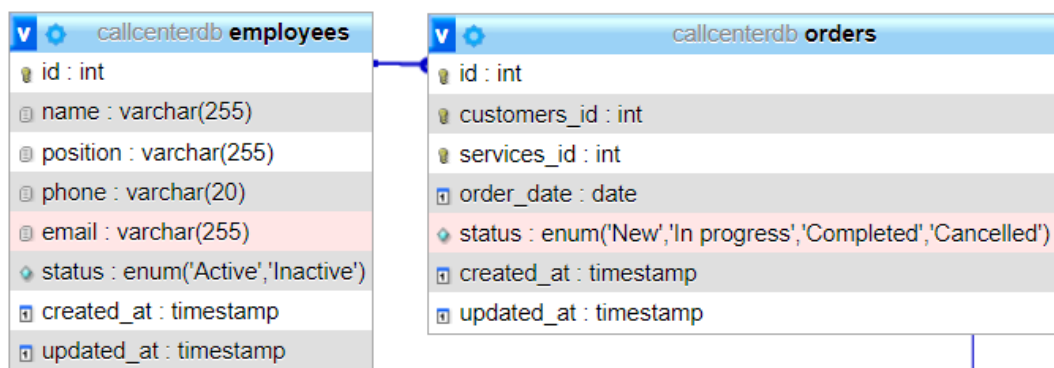


Рисунок 2.20. Зв'язок сутностей Співробітники - Замовлення

Між сутностями Customers (клієнти) – CallRecordings (запис дзвінків) існує зв'язок, адже один клієнт може мати багато записів дзвінків.

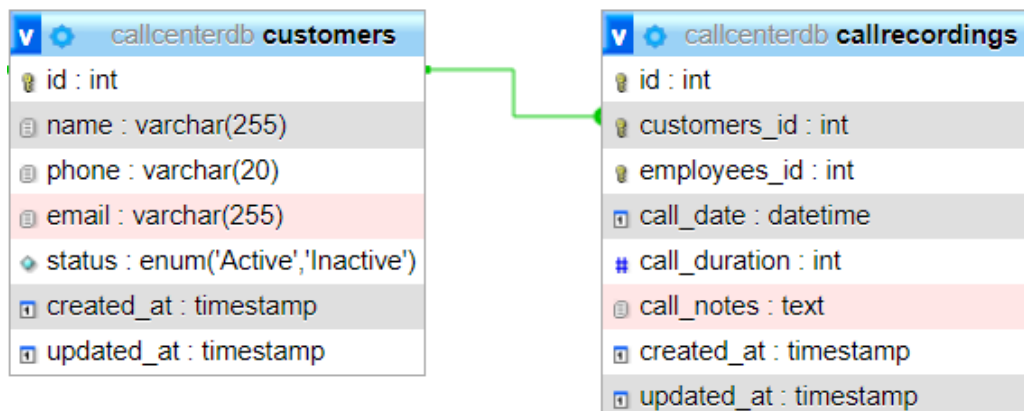


Рисунок 2.21. Зв'язок сутностей Клієнти – Записи дзвінків

Між сутностями Employees (співробітники) – CallRecordings (записи дзвінків) утворюється зв'язок, тому що один співробітник може приймати участь у багатьох дзвінках.

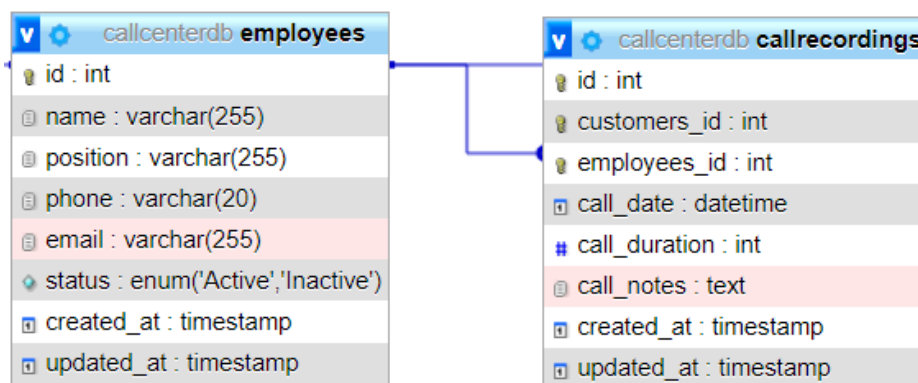


Рисунок 2.22. Зв'язок сутностей Співробітники – Записи дзвінків

Між сутностями Orders (замовлення) – OrderStatusChanges (зміна статусу замовлення) з'являється зв'язок, адже одне замовлення може мати декілька змін статусу.

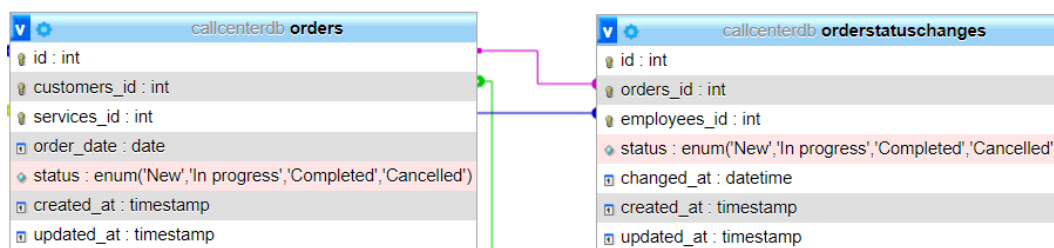


Рисунок 2.23. Зв'язок сутностей Замовлення - Зміна статусу замовлення

Між сутностями Employees (співробітники) і OrderStatusChanges (зміна статусу замовлення) існує зв'язок, тому що один співробітник може змінювати статус багатьох замовлень.

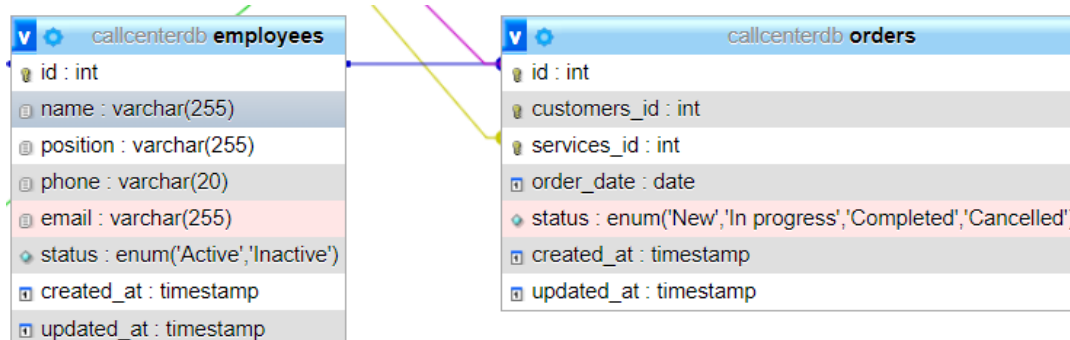


Рисунок 2.24. Зв'язок сутностей Співробітники - Зміна статусу замовлення

Сучасні системи все частіше використовують формат JSON для обміну даними. База даних документів JSON пропонує зручне та гнучке рішення для зберігання та обробки такої інформації.

Нативною підтримкою JSON є те, що база даних документів JSON розуміє мову JSON, забезпечуючи просту інтеграцію з вашими існуючими системами. Немає обмежень схемою - можна зберігати дані у будь-якому форматі JSON, не турбуючись про визначення структури. Дані зберігаються у стандартних типах даних SQL (VARCHAR2, CLOB, BLOB), що забезпечує гнучкість та зручність доступу [14].

Для взаємодії з JSON-даними використовуються стандартні інструменти, такі як OCI, .NET та JDBC. База даних оптимізована для роботи з даними JSON, забезпечуючи швидкий доступ та обробку запитів. Практичними перевагами є:

1. Персоналізація - можна зберігати детальну інформацію про клієнтів, що дозволяє надавати їм персоналізовані пропозиції та покращувати взаємодію.
2. Ефективність - дані JSON можна легко реплікувати, що спрощує розширення системи та покращує її продуктивність.

База даних документів JSON - це потужний інструмент, який дозволяє легко зберігати, обробляти та використовувати дані JSON, роблячи систему більш гнучкою та ефективною.

2.5. Вимоги для розробки та розробка дизайну веб-сайту

Розробка веб-сайту для колл-центру вимагає врахування різноманітних вимог, що допоможуть забезпечити ефективну та зручну роботу колл-центру. Основні вимоги до розробки веб-сайту для колл-центру можуть включати:

1. Функціональність - веб-сайт повинен мати всі необхідні функції для роботи колл-центру, такі як система керування дзвінками, обробка

запитів користувачів, управління клієнтською базою даних, графіки роботи операторів, інтеграція з іншими системами (наприклад, CRM або Helpdesk).

2. Керованість - веб-сайт повинен мати зручний та інтуїтивно зрозумілий інтерфейс для керування всіма функціями колл-центру. Операторам має бути легко навчатися та використовувати систему без особливих зусиль.
3. Користувацький досвід - веб-сайт повинен бути легким у використанні та мати привабливий та зручний дизайн. Користувачам має бути зручно шукати інформацію, звертатися до операторів та отримувати відповіді на свої запитання.
4. Масштабованість - веб-сайт повинен бути готовим до масштабування, оскільки кількість операторів та обсяг роботи колл-центру можуть зростати з часом. Система повинна бути готовою обробляти багато запитів одночасно та масштабуватися для задоволення потреб колл-центру.
5. Безпека - веб-сайт повинен мати високий рівень безпеки, оскільки колл-центри обробляють чутливу інформацію про клієнтів. Система повинна мати захист від несанкціонованого доступу, а також механізми резервного копіювання та відновлення даних.
6. Інтеграція - веб-сайт повинен бути готовим для інтеграції з іншими системами, які використовуються в колл-центрі, наприклад, CRM-системами, електронною поштою або системами онлайн-чату.
7. Аналітика - веб-сайт повинен мати можливості аналітики та статистики, щоб дозволити керівництву колл-центру аналізувати продуктивність операторів, тренди запитань користувачів та іншу корисну інформацію.
8. Підтримка - розробник повинен забезпечити підтримку та поновлення програмного забезпечення, щоб уникнути проблем з безпекою та недоступністю.

Розглянемо більш детально для нашого проєкту необхідні вимоги.

Функціональні вимоги повинні визначати основні функції та можливості. Вони включають в себе:

- а) система керування дзвінками – сайт повинен мати можливість приймати та маршрутизувати вхідні дзвінки до доступних операторів;
- б) обробка запитів користувачів – необхідно щоб сайт приймав запити користувачів через різні канали, такі як телефон, електронна пошта або онлайн-чат;
- в) управління клієнтською базою даних - сайт повинен мати можливість зберігати та керувати інформацією про клієнтів, включаючи контактну інформацію, історію запитів та іншу корисну інформацію, система повинна також мати можливість створювати та керувати базами знань для операторів;
- г) інтеграція з іншими системами - сайт повинен мати можливість інтегруватися з іншими системами, які використовуються в колл-центрі, crm-системами проєктів, електронною поштою та системами онлайн-чату;
- д) система запису розмов – сайт матиме функціонал запису розмов, що дозволяє зберегти запис розмови між оператором та клієнтом для подальшого аналізу або навчання.

Доступність:

- 1. Вибір надійного хостингу, який забезпечує безперебійну роботу веб-сайту.
- 2. Здатність веб-сайту обробляти велику кількість запитів, особливо під час пікових навантажень.
- 3. Можливість доступу з різних пристроїв: зручний доступ до веб-сайту з різних пристроїв (комп'ютери, планшети, мобільні телефони).

4. Швидкість завантаження: швидке завантаження сторінок веб-сайту для забезпечення комфортного користування.

Користувацький досвід:

- 1) інтерфейс веб-сайту має бути простим та інтуїтивно зрозумілим, навіть для користувачів, які не мають досвіду роботи з подібними системами;
- 2) інформація на веб-сайті має бути структурована таким чином, щоб користувачі могли легко знайти потрібну інформацію;
- 3) зручний та логічний механізм навігації по веб-сайту;
- 4) естетично привабливий дизайн, який відповідає бренду компанії.

Можливість налаштування:

1. Здатність налаштовувати інтерфейс веб-сайту під потреби колл-центру, в залежності від вимог клієнта (змінювати колірну гамму, додавати логотипи, налаштовувати функціональність).
2. Можливість налаштовувати права доступу до різних розділів веб-сайту для різних груп користувачів.
3. Здатність інтегрувати веб-сайт з іншими системами, що використовуються в колл-центрі, у даному випадку, з CRM та системою аналітики.

Безпека – має бути налаштовано шифрування передачі даних між сайтом та клієнтами; забезпечення безпеки шляхом регулярного оновлення. Також, має бути проведення регулярних аудитів безпеки для виявлення та усунення можливих вразливостей.

Ефективний веб-сайт для колл-центру може значно покращити його роботу, підвищити задоволеність клієнтів та збільшити продуктивність.

Розробка дизайну веб-сайту є надзвичайно актуальним процесом у сучасному світі, оскільки Інтернет став невід'ємною частиною нашого життя.

Веб-сайти використовуються як основний засіб комунікації та представлення інформації для бізнесу, уряду, неприбуткових організацій, освітніх установ, медичних закладів та багатьох інших сфер діяльності.

Основна актуальність розробки дизайну веб-сайту полягає в наступних аспектах:

1. Перші враження: дизайн веб-сайту є першим, з чим користувач зустрічається при відвідуванні сайту. Враження, яке вони отримують, може вплинути на їхнє рішення залишитися на сайті або покинути його. Гарний дизайн забезпечує професійний та привабливий вигляд сайту, що стимулює користувачів залишатися та досліджувати додатковий контент.
2. Використання бренду: дизайн веб-сайту може бути ефективним засобом показати брендову ідентичність компанії. Використання корпоративного стилю, кольорових схем, логотипів та інших елементів брендингу допомагає створити єдиний образ компанії та підкреслити її цінності.
3. Зручність використання: добре розроблений дизайн сприяє зручності використання сайту. Інтуїтивно зрозуміле розташування елементів, легка навігація та швидкий доступ до потрібної інформації роблять сайт приємним для користувача. Це особливо важливо для колл-центрів, які мають багато клієнтів та потребують швидкого доступу до інформації.
4. Адаптивний дизайн: зростання використання мобільних пристроїв для доступу до Інтернету зумовлює необхідність адаптивного дизайну. Веб-сайти повинні добре виглядати та працювати на різних пристроях та розмірах екрану. Адаптивний дизайн забезпечує оптимальне відображення контенту на будь-якому пристрої, що покращує користувацький досвід.
5. Конкурентна перевага: у сучасному цифровому світі конкуренція серед веб-сайтів є дуже великою. Гарний дизайн може стати конкурентною

перевагою, яка виділить ваш сайт серед інших. Привабливий та функціональний дизайн може залучити нових клієнтів та покращити їх задоволення від користування сайтом.

Усе це показує, що розробка дизайну веб-сайту є надзвичайно актуальною, оскільки вона сприяє поліпшенню користувацького досвіду, покращенню конкурентоспроможності та позиціонуванню бренду компанії.

Саме через це, було розроблено дизайн для даного проєкту. Його часткову реалізацію для версії комп'ютера (десктопної версії) можна переглянути на рис. 2.25, адже більшість блоків повторюються і змінюється лише наповнення.

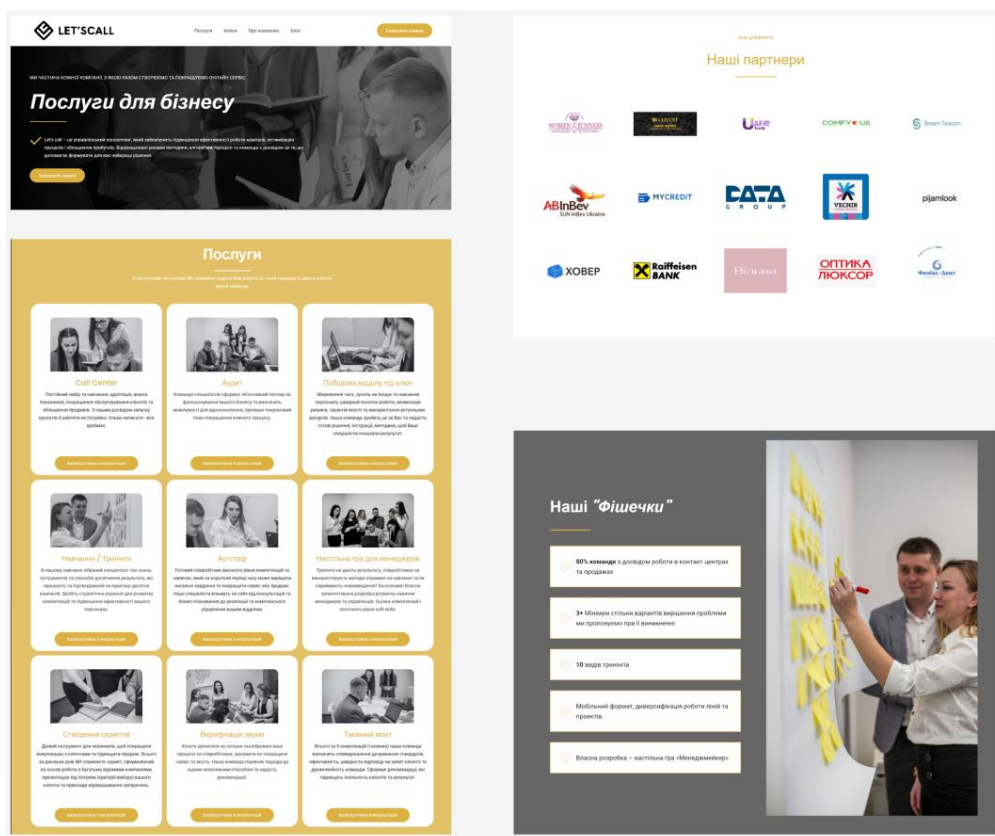


Рисунок 2.25. Дизайн блоків сторінки для десктопної версії

Реалізація для розмірів планшету (рис. 2.26).

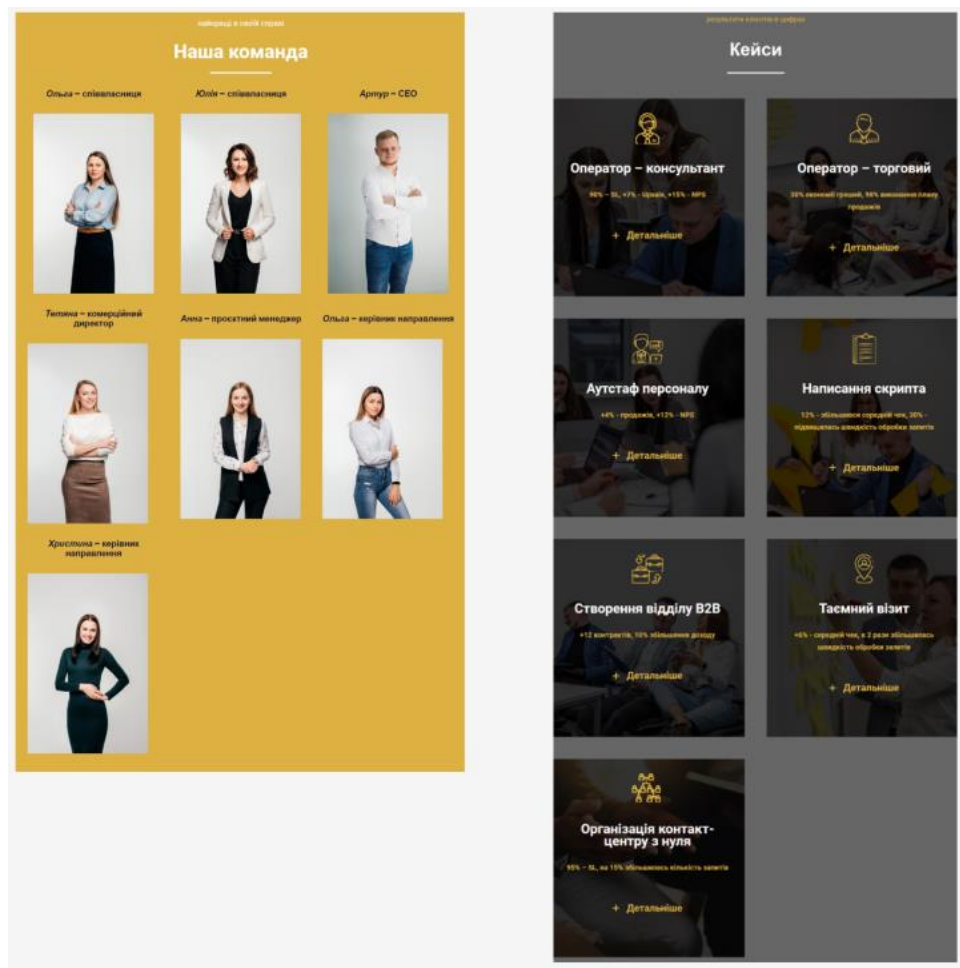


Рисунок 2.26. Дизайн блоків сторінки для планшетної версії

Також було реалізовано дизайн для мобільної версії проєкту. Дизайн адаптивний та зручний. В цьому можна переконатись переглянувши повну версію проєкту.

Висновки до другого розділу

У другому розділі були спроектовані схеми роботи сайту для клієнта, адміністраторів та співробітників компанії. Це необхідно, щоб зрозуміти логіку, за якою здійснюватиметься обробка даних при підключенні до бази даних. Також, було визначено основні алгоритми, які використовуються при створенні бази даних, для досягнення результату розв’язання задачі за скінченну кількість дій.

Спроектовано базу даних, яка відповідає за запис даних, їх введення та виведення. Було визначено сутності та створення зв'язків між ними для подальшого використання. В результаті чого створено логічну модель, що відповідає реляційній моделі даних. Також була створена ER-модель бази даних.

Проаналізовано основні вимоги для створення сайту та розроблено дизайн веб-сайту.

РОЗДІЛ 3. АНАЛІЗ СЕРЕДОВИЩ РОЗРОБКИ ТА РОЗРОБКА ВЕБ САЙТУ

3.1. Аналіз обраної середовища програмування

Створення сучасного веб-сайту потребує поєднання різних технологій, які відповідають за структуру, дизайн, функціональність та взаємодію з користувачем. Як правило, такі рішення базуються на серверних технологіях типу Python, Java, PHP, JavaScript або використовують готові потужні засоби для створення корпоративних сайтів, що орієнтовані на впровадження вказаних технологій. Розглянемо детальніше популярні інформаційні технології [3].

Мова програмування Python одна із найбільш популярніших з великою кількістю бібліотек, що робить її придатною для різноманітних завдань [10]. У веб-розробці використовується за допомогою двох основних фреймворків - Django та Flask.

Django - безкоштовний фреймворк високого рівня з відкритим кодом, який призначений для розробки веб-додатків на Python. Він дозволяє пришвидшити процес розробки та створити більш чистий дизайн. Його метою є позбавити програмістів від зайвого клопоту, пов'язаного з розробкою веб-додатків, а значить писати менше коду.

Flask - це невеликий легкий веб-фреймворк, написаний на мові Python, що пропонує корисні інструменти і функції для полегшення процесу створення веб-додатків з використанням Python. Він забезпечує гнучкість і є більш оптимальним фреймворком для новачків, так як дозволяє створити веб-додаток швидко, використовуючи тільки один Python файл [10]. Flask - це розширювана система, яка не зобов'язує використовувати конкретну структуру директорій і не вимагає складного шаблонного коду перед початком використання.

У нього в комплекті немає ні набору інструментів, ні бібліотек, якими славляться іншим популярні фреймворки Python: Django або Pyramid. Але він створений з потенціалом для розширення. У фреймворку є набір базових можливостей, а розширення відповідають за все інше. «Чистий» Flask не вміє підключатися до бази даних, перевіряти дані форми, завантажувати файли і так далі. Для додавання цих функцій потрібно використовувати розширення. Це допомагає використовувати тільки ті, які насправді потрібні.

Основними перевагами Python є простота розробки, характерна для шаблонів підходу, наявність великої кількості бібліотек, легкість їх використання, потужні і різноманітні середовища розробки, висока продуктивність, що робить його придатним для великих і складних проєктів. Простий та швидкий у вивченні, високопродуктивний, широко використовуваний - завдяки всім цим факторам Python є потужною і універсальною мовою програмування, яка може використовуватись для різних завдань.

Технологія .NET є новітньою розробкою компанії Microsoft і заявлена як новий етап у розвитку засобів взаємодії між прикладними програмами. На даний момент вона доступна в якості доповнення [14]. .NET Framework до сімейства операційних систем Microsoft Windows, а також у новому продукті Windows Server 2003. Також ведуться роботи зі створення .NET Framework на інших операційних системах. Платформа .NET спрощує розробку прикладних програм і підвищує надійність коду. Зокрема, вона забезпечує автоматичне керування часом життя об'єктів, нейтральні до мов бібліотеки класів та перетинають кордони мов спадкування, обробку виключень і налагодження.

У ASP.NET закладено все, для того, щоб зробити весь цикл розробки web-орієнтовані програми більш швидким, а підтримку простіше. Нижче наведені основні можливості та принципи роботи ASP.NET.

- компілювання коду при першому зверненні;
- широкий вибір бібліотек компонентів, що поставляються з .NET;
- підтримка потужного засобу розробки - Visual Studio .NET;

- мовна незалежність у межах платформ для яких реалізована загальна мовне середовище виконання CLR;
- можливості розширення за допомогою мультипроцесорних і кластерних рішень;
- нові можливості з обробки помилок;
- об'єктно-орієнтовані мови розробки (нову мову C #);
- розширені можливості повторного використання компонент.

Очевидно, що платформа. NET і ASP.NET надали нові можливості по розробці Web-систем. Вони відповідають всім сучасним вимогам і дозволяють значно прискорити і спростити розробку складних прикладних програм. Однак, на даний момент, NET в повному обсязі існує тільки для платформи Windows. Розробки по перенесенню на інші системи ведуться, але ще не завершені і їхні майбутні результати важко оцінити.

Що стосується розробки сайтів, то ASP.NET сильно прив'язана до сервера IIS, і, хоча архітектура. NET дозволяє перенести прикладну програму ASP.NET на іншу платформу, на даний момент реальна можливість відсутня. Таким чином найважливіше - багатоплатформенність поки що не може бути задоволено платформою. NET, а значить її використання для такої системи поки не виправдано. Однак необхідно відзначити, що така система повинна мати можливості інтеграції з платформою. NET (особливо Web-сервіси), оскільки її майбутнє широке використання не викликає сумнівів.

3.2. Обґрунтування вибору технологій для розробки

Vue.js — це прогресивний фреймворк для JavaScript, який використовується для побудови веб-інтерфейсів та односторінкових додатків. Не тільки для веб-інтерфейсів, Vue.js також використовується як для розробки настільних ПК, так і для мобільних додатків з фреймворком Electron [4].

Vue.js є одним із найлегших для вивчення через його інтенсивну простоту. Цей факт зробив великий внесок у його популярність. Vue.js

дозволяє розробникам створювати користувацькі інтерфейси [4]. Ця відкрита версія JavaScript має функцію адаптованої архітектури, яка дозволяє інтегрувати проект з різними бібліотеками JavaScript. Розробникам потрібно покласти мінімальні зусилля для оптимізації веб-розробок.

JavaScript та його фреймворки часто використовуються для створення додатків і різноманітних інтерфейсів. Вибір фреймворків для JS настільки широкий, що іноді розробники-початківці не знають, який обрати. Водночас є серед усіх фреймворків найпопулярніший інструмент — і це React JS [13].

React JS — використовується для розробки користувацького інтерфейсу веб-додатків. Основною метою React Frontend є створення інтерактивних, динамічних та відзивчивих інтерфейсів для користувачів. На думку багатьох розробників, одна з головних переваг цього фреймворку — дружність до SEO. Це робить його ідеальним рішенням для створення сайтів та інтернет-магазинів.

Також React JS пропонує чималу кількість можливостей для розробників, а ось чи зможуть вони ними скористатися — завжди залежить від компетенції.

Певною мірою причиною популярності React JS є простота вивчення. На освоєння React JS не потрібно витрачати багато часу: достатньо лише конструктивно підійти до вивчення матеріалу. Також він має велику та активну спільноту, що надає широкий спектр бібліотек, документації та підтримки. Водночас опанувати всі можливості React JS можна виключно завдяки досвіду, який ще потрібно отримати.

React JS має велику кількість переваг перед іншими фреймворками. До переліку причин, чому саме React JS обирають розробники, сміливо можна віднести наступні:

- швидкість та продуктивність – віртуальний DOM зменшує кількість оновлень UI, що збільшує швидкість та продуктивність додатків;
- компоненти можуть бути легко використані в різних частинах, що зменшує кількість коду і спрощує розробку;

-дозволяє створювати великі і складні додатки та сайти завдяки модульному підходу та легкій інтеграції з іншими технологіями.

Оцінюючи React JS та аналізуючи його недоліки, більшість спеціалістів говорять про некомплектність технології та її використання потребує додаткових інструментів та навичок

Підбиваючи підсумки аналізу React JS, можна оцінити його, як вельми гарний інструмент для розробки. Проте, React залишається ефективним інструментом для досвідчених розробників, які цінують його гнучкість, продуктивність та велике ком'юніті.

На відміну від React, Angular - це повний фреймворк від Google, побудований за принципом MVVM. Він надає готове рішення з CLI для створення структури проекту, різноманітними бібліотеками, засобами лінтування та тестування. Це спрощує процес розробки та робить Angular привабливим для новачків. Містить зокрема такі можливості, як:

- CLI для генерування структури файлів різних Angular-конструкцій, структури прм бібліотек, тестування, лінтування і т.д.
- Ряд бібліотек для самих різноманітних завдань — router, бібліотеки для роботи з формами, анімацією й інші.
- При ініціалізації проєкту можна доволі тонко налаштовувати те, що з доступних бібліотек буде додано до нього.
- Засоби лінтування.
- Karma + Jasmine для Unit і End to End тестування.

Angular є фактично оновленою версією AngularJS. Щоб уникнути плутанини, команда розробки вирішила назвати першу версію AngularJS, а другу – просто Angular. Як наслідок, команда розробки фреймворку почала використовувати назви Angular першої версії (для AngularJS) і Angular другої версії (для Angular), аби чіткіше їх розрізнати.

Якщо порівняти Angular з React, то його структура є значно складнішою. Окрім компонентів, тут також з'являються наступні елементи, кожен зі своїм призначенням:

- Пайпи — свого роду допоміжні функції для трансформації візуального вигляду даних у HTML темплейтах [11]. Наприклад: форматування чисел, дат, валют тощо.

- Директиви — класи, котрі використовуються для надання додаткового функціонала іншим елементам застосунку. І також поділяються на кілька видів, в залежності від форми взаємодії з елементами (цей момент я пропущу, аби не вдаватись в подробиці).

- В основному застосовуються у вигляді користувацьких атрибутів елементів розмітки.

- Компоненти — є елементом для візуального зображення даних. Поєднують в собі стилі, HTML-файл з розміткою, а також JS або TS з логікою компонента.

- Сервіси — для можливості централізованого зберігання даних і утилітарних методів. Можуть використовуватись Пайпами, Директивами, Компонентами й навіть іншими Сервісами, додаючи їх за допомогою Dependency Injection.

- Модулі — спосіб згрупувати всі вище перераховані елементи. В більшості випадків, по функціоналу або ж сторінках.

Angular – повноцінний фреймворк, що пропонує структурований підхід до розробки, забезпечує чітку структуру проєкту та стандартні рішення, надає широкі можливості для розробки. Тим не менш, його крива вивчення крута, тому для новачків цей фреймворк буде дуже складним і тривалим, а також може бути важким для невеликих проєктів.

React – досить «дорослий» продукт, який сьогодні користується найбільшою популярністю і вдячністю розробників. Гнучкість та можливість простої інтеграції з іншими бібліотеками – одна з ключових особливостей продукту. Має високу продуктивність, тобто завдяки VirtualDOM відбувається висока швидкість рендерингу. На даний момент найуніверсальніший варіант серед трьох порівнюваних.

Vue існує вже кілька років, але за фактом новачок на ринку. Тим не менш, на його використання перейшли більшість великих азіатських компаній, що говорить про його якість. Особливо добре підійде для тих, кому потрібно швидко опанувати його та потрібне швидке зростання. Наприклад, стартапу з молодого командою варто було звернути увагу саме на це середовище розробки [4].

У висновку можна зазначити, що Angular – потужний та структурований, кращий для великих проєктів; React – більш гнучкий та швидкий, ідеальний для складних проєктів; Vue – легкий в освоєнні, підходить для швидкої та легкої розробки.

3.3. Реалізація веб-сайту

Для кращого розуміння, що потрібно буде реалізувати, розглянемо існуючий сайт компанії [7] та внесемо зміни до нього. Також розробимо новий дизайн та функціонал. Зараз основна інформація на сайті є розробленою на лендінгу – односторінковому сайті. Тобто, інформація про послуги, кейси – саме та інформація, яка цікавить клієнта знаходиться на головній сторінці і в ній можна заплутатись, адже реалізації переходу на даний пункт немає (рис. 3.1).

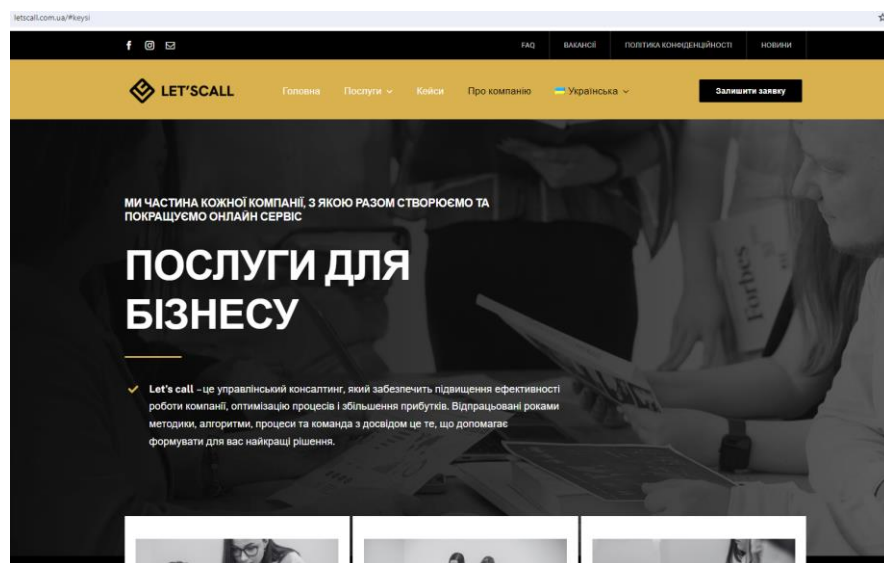


Рисунок 3.1. Приклад реалізації на існуючому сайті

На окрему сторінку винесена не основна інформація, а допоміжна – Питання/Відповідь, Вакансії, Політика конфіденційності (рис. 3.2). Дану інформацію було б краще не розміщати на різних сторінках, а перенести на одну сторінку, адже ця інформація – вона є додатковою для клієнта і на неї необхідно менше звертати увагу.

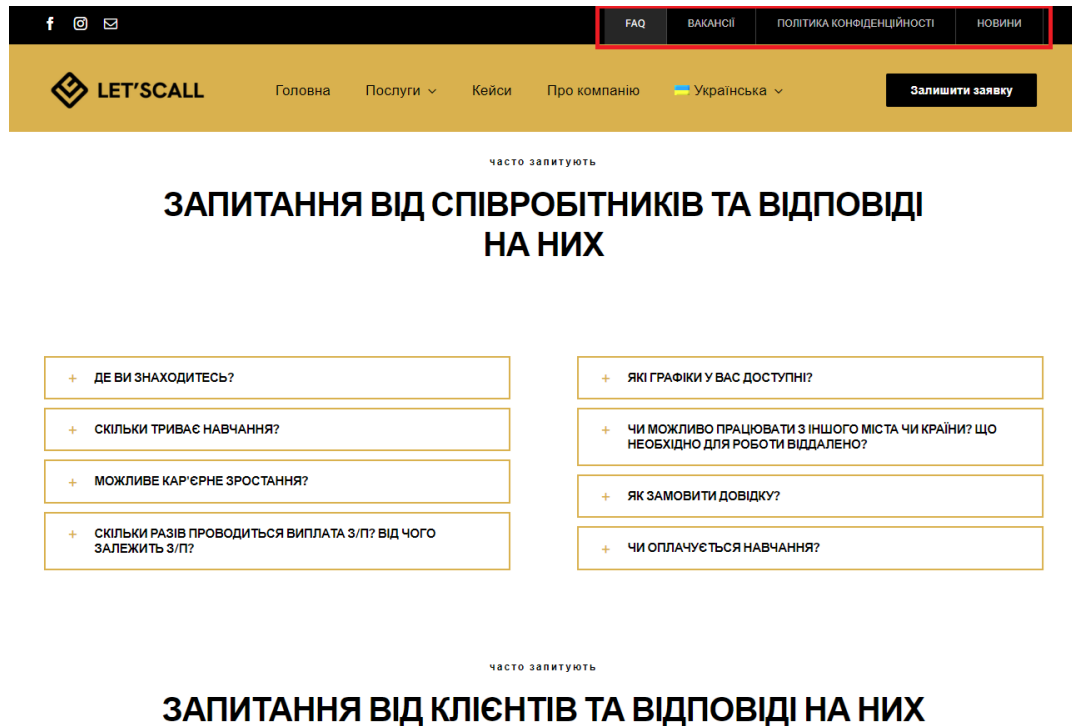


Рисунок 3.2. Приклад інформації, винесеної на різні сторінки

Для оптимізації роботи сайту було запропоновано та реалізовано наступну структуру:

1. Розділити та винести на різні блоки наступні пункти: Послуги, Кейси, Про компанію.
2. На сторінці Головна залишаємо інформацію про Фішки, Команду, Партнери.
3. На окрему сторінку виносимо інформацію Питання/Відповідь, Вакансії, Політика конфіденційності.

Сайт складається з чотирьох основних блоків. Вони призначені для клієнтів.

Структура головної сторінки:

- index.html головна сторінка, куди потрапляє користувач при першому зверненні на сайт. Тут він може ознайомитися з сервісами (послугами) які надає компанія, командою, партнерами, із якими співпрацювала та співпрацює колл-центр, і контактною інформацією;
- index.scss містить у собі загальні стилі для усього проєкту;
- App.jsx приймає та компілює основні блоки сайту.

Головна сторінка сайту представлена рисунку 3.3. Вона має простий зовнішній вигляд. Має чотири активні області: Послуги, Кейси, Про компанію, Блог. При відвідуванні клієнт одразу потрапить на головну сторінку та ознайомитись із вмістом сайту.

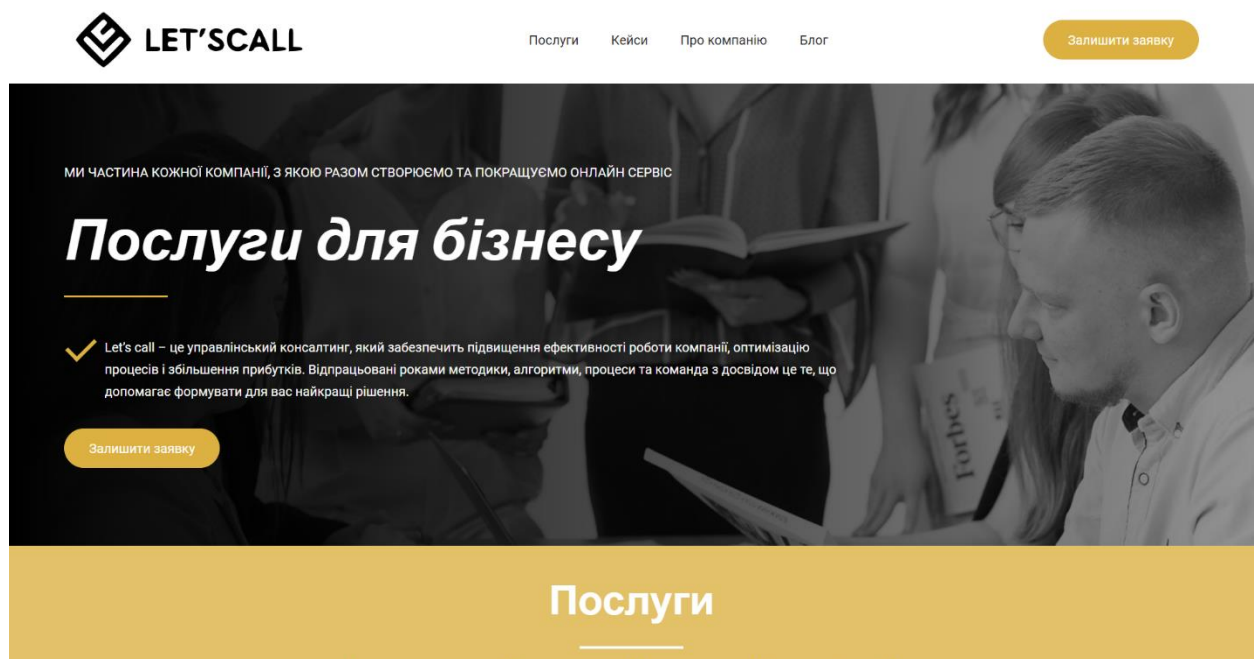


Рисунок 3.3. Частина головної сторінки

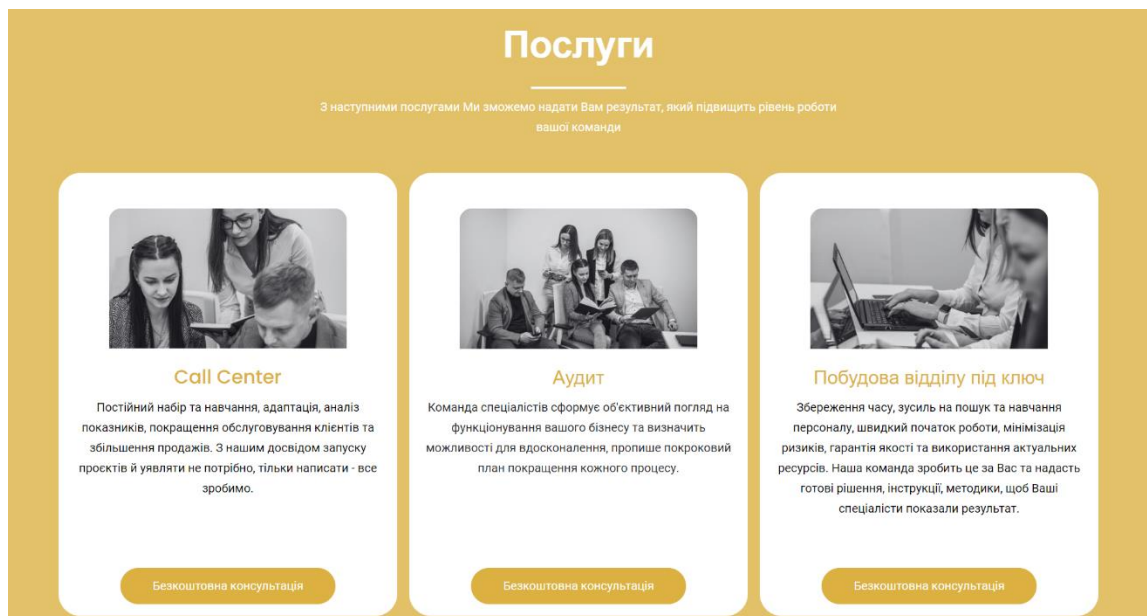


Рисунок 3.4. Частина головної сторінки

Сайт було реалізовано за допомогою React.js, що полегшило розробку кожного блоку та компілювання інформації у загальний вигляд, що зменшує навантаження для просування. Також, для подальшого покращення сайту або змінення інформації, не потрібно буде вишукувати необхідний блок на всій сторінці, достатньо подивитись на структуру проєкту та у відповідному компоненті знайти необхідний блок для редагування (рис. 3.5).

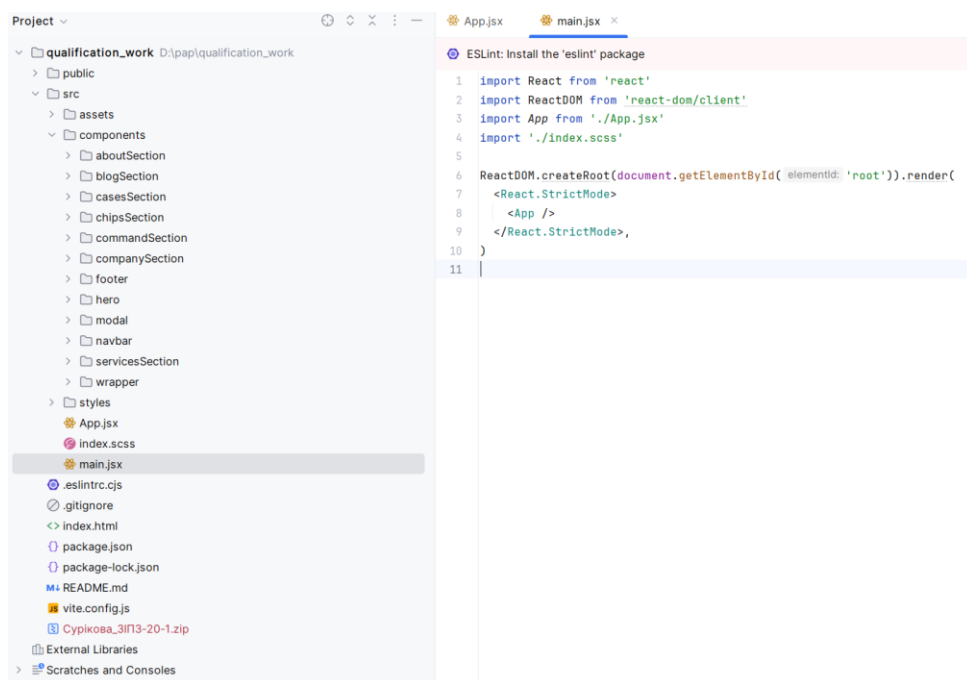
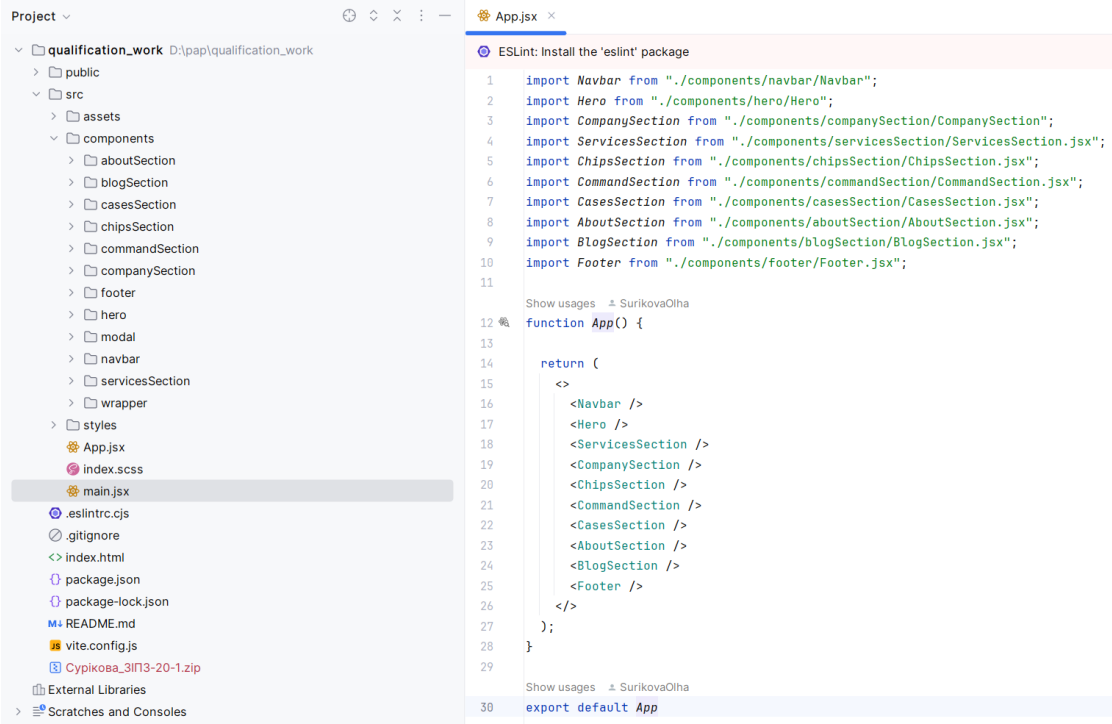


Рисунок 3.5. Структура розробленого проєкту

Всі блоки реалізовані в окремих компонентах та збираються в одному файлі – App.jsx (рис. 3.6).



```
1 import Navbar from "../components/navbar/Navbar";
2 import Hero from "../components/hero/Hero";
3 import CompanySection from "../components/companySection/CompanySection";
4 import ServicesSection from "../components/servicesSection/ServicesSection.jsx";
5 import ChipsSection from "../components/chipsSection/ChipsSection.jsx";
6 import CommandSection from "../components/commandSection/CommandSection.jsx";
7 import CasesSection from "../components/casesSection/CasesSection.jsx";
8 import AboutSection from "../components/aboutSection/AboutSection.jsx";
9 import BlogSection from "../components/blogSection/BlogSection.jsx";
10 import Footer from "../components/footer/Footer.jsx";
11
12 function App() {
13
14   return (
15     <>
16       <Navbar />
17       <Hero />
18       <ServicesSection />
19       <CompanySection />
20       <ChipsSection />
21       <CommandSection />
22       <CasesSection />
23       <AboutSection />
24       <BlogSection />
25       <Footer />
26     </>
27   );
28 }
29
30 export default App
```

Рисунок 3.6. Реалізація App.jsx

Кожен блок має схожу структуру. В ній імпортуються стилі, необхідні фото, що полегшує подальшу розробку проєкту. Для прикладу – переглянемо вигляд блоку Новини (рис. 3.7) та його реалізацію (рис. 3.8).

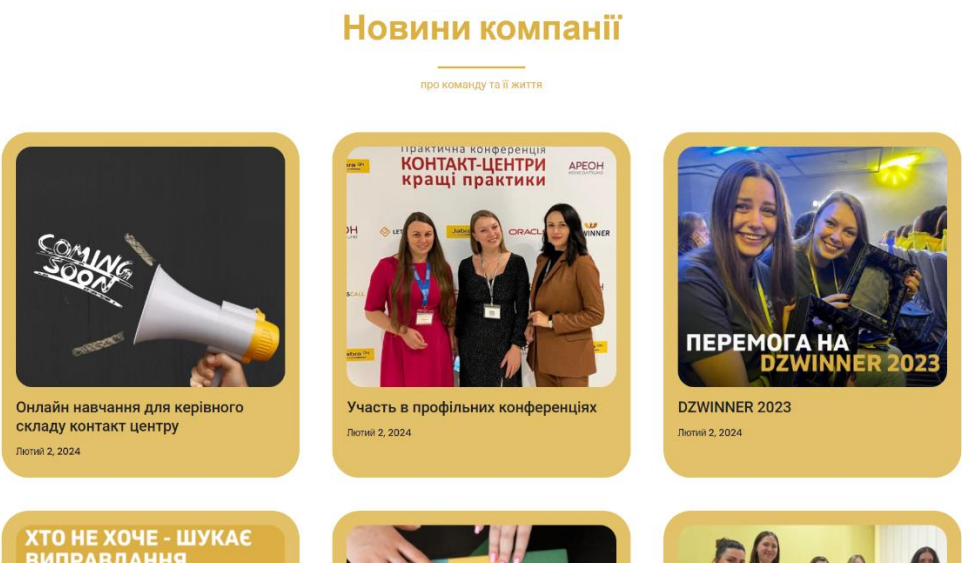


Рисунок 3.7. Реалізований блок Новини

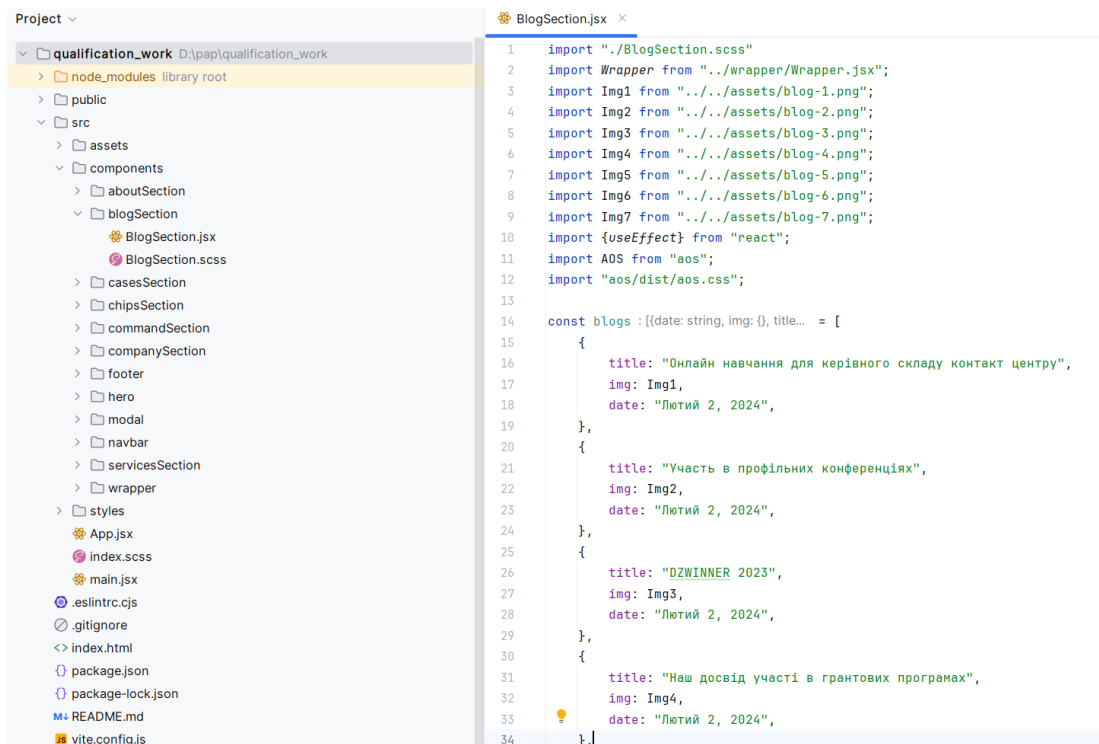


Рисунок 3.8. Реалізація блоку Новини за допомогою React.js

Також, сайт було адаптовано для менших розмірів – для планшета, міні-паду та телефону. На основному сайті компанії такої реалізації не було, що погіршувало показники його ефективності.

З кожним днем зростає кількість мобільних користувачів. Якщо сайт не адаптивний – він просто не зручний. Адаптивна верстка дозволяє не втрачати користувачів із мобільних пристроїв та отримувати більше заявок та замовлень.

Адаптивність сайту має свої переваги:

1. Зручний інтерфейс на будь-якому пристрої.
2. Лояльність з боку пошукових систем.
3. Переваги перед конкурентами.
4. Збільшення швидкості завантаження сайту.
5. Підвищення довіри користувачів.
6. Ріст конверсій з мобільного трафіку.

Приклад реалізації на основному сайті компанії (рис. 3.9):

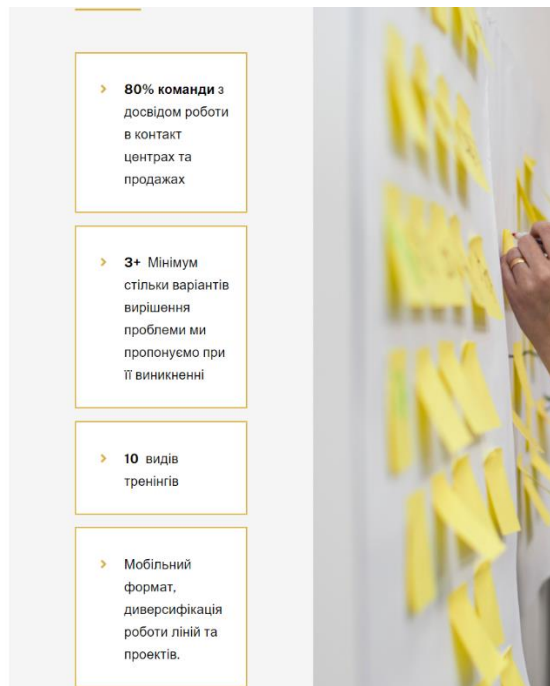


Рисунок 3.9. Реалізація адаптації для планшету да основному сайті компанії

Саме тому, було запропоновано та реалізовано адаптацію сайту для різних розширень. Для того, щоб не прописувати зайвий код для окремих розмірів, було зроблено максимально адаптивно нову версію сайту, який коректно на гарно відображається на будь-якому розмірі. було зроблено максимально адаптивно нову версію сайту, який коректно на гарно відображається на будь-якому розмірі (рис. 3.10).

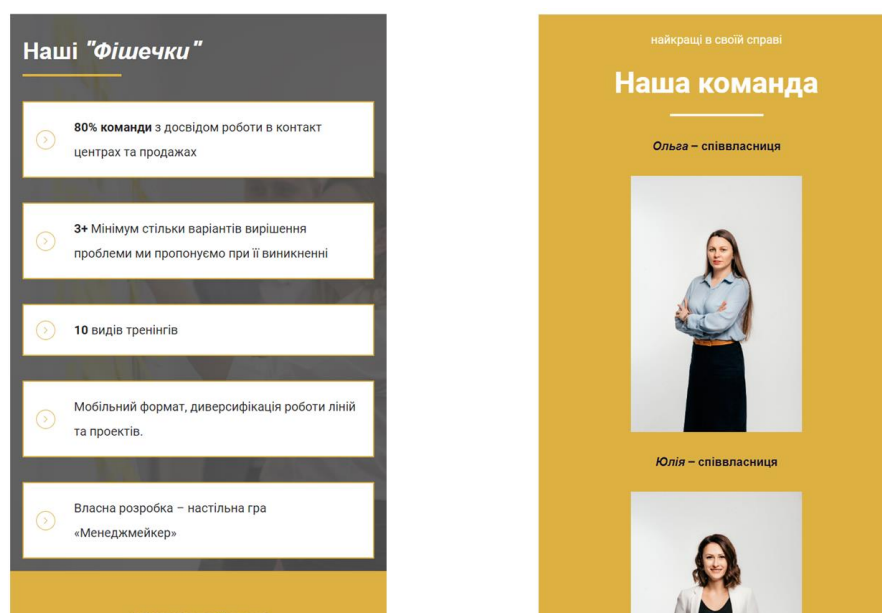


Рисунок 3.10. Реалізація для мобільного телефону

Також, для більш приємного сприйняття клієнтами, було реалізовано нескладну анімацію, яка не перенавантажує сайт, проте робить його візуально приємним.

Частина коду, для прикладу, буде розміщено у додатку Б.

Висновки до третього розділу

Під час виконання цього розділу було проаналізовано, які методи найчастіше використовують при розробці сайтів. Для розробки сайту було обрано React.js, JSON, HTML та SCSS

Також, було розроблено веб-сайт за попередньо розробленим дизайном, проаналізовано та протестовано розроблений сайт. Наведено приклади сторінок, які були розроблені, також було представлено часткові приклади реалізації коду.

ВИСНОВКИ

Випускна кваліфікаційна робота бакалавра являє собою опис, проєктування та розробку сайту для аутсорсингового колл-центру «ЛЕТСКОЛЛ», що допоможе клієнтам більш детально ознайомитись із послугами, які надаються. Також, упорядкувати та прискорити діяльність співробітників. На початку роботи було представлено основні задачі та цілі роботи. Був створений план вирішення поставленої задачі.

Для реалізації поставленої задачі було обрано наступні мови програмування та технології

- React.js - відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків;

- HTML – HyperText Markup Language, мова розмітки веб-сторінки, використовується для побудови структури інтерфейсу [1];

- SCSS – скриптова метамова, яка інтерпретується в каскадні таблиці стилів [12];

- JSON - текстовий формат обміну даними між комп'ютерами.

Презентація виконаної роботи підтвердила працездатність сайту. В цілому, мета, що була поставлена на початку роботи, була виконана.

Було розроблено працездатний сайт. На основі проведеної роботи, можна з успіхом розвивати та вдосконалювати розроблений сайт та надавати йому нової функціональності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дакетт Д. HTML та CSS. Розробка та дизайн веб-сайтів / Джон Дакетт., 2017. – 480 с. – (1).
2. Конноллі. Бази даних, проектування, реалізація та супровід / Конноллі, Бегг, Страчан. – 1112 с. – (1).
3. Дакетт Д. Javascript та jQuery. Інтерактивна веб-технологія. / Джон Дакетт., 2021. – 351 с. – (1).
4. vuejs.org. Vue.js. Tutorial [Електронний ресурс] / vuejs.org – Режим доступу до ресурсу: <https://vuejs.org/tutorial/#step-1>.
5. Дронов. HTML, JavaScript, PHP та MySQL. Джентльменський набір Web-майстра / Дронов, Прохоренок., 2019. – 463 с. – (5).
6. Янк К. PHP та MySQL. Від новачка до професіонала / Кевін Янк., 2013. – 256 с.
7. LET'SCALL - Послуги для бізнесу [Електронний ресурс] – Режим доступу до ресурсу: <https://www.letscall.com.ua/>.
8. Послуги аутсорс колл-центру [Електронний ресурс] – Режим доступу до ресурсу: <https://grace-call.com>.
9. ShiftON - автоматизований сервіс планування робочих графіків [Електронний ресурс] – Режим доступу до ресурсу: <https://shifton.com/uk>.
10. Python - що це та як використовувати [Електронний ресурс] – Режим доступу до ресурсу: <https://www.python.org/>.
11. HTML конструювання [Електронний ресурс] – Режим доступу до ресурсу: <http://htmlbook.in.ua/>.
12. Scss Вкладені правила та властивості [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/sass/sass_nesting.html#gsc.tab=0.

13. Фленаган Д. JavaScript повне керівництво; довідник з найпопулярнішої мови програмування / Девід Фленаган., 2022. – 722 с. – (7).

14. Технологія ASP.NET - основні характеристики, можливості використання [Електронний ресурс] – Режим доступу до ресурсу: <https://dotnet.microsoft.com/en-us/apps/aspnet>.

15. contactis - Аутсорсинговий Контакт-Центр [Електронний ресурс] – Режим доступу до ресурсу: <https://contactis.ua/>.

ДОДАТКИ

				ПЗ.КР.Б – 121 – 24 – ПЗ	

Код стилів SCSS

index.scss

```
@import "../styles/mixins.scss";

*,
*::after,
*::before {
  padding: 0;
  margin: 0;
  box-sizing: inherit;
  outline: 0;
  text-decoration: none;
  list-style: none;
}

body {
  color: #212123;
  background-color: #fff;
  font-family: "Roboto", sans-serif;
  font-size: 16px;
  font-weight: 400;
  line-height: 1.7em;
}

img {
  display: block;
}

button {
  display: inline-block;
  outline: none;
  border: none;
  background: none;
}

h1 {
  color: #fff;
  //letter-spacing: -1.7em;
  font-family: "Poppins", sans-serif;
```

```
font-size: 70px;  
font-weight: 700;  
line-height: 1.1em;  
margin-bottom: 10px;
```

```
@include respond(tab-port) {  
  font-size: 40px;  
}  
}
```

```
h4 {  
  color: #110d34;  
  font-family: "Poppins", sans-serif;  
  font-size: 36px;  
  font-weight: 700;  
  line-height: 1.1em;
```

```
@include respond(tab-port) {  
  font-size: 25px;  
}  
}
```

```
h3 {  
  color: #110d34;  
  margin-bottom: 10px;  
  font-family: "Poppins", sans-serif;  
  font-size: 50px;  
  font-weight: 700;  
  line-height: 1.1em;
```

```
@include respond(tab-land) {  
  font-size: 30px;  
}  
}
```

```
h6 {  
  color: #110d34;  
  margin-bottom: 10px;  
  font-family: "Poppins", sans-serif;  
  font-size: 20px;  
  font-weight: 700;
```

```

    line-height: 1.1em;
}

.button-primary {
    color: #fff;
    text-align: center;
    background-color: #dcb142;
    border: 1px solid #dcb142;
    border-radius: 30px;
    padding: 10px 30px;
    transition: background-color 0.3s ease-in-out;

    &:hover {
        background-color: #fff;
        color: #dcb142;
        border: 1px solid #dcb142;
    }
}

.button-outline {
    color: #110d34;
    text-align: center;
    background-color: rgba(0, 0, 0, 0);
    border: 1px solid #110d34;
    border-radius: 30px;
    padding: 10px 30px;
    transition: all 0.3s ease-in-out;

    &:hover {
        background-color: #110d34;
        color: #fff;
    }
}

.button-outline-2 {
    color: #fff;
    text-align: center;
    border: 1px solid #fff;
    border-radius: 30px;
    padding: 10px 30px;
    transition: all 0.3s ease-in-out;
}

```

```
&:hover {  
  background-color: #fff;  
  color: #dcb142;  
}  
}
```

Лістинг програми

```
App.jsx
import Navbar from "../components/navbar/Navbar";
import Hero from "../components/hero/Hero";
import CompanySection from "../components/companySection/CompanySection";
import ServicesSection from "../components/servicesSection/ServicesSection.jsx";
import ChipsSection from "../components/chipsSection/ChipsSection.jsx";
import CommandSection from
"/components/commandSection/CommandSection.jsx";
import CasesSection from "../components/casesSection/CasesSection.jsx";
import AboutSection from "../components/aboutSection/AboutSection.jsx";
import BlogSection from "../components/blogSection/BlogSection.jsx";
import Footer from "../components/footer/Footer.jsx";
function App() {
  return (
    <>
      <Navbar />
      <Hero />
      <ServicesSection />
      <CompanySection />
      <ChipsSection />
      <CommandSection />
      <CasesSection />
      <AboutSection />
      <BlogSection />
      <Footer />
    </>
  );
}
export default App
```

Navbar.jsx

```
import './Navbar.scss';
import Wrapper from '../wrapper/Wrapper';
import Logo from '../assets/logo.svg'
import { FaBars } from 'react-icons/fa';
import { useState } from 'react';
import Modal from '../modal/Modal.jsx';
const Navbar = () => {
  const [showNav, setShowNav] = useState(false);
  const [modalActive, setModalActive] = useState(false);
  return (
    <nav className="navbar">
      <Wrapper className="navbar__container">
        <a href="#" className="navbar__logo" onClick={() =>
setShowNav(false)}>
          <img src={Logo} alt=""/>
        </a>
        <ul className={`navbar__links ${showNav ? "show-nav" : ""}`>
          <li onClick={() => setShowNav(false)}><a href="#">Послуги</a></li>
          <li onClick={() => setShowNav(false)}><a href="#">Кейси</a></li>
          <li onClick={() => setShowNav(false)}><a href="#">Про
компанію</a></li>
          <li onClick={() => setShowNav(false)}><a href="#">Блог</a></li>
        </ul>
        <a href="#" className="button-primary navbar__btn" onClick={() =>
setModalActive(true)}>Залишити заявку</a>
        <Modal active={modalActive} setActive={setModalActive}/>
        <div className={`navbar__menubar ${showNav ? "bg-color" : ""}`>
          <div className="navbar__menubar__icon" onClick={() => setShowNav(!showNav)}>
            <img alt="hamburger menu icon" />
          </div>
        </div>
      </Wrapper>
    </nav>
  )
}
```

```

        <FaBars />
      </div>
    </Wrapper>
  </nav>)
};
export default Navbar

```

Hero.jsx

```

import './Hero.scss';
import Wrapper from "../wrapper/Wrapper";
import {useState} from "react";
import Modal from "../modal/Modal.jsx";
const Hero = () => {
  const [modalActive, setModalActive] = useState(false);
  return (
    <section className="hero">
      <Wrapper className='hero__container'>
        <div className="hero__left">
          <p>МИ ЧАСТИНА КОЖНОЇ КОМПАНІЇ, З ЯКОЮ РАЗОМ
СТВОРЮЄМО ТА ПОКРАЩУЄМО ОНЛАЙН СЕРВІС
          </p>
          <h1>Послуги для бізнесу</h1>
          <div className="description">
            <div className="wrap">
              <div id="check-part-1" className="check-sign"></div>
              <div id="check-part-2" className="check-sign"></div>
            </div>
            <p>Let's call – це управлінський консалтинг, який забезпечить
підвищення ефективності роботи

```

компанії, оптимізацію процесів і збільшення прибутків.

Відпрацьовані роками методики, алгоритми,

процеси та команда з досвідом це те, що допомагає формувати для вас найкращі рішення.</p>

</div>

<div className="hero__btn-container">

 setModalActive(true)}>Залишити
заявку

<Modal active={modalActive} setActive={setModalActive}/>

</div>

</div>

</Wrapper>

</section>

)

};

export default Hero

BlogSection.jsx

import './BlogSection.scss'

import Wrapper from '../wrapper/Wrapper.jsx';

import Img1 from '../assets/blog-1.png';

import Img2 from '../assets/blog-2.png';

import Img3 from '../assets/blog-3.png';

import Img4 from '../assets/blog-4.png';

import Img5 from '../assets/blog-5.png';

import Img6 from '../assets/blog-6.png';

import Img7 from '../assets/blog-7.png';

```
const blogs = [
  {
    title: "Онлайн навчання для керівного складу контакт центру",
    img: Img1,
    date: "Лютий 2, 2024",
  },
  {
    title: "Участь в профільних конференціях",
    img: Img2,
    date: "Лютий 2, 2024",
  },
  {
    title: "DZWINNER 2023",
    img: Img3,
    date: "Лютий 2, 2024",
  },
  {
    title: "Наш досвід участі в грантових програмах",
    img: Img4,
    date: "Лютий 2, 2024",
  },
  {
    title: "Бізнес Гра МЕНЕДЖмейкер",
    img: Img5,
    date: "Січень 19, 2024",
  },
  {
    title: "Кращий продавець 2023",
    img: Img6,
    date: "Січень 19, 2024",
  },

```

```

    },
    {
      title: "DZWinner 2022",
      img: Img7,
      date: "Січень 19, 2023",
    },
  ];
const BlogSection = () => {
  return (
    <section className="blog">
      <Wrapper>
        <div className="blog__text">
          <h3>Новини компанії</h3>
          <p>про команду та її життя</p>
        </div>
        <div className="blog__container">
          {blogs.map((blog) => (
            <div className="blog__card">
              <div className="blog__card-img">
                <img src={blog.img} alt=""/>
              </div>
              <div className="blog__title">
                {blog.title}
              </div>
              <h6>{blog.date}</h6>
            </div>
          ))}
        </div>
        <div className="blog__btn">
          <a href="#" className="button-primary">Дивитись все</a>
        </div>
      </Wrapper>
    </section>
  );
};

```

```
        </div>
    </Wrapper>
</section>
)
}
export default BlogSection
```