Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Створення CRM-системи для інстаграм-магазину "jolli.store.ua" засобами HTML5, CSS, JS та MongoDB

Виконав: студент VI курсу, групи САм-61
спеціальності 124 Системний аналіз
(шифр і назва спеціальності)

(підпис) Серкіз С.С. (прізвище та ініціали)

Керівник (підпис) Гром'як Р.С. (прізвище та ініціали)

Нормоконтроль (підпис) Мацюк О.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Луцик Н.С. (прізвище та ініціали)

Тернопіль
2022

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«____» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 124 Системний аналіз
(шифр і назва спеціальності)

Студенту Серкізу Станіслав Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Створення CRM-системи для інстаграм-магазину "jolli.store.ua" засобами HTML5, CSS, JS та MongoDB

Керівник роботи Гром'як Роман Сельвесторович, к.ф.-м.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 22 » листопада 2022 року № 4/7-947

2. Термін подання студентом завершеної роботи 21 грудня 2022р.

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ.1 Аналіз предметної області. 2 Аналіз інструментів розроблення CRM-систем.
3. Створення CRM-системи
4 Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Мацюк О.В., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання 14 листопада 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	14.11.2022-15.11.2022	*Виконано*
2.	Підбір наукових джерел про CRM-систему	16.11.2022-20.11.2022	*Виконано*
3.	Переклад та опрацювання наукових джерел про	21.11.2022-23.11.2022	*Виконано*
4.	Виконання дослідження щодо методів розробки	24.11.2022-27.11.2022	*Виконано*
	системи		
5.	Оформлення розділу «АНАЛІЗ ПРЕДМЕТНОЇ	28.11.2022-30.11.2022	*Виконано*
	ОБЛАСТІ»		
6.	Оформлення розділу «АНАЛІЗ ІНСТРУМЕНТІВ	01.12.2022-04.12.2022	*Виконано*
	РОЗРОБЛЕННЯ CRM-СИСТЕМИ»		
7.	Оформлення розділу «СТВОРЕННЯ	05.12.2022-07.12.2022	*Виконано*
	CRM-СИСТЕМИ»		
8.	Виконання завдання до підрозділу «Охорона праці»	08.12.2022-09.12.2022	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	10.12.2022-11.12.2022	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	12.12.2022-13.12.2022	*Виконано*
11.	Нормоконтроль	14.12.2022-15.12.2022	*Виконано*
12.	Перевірка на плагіат	15.12.2022	*Виконано*
13.	Попередній захист кваліфікаційної роботи	16.12.2022	*Виконано*
14.	Захист кваліфікаційної роботи	21.12.2022	

Студент _____ Серкіз С.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Гром'як Р.С.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення CRM-системи для інстаграм-магазину “jolli.store.ua” засобами HTML5, CSS, JS та MongoDB // Кваліфікаційна робота освітнього рівня «Магістр» // Серкіз Станіслав Сергійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група САм-61 // Тернопіль, 2022 // С.66, рис. – 59, табл. – 0, кресл. – 0, додат. – 2, бібліогр. – 50.

Ключові слова: CRM-система, база даних , проект

Кваліфікаційна робота присв’ячена розробці CRM-системі для магазину одягу. В першому розділі кваліфікаційної роботи описані поняття про CRM-систему та про бази дани . Розглянуто інсуючі системи та проаналізований їх функціонал . В другому розділі кваліфікаційної роботи було проаналізовано та вибрано мову програмування та базу даних для розробки системи.

В третьому розділі кваліфікаційної роботи описано розроблення CRM-системи для магазину одягу.

ANNOTATION

CRM System Development for Instagram-based Store "jolli.store.ua" with HTML5, CSS, JS and MongoDB// Qualification work of the educational level "Master" // Serkiz Stanislav// Ternopil National Technical University named after Ivan Pulyuy, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, Sam-61 group // Ternopil, 2022 // P. 66, fig. – 59 , tables – 0 , chair. - 0, annexes – 2 , references. - 50.

Key words: CRM system, database, project

The qualification work is devoted to the development of a CRM system for a clothing store. The first section of the qualification work describes the concepts of the CRM system and databases. The existing systems were considered and their functionality was analyzed. In the second section of the qualification work, the programming language and database for system development were analyzed and selected.

The third section of the qualification work describes the development of a CRM system for a clothing store.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних.

СУБД – система управління базами даних

НС – надзвичайні ситуації

CRM (англ. Customer Relationship Management) – управління відносинами з клієнтами.

ЗМІСТ

ВСТУП 6
1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 7
1.1 Поняття CRM-системи 7
1.2 Розбір існуючих CRM-систем 10
1.3 Поняття та аналіз систем керування базами даних 14
1.4 Висновок до першого розділу 17
2 АНАЛІЗ ІНСТРУМЕНТІВ РОЗРОБЛЕННЯ CRM-СИСТЕМИ 18
2.1 Постановка завдання 18
2.2 Вибір інструментів для розроблення CRM-систем 19
2.3 Додаткові сервіси для розроблення CRM-систем 23
2.4 Вибір системи управління базами даних 24
3 СТВОРЕННЯ CRM-СИСТЕМИ 27
3.1 Архітектура програми 27
3.2 Розроблення серверної сторони проекта 28
3.3 Розроблення клієнтської сторони проекту сторони проекта 35
3.4 Огляд створеної системи 46
4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ 51
4.1 Організація безпечних умов праці при монтажі комп'ютерної мережі 51
4.2 Підвищення стійкості роботи об'єктів торгівлі в воєнний час 54
ВИСНОВКИ 58
ПЕРЕЛІК ДЖЕРЕЛ 59
ДОДАТКИ

ВСТУП

Актуальність теми. Кожна компанія, яка надає послуги для бізнесу чи окремих осіб, намагається зробити свою діяльність максимально результативною та знайти якомога більше точок дотику з клієнтами. За допомогою CRM-систем допоможе відмовитися від рутинних робіт та допоможе оптимізувати роботу бізнес та знизить його витрати.

Автоматизація бізнес-процесів – украй важливе завдання для будь-якої компанії, орієнтованої на розвиток. Критерії, що визначають потребу в її запровадженні, можуть бути різноманітними і залежати від масштабів діяльності організації та стадії її розвитку. Цими ж критеріями слід керуватися й у виборі інструментів для автоматизації. Водночас важливо обрати правильну систему, яка дасть змогу організувати злагоджену роботу всіх підрозділів та рівнів комунікації і визначить успіх бізнесу.

Сьогодні найефективнішими інструментами стали CRM-системи для оптимізації бізнес-процесів

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття CRM-системи

Система управління відносинами з клієнтами (CRM) – це інструмент або система інструментів, які допомагають підприємствам збирати, організовувати та аналізувати дані про клієнтів для кращого управління відносинами з клієнтами.

Коли дані про потенційних і існуючих клієнтів збираються та зберігаються в системі CRM, відстеження інформації про кожну точку контакту на шляху клієнта (подробиці, надані через форми, взаємодія з маркетинговими кампаніями, взаємодія зі службою підтримки клієнтів, моделі купівлі) може допомогти вам задовольнити їх потреб, будувати кращі відносини та розумніший ринок.

Програмне забезпечення CRM може допомогти вашому бізнесу розвиватися, і організації в усіх галузях застосовують технології CRM і пожинають плоди; більше 64% компаній вважають, що інструменти CRM впливають або дуже впливають.

CRM-системи використовують за різним призначенням , їх можна поділити на такі види:

- системи для управлінням продажами. В них вбудовану функцію аналізу витрат та прибутку, можна проаналізувати прогноз продаж за період часу;
- системи управління маркетингом. В такій системі відображаються результати просування продукту , за допомогою цих даних аналізуються стратегії та поточні, та майбутні рекламні кампанії;
- системи для клієнтського обслуговування.

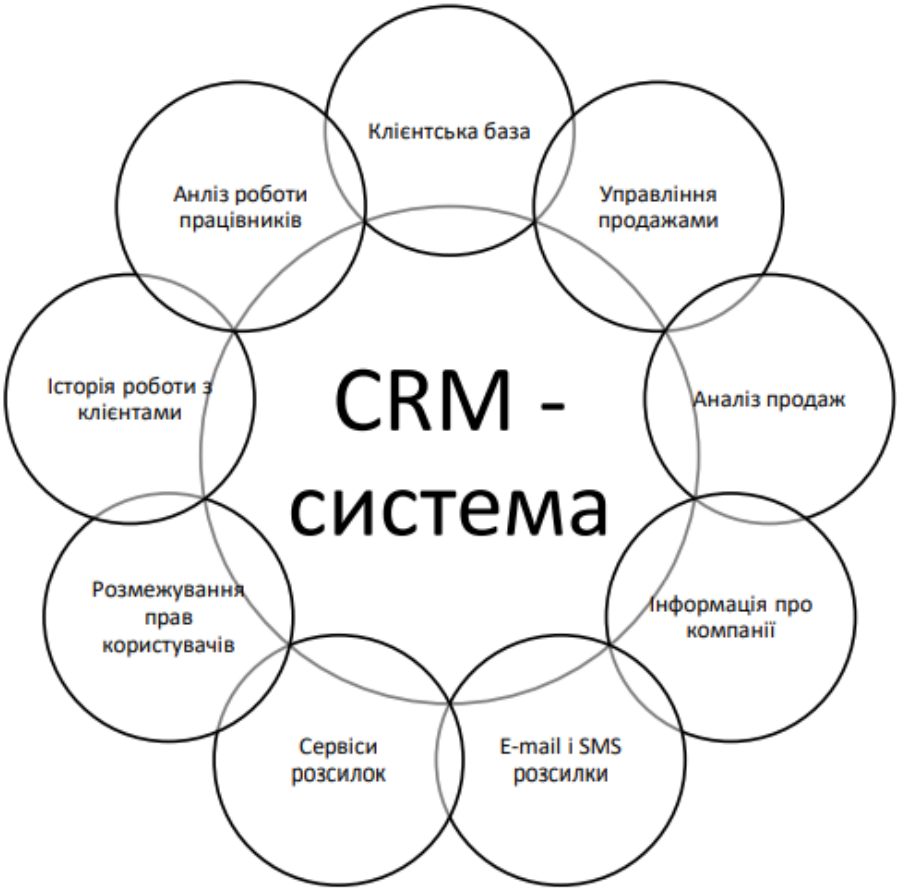

Рисунок 1.1 – Огляд CRM-системи

Переваги встановлення CRM-системи :

- значне пришвидшення та удосконалення міркування
- поділ клієнтів за різними даними ;
- визначення лідеру товару на ринку
- підвищення кількості нових лідів;
- зменшення збитків за допомогою оптимізації
- зменшення витрат часу на опрацювання клієнта
- повнота інформації клієнта, пошук підходу до нього
- навчання та вникнення в нюанси роботи нових працівників за допомогою вже опрацьованих звітах по клієнтах;
- контроль ефективності праці робітників;
- оцінка виконаної роботи за проміжок часу.

Слід ретельно підбирати CRM-систему , якщо не вірно буде вибрана система , вона не дасть максимального результату. Спочатку потрібно розібратися , які функції повинна виконувати система. Функції по яким можна обрати систему:

- збереження товару та збереження інформації про клієнтів;
- автоматична зміна статусу посилки;
- встановлення системи для аналізу статистики;
- налаштування розсилок на пошту та телефон;
- історія перетинання з кожним клієнтом

Підключення CRM-системи точно слід почати роботи , коли всі помилки виникають з клієнтами в області бізнес-процесів. Такі проблеми можуть бути:

- вказано не вірно дату доставки товару;
- неправильний облік товару;
- відсутність точних фото товару;
- відсутня комунікація з клієнтами та потенційними клієнтами;
- відсутня лінія зв’язку та комункації з менеджером.

CRM-система автоматизує рутинні процеси та зводить до мінімуму можливий негативний вплив людського фактора (не додивилися, не встигли, забули, помилилися).

За допомогою CRM-системи можна оптимізувати буденні процеси , такі як розсилка накладних клієнтам, зміна статусу накладних , нагадування клієнтам про прибуття посилки у відділення пошти.

На початковому етапі інтернет-магазину, можливе введення обліку товарів , обліку замовлень самостійно або за допомогою Excel таблиць, але для точного аналізу і кращої ефективності роботи інтернет-магазину та ефективності робітників , потрібні додаткові сервіси , які будуть удосконалювати та спрощувати комунікацію з людьми , облік замовлень та товарів та аналіз ефективності бізнес процесів.

1.2 Розбір існуючих CRM-систем

Проаналізуємо такі CPM системи: Sales Drive , MyDrop, Voiptime CRM.

Sales Drive – CRM-система розроблена для полегшення роботи з онлайн продажу товарів. Систему можна інтегрувати з такими сервісами як: “Нова пошта”, “Укрпошта”, “Приват Банк ”, “Монобанк”, “Prom”, “Rozetka”, “OpenCart”, “WordPress”.

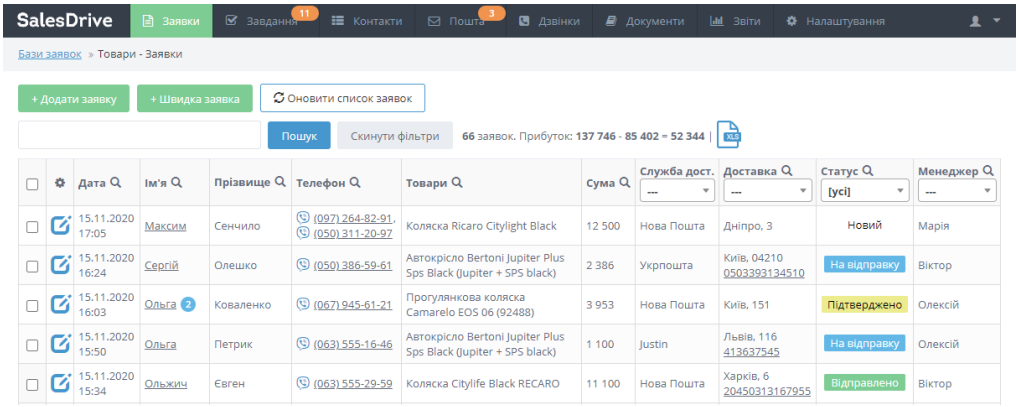

Рисунок 1.2 – Огляд системи SalesDrive

Інтеграція системи з сервісами полегшує та пришвидшує роботу, так як заявки клієнтів автоматично перенаправляються з сайту до системи, автоматично розподілюються між менеджерами, автоматично формується експрес-накладна, миттєвий друк накладної , система самостійно заповнює розмір , вагу та опис відправлення з каталогу , зручне відслідкування статуса посилки, фільтрація посилок по статусу. Також в систему можна інтегувати автоматичне відправлення повідомлень для таких дій як: підтвердження замовлень, розсилка клієнтам накладних, про зміну статусу посилки на поштових відділеннях.

MyDrop – система для роботи в CRM по системі дропшипінг.

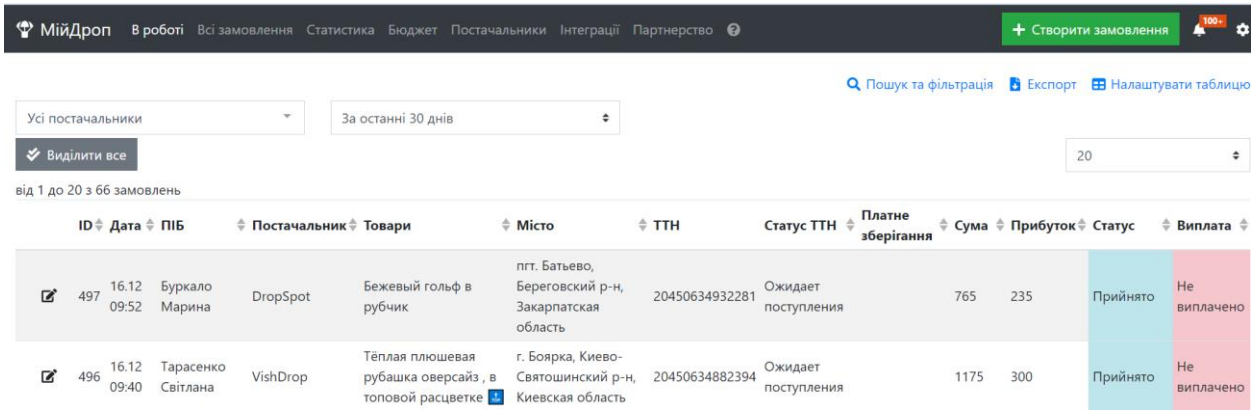

Рисунок 1.2 - Огляд системи MyDrop

В системі вбудоване зручне вікно для перегляду статистика замовлень , де можна переглянути кількість замовлень, прибуток , збитки , кількість успішних замовлення, кількість невдалих замовлень , середня ціна прибутку.

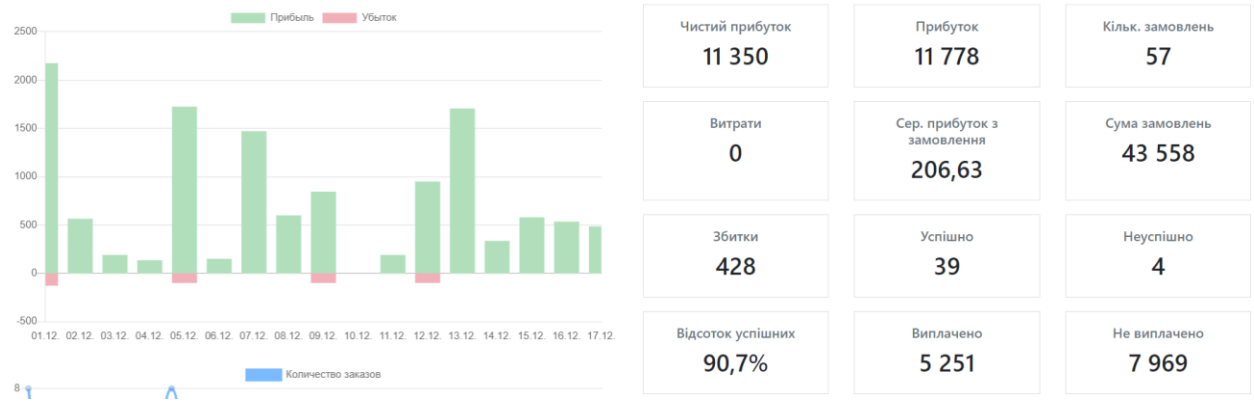

Рисунок 1.3 - Огляд розділу «Статистика»

Також вбудована розділ «Бюджет», де вказуються додаткові витрати на рекламу на різних джерелах трафіку та можна обирати товар на який, було витрачено кошти на рекламу.

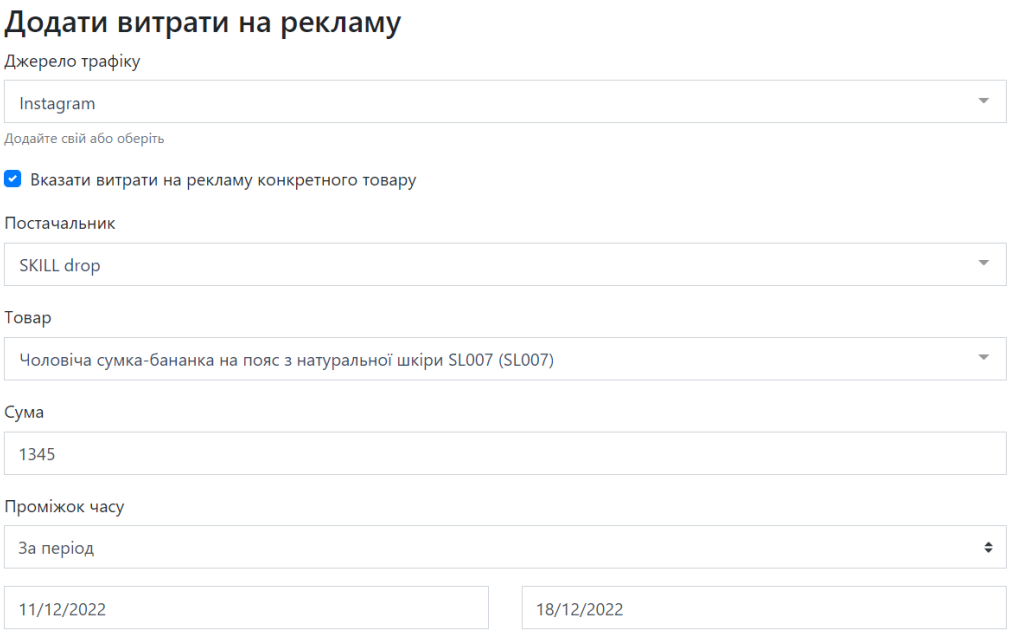

Рисунок 1.3 - Огляд розділу «Бюджет»

В системі можна виділити такі переваги:

- Скорість роботи
- Прости та зрозумілий інтерфейс
- Підключення сповіщень про зміну статусу посилки в соціальних мережах
- Облік фінансів, за допомогою системи можна контролювати прибутки та збитки за допомогою функції аналітика
- Інтеграція з програмами: “Нова пошта”, “Prom”, “Checkbox”, “Woo Commerce ” ,“Sms club ”,“УкрПошта”
- Контроль актуальної наявності товару
- Самостійне налаштування таблиць для контролю замовленнями
- Швидка техпідтримка
- Оновлення з оптимізацією системи
- Добавлення нових функцій

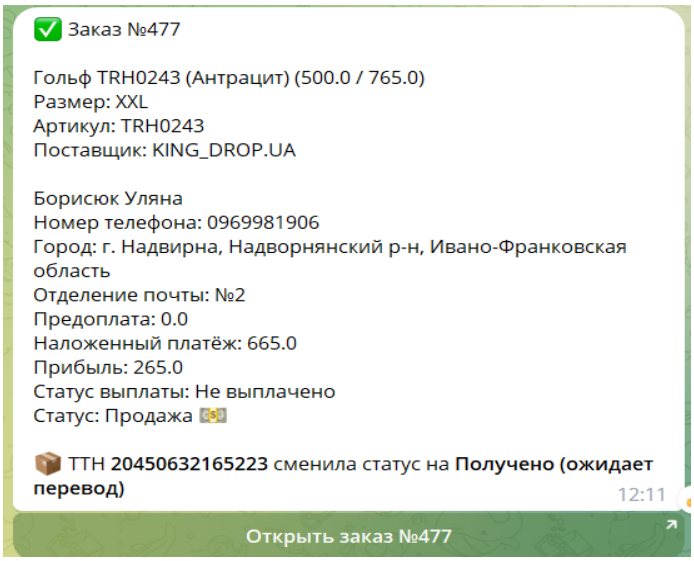

Рисунок 1.3 – Сповіщення про стан посилки

Voiptime CRM - зручна CRM-система для товарного бізнесу, яка дозволяє оптимізувати продаж в інтернет-магазині або маркетплейсі. Інтеграція з сайтом, автообзвін, тригерні SMS-розсилки, чати, обробка звернень з Facebook та Instagram, складський облік та аналітика.

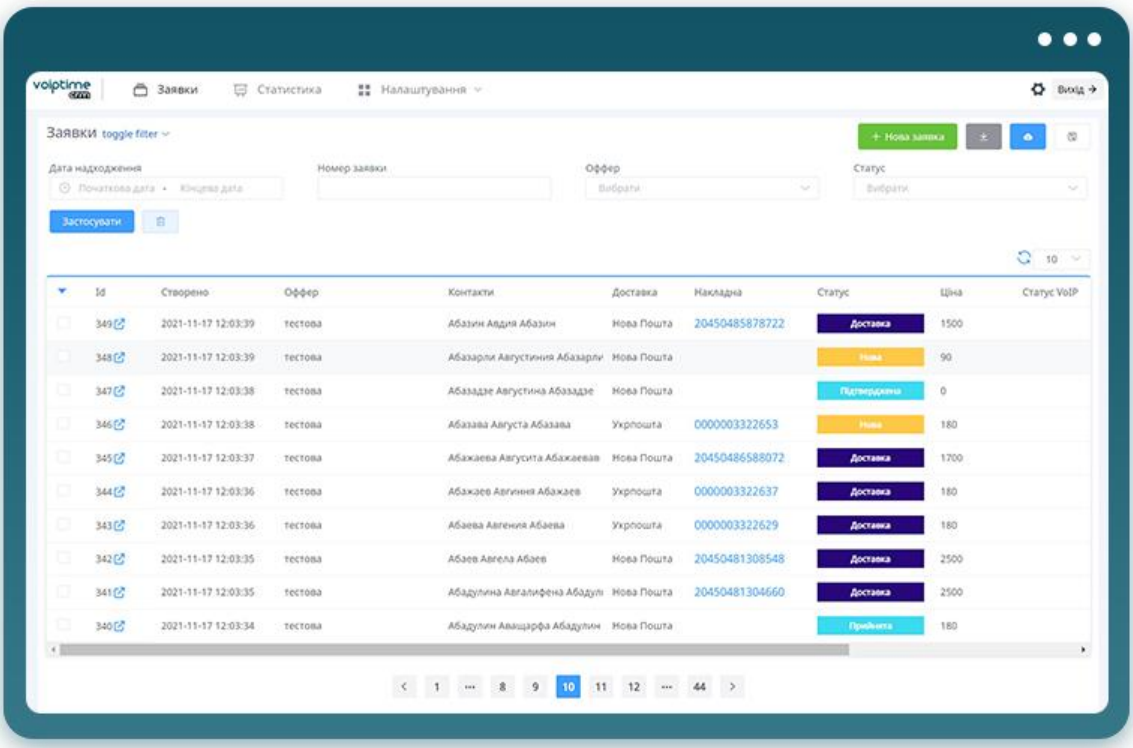

Рисунок 1.4 – Сповіщення про стан посилки

Можливості системи:

- Гнучке налаштування полів замовлення
- Історія замовлень клієнта
- Інтеграція з поштовими та кур'єрськими службами:
- Нова пошта
- Укрпошта

Детальна аналітика:

- Панель приладів менеджера та оператора
- Аналіз роботи кожного співробітника
- Контроль ефективності вебмайстрів
- Фінансова аналітика та контроль ефективності продажів
- Автоматизація та комунакація:
- Вхідні та вихідні дзвінки з VoIPTime CCSMS повідомлення
- Налаштування тригерів
- Інтеграція з Instagram, Facebook

1.3 Поняття та аналіз систем керування базами даних

База даних — це інформація, налаштована для легкого доступу, керування та оновлення. Комп’ютерні бази даних зазвичай зберігають сукупність записів даних або файлів, які містять інформацію, таку як операції з продажу, дані про клієнтів, фінанси та інформацію про продукт.

Бази даних використовуються для зберігання, підтримки та доступу до будь-яких даних. Вони збирають інформацію про людей, місця або речі. Ця інформація збирається в одному місці, щоб її можна було спостерігати та аналізувати. Бази даних можна розглядати як організований збір інформації.

Компанії використовують дані, що зберігаються в базах даних, для прийняття обґрунтованих бізнес-рішень. Деякі зі способів використання баз даних організаціями включають наступне:

- удосконалити бізнес-процеси. Компанії збирають дані про бізнес-процеси, такі як продажі, обробка замовлень і обслуговування клієнтів. Вони аналізують ці дані, щоб покращити ці процеси, розширити свій бізнес і збільшити дохід.
- слідкуйте за клієнтами. Бази даних часто зберігають інформацію про людей, наприклад клієнтів або користувачів. Наприклад, платформи соціальних мереж використовують бази даних для зберігання інформації про користувачів, такої як імена, адреси електронної пошти та поведінка користувачів. Дані використовуються, щоб рекомендувати вміст користувачам і покращувати взаємодію з ними.
- захистіть особисту інформацію про здоров'я. Постачальники медичних послуг використовують бази даних для безпечного зберігання особистих даних про здоров'я для інформування та покращення догляду за пацієнтами.
- зберігати особисті дані. Бази даних також можна використовувати для зберігання особистої інформації. Наприклад, особисте хмарне сховище доступне для окремих користувачів для зберігання мультимедійних даних, наприклад фотографій, у керованій хмарі.

СУБД (Система управління базами даних) - це система, яка дозволяє користувачеві робити операції з даними.

СУБД позволяє користувачам:

- Розробляти структуру та вносити інформацію та зберігати її на хмарі
- Отримувати результати
- Виводити інформацію для користувача у термінал
- Надання доступу до даних іншим користувачам

СУБД дозволяє усунути недоліки, властиві раніше базам даних:

- зменшення використання полів, за рахунок нормалізації таблиць, у яких зберігаються дані

- можна підтримувати цілісність і надійність даних;
- більш надійний захист
- незалежність даних

СУБД можна розділити на такі групи: за моделями даних , за розміщенням , за способ доступу до БД.

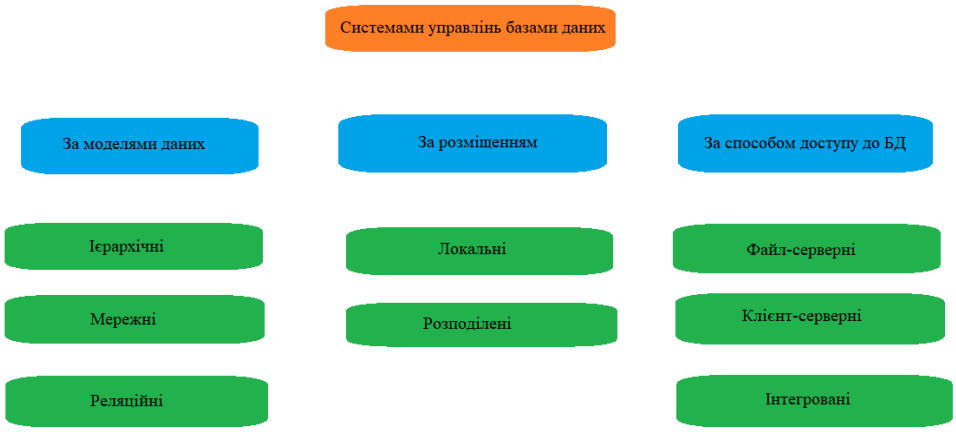

Рисунок 1.5 – Система управління базами даних

За моделями даних СУБД поділяється на:

- ієрархічні - модель відображенням бази даних у вигляді дерева. Структура ієрархічної моделі складається так, що кожен об’єкт має лише одного батька, для якого він є дочірнім (батьком), і може мати кілька дочірніх (дітей) об’єктів. Винятком може бути елемент найвищої ієрархії – у нього немає батьківського елемента. Прикладом такої моделі може бути файлова структура FAT32, в цій структурі існує основний батьківський елемент, в якій зберігаються йому підпорядковані папки та файли;
- мережні – у кожного об’єкта можливе існування декількох батьківських об’єктів , а також декількох об’єктів нащадків.
- реляційні – фундамент цеї моделі є таблиці. В цій моделі потрібно дотримуватися таких правил: обов'язкова індефікація об’єкта за допомогою властивостей набору значень заданих користувачем і також мають бути правильно задані зв’язки між таблицями

- об’єктно-реляційні – це модель в якій об’єкти мають певні властивості та можуть взаємодіяти з певними об’єктами та також дані зберігаються як абстрактні

Серед популярних систем управління базами даних можна виокремити такі системи: MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase, Interbase, Firebird і IBM Db2, MongoDb.

Перед створення БД , потрібно спроектувати її моделі. Для цього знадобиться зрозуміти:

- кількість даних, які будуть зберігатися
- число таблиць в якій будуть зберіатися дані
- структуру таблиць
- визначити імена та типи полів
- основні поля для таблиць

1.4 Висновок до першого розділу

В першому розділі розглянули кваліфікаційної робити було розглянуто поняття CRM-системи , їх використання та їх призначено.

Проаналізувавши ринок систем , ми обрати такі системи для аналізу: Sales Drive , MyDrop, Voiptime CRM та визначали їх сильна та слабкі сторони та проаналізували їх роботу .

Також було опрацьоване поняття про системи керування бази даних та було визначено поділ СУБД на моделі та чим вони відрізняються між собою.

2 АНАЛІЗ ІНСТРУМЕНТІВ РОЗРОБЛЕННЯ CRM-СИСТЕМИ

2.1 ПОСТАНОВКА ЗАВДАННЯ

Проаналізовавши доступні CRM-системи на ринку було прийнято розробити власну CRM-систему для магазину одягу.

Для виконання завдання потрібно:

- проаналізувати потрібний функціонал системи
- обрати СУБД для зберігання та керуванням даних
- розробити функціонал сайту
- зробити простий та зручний інтерфейс

В CRM-систему повинні бути такі функції:

- реєстрація та авторизація акаунта для подальшого розвитку філіалів магазину;
- добавлення сторінки «Асортимент» , а в ній розділу «Позиції» для створення нових категорій одягу , а та позицій в розділі категорій;
- добавлення на сторінку вкладки «Добавити замовлення» в якій відображаються всі актуальні категорії та в категоріях всі актуальні позиції , також добавити кнопку завершити , яка відповідає підтвердженю подачі замовлення та добавленню інформації щодо замовлення;
- добавити вкладку «Історія» в якій відображаються всі затверджені замовлення в яки вказаний номер , дата , час, сума і також добавити змогу відкрити замовлення і переглянути її всю інформацію. Також потрібно розробити фільтр , який буде знаходити замовлення по номерації заказів.

2.2 Вибір інструментів для розроблення CRM-систем

Для розроблення програмного забезпечення є Велкількість мов програмування , кожна з них є унікальною та має переваги та недоліки. Вибір мови програмування залежить від поставленої задачі та її виконання.

Для розроблення веб CRM-системи ми обрали мову – JavaScript. Ця мова відноситься до мультипарадигменних мов , з нею можна використовувати різні підходити до програмування , враховуючи складність та специфіку майбутньої програми. JavaScript по думці багатьох аналітиків рахується одніє з найбільш популярнішою в світі.

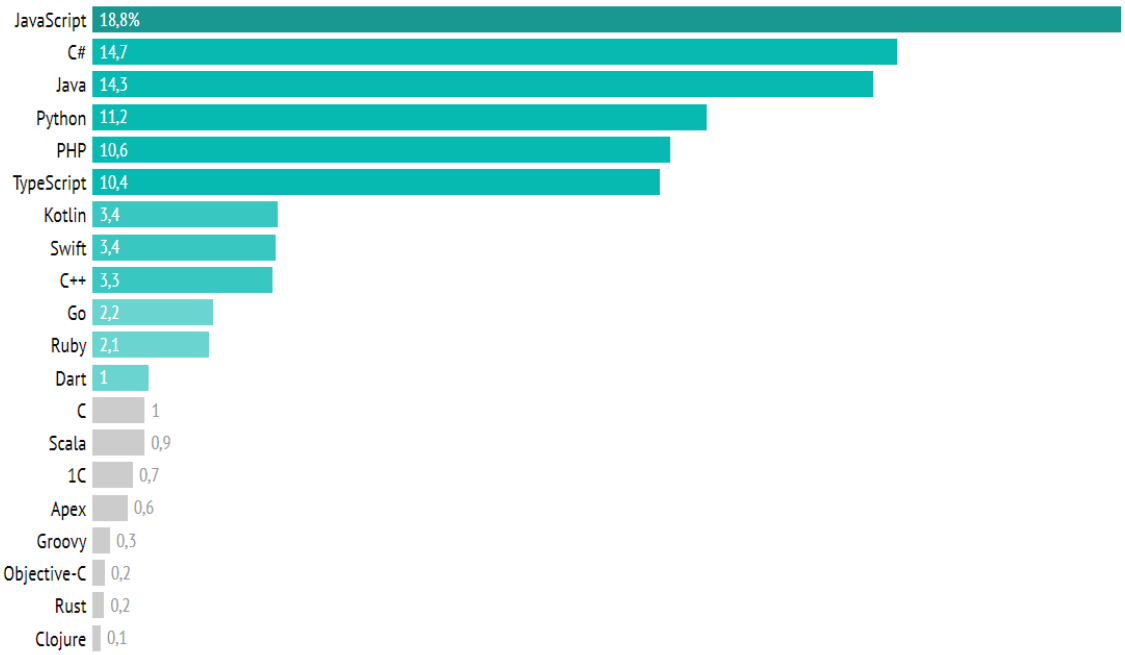

Рисунок 2.1 – Рейтинг мов програмування

JavaScript – універсальна мова програмування, любий браузер та будь-який комп’ютерний пристрій читає код на JS (JavaScript) і навіть мобільні програми. Також на цій мові написано багато фреймворків , які допомагають в розробці.

Великим плюсом JS є багато корисних та ефективних фреймоврків. Фреймворки JS – це бібліотеки програмування JavaScript , в яких є початково

написаний код для використання в стандартних функціях і задач програмування.

Основним плюсом використання фреймворку збільшення швидкості виробництва, за допомогою нього потрібно писати менше коду вручну, тому що вже є написані готові функції та шаблони.

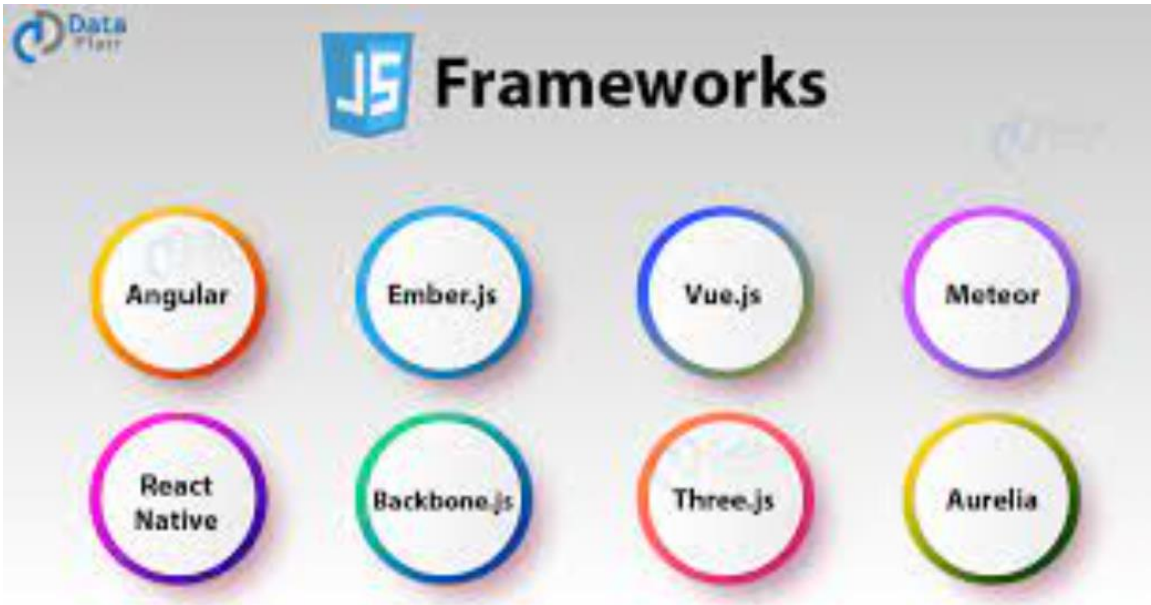

Рисунок 2.2 – Фреймворки JavaScript

В розробці проекту використаємо такі фреймворки як: Angular, Vue.Js, Node.Js.

Angular - фреймворк JavaScript, що допомагає розробникам створювати веб-програми. Технічно його можна використовувати будь-де, але найкраще він працює в нестандартних додатках з даними, там, де потрібна комплексна маршрутизація та анімація. Також на його основі створено безліч ігор та додатків із доповненою реальністю.

Node.js (Node) - це платформа з відкритим вихідним кодом для роботи з мовою JavaScript, побудована на Chrome V8. Вона дозволяє писати серверний код для веб-застосунків і динамічних веб-сторінок, а також програм командного рядка. В основі платформи - подієво-керована модель з неблокуючими операціями введення-виведення, що робить її ефективною та

легкою. До появи Node.js програми, написані мовою програмування JavaScript, можна було запускати лише у браузері. З появою платформи можна було писати на JavaScript не тільки в браузері, але і на сервері. Платформу використовують fronted-розробники, backend-розробники та , вона дозволяє написати програму для різних операційних систем : Linux, mac OS та Windows може використовуватися для створення API. Також Node.js застосовується для розробки крос-платформних програм: наприклад, списку завдань, який повинен працювати на різних платформах, синхронізувати дані в реальному часі та відправляти на мобільний пристрій. Node.js використовується при створенні сервісів з постійним обміном інформацією з користувачем: соціальних мереж, онлайн-ігор, чатів, систем спільної роботи над проектом, онлайн-редакторів тексту.

Плюси використання Node.Js

- Висока швидкість. JavaScript-код, який виконується в середовищі Node.js, може бути в кілька разів швидше, ніж написаний мовами, наприклад Ruby або Python. У Node.js використовується модель асинхронного програмування. Модель дозволяє продовжити обробку інших завдань, не чекаючи завершення передачі. Коли потрібно виконати операцію введення-виведення на кшталт доступу до файлової системи або бази даних, Node.js не блокує головний потік очікуванням результатів. Платформа ініціює її виконання та продовжує виконувати інші завдання, доки результати попередньої операції не будуть отримані.

- Універсальність та гнучкість. У Node.js виконується код написаний на JavaScript. Це означає, що frontend-розробники, які вже використовують JavaScript у браузері, можуть писати і клієнтський, і серверний код звичною мовою програмування, не вивчаючи інструмент з нуля. У Node.js можна швидко переходити на нові стандарти ECMAScript у міру їхньої реалізації. Нові можливості мови стають доступними відразу після встановлення версії Node.js, що підтримує їх.

- Велика кількість модулів та бібліотек. Екосистема Node.js швидко розвивається завдяки менеджеру пакетів NPM. Він містить понад 500 000 модулів та бібліотек open-source, які знаходяться у вільному доступі. Також постійно з'являються нові.
- Робота на Chrome V8. Node.js працює на JavaScript V8 від Google. V8 - д JavaScript з відкритим вихідним кодом, що розповсюджується за ліцензією BSD. Він застосовується у браузерах на основі Chromium. Це означає, що в Node.js використано напрацювання тисяч інженерів. Двигун написаний C++, має відкритий вихідний код і просунуті бібліотеки.
- Vue.js - це прогресивний JavaScript фреймворк з відкритим вихідним кодом, призначений для розробки інтерфейсу користувача. Він є одним із найпопулярніших фреймворків для спрощення веб-розробки. VueJS працює в основному з рівнем представлення. Його з легкістю можна інтегрувати у великі проекти для фронт-енд розробки.

Vue.js використовується при розробці:

- швидких веб-сайтів та додатків, блогів невеликого розміру;
- сайтів з високим навантаженням - інтернет-магазинів, інформаційних порталів;односторінкових (SPA) додатків - соціальних мереж, мікроблогів-сервісів, CMS ;
- адаптивних інтерфейсів;
- розділів особистих кабінетів та сторінок користувача;
- інтерфейсів авторизації, онлайн-чатів, форм заявки та інших функціональних блоків

Можна виділити такі переваги фреймворка:

- прогресивність, ядро Vue.js ідеально підходить для впровадження у існуючий проект. Так, сайт готового продукту може продовжувати працювати, наприклад, на jQuery (раніше використовуваній бібліотеці), але частина модулів поступово переписуватиметься на Vue до повноцінного переходу;

- простота , почати працювати з фреймворком можна без базових знань у веб-розробці. Низький поріг входження — причина популярності у розробників-початківців;
- невелика вага, фреймворк займає близько 20 кБ, тому реалізовані у ньому проекти швидше завантажуються і краще ранжуються пошуковими роботами.

2.3 Додаткові сервіси для розроблення CRM-систем

Mongoose — це інфраструктура JavaScript, яка зазвичай використовується в Node.js з базою даних MongoDB.

Mongoose надає неймовірну кількість функцій для створення та роботи зі схемами. В даний час Mongoose містить вісім типів SchemaType, які зберігаються для якості, коли воно зберігається в MongoDB.

Кожен тип даних дозволяє вам вказати:

- значення за замовчуванням користувальницька функція перевірки
- вказати поле, обов'язкове для заповнення
- функція get, яка дозволяє вам маніпулювати даними перед тим, як вони будуть повернуті як об'єкт
- функція set, яка дозволяє вам маніпулювати даними перед їх збереженням у базі даних
- створювати індекси, що дозволяють швидше отримувати дані

На додаток до цих загальних параметрів, певні типи даних дозволяють додатково налаштовувати, як дані зберігаються і вилучаються з бази даних.

2.4 Вибір системи управління базами даних

В якості бази даних було обрано MongoDb. MongoDB — кросплатформна документоорієнтована база даних з відкритим програмним забезпеченням і провідна база даних NoSQL, написана на C++. Вона спирається на концепції колекцій та документів.

Головні особливості MongoDB:

- це кроссплатформенна документоорієнтована база даних NoSQL з відкритим вихідним кодом.
- вона не вимагає опису схеми таблиць, як у реляційних БД. Дані зберігаються у вигляді колекцій та документів.
- між колекціями немає складних з'єднань типу JOIN, як між таблицями реляційних БД. Зазвичай з'єднання здійснюється за збереження даних шляхом об'єднання документів.
- дані зберігаються у форматі BSON (бінарні JSON-подібні документи).
- у колекцій не обов'язково має бути схожа структура. Один документ може мати один набір полів, тоді як інший документ — зовсім інший (як тип, і кількість полів).

В одному документі можуть бути поля різних типів даних, дані не потрібно наводити до одного типу. Основна перевага MongoDB полягає в тому, що вона може зберігати будь-які дані, але ці дані мають бути у форматі JSON.

СУБД MongoDB покладається на концепцію бази даних, колекцій та документів. Розглянемо основні терміни:

- База даних – це фізичний контейнер для колекцій.
- Колекція – група документів MongoDB.
- Документ — запис у колекції MongoDB, набір пар ключ-значення.

- Поле – ключ у документі.

Переваги використання MongoDb:

- документоорієнтована база — збереження даних у форматі документів замість формату реляційного типу, що робить MongoDB дуже гнучкою та адаптованою до бізнес-вимог. Можливість зберігання різних типів даних особливо важлива при роботі з великими даними, які збираються з різних джерел і не лягають на одну структуру.
- спеціальні запити — MongoDB підтримує пошук полями, діапазонні запити та пошук регулярних виразів. Можуть бути зроблені запити повернення певних полів у документах.
- індексація — ви можете створити індекси для покращення продуктивності пошуку в MongoDB. Будь-яке поле в документі може бути проіндексовано. Це забезпечує високу швидкість роботи СУБД.
- реплікація - ця СУБД може забезпечити високу доступність за допомогою наборів реплік. Набір реплік складається з двох або більше екземплярів MongoDB. Кожна репліка набору може бути первинної чи вторинної. Первинна репліка – головний сервер, який взаємодіє з клієнтом та виконує всі операції читання/запису. Повторні репліки зберігають копію даних первинної репліки за допомогою вбудованої реплікації. Якщо з первинною реплікою щось трапилося, відбувається автоматичне перемикання на вторинну репліку, потім стає основним сервером.
- Балансування навантаження MongoDB використовує концепцію шардингу для горизонтального масштабування за допомогою поділу даних між декількома екземплярами БД. Вона може працювати на декількох серверах, балансуючи навантаження та/або дублюючи дані, щоб підтримувати працездатність системи у разі апаратного збою.
- Можливість розгорнути у хмарі – ви отримуєте готову до роботи, оптимально налаштовану, масштабовану та керовану базу даних

- Доступність - MongoDB підтримує всі популярні мови програмування, її можна використовувати безкоштовно.

3 СТВОРЕННЯ CRM-СИСТЕМИ

3.1 Архітектура програми

Взаємозв’язок програми буде побудований за допомогою серверної сторони, клієнтської сторони та бази даних.

До серверної сторони будуть входити бібліотеки JavaScript такі як Express, Mongoose, Passport. За допомогою бібліотеки Mongoose буде виконаний зв’язок між базою даних та серверною стороною, клієнтська частина Mongoose буде розроблена серверна частина програми, за допомогою Angular буде розроблено клієнтську частину програми. MongoDb буде виконувати систему управління базою даних і також за допомогою бібліотеки Passport буде розроблене вікно авторизації та реєстрації.

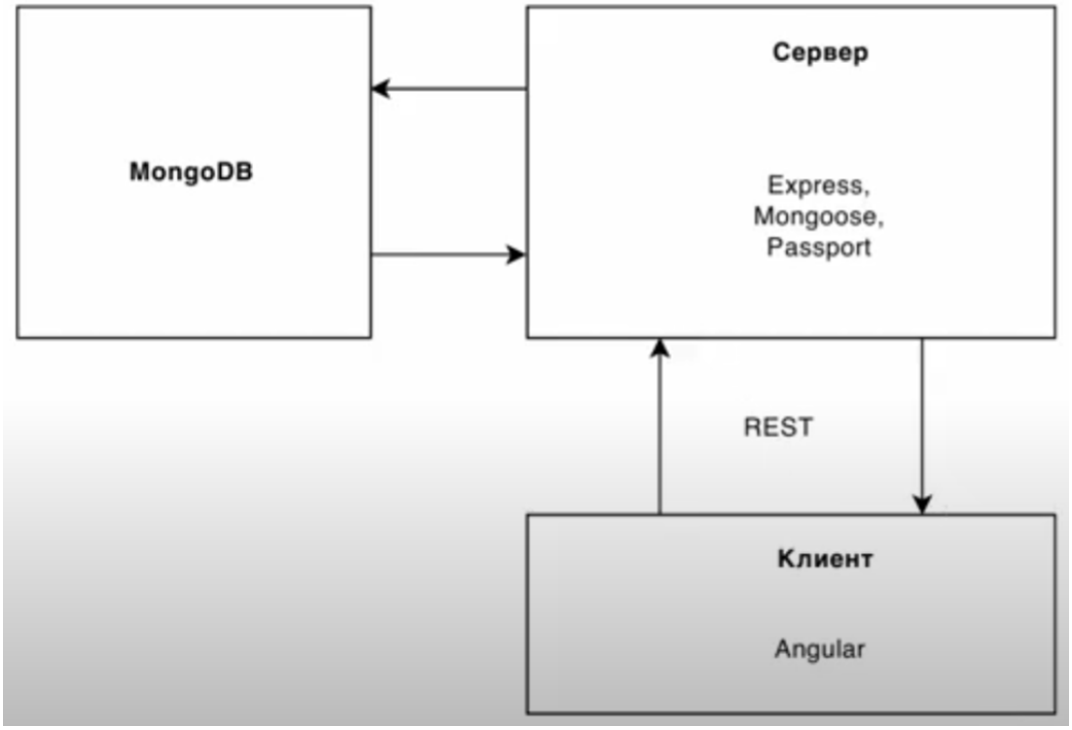

Рисунок 3.1.1 – Діаграма виконання CRM-системи

3.2 Розроблення серверної сторони проекта

Для початку роботи з серверною стороною проекту потрібно підключити бібліотеки Experess, Mongoose та Passport до проекту та запустити сервер.

JS index.js 6

JS index.js > ...

```
1   const app = require('./app')
2   const port = process.env.PORT || 5000
3
4
5   app.listen(port, () => console.log('Server has been
    started on ${port}'))
```

JS app.js 9+

JS app.js > authRoutes

```
1   const express = require('express')
2   const passport = require('passport')
3   const mongoose = require('mongoose')
```

Рисунок 3.1 – Підключення бібліотеки Express

Створюємо маршрутизацію для навігації по сайту. Створюємо маршрутизацію для вкладки авторизації та реєстрації. Для роботи з HTTP запитами використовують методи POST, GET, DELETE , PATCH.

Метод POST використовується для створення запиту, при якому сервер приймає дані.

Метод GET використовується для виконання запита на сервес для доступу до файлів та елементів.

Метод DELETE використовується для видалення вказаного ресурса.

Метод PATH використовується для редагування вказаного ресурса.

```
const express = require('express')
const controller = require('../controllers/auth')
const router = express.Router()
const passport = require('passport')

router.post('/login', controller.login)

router.post('/register', controller.register)

module.exports = router
```

Рисунок 3.2 – Створення маршрутизації для авторизації та реєстрації

Створюємо маршрутизацію для вкладки категорії:

```
JS category.js 9+
routes > JS category.js > ...
1   const express = require('express')
2   const upload = require('../middleware/upload')
3   const controller = require('../controllers/category')
4   const passport = require('passport')
5   const router = express.Router()
6
7   router.get('/', passport.authenticate('jwt',
    {session:false}),controller.getAll)
8   router.get('/:id',passport.authenticate('jwt',
    {session:false}), controller.getById)
9   router.delete('/:id',passport.authenticate('jwt',
    {session:false}), controller.remove)
10  router.post('/',passport.authenticate('jwt',
    {session:false}), upload.single('image'), controller.create)
11  router.patch('/:id ',passport.authenticate('jwt',
    {session:false}), upload.single('image'), controller.update)
12
13  module.exports = router
```

Рисунок 3.3 – Створення маршрутизації для категорій

Створюємо маршрутизацію для вкладки позиції:

JS position.js 9+

routes > JS position.js > ...

```
const express = require('express')
const controller = require('../controllers/position')
const router = express.Router()
const passport = require('passport')

router.get('/:categoryId',passport.authenticate('jwt',
{session:false}), controller.getByCategoryId)
router.post('/', passport.authenticate('jwt',
{session:false}), controller.create)
router.patch('/:id', passport.authenticate('jwt',
{session:false}), controller.update)
router.delete('/:id', passport.authenticate('jwt',
{session:false}), controller.remove)

module.exports = router
```

Рисунок 3.4 – Створення маршрутизації для вкладки позиції

Створюємо маршрутизацію для вкладки замовлення:

JS order.js 9+

routes > JS order.js > ...

```
const express = require('express')
const passport = require('passport')
const controller = require('../controllers/order')
const router = express.Router()

router.get('/', passport.authenticate('jwt',
{session:false}), controller.getAll)
router.post('/', passport.authenticate('jwt',
{session:false}), controller.create)

module.exports = router
```

Рисунок 3.5 – Створення маршрутизації для вкладки замовлення

Створюємо контролери для обробки HTTP запитів, вони використовуються для оброблення моделі і представлення і відправляють відповідповіль клієнту про результат обробки.

Створюємо контролер для реєстрації користувача:

```
module.exports.register = async function(req,res) {

  const candidate = await User.findOne({email: req.body.
  email})

  if(candidate) {
    //Користувач вже створений
    res.status(409).json({
      message: 'Такий email вже зайнятий'
    })
  } else {
    const salt = bcrypt.genSaltSync(10)
    const password = req.body.password
    const user = new User({
      email: req.body.email,
      password: bcrypt.hashSync(password, salt)
    })

    try {
      await user.save()
      res.status(201).json(user)
    }
    catch(e) {
      // Помилку опрацювання
      errorHandler(res, e)
    }
  }
}
```

Рисунок 3.6 - Контролер для реєстрації користувача

За допомогою функції «FindOne» , шукаємо дані для моделі. Якщо змінна «candidate» існує , це означає , що такий користувач вже існує і з’явиться відповідь від сервера про помилку. Якщо провірка email адресу та паролю пройшла успішно , тоді за допомогою методу «save()» , збережемо дані про користувача в базу даних, якщо ні , тоді сервер дасть в відповідь помилку.

Створюємо контролер для авторизації користувача:

```
module.exports.login = async function (req,res) {
  const candidate = await User.findOne({email: req.body.
  email})

  if (candidate) {
    // проверка пароль
    const passwordResault = bcrypt.compareSync(req.body.
    password, candidate.password)
    if (passwordResault) {
      const token = jwt.sign({
        email: candidate.email,
        userId: candidate._id
      }, keys.jwt, {expiresIn: 60 * 60})
      res.status(200).json({
        token: `Bearer ${token}`
        })
    } else {
      //Паролі не співпадають
      res.status(401).json({
        message:'Паролі не співпадають'
      })
    }

  } else {
    res.status(404).json({
      message: 'Користувач не знайдений з таким email'
    })
  }
}
```

Рисунок 3.7 - Контролер для авторизації користувача

За допомогою функції FindOne(), ми робимо пошук користувача в базі. Наступним кроком буде пошук паролю користувача в базі даних. Якщо користувача не буде знайдено або буде не вірний пароль сервер дасть помилку.

Створюємо контролер для вкладки категорія, в вкладці категорії в нас буде використовуватися 5 методів

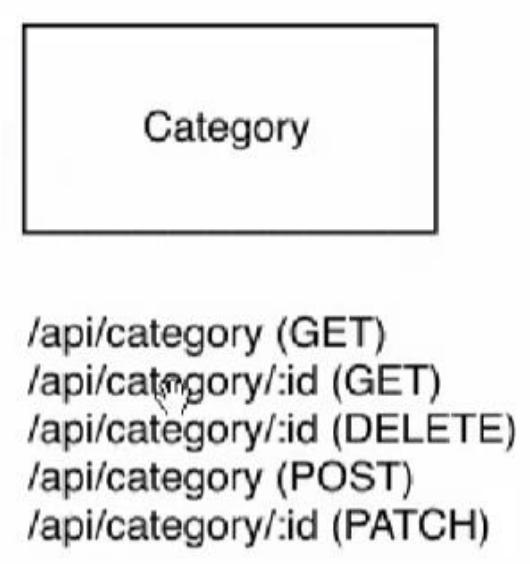

Рисунок 3.8 – Методи вкладки «Категорія»

Метод «getAll» буде відповідати за пошук існуючих категорій в базі

```
module.exports.getAll = async function(req,res) {
  try {
    const categories = await Category.find({
      user: req.user.id
    })
    res.status(200).json(categories)
  } catch (e) {
    errorHandler(res,e)
  }
}
```

Рисунок 3.9 – Метод «getAll»

Метод «getById» буде відповідати за пошук існуючого елемента по id

```
module.exports.getById = async function(req,res) {
  try {
    const category = await Category.findById(req.params.
    id)
    res.status(200).json(category)
  } catch (e) {
    errorHandler(res,e)
  }
}
```

Рисунок 3.10 – Метод «getById»

Метод «remove» відповідає за видалення існуючого елемента з бази

```
module.exports.remove = async function(req,res) {
  try {
    await Category.remove({_id: req.params.id})
    await Position.remove({category: req.params.id})
    res.status(200).json({
      message: 'Категорія видалена'
    })
  } catch (e) {
    errorHandler(res,e)
  }
```

Рисунок 3.11 – Метод «remove»

Метод «create» відповідає за створення нової категорії

```
module.exports.create = async function(req,res) {
    const category = new Category({
    name: req.body.name,
    user: req.user.id,
    imageSrc: req.file ? req.file.path : ''
  })
  try {
    await category.save()
    res.status(201).json(category)
  } catch (e) {
    errorHandler(res,e)
  }
}
```

Рисунок 3.12 – Метод «remove»

Метод «update» відповідає за редагування категорії

```
module.exports.update = async function(req,res) {
  const updated = {
    name: req.body.name
  }

  if (req.file) {
    updated.imageSrc = req.file.path
  }

  try {
    const category = await Category.findOneAndUpdate(
      {_id: req.params.id},
      {$set: updated},
      {new: true}
    )
    res.status(200).json(category)
  } catch (e) {
    errorHandler(res,e)
  }
}
```

Рисунок 3.13 – Метод «update»

3.3 Розроблення клієнтської сторони проекту сторони проекта

За клієнтську сторону проекту буде відповідати бібліотека JavaScript – Angular. Її можна встановити за допомогою команди в консолі npm install –g angular/cli. За допомоги консольної команди ng new client створюємо папку «client» в якій буде створений вміст клієнтської сторони проекту.

Створюємо консольну команду для запуску серверної та клієнтської сторони

```
▷ Отладка
"scripts": {
  "start": "node index.js",
  "server": "nodemon index",
  "client-install": "npm install --prefix client",
  "client": "npm run start --prefix client",
  "dev": "concurrently \"npm run server\" \"npm run
  client\""
},
```

Рисунок 3.14 – Запуск проекта

Створюємо маршрутизацію в папці «client»

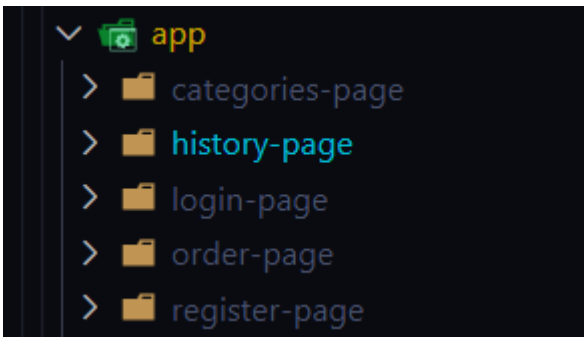

Рисунок 3.15 – маршрутизація в папці «client»

Підключаємо усі створені модулі та компоненти до проекту

```
@NgModule({
  declarations: [
    AppComponent,
    LoginPageComponent,
    AuthLayoutComponent,
    SiteLayoutComponent,
    RegisterPageComponent,
    OverviewPageComponent,
    AnalyticsPageComponent,
    HistoryPageComponent,
    OrderPageComponent,
    OrderPositionsComponent,
    CategoriesPageComponent,
    CategoriesFormComponent,
    PositionsFormComponent,
    OrderCategoriesComponent,
    HistoryListComponent,
    HistoryFilterComponent,
```

```
imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule
],
```

Рисунок 3.16 – Підключення модулі та компонентів

Створення візуальної сторони сайту сторінки логін та реєстарції

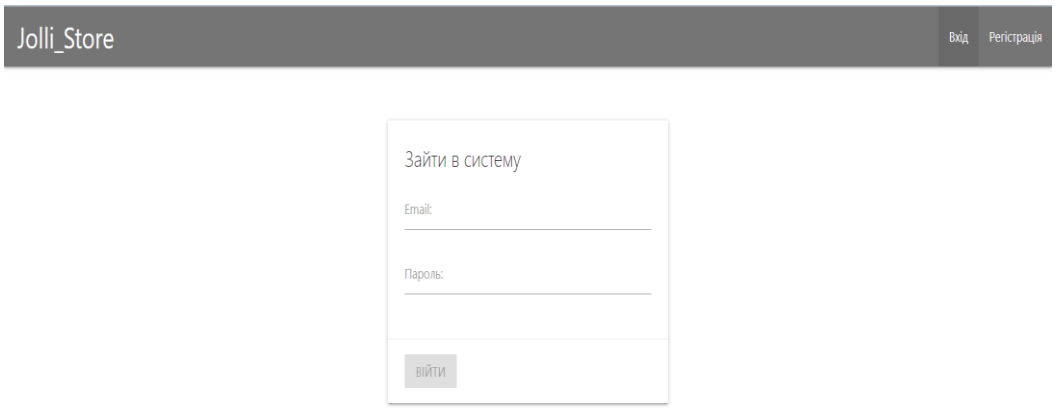

Рисунок 3.17 – Сторінка логіна та реєстрації

Оптимазація форми логіну та реєстрації. Розробляємо правила валідації для для форми.

Задаємо початкові значення «null» , бо при вході на сторінку не мають бути встановлені дані. За допомогою валідатора «Validators.email» провіряємо чи коректно задане значення email в строку, також за допомогою валідатора «Validators.minLenght(6)» вказуємо мінімальні довжину пароля в формі.

```
ngOnInit() {
  this.form = new FormGroup({
    email: new FormControl(null, [Validators.required, Validators.email]),
    password: new FormControl(null, [Validators.required, Validators.minLength(6)])
  })
```

Рисунок 3.18 – Валідація форми

Наступним кроком потрібно задати дію кнопці «Війти»

```
<div class="card-action">
    <button
    type="submit"
    class="modal-action btn waves-effect"
    [disabled] = "form.invalid || form.disabled">
    Війти</button>
</div>
```

Рисунок 3.19 – Кнопка «Війти»

Створимо додатковий сервіс з методом «POST» для запросу інформації з бази даних

```
login (user: User):Observable<{token: string}>{
 return this.http.post<{token: string}>('/api/auth/login', user)
   .pipe(
     tap(
       ({token}) => {
       localStorage.setItem('auth-token', token)
       this.setToken(token)
         }
       )
   )
}
```

Рисунок 3.20 – Додатковий сервіс

В момент нажаття на кнопку , якщо форма проходить валідність , ми перехедомо на сторінку «Огляд».

```
  }
  onSubmit() {
    this.form.disable()

    this.aSub = this.auth.login(this.form.value).subscribe(
      () => this.router.navigate(['/overview']),
    (error: any) => {
        console.warn(error)
        this.form.enable()
      }

    )
  }
```

Рисунок 3.21 – Кнопка «Вхід»

Наступний компонент буде форма реєстрації.

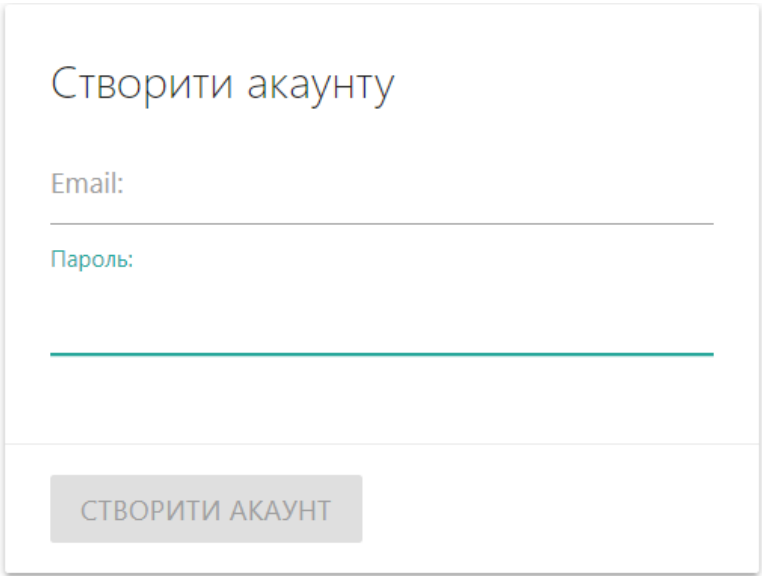

Рисунок 3.22 – Форма «Реєстрації»

Для початку нам потрібно реалізувати метод «POST» для отримання даних з бази даних.

```
register (user:User): Observable<User>{ ···
}

login (user: User):Observable<{token: string}>{
 return this.http.post<{token: string}>('/api/auth/login', user)
   .pipe(
     tap(
       ({token}) => {
       localStorage.setItem('auth-token', token)
       this.setToken(token)
         }
       )
   )
}
```

Рисунок 3.23 – Метод «POST» для реєстрації

Провіряємо чи введені коректно і також чи ці дані не находяться в системі вже зареєстровані. Якщо дані введені валідні та на находяться зареєстровані в системі , користувача перенаправлено в форму авторизації.

```
onSubmit() {
  this.form.disable()
  this.aSub = this.auth.register(this.form.value).subscribe(
    () => {
      this.router.navigate(['/login'], {
        queryParams: {
          registred: true,
        }
      })
    },
    (error: any) => {
      console.warn(error)
      this.form.enable()
    }
  )
}
```

Рисунок 3.24 – Провірка даних при реєстрації

Також потрібно захистити сайт від перегляду без авторизції , повинно бути дозволена сторінка без захисту – авторизації а реєстрації.

```
export class AuthGuard implements CanActivate, CanActivateChild{
constructor(private auth: AuthService,
            private router: Router) {

}
  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<boolean> {
   if(this.auth.isAuthenticated()){
    return of(true)
   } else {
    this.router.navigate(['/login'], {
      queryParams: {
        accessDenied: true
      }
    })
    return of(false)

   }
  }
  canActivateChild(route: ActivatedRouteSnapshot, state: RouterStateSnapshot):  Observable<boolean> {
    return this.canActivate( route, state)
  }
}
```

Рисунок 3.25 – Захист проекту без реєстрації

Створюємо компонент для переходу між сторінками

```
  @ViewChild('floating') floatingRef: ElementRef
links = [
  {url:'/overiew', name:"Обзор"},
  {url:'/analytics', name:"Аналітика"},
  {url:'/history', name:"Історія"},
  {url:'/order', name:"Добавити замовлення"},
  {url:'/categories', name:"Асортимент"},
]
```

Рисунок 3.26 – Компонент SiteLayOutComponent

Розроблення вкладки категорії та позиції. За допомогою методу fetch(), получимо ресурси по мережі асинхронно

```
fetch(): Observable<Category[]> {
  return this.http.get<Category[]>('/api/category')
}
```

Рисунок 3.27 – Використання методу fetch()

Відображення списку всіх категорій

```
ngOnInit() {
  // this.categories$ = this.categoriesService.fetch()
  // console.log(this.categories$);
  this.categoriesService.fetch().subscribe((data) => {
    this.arrCategory = data;
  })
}
}
```

Рисунок 3.28– Список всіх категорій

Потрібно розробити в вкладці «Асортимент» зробити перехід на нову сторінку , де є можливість редагувати категорію та добавляти нову позицію в категорію. За допомогою дерективи routerLink робимо перехід на нову сторінку по вказаному url адресу

```
<div class="row" *ngIf="arrCategory.length > 0">
    <div class="col s12">
        <div class="collection">
          <a
            *ngFor = "let category of arrCategory"
            [routerLink]="['/categories', category._id]"
            class="collection-item"
          >
          {{category.name}}
          </a>
        </div>
    </div>
</div>
```

Рисунок 3.29 – Використання routerLink

Розроблення функцій для добавлення нових категорій та редагування поточних категорій. Для редагування наявної категорії розробимо метод getById для отримання інформації про поточну категорію.

```
getById(id: string): Observable<Category> {
  return this.http.get<Category>(`/api/category/${id}`)
}
```

Рисунок 3.30 – Отримання інформації про категорію

За допомогою форми getById получили поточну інформацію про категорію та тепер можемо змнити ці дані за допомогою методу patchValue.

```
  ).subscribe({
    next: (category) => {
      if(category){
      this.category = category
      this.form.patchValue({
        name: category.name
      })
      this.imagePreview = category.imageSrc
    }},
    error: (e) => MaterialService.toast(e.error.message)
})
```

Рисунок 3.31 – Заміна даних наявної категорії

Для видалення поточної категорії розробимо метод delete()

```
  delete( id: string ): Observable<Message> {
    return this.http.delete<Message>(`/api/category/${id}`)
  }
}
```

Рисунок 3.32 – Видалення категорії

Для видалення категорії створюємо змінну , в разі підтверження видаляємо категорію

```
deleteCategory(){
  const decision = window.confirm("Ви впенені , що хочете видалити категорію ? ")

  if(decision) {
    this.categoriesService.delete(this.category._id)
    .subscribe({
    next: (response: { message: string; }) => {
      MaterialService.toast(response.message)
    },
    error: (e) => {
      MaterialService.toast(e.error.message)
    },
    complete:() => {
      this.router.navigate(['/categories'])
    }
  })
```

Рисунок 3.33 – Конструкція для видалення категорії

Для того щоб добавляти та редагувати позиції скористаємося мотодом fetch() для получення інформації

```
fetch(categoryId: string): Observable<Position[]> {
 return this.http.get<Position[]>(`/api/position/${categoryId}`)
}

 ngOnInit() {
   // this.categories$ = this.categoriesService.fetch()
   // console.log(this.categories$);
   this.categoriesService.fetch().subscribe((data) => {
     this.arrCategory = data;
   })
 }
}
```

Рисунок 3.34 – Получення інформації

```
ngAfterViewInit() {
 this.modal = MaterialService.initModal(this.modalRef)

}

onSelectPosition(position: Position) {
  this.modal.open()
}
onAddPosition() {
  this.modal.open()

}
```

Рисунок 3.35 – Відкриття модального вікна позицій

Опрацьовуємо кнопку «Зберегти» за допомогою методу create() та зберігаємо інформацію модального вікна в базу даних

```
create(position: Position): Observable<Position> {
  return this.http.post<Position>('/api/position',
  position)
}

  this.positionsService.create(newPosition)
  .subscribe({
    next: (position) => {
      MaterialService.toast("Позиція створена")
      this.positions.push(position)
    },
    error: (e) => {
      this.form.enable()
      MaterialService.toast(e.error.message)
    }
  })
 }
```

Рисунок 3.36 – Збереження інформації форми

Реалізація вкладки «Добавити замовлення», створюємо масив list[] в який буде добавляти всі створені позиції

```
add(position: Position) {
  const orderPosition: OrderPosition = Object.assign({},{
    name: position.name,
    cost: position.cost,
    quantity: position.quantity,
    _id: position._id
  })

  const candidate = this.list.find(p => p._id === orderPosition._id )
  if(candidate) {
    candidate.quantity += orderPosition.quantity
  } else {
    this.list.push(orderPosition)
  }
  this.computerPrice()

}
```

Рисунок 3.37 – Добавлення замовлення в масив

Для збереження замовлення в базі , використаємо метод create()

```
create(order:Order): Observable<Order> {
  return this.http.post<Order>('/api/order', order)
}
```

Рисунок 3.38 – Метод create()

```
  submit() {
    this.modal.close()
    const order: Order = {
      list: this.order.list.map(item => {
        delete item._id
        return item
      })
    }

    this.ordersService.create(order).subscribe({
      next:(newOrder: any) => {
        MaterialService.toast(`Заказ №${newOrder.order} був добавлений`)
        this.order.clear()
      },
      error:(error: any) => {
        MaterialService.toast(error.message)
      },
      complete: () => {
        this.modal.close()
      }

    })
  }
}
```

Рисунок 3.39 – Збереження замовлення в базі

Для відображення історії замовлення витягнемо, візьмемо інформацію про створення замовлення та дані замовлення з бази даних

```
<tr *ngFor =" let order of orders">
    <td>{{order.order}}</td>
    <td>{{order.date | date:'dd.MM.yyyy'}}</td>
    <td>{{order.date | date: 'HH.mm'}}</td>
    <td>{{computePrice(order)}} грн</td>
    <td>
        <button class="btn btn-small  grey darken-1" click="selectOrder(order)">
            <i class="material-icons">open_in_new</i>
        </button>
    </td>
</tr>
```

Рисунок 3.40 – Відображення замовлень

3.4 Огляд створеної системи

Створене вікно авторизації.

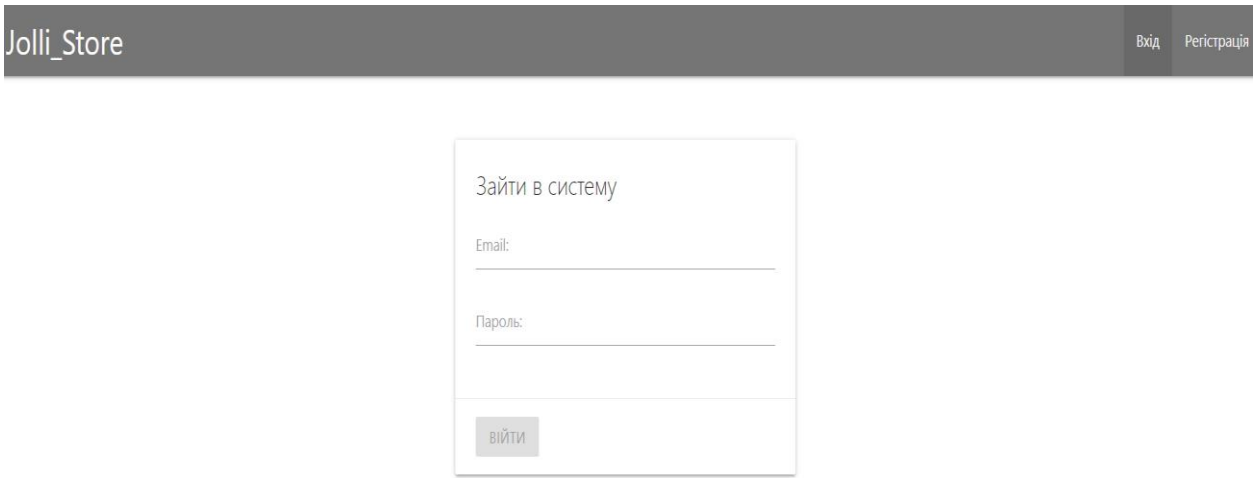

Рисунок 3.41 – Вікно авторизації

Створене вікна реєстрації.

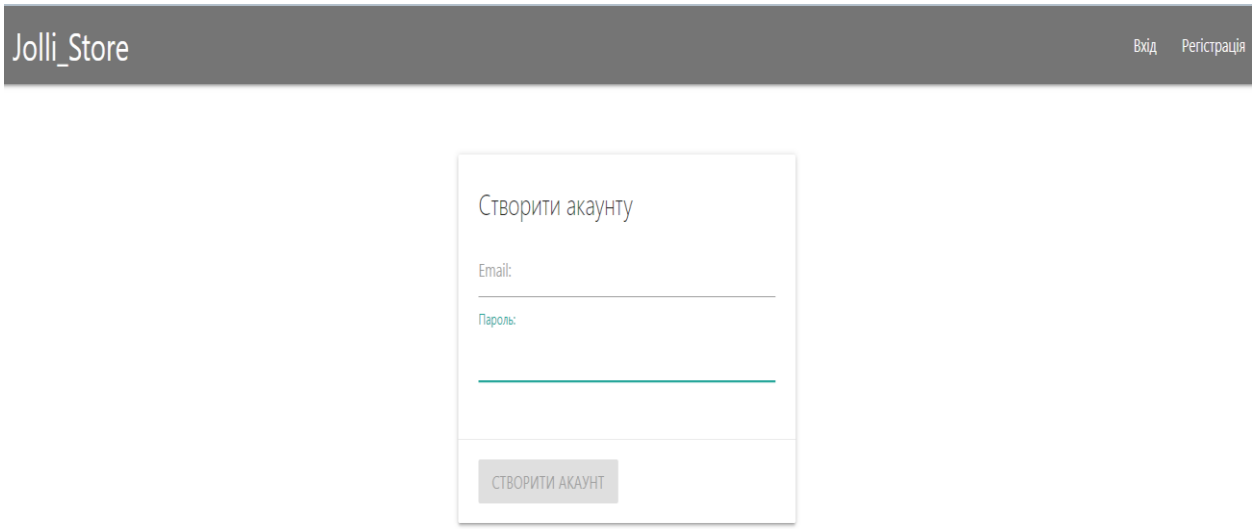

Рисунок 3.42 – Вікно реєстрації

Дизайн сторінки створений з бокового меню в яке входити 3 пункти: «Історія», «Добавити замовлення», «Асортимент»

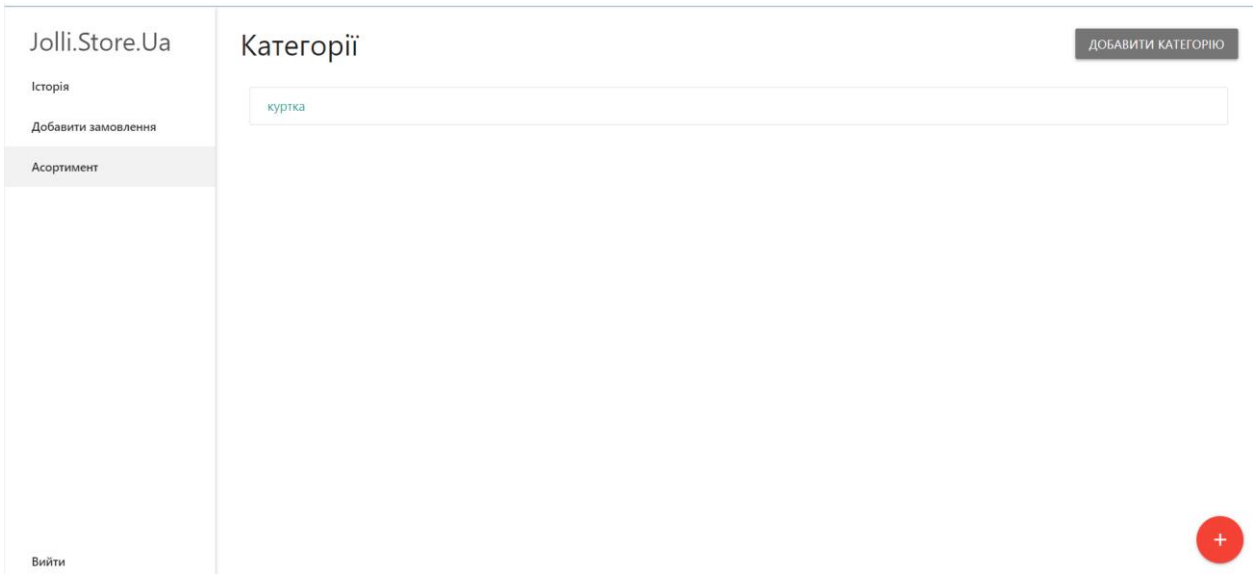

Рисунок 3.43 – Огляд дизайну

В пункті «Категорія» відображаються наявність категорії , які ми створили за допомогою кнопки «Добавити категорію».

При нажатті на кнопку «Добавити категорію» відкривається модальне вікно, де ми можемо задати назву , загрузити фотографію нової категорії.

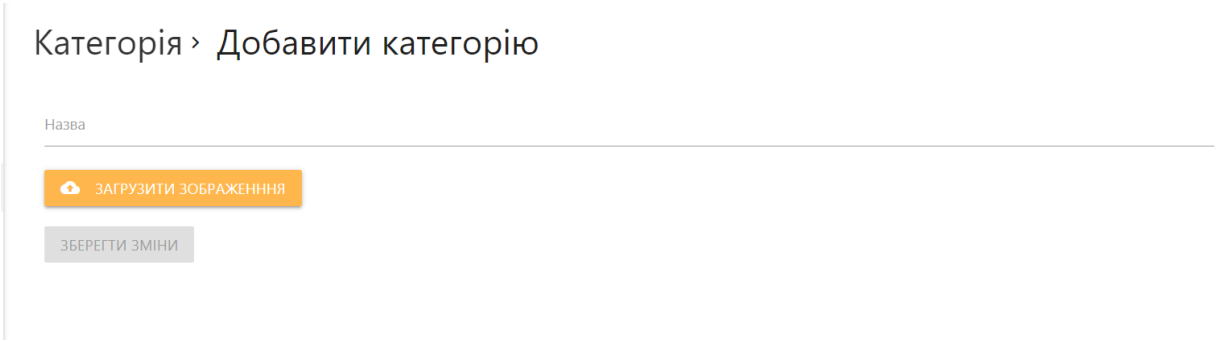

Рисунок 3.44 – Модальне вікно

При нажатті на наявну категорію , ми можемо відредагувати її , задавши їй нове ім’я та фото, а також можемо добавити позиції для категорій.

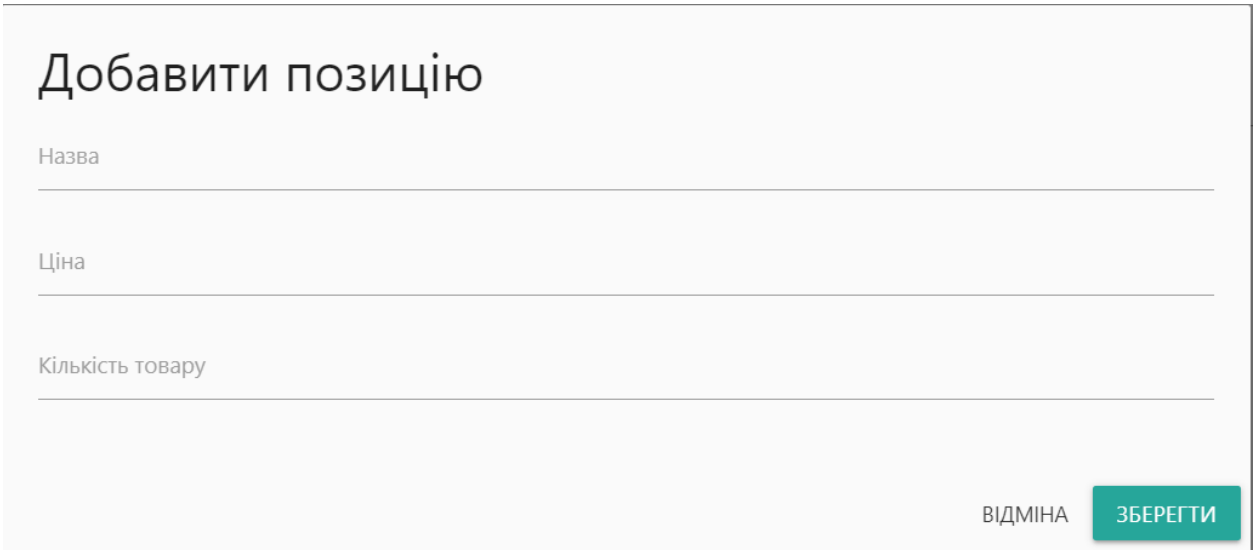

Рисунок 3.45 – Добавити позицію

В розділі «Добавити замовлення» відображаються всі наявні категорії.

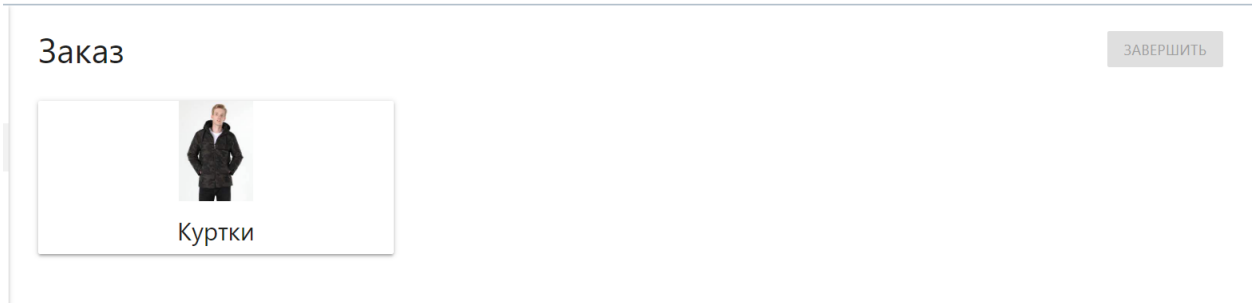

Рисунок 3.46 – Дизайн сторінки Добавити замовлення

Під час нажаття на категорію , ми можемо обрати позицію та добавити її до замовлення

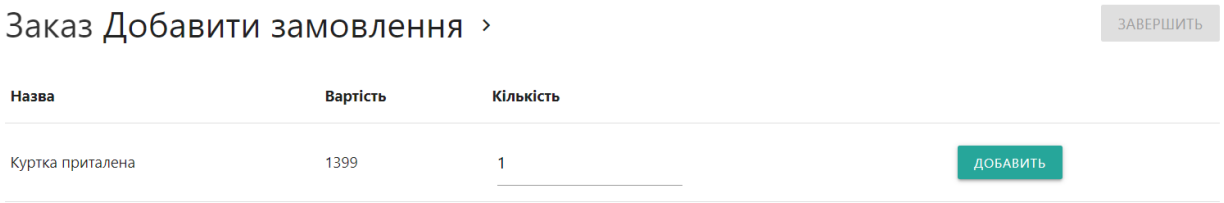

Рисунок 3.47 – Етап оформлення замовлення

Кнопка «Завершити» відповідає за підтвердження замовлення

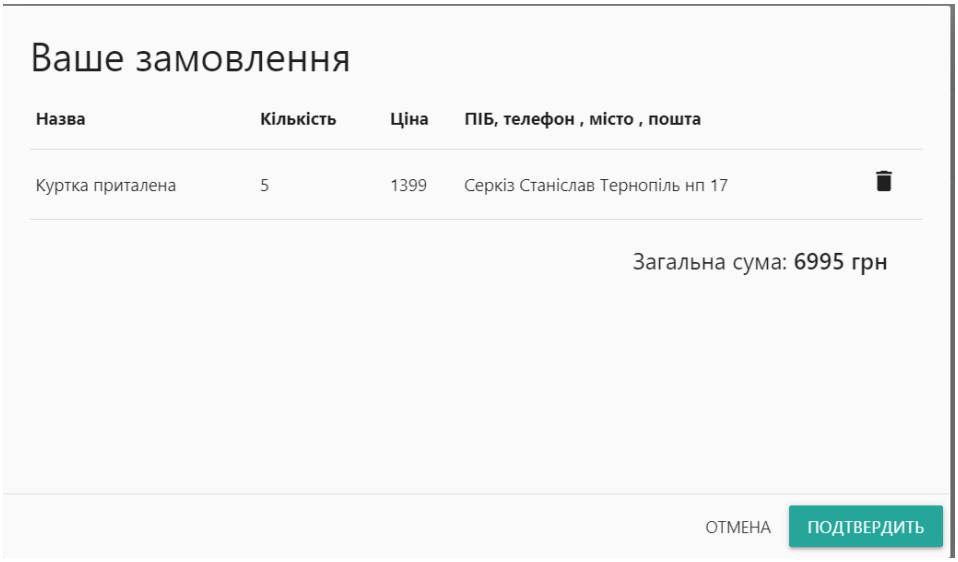

Рисунок 3.48 – Кнопка підтвердження замовлення

В розділі «Історія» відображаються всі минулі створені замовлення, також є фільтр для пошуку замовлень

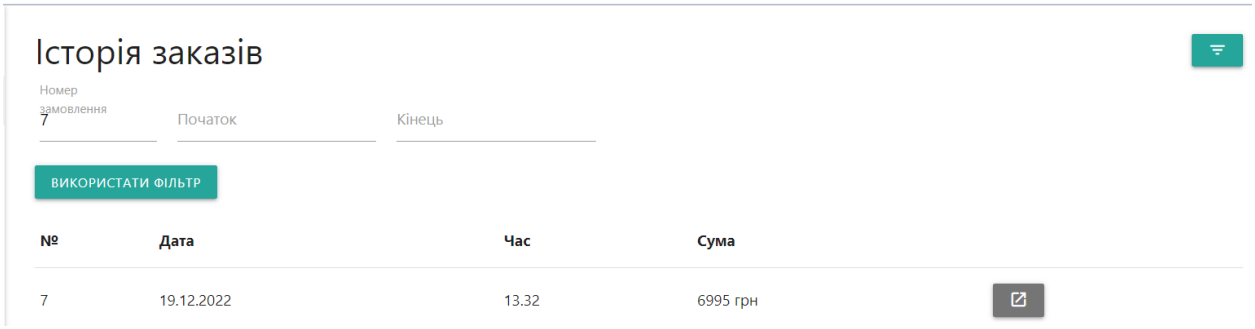

Рисунок 3.49 – Розділ «Історія»

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Організація безпечних умов праці при монтажі комп'ютерної мережі

Для безпечного монтажу комп'ютерної мережі робітники повинні бути проінструктовані щодо розпорядку на робочому місці, порядку переміщення по території об'єкта, місце відпочинку під час технологічних та обідньої перерв, порядок закінчення роботи.

Під час монтажу комп'ютерної мережі необхідно дотримуватись таких правил:

- не допускати деформації кабелю;
- не допускати зайвого натягнення кабелю;
- не допускати надмірного скупчення кабелю в одному місці;
- радіус вигину кабелю не повинен бути меншим ніж 4 – 5 см;
- уникати прокладки кабелю біля електрощитів;
- не прокладати кабель біля опалювальних елементів.

Якщо мережа розташована в декількох приміщеннях або на декількох поверхах будинку, найкращим варіантом буде використання монтажної шафи та мережевих розеток.

Правила кріплення коробів. Для кріплення коробів до стіни використовуються шурупи та пластмасові наповнювачі. Чим більший короб, тим щільніше повинні розташовуватися шурупи або замінити на шурупи більшого розміру.

В разі необхідності частину короба можна відрізати. У місцях поворотів короб необхідно кріпити ретельніше та акуратніше.

При стиковці коробів різної ширини необхідно використовувати спеціальні перехідники.

Розглянемо правила монтажу мережевих розеток. Для прикладу розглянемо монтаж мережевої розетки, яка кріпиться на стіні за допомогою шурупа або клейкого двостороннього скотча. Така розетка складається з трьох частин: основи, кришки та плати з контактною групою.

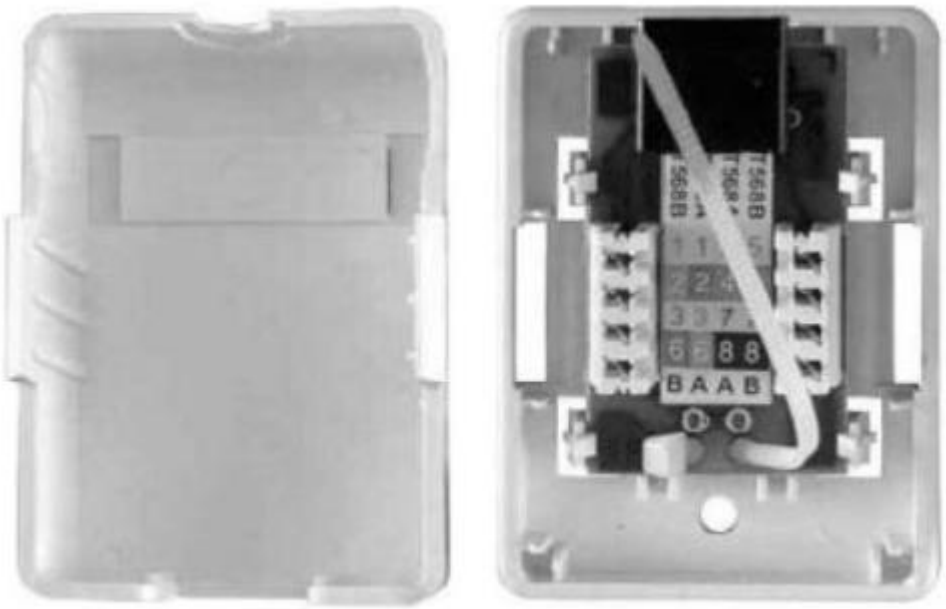

Рисунок 4.1 - Будова мережевої розетки для кріплення на стіні

Під час монтажу такої розетки необхідно дотримуватися наступних правил:

- розібрати розетку на складові частини;
- закріпити основу на стіні;
- затиснути провідники на контактній групі плати згідно маркування;
- закріпити плату на основі;
- закрити розетку кришкою.

Для затиску провідників в розетках використовується спеціальний ніж-вставка.

Встановивши провідники в своїх контактах, натисненням ножа на кожному з провідників потрібно їх зафіксувати

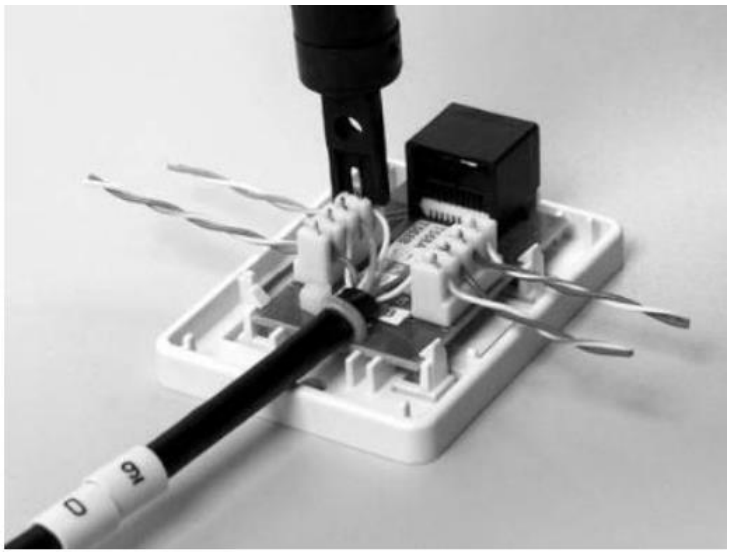

Рисунок 4.2 - Фіксація провідників в контактній площадці.

Розглянемо правила обтиску кабелю. Варто сказати, що у великих мережах використовують готові патч-корди та крос-корди. Проте, знання правил обтиску кабелю та набуття такого досвіду є необхідним, оскільки часто до-водиться створювати додаткові кабелі для підключення обладнання. Розглянемо даний процес детальніше на прикладі створення патч-корда. Для обтиску кабелю «звита пара» використовуються конектори RJ-45.

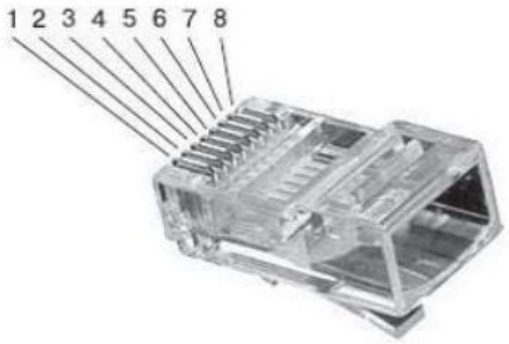

Рисунок 4.3 - Нумерація контактів конектора

4.2 Підвищення стійкості роботи об’єктів торгівлі в воєнний час

Ефективність економіки держави залежить від того, наскільки окремі галузі торгівлі здатні стійко працювати не тільки у звичайних умовах, а й в умовах воєнного часу.

Значні руйнування, пожежі та втрати серед населення, викликані наслідками небезпечних ситуацій (НС), можуть стати причиною різкого скорочення випуску промислової та сільськогосподарської продукції, а отже і зниження торгівельного потенціалу держави. Виникає потреба завчасного вживання заходів щодо забезпечення стійкої роботи промислових об’єктів на випадок виникнення небезпечної ситуації.

На стійкість роботи об’єкта впливають такі фактори:

– захищеність робітників та службовців від уражальних факторів у НС;

– здатність інженерно-технічного комплексу об’єкта (будівель, споруд, обладнання та комунально-енергетичних мереж) протистояти руйнівній дії уражальних факторів аварій, катастроф, стихійного лиха та сучасної зброї;

– надійність постачання об’єкта електроенергією, водою, паливом, комплектуючими та сировиною;

– підготовленість об’єкта до проведення аварійно-рятувальних та відновлюваних робіт;

– оперативність управління виробництвом

Підвищення стійкості об’єкта досягають проведенням комплексу інженернотехнічних, технологічних, організаційних заходів.

До інженерно-технічних заходів належать роботи, що забезпечують стійкість виробничих будівель і споруд, обладнання та комунально-енергетичних систем. Технологічні заходи забезпечують підвищення стійкості об’єкта спрощенням технологічного процесу виробництва кінцевої продукції та виключенням або обмеженням розвитку аварій.

Заходи щодо підвищення стійкості об’єктів здійснюють відповідно до вимог Норм проектування інженерно-технічних заходів цивільного захисту. Дані вимоги призначені для того, щоб в умовах надзвичайних ситуаціях:

– забезпечити захист населення та знизити масштаби руйнувань (пожеж, затоплень, заражень);

– підвищити стійкість роботи об’єктів і галузей економіки;

– створити умови для успішного проведення робіт з ліквідації наслідків НС

Для забезпечення надійного постачання об’єкта господарювання електроенергією, водою та газом в комунально-енергетичних системах слід передбачати:

– дублювання джерел постачання;

– кільцювання систем;

– прокладання комунікацій під землею;

– створення резервних джерел постачання або резервних запасів;

– використання пристроїв для автоматичного вимикання пошкодженої дільниці.

Електропостачання має здійснюватися від енергосистем, до яких входять електростанції на різних видах палива. Електроенергію до дільниць виробництва слід подавати окремими електрокабелями, прокладеними під землею. Також мають існувати автономні резервні джерела електропостачання. Для підвищення надійності водопостачання, крім вище вказаних способів, передбачають повторне використання води для технічних потреб.

Виконання вимог норм проектування сприяє не тільки безпечному та безперебійному функціонуванню об’єктів, але й покращенню умов праці та проживання в певному районі.

Підвищення стійкості системи електропостачання досягається базуванням підприємства на двох і більше джерелах, віддалених на таку

відстань, щоб виключалася можливість руйнування їх одним ЯВ. При відсутності можливості живлення від двох джерел на випадок виходу з ладу основного джерела електропостачання готується резервний автономне джерело. Доцільно також провести заходи щодо захисту існуючих і будівництва резервних підстанцій, а розподільну апаратуру і прилади розмістити в захисних спорудах. Електропостачання слід перевести з повітряної на підземно-кабельне.

Для запобігання виходу з ладу електричних мереж слід встановлювати пристрої автоматичного відключення їх при утворенні перенапруг, які можуть бути створені електромагнітними полями.

На багатьох об'єктах економіки газ може використовуватися як паливо, а на хімічних підприємствах - для технологічних цілей. При руйнуванні газових мереж газ може бути причиною вторинних вражаючих факторів .

На випадок пошкодження джерел газопостачання або газопроводів на великих підприємствах рекомендується мати підземні ємності, службовці акумуляторами газу. Газ під великим тиском закачується в підземні ємності - і служить резервом. Крім того, необхідно готувати підприємство до роботи на різних видах палива і створювати їхні запаси. На газопроводах слід встановити запірну арматуру та крани з дистанційним управлінням, що дозволяє автоматично перемикати потік газу при розриві труб.

Пар використовують багато підприємств. Паропровід повинен бути проведений під землею у спеціальній траншеї, що забезпечує захист труб при дії ударної хвилі.

Котельні зазвичай розміщуються в підвальних приміщеннях, які можуть бути відповідним чином укріплені.

Вихід з ладу системи водопостачання тягне за собою зупинку підприємства і припинення випуску продукції. Для забезпечення сталої роботи об'єктів необхідно:

- створення резервних джерел водопостачання;

- заглиблення в грунт всіх ліній водопроводів;
- оборотне водопостачання з повторним використанням води для технічних цілей.

Підвищення стійкості мереж комунального господарства. Теплову мережу доцільно будувати по кільцевій системі і прокладати труби опалювальної системи в спеціальних каналах під землею. Для підвищення стійкості системи каналізації слід будувати окремі системи каналізації: одна для зливових, інша для промислових і господарських (фекальних) вод.

Захист об'єктів від вторинних факторів ураження. Для захисту об'єктів від вторинних факторів ураження передбачаються наступні заходи:

- підвищення вогнестійкості дерев'яних конструкцій (вогнезахисна фарбування, побілка та ін);
- спорудження водоймищ для гасіння пожеж;
- будівництво сховищ для ЛЗР, нафти, бензину, мазуту, отрутохімікатів за межами території об'єкта.

Підготовка до відновлення порушеного виробництва. По кожному варіанту можливого ураження розробляється план відновлення об'єкта. При цьому складаються розрахунки потрібних матеріалів, механізмів і сил.

В основу планів і проектів відновлення повинно бути закладена вимога - якомога швидше відновити випуск продукції. Тому в проектах відновлення допустимі (в розумних межах) відступу від прийнятих будівельних, технічних та інших норм.

ВИСНОВКИ

В першому розділі кваліфікаційної роботи освітнього рівня «Магістр»: поняття про CRM-систему їх переваги у використанні інтернет магазину та як правильно обирати систему для власного використання. Також було розгянуто правильність підбору CRM систем. Розібрали існуючі системи та проаналізували їх переваги.

Розібрали поняття база даних та системи управління базами даних .

В другому розділі кваліфікаційної роботи:

– Проаналізували функції майбутньої CRM-системи

– Обрали інструменти для розробки системи

– Проаналізували додаткові сервіси, які допоможуть в розробці системи

В третьому розділі кваліфікаційної роботи:

– Змоделювали роботу системи

– Розробили серверну частину проекту

– Розробили клієнтську частину проекту

У розділі «Охорона праці та безпека в надзвичайних ситуаціях» проаналізовано правильний монтаж інтернет кабелю та як підвищити стійкість у сфері торгівлі у воєнний час.

ПЕРЕЛІК ДЖЕРЕЛ

1. What is CRM management? [Електронний ресурс] – Режим доступу до ресурса: URL: https://monday.com/blog/crm-and-sales/crm-management/

2. Customer Relationship Management [Електронний ресурс] – Режим доступу до ресурса: URL: https://www.divaportal.org/smash/get/diva2:1020146/FULLTEXT01.pdf

3. Системи CRM [Електронний ресурс] – Режим доступу до ресурса: URL: http://texnonews.com/systemy-crm.

4. CRM-системи [Електронний ресурс] – Режим доступу до ресурса: URL: https://works.doklad.ru/view/-ZCAAokfZMo/all.html

5. Об’єкт торгівлі [Електронний ресурс] – Режим доступу до ресурса: URL: https://zakon.rada.gov.ua/rada/show/v_320569-02#Text

6. Техніка безпеки під час монтажу розподільних пристроїв [Електронний ресурс] – Режим доступу до ресурса: URL: https://studfile.net/preview/3741178/page:7/

7. Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроям НПАОП 0.00-7.15-18 [Електронний ресурс] – Режим доступу до ресурса: URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text

8. MongoDB [Електронний ресурс] – Режим доступу до ресурса: URL: https://medium.com/nuances-of-programming?source=post_page-----6e844437b0de----------

9. CRM-системи [Електронний ресурс] – Режим доступу до ресурса: URL: https://works.doklad.ru/view/-ZCAAokfZMo/all.html

10. MongoDB: слабкі сторони типової нереляційної бази банних [Електронний ресурс] – Режим доступу до ресурса: URL: http://www.rusnauka.com/35_NII_2017/Informatica/4_229459.doc.htm

11. Переваги MongoDb [Електронний ресурс] – Режим доступу до ресурса: URL: https://blog.skillfactory.ru/glossary/mongodb/

12. Робота з MongoDb [Електронний ресурс] – Режим доступу до ресурса: URL: https://works.doklad.ru/view/-ZCAAokfZMo/all.html

13. JavaScript [Електронний ресурс] – Режим доступу до ресурса: URL: https://uk.wikipedia.org/wiki/JavaScript

14. Чому javascript універсальна мова програмування [Електронний ресурс] – Режим доступу до ресурса: URL: https://dou.ua/forums/topic/35184/

15. Introduction to angular [Електронний ресурс] – Режим доступу до ресурса: URL: https://angular.io/guide/providers

16. Communicating with backend services using HTTP [Електронний ресурс] – Режим доступу до ресурса: URL: https://angular.io/guide/http

17. Топ 5 популярних системи керування базами даних [Електронний ресурс] – Режим доступу до ресурса: URL: URL: https://info-comp.ru/toppopular-database-management-systems

18. Management Tools - Customer Relationship Management - Bain & Company [Електронний ресурс] – Режим доступу до ресурса: URL: www.bain.com.

19. Industry Specific/Vertical Market CRM Solutions [Електронний ресурс] – Режим доступу до ресурса: URL: smallbizcrm.com

20. International Book Series "Information Science and Computing [Електронний ресурс] – Режим доступу до ресурса: URL: http://www.foibg.com/ibs_isc/ibs-07/IBS-07-p25.pdf

21. Все про node.js [Електронний ресурс] – Режим доступу до ресурса: URL: https://ru.hexlet.io/blog/posts/zachem-izuchat-node-js-ili-o-perspektivah-bekenda-na-javascript

22. Про node.js [Електронний ресурс] – Режим доступу до ресурса: URL: https://nodejs.org/ru/about/

23. Що таке Node.js[Електронний ресурс] – Режим доступу до ресурса: URL: https://goit.global/ua-ru/blog/node-js-chto-eto-kak-uchit/

24. АУТЕНТИФИКАЦИЯ API ПО JWT И PASSPORT [Електронний ресурс] – Режим доступу до ресурса: URL: https://www.8host.com/blog/autentifikaciya-api-po-jwt-i-passport/

25. Як використати JWT [Електронний ресурс] – Режим доступу до ресурса: URL: https://works.doklad.ru/view/-ZCAAokfZMo/all.html

26. API аутентифікації [Електронний ресурс] – Режим доступу до ресурса: URL: https://laravel.su

27. Авторизація в Node [Електронний ресурс] – Режим доступу до ресурса: URL: https://dou.ua/calendar/33266/

28. Сучасна авторизація [Електронний ресурс] – Режим доступу до ресурса: URL: itnan.ruCRM-системи [Електронний ресурс] – Режим доступу до ресурса: URL: https://works.doklad.ru/view/-ZCAAokfZMo/all.html

29. observable [Електронний ресурс] – Режим доступу до ресурса: URL: https://rxjs.dev/guide/observable

30. Using observables to pass values [Електронний ресурс] – Режим доступу до ресурса: URL: https://angular.io/guide/observables

31. Understanding, creating and subscribing to observables in Angular [Електронний ресурс] – Режим доступу до ресурса: URL: https://luukgruijs.medium.com/understanding-creating-and-subscribing-to-observables-in-angular-426dbf0b04a3

32. Метод subscribe [Електронний ресурс] – Режим доступу до ресурса: URL: https://ru.stackoverflow.com/questions/821911/Что-делает-метод-subscribe-в-angular5

33. International Book Series "Information Science and Computing [Електронний ресурс] – Режим доступу до ресурса: URL: http://www.foibg.com/ibs_isc/ibs-07/IBS-07-p25.pdf

34. Інформаційний портал CRM [Електронний ресурс] – Режим доступу до ресурса: URL: https://works.doklad.ru/view/-ZCAAokfZMo/all.html

35. Mongoose [Електронний ресурс] – Режим доступу до ресурса: URL: https://mongoosejs.com

36. CRM-системи [Електронний ресурс] – Режим доступу до ресурса: URL: https://works.doklad.ru/view/-ZCAAokfZMo/all.html

37. Express [Електронний ресурс] – Режим доступу до ресурса: URL: https://developer.mozilla.org/ru/docs/Learn/Server-side/Express_Nodejs/mongooseCRM-системи [Електронний ресурс] – Режим доступу до ресурса: URL: https://works.doklad.ru/view/-ZCAAokfZMo/all.html

38. Вступ до Mongoose для MongoDB та Node.js [Електронний ресурс] – Режим доступу до ресурса: URL: https://code.tutsplus.com/uk/articles/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527

39. Використання Mongoose [Електронний ресурс] – Режим доступу до ресурса: URL: https://works.doklad.ru/view/-ZCAAokfZMo/all.html

40. Mongoose [Електронний ресурс] – Режим доступу до ресурса: URL: https://my-js.org/docs/guide/mongoose/CRM-системи [Електронний ресурс] – Режим доступу до ресурса: URL: https://works.doklad.ru/view/-ZCAAokfZMo/all.html

41. Переваги Node.js [Електронний ресурс] – Режим доступу до ресурса: URL: https://artjoker.ua/ru/blog/v-chem-preimushchestva-nodejs/

42. Node.js or java[Електронний ресурс] – Режим доступу до ресурса: URL: https://medium.com/webbdev/js-9ca13173577b

43. Node.js або Java: продуктивність, ресурси, управління потоками, популярність і особистий досвід [Електронний ресурс] – Режим доступу до ресурса: URL: https://devzone.org.ua/post/nodejs-abo-java-produktivnist-resursi-upravlinnya-potokami-populyarnist-i-osobistiy-dosvid

44. Чому варто порівнювати node I java [Електронний ресурс] – Режим доступу до ресурса: URL: https://senior.ua/articles/nodejs-proti-java-chomu-yak--chi-varto-h-porvnyuvati

45. Js фреймворки[Електронний ресурс] – Режим доступу до ресурса: URL: https://www.internet-technologies.ru

46. 10 фреймворків JS [Електронний ресурс] – Режим доступу до ресурса: URL: https://tproger.ru/articles/10-javascript-frejmvorkov-kotorye-stoit-vyuchit-v-2021-godu/

47. Кращі фреймворки [Електронний ресурс] – Режим доступу до ресурса: URL: https://appmaster.io/ru/blog/luchshie-javascript-freimvorki-dlia-ispol-zovaniia-v-2022-godu-polnoe-rukovodstvo

48. Angular HTTP POST Example [Електронний ресурс] – Режим доступу до ресурса: URL: https://www.tektutorialshub.com/angular/angular-http-post-example/

49. Angular Http POST request example [Електронний ресурс] – Режим доступу до ресурса: URL: https://stackblitz.com/edit/angular-http-post-request-example?file=src%2Fapp%2Fapp.component.ts

50. $http [Електронний ресурс] – Режим доступу до ресурса: URL: https://docs.angularjs.org/api/ng/service/$http

ДОДАТКИ

Додаток А

Тези конференції

УДК 004.45

Р.С.Гром'як, к.ф.-м.н,, С.С.Серкіз, студент магістр, група САм-61, Тернопільський національний технічний університет імені Івана Пулюя, Україна

CRM-СИСТЕМА ЯК ІНСТРУМЕНТ УДОСКОНАЛЕННЯ ВЗАЄМОВІДНОСИН З КЛІЄНТАМИ

R.S.Gromyak, Ph.D., Doctor of Physical and Mathematical Sciences, S.S Serkiz, master's student, group SAm-61,

CRM SYSTEM AS A TOOL FOR IMPROVING CUSTOMER RELATIONS

За допомогою CRM-системи реалізується стратегія і тактика формування і розвитку клієнтської бази. Ефективне задоволення потреб клієнтів цільового ринку сприяє залученню значної кількості нових клієнтів, утриманню існуючих клієнтів і, як наслідок, формуванню постійної клієнтської бази і, як наслідок, зростанню клієнтури з прибутковістю ринкової діяльності компаній. Професійно впроваджені системи CRM пропонують багато переваг для продажів, маркетингу, підтримки тощо. І ось найважливіші з них: підвищення продуктивності, автоматизація, обробка даних клієнтів, ефективне планування та відстеження, інтеграція з іншими інформаційними продуктами, доступність з будь-якого місця, покращення відносин із клієнтами.

CRM-системи базуються на певних компонентах, які частково дозволяють поліпшити відносини з клієнтами, серед них:

1. Автоматизація маркетингу – система, яка автоматизує маркетингові операції, спрощує інформаційні процеси та забезпечує більш ефективне маркетингове планування та аналіз результатів. Функціональність МА включає:

- засоби аналізу та формування цільової аудиторії, створення списків потенційних клієнтів.

– засоби планування та проведення маркетингової кампанії, аналізу її результатів для кожної цільової групи, виду товару, регіону;

– Виявлення та аналіз вимог клієнтів

– Управління потенційними угодами;

2. Автоматизація обслуговування клієнтів – система автоматизації підтримки та обслуговування клієнтів, що включає базу даних договорів клієнтів; моніторинг замовлень; інструменти контролю обслуговування клієнтів; База знань типових проблем, пов'язаних з використанням товарів (послуг), засобів їх вирішення.

Функціонал системи включає: база даних контактів із замовником, можливість групової роботи із замовником; управління відносинами з потенційними клієнтами, інтерактивна підтримка клієнтів, надання послуг; моніторинг надходження заявок контроль процесів обробки запитів і заявок, надання відповідей на них, звітування про результати обслуговування; обслуговування клієнтів і бізнес-партнерів в режимі реального часу;

Література

Шарапа О.М. Управління взаємовідносинами з клієнтами – 2009. – № 7 (97). – С. 175-183.

CRM-система як інструмент удосконалення взаємовідносин з клієнтами http://www.economy-confer.com.ua/full-article/3102/

УДК 004.45

Р.С.Гром'як, к.ф.-м.н, С.С.Серкіз, студент магістр, група САм-61, Тернопільський національний технічний університет імені Івана Пулюя, Україна

Взаємодія бізнес процесів з клієнтами за допомогою CRM-СИСТЕМ

R.S.Gromyak, Ph.D., Doctor of Physical and Mathematical Sciences, S.S Serkiz, master's student, group SAm-61,

INTERACTION OF BUSINESS PROCESSES WITH CUSTOMERS USING CRM-SYSTEMS

Використання CRM-систем стало незамінним у веденні бізнесу. Основним завданням CRM є збір даних про покупців і постійний інформаційний зв'язок з ними. Щоб зрозуміти сенс і важливість впровадження CRM-системи, варто звернути увагу на те, які основні функції вона виконує, а саме: визначення плану продажів; отримання та відправка замовлення на продаж; Створення плану продажів для споживачів; конфігурація продукту; Аналіз обсягів продажів по групах клієнтів і товарних групах; Управління збутовими ресурсами ; Комунікація з клієнтами [1].

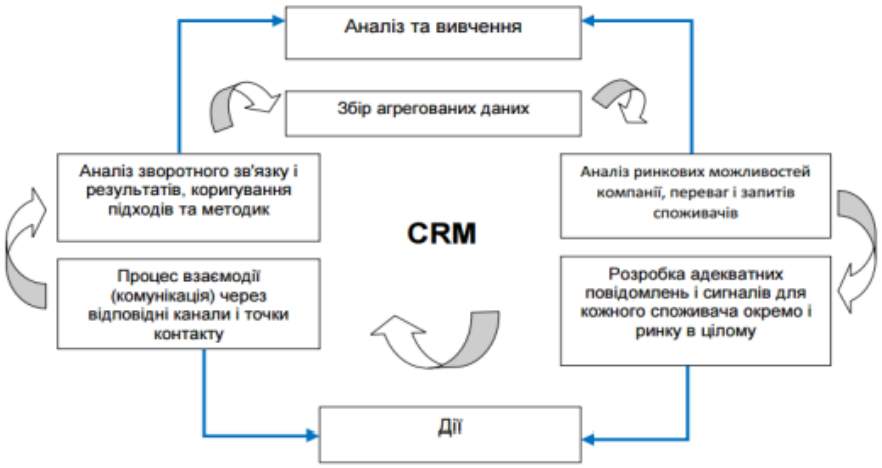

Рисунок 1 – Цикл інформаційних процесів в CRM

В CRM-системі зберігається інформація, яка може сприяти підвищенню ефективності кампанії вихідного обзвону. Це може бути детальна контактна інформація, дані про найбільш зручний час зв'язку з клієнтом.

Також в деяких CRM-системах можлива функція під'єднання декількох соціальних мереж, за допомогою цього спілкування з клієнтами стає швидшим, зручнішим та якіснішим [2].

Література

Застосування інформаційних технологій в управлінні підприємством http://ape.fmm.kpi.ua/article/view/102782

Найкращі CRM-системи для ведення бізнесу https://esputnik.com/uk/blog/oglyad-dvadcyatki-najkrashih-crm-sistem-dlya-biznesu

Додаток Б

```
const express = require('express')
const passport = require('passport')
const mongoose = require('mongoose')
const bodyParser = require('body-parser')
const app = express()
const authRoutes = require('./routes/auth')
const analyticsRoutes = require('./routes/analytics')
const categoryRoutes = require('./routes/category')
const orderRoutes = require('./routes/order')
const positionRoutes = require('./routes/position')
const keys = require('./config/keys')

mongoose.connect(keys.mongoURL)
  .then(() => console.log('MongoDB connected'))
  .catch(error => console.log(error))

app.use(passport.initialize())
require('./middleware/passport')(passport)

app.use(require('morgan')('dev'))
app.use('/uploads', express.static('uploads'))
app.use(require('cors')())
app.use(bodyParser.urlencoded({extended: true}))
app.use(bodyParser.json())

app.use('/api/auth' , authRoutes)
app.use('/api/analytics' , analyticsRoutes)
app.use('/api/category' , categoryRoutes)
app.use('/api/order' , orderRoutes)
app.use('/api/position' , positionRoutes)

module.exports = app
```