Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«CRM-система управління взаємодією з клієнтами книжкового Інтернет-магазину»

Студента 1 курсу, 3м групи, спеціальності 122 «Комп'ютерні науки» _____ *підпис студента* — Кулініча Анатолія Миколайовича

Науковий керівник кандидат фізико-математичних наук, доцент _____ *підпис керівника* — Самойленко Ганна Тимофіївна

Гарант освітньої програми доктор фізико-математичних наук, професор _____ *підпис керівника* — Пурський Олег Іванович

Київ 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій

Кафедра комп'ютерних наук та інформаційних систем

Спеціальність 122 «Комп'ютерні науки»

Затверджую

Зав. кафедри _____Пурський О.І.

«20» грудня 2020р.

Завдання
на випускний кваліфікаційний проект студенту

Кулініч Анатолію Миколайовичу

(прізвище, ім'я, по батькові)

1.Тема випускної кваліфікаційної роботи (проекту)

«CRM-система управління взаємодією з клієнтами книжкового Інтернет-магазину.»

Затверджена наказом ректора від *«02» грудня 2020 р. № 4110*

2. Строк здачі студентом закінченої роботи 2*6 листопада 2021 року*

3. Цільова установка та вихідні дані до роботи

Мета роботи: обґрунтування та розробка CRM-системи управління взаємодією з клієнтами книжкового Інтернет-магазину, з урахуванням сучасних тенденцій побудови організаційних та функціональних інформаційних структур підприємств.

Об'єкт дослідження: процес проектування CRM-системи управління взаємодією з клієнтами книжкового Інтернет-магазину

Предмет дослідження: засоби створення CRM-система управління взаємодією з клієнтами книжкового Інтернет-магазину

4. Перелік графічного матеріалу_____
_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Самойленко Г.Т.	5.12.2020р.	5.12.2020р.
2	Самойленко Г.Т.	5.12.2020р.	5.12.2020р.
3	Самойленко Г.Т.	5.12.2020р.	5.12.2020р.

6. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1

Теоретичні основи діяльності Web-орієнтованої інформаційної системи

1.1. Основні поняття та класифікація CRM-систем

1.2. Аналіз існуючих CRM-систем

1.3. Інтернет-магазин як різновид Web орієнтованої CRM-системи

Висновки до Розділу 1

РОЗДІЛ 2

Організація розробки Web орієнтованої CRM-системи

2.1 Аналіз особливостей роботи веб - додатку

2.2 Огляд фреймворків для розробки системи

2.3 Вибір фреймворку для розробки серверної частини

2.4 Вибір фреймворку для розробки клієнтської частини

Висновки до Розділу 2

РОЗДІЛ 3.

Програмна реалізація CRM-системи управління взаємодією з клієнтами книжкового Інтернет-магазину

3.1. Проектування логічної та фізичної структури сайту

3.2 Проектування API

3.3 Проектування структури інтерфейсу

3.4 Проектування бази даних

3.5 Розробка модуля авторизації

3.6 Розробка адміністративної частини

Висновки до розділу 3

ВИСНОВКИ ТА РЕЗУЛЬТАТИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*01.11.2020*	*01.11.2020*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*05.12.2020*	*05.12.2020*
3	*Вступ*	01.06.2020	
4	*РОЗДІЛ 1. Теоретичні основи діяльності Web-орієнтованої інформаційної системи*	25.06.2020	
5	*РОЗДІЛ 2. Організація розробки Web орієнтованої CRM-систем*	02.09.2020	
6	*РОЗДІЛ 3. Програмна реалізація CRM-система управління взаємодією з клієнтами книжкового Інтернет-магазину*	09.09.2020	
7	*Висновки*	02.11.2020	
8	*Підготовка статті у збірник наукових статей магістрів*	*09.09.2021*	
9	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	18.06.2021	
10	*Попередній захист випускної кваліфікаційної роботи*	01.12.2021	
11	*Виправлення зауважень, зовнішнє*	22.11.2021	

	рецензування випускної кваліфікаційної роботи		
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*26.11.2021*	
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «5» грудня 2020 р.

Керівник випускної кваліфікаційної роботи

Самойленко Г.Т.
(прізвище, ініціали, підпис)

Гарант освітньої програми Пурський О.І.
(прізвище, ініціали, підпис)

Завдання прийняв студент-дипломник Кулініч А.М.
(прізвище, ініціали, підпис)

9. Відгук керівника випускної кваліфікаційної роботи

_____

_____

_____

_____

_____

_____

Керівник випускного кваліфікаційного проекту

_____ __._____ 20__ р.

(підпис, дата)

13. Висновок про випускний кваліфікаційний проект

Випускний кваліфікаційний проект студента _____

(прізвище, ініціали)

може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми_____ Пурський О.І.

(підпис, прізвище, ініціали)

Завідувач кафедри _____ Пурський О.І.

(підпис, прізвище, ініціали)

«_____»_____2021 р.

Анотація

У випускному кваліфікаційному проекті розглянуто основні поняття CRM-системи, їх різновиди та функції. Проаналізовано доцільність розробки, розглянуто існуючі аналоги та виявлено їх недоліки. Проведено аналіз засобів розробки, спроектовано структуру та моделі системи, розроблено програмні модулі системи.

При виконанні другого розділу дипломної роботи було здійснено проектування CRM-системи, яке складалося з таких етапів: постановка завдання, вибору засобів розробки та СУБД та розробки ER-моделі бази даних.

В третьому розділі представлено розробку CRM-системи, яка в себе включає: створення БД, таблиць та зв'язків між ними, представлень та збережених процедур, а також додатку для управління БД у якому будуть реалізовані функції CRM-системи. У даному розділі також розглянуто інтерфейс додатку, основні функції та алгоритм дій користувача, а також процес та етапи впровадження CRM-системи в інтернет-магазині.

Ключові слова: crm-система, електронна комерція, автоматизація, бізнес-процеси, web-технології.

Abstract

The final qualification project considers the basic concepts of CRM-system, their varieties and functions. The expediency of development is analyzed, the existing analogues are considered and their shortcomings are revealed. The analysis of development means is carried out, the structure and models of system are designed, software modules of system are developed.

During the second section of the thesis was designed CRM-system, which consisted of the following stages: setting the task, the choice of development tools and DBMS and the development of ER-database model.

The third section presents the development of a CRM-system, which includes: the creation of databases, tables and relationships between them, views and stored procedures, as well as an application for database management in which the functions of the CRM-system will be implemented. This section also discusses the interface of the

application, the main functions and algorithm of user actions, as well as the process and stages of implementation of the CRM-system in the online store.

Keywords: crm-system, e-commerce, automation, business processes, web-technologies.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

WWW (англ. World Wide Web) - це розподілена мережева гіпертекстова інформаційна система.

БД – база даних.

СУБД – система управління базами даних.

SQL (англ. Structured Query Language) - мова структурованих запитів.

CSS – (англ. Cascading Style Sheets), каскадні таблиці стилів.

HTML – (англ. HyperText Markup Language), мова розмітки гіпертексту.

API – (англ. Application Programming Interface), набір чітко визначених методів для взаємодії різних компо

ВСТУП

З розвитком мережі Інтернет широкого використання набувають онлайн сервіси, основну частину з яких складають системи управління для різного роду бізнесу та комерції. Вже складно уявити собі складський або бухгалтерський облік без застосування спеціалізованого програмного забезпечення, торгові представники використовують спеціальні програми для оформлення і відправлення замовлення в офіс прямо з планшета або мобільного телефону, достатньо велика частина замовлень приходить з сайту вже у вигляді готових до обробки документів. Але при цьому взаємини з клієнтами, принаймні, в середньому і малому бізнесі, чомусь дуже часто ведуться без впровадження автоматизації і достатньої уваги до обліку.

Основною стратегією успішного існування та подальшого розвитку сучасних книжкових магазинів, нарівні з компаніями в інших сферах діяльності, поступово стає ефективне управління взаємодією з клієнтами. Орієнтація на удосконалення відносин з клієнтами обумовлена низкою тенденцій, зокрема посиленням конкуренції і відповідним підвищенням вимог до якості обслуговування [1]. Знання своїх клієнтів, задоволення запитів і потреб кожного з них дозволить в значній мірі підвищити загальну ефективність діяльності інтернет-магазину.

Одним з ефективних засобів вирішення описаних проблем є впровадження системи управління взаємодією з клієнтами, або як її часто називають CRM - системи. CRM - це прикладне програмне забезпечення, призначене для автоматизації стратегій взаємодії з замовниками (клієнтами), зокрема, для підвищення рівня продажів і якості обслуговування шляхом збереження інформації про клієнтів та історії взаємин з ними [2]. Навіть якщо компанія веде історію дзвінків і контактів на папері або в Excel - це можна вважати CRM - системою в тому випадку, якщо розроблена схема обліку працює і дозволяє контролювати всі варіанти взаємодії з клієнтами [3].

Актуальність теми. Готові CRM-системи непогано підходять для малого бізнесу завдяки своїй доступності та простоті в обслуговуванні. Однак вони не припускають можливостей гнучкого налаштування під конкретні цілі бізнесу, у

зв’язку з чим все одно змушують співробітників використовувати xlsx- таблиці, органайзери та інше додаткове ПЗ. Розробка індивідуальної CRM-системи виключає вищеописані проблеми, але коштує дорожче і вимагає деякий час на розробку та оптимізацію. У той же час, маючи власну CRM-систему, замовник отримує довгостроковий інструмент, який, при необхідності, завжди можна модифікувати під нові завдання і вимоги.

Мета роботи – дослідити та провести аналіз існуючих CRM-систем та на їх основі враховуючи усі переваги та недоліки розробити власну CRM-систему для книжкового інтернет-магазину.

Об’єкт дослідження – CRM-система книжкового інтернет-магазину.

Предмет дослідження – сукупність теоретичних та практичних засад реалізації CRM-системи для електронної комерції. Результатом дипломної роботи магістра повинна бути повністю функціонуюча CRM-система для електронної комерції за допомогою якої працівники магазину зможуть вести облік клієнтів для подальшої взаємодії з ними, здійснювати продаж товарів, комунікацію з клієнтами за допомогою e-mail розсилки, вести облік товарів на складі, керувати поставками використовуючи зручний додаток, що працює з єдиною БД та дозволяє отримати доступ до необхідної інформації в будь-який момент часу.

Для досягнення поставленої мети були визначені наступні задачі:

- провести аналіз предметної області;
- оцінити можливості існуючих рішень;
- проаналізувати сучасні методи веб-розробки;
- спроектувати логічну та фізичну структури системи;
- розробити структуру інтерфейсу;
- спроектувати базу даних;
- розробити програмні модулі, що реалізовують усі необхідні в системі функції;

Для проектування і реалізації цього проекту були розглянуті різні технології: SPA, фреймворки для написання серверної та клієнтської частин веб-додатку

(Django, Laravel, Vue.js) тощо. Також були отримані деякі знання відносно підходів по розробці баз даних.

Наукова новизна одержаних результатів полягає в використанні найактуальніших методів розробки та отримання інструменту, який може працювати з найсучаснішими інтернет-магазинами.

Практичне значення. Матеріали дипломного проекту можуть бути використані для управління взаємовідносинами з клієнтами книжкового інтернет-магазину.

Публікації. Результати дослідження опубліковано у збірнику наукових статей студентів, які здобувають освітній ступінь магістра за спеціалізацією «Комп'ютерні науки» КНТЕУ на тему: «CRM-система управління взаємодією з клієнтами книжкового Інтернет-магазину», 2021 р.

У відповідності до поставлених мети та конкретних завдань дослідження, визначено структуру роботи. Вона складається зі вступу, трьох розділів, висновків та списку використаної літератури.

РОЗДІЛ 1.

Теоретичні основи діяльності Web-орієнтованої інформаційної системи

1.1. Основні поняття та класифікація CRM-систем

Сучасна ринкова економіка являє собою складну динамічно розвиваючу систему. Одним із ключових понять на ринку є поняття конкуренції. Для того щоб компанія могла виграти конкурентну боротьбу, вона повинна залучати нових клієнтів, утримувати вже існуючих, виконувати свою роботу якісно і в короткі терміни.

Одним з найбільш раціональних рішень цього завдання є впровадження CRM-системи. В даний час число лояльних клієнтів у організації є одним з головних чинників успішності, стійкості і процвітання бізнесу. Завдяки новим технологіям функціонують ще недавно малі підприємства, розвиваються стрімкими темпами і стають одними з головних конкурентами великих компаній.

На сьогоднішній день термін CRM став досить популярним в Україні. Як правило під CRM розуміють клієнто-орієнтовану стратегію, створену на основі використання передових управлінських і інформаційних технологій, з метою формування взаємовигідних відносин з клієнтами. У свою чергу CRM визначають як технологію, спрямовану на завоювання, задоволення і збереження платоспроможних клієнтів. [21]

Основним завданням CRM-системи є підвищення ефективності бізнеспроцесів з пошуку і утриманню покупців, в ході продажів. Управління взаємовідносинами з клієнтами дозволяє залучити нових, а також перетворити нейтральних в лояльних [22].

Історія виникнення CRM-систем почалася ще в 80-і роки минулого століття, вже тоді компанії виділяли пріоритетну групу клієнтів, яким приділялася особлива увага. Керівництво комерційних структур ставили перед компанією питання: «Чому люди вибирають саме їх продукт або послугу?».

Системи управління взаємовідносинами з клієнтами стали широко відомі на початку 90-х на заході. Стрімкий розвиток інформаційних технологій і обчислювальної техніки надавало можливість працювати з серйозними базами

даних. Саме в кінці 20 століття стало можливим обробляти і аналізувати великі обсяги інформації для прийняття рішень. Таким чином, з'явилися перші розробники CRM-систем.

На відміну від заходу, який пройшов довгий шлях становлення CRM - систем, що нараховує більше тридцяти років, вітчизняний ринок почав активно їх використовувати трохи більше десяти років тому. Причина відставання вітчизняних компаній спричинена багато в чому через кризу 90-х років. У той період попит на товари значно перевищував пропозицію, потенційних клієнтів було багато і основним завданням компаній було розширення, створення нових продуктів і послуг [23].

За допомогою CRM-систем підприємство отримує повну інформацію про своїх клієнтів і про те чого вони хочуть, дозволяє докладно вивчити свою цільову аудиторію. Далі ці дані керівництво компанії використовує для вибору стратегії розвитку підприємства.

Для того, щоб всі дані про клієнтів і їх покупки зберігалися в одному місці, підприємство розробляє власну CRM-систему або використовує уже готову, що пропонують сторонні розробники програмного забезпечення.

На основі підходів ведення бізнесу, масштабів організації і видів діяльності компанії CRM системи різні за своєю структурою і функціоналом.

Системи управління взаємовідносинами з клієнтами можна класифікувати за рівнем обробки інформації, функціональної спрямованості, технології функціонування та вартості рішення.

За рівнем обробки CRM поділяють на:

- операційні, які в автоматичному режимі фіксують дані про контакти з клієнтами, впорядковують їх в базах і дають можливість отримувати до них доступ;
- аналітичні, які дозволяють створювати звіти і аналізувати інформацію в базах даних;

• колаборaційні, що включають в себе можливості операційних і аналітичних CRM систем, а також надають можливість встановлювати більш щільну зв’язку з клієнтською базою, аж до їх впливу на прийняття рішень.

В даному випадку автоматизація контакту з клієнтами здійснюється по різних каналах зв’язку (телефон, особистий контакт, електронні канали).

За функціональною спрямованістю CRM поділяють на системи управління контактами (call центр, зворотний зв’язок), маркетингом (опитування, аналіз результатів), продажами (Інтернет-магазин, замовлення по телефону), комплексні.

Залежно від технології функціонування CRM-системи поділяють на:

• SaaS (Software as a service «програмне забезпечення як послуга»), де доступ до системи здійснюється як до хмарного сервісу.

• Stand-Alone, що здійснюються на основі програмних комплексів з ліцензією на установку і використання.

За вартістю системи поділяють на бюджетні версії, системи середнього рівня і дорогі системи. Варто відзначити, що ціна на продукт залежить від виконуваних функцій системи.

При впровадженні і експлуатації компанія ставить перед CRM-системою наступні завдання:

• дослідження ринку;

• обґрунтування необхідності створення нових товарів і послуг, а також виведення їх на ринок;

• формування ефективних методів просування товарів і послуг;

• моніторинг та контроль показників роботи персоналу;

• навчання і підвищення кваліфікації та рівня професіоналізму співробітників;

• збір інформації про клієнтів, партнерів, а також їх систематизація в єдиній базі [24].

Цілі CRM-системи:

• Скорочення чисельності управлінського персоналу.

• Підвищення рівня продаж.

- Оптимізація роботи з клієнтами.
- Збереження історії взаємовідносин працівників компанії з клієнтами. 13
- Оптимізація внутрішньої роботи підприємства.
- Контроль та оцінка ефективності кожного працівника підприємства.

CRM-системи можуть бути як загальними, які оптимізують всі основні бізнес-процеси в одній системі, зберігають інформацію в одній базі даних, так і спеціальними, які використовують для оптимізації конкретного бізнес-процесу, наприклад, тільки для складського обліку, бухгалтерії, кредитного відділу і так далі[25].

Такі системи наводять «порядок» у внутрішній роботі конкретного відділу.

Переваги CRM:

1. Робота з вже наявними клієнтами, або з тими, хто колись цікавився компанією. Витрати на залучення нового клієнта набагато більші, ніж на утримання вже наявного. Система нагадує працівникові, коли йому потрібно зв'язатися з клієнтом, коли розповісти про нові вигідні пропозиції, інформує автоматично SMS і E-mail розсилкою.
2. Ефект сарафанного радіо. Клієнт, потреби якого були задоволені, який залишився задоволений сервісом, розповість про компанію свому оточеню. В середньому, за статистикою, про вдалу покупку розповідають п'ятьом своїм знайомим. Якщо ж клієнт залишився незадоволеним, то десятьом.[25]
3. Підвищення прибутку. При утриманні клієнтів, які цікавилися продукцією компанії, прибуток компанії збільшується на 50-70%.
4. Підвищення конверсії. Після впровадження CRM-системи на підприємстві значно підвищується конверсія, отже витрати на рекламу і просування скорочуються і можуть використовуватися для удосконалення самого продукту, поліпшення якості.

1.2. Аналіз існуючих CRM-систем

Аналізучи світовий рейтинг CRM-систем було виділено такі системи: Salesforce.com, Infor, Oracle Cloud, Bpm' online, Microsoft Dynamics 365[23].

Salesforce CRM – система компанії Salesforce основу якої складають хмарні (онлайн) технології. Вона розроблена для управління аналітикою, бізнес- 14 процесами, взаємовідносинами з клієнтами, а також продажами і маркетингом. Salesforce CRM працює, ґрунтуючись на моделі SaaS. Основними особливостями системи є:

- управління контактами;
- генерація лідів;
- управління можливостями;
- прогнозування продажів;
- автоматизація документообігу;
- інструментарій для роботи в групі.

Salesforce надає найширшу систему підтримки клієнтів. На веб-сайті компанії активні функції живого чату, є безліч довідкових ресурсів в режимі реального часу, також можна зв'язатися з торговим представником Salesforce і відділом продажів[26].

Система **Infor CRM** (SalesLogix) використовують більш ніж в 10 тисячах організацій по всьому світу. Вона займає одне з лідируючих місць в СНД за кількістю інсталяцій завдяки таким особливостям:

- забезпечує досить швидке повернення інвестицій (ROI);
- легкість використання, так як не вимагає кваліфікованих IT-ресурсів;
- має модульну масштабовану побудову («росте» разом з організацією);
- легкість налаштувань (в базовій конфігурації – відразу після придбання, а під конкретні вимоги замовника – протягом 10-20 тижнів);

• перевершує аналогічні продукти по співвідношенню ціни та якості [27].

Система Infor CRM є гнучкою і адаптивною завдяки модульній структурі, яка включає в себе наступні модулі:

• модуль продажів;

• модуль маркетингу;

• модуль сервісного супроводу;

• модуль електронної комерції.

Залежно від вимог організації кожен з модулів системи може використовуватися автономно, що є великою перевагою для даної системи. Отже, представляється можливим використовувати тільки ті компоненти, які необхідні споживачеві в даний момент, потім можна розширювати систему в залежності від пропонованих вимог [28].

Компанія Oracle, яка є лідером серед розробників програмних рішень, представляє власну модульну систему – **Oracle CRM**. Система включає в себе наступні модульні підгрупи:

• Oracle CRM on Demand.

• Oracle Sales Cloud.

• Oracle Siebel CRM.

Програмний комплекс використовує принцип CRM on Demand, який дозволяє користувачеві вибрати свій набір модулів, що підходить для вирішення конкретних завдань обраної організації/бізнесу. Система надає для скачування наступні рішення, розділені по галузях:

• природні науки (фармацевтичний напрямок і медицина);

• Hi-Tech і IT-сектор CRM on Demand;

• система роботи зі страхуванням;

- управління приватними інвестиціями в рамках біржі або банку (Financial Services Edition);
- автомобільне виробництво.

За допомогою Oracle Sales Cloud користувач може отримати інформацію про клієнта з сформованого банку даних в будь-який час в будь-якому місці, ця можливість дозволяє поліпшити і підвищити рівень продажів.

Модуль Oracle Siebel CRM дозволяє розгорнути власний сервер для обслуговування додатків і банку даних компанії [29].

BPM online CRM – SaaS-рішення, розроблене компанією Terrasoft, що дозволяє об'єднати функціональні можливості системи управління взаємовідносинами з клієнтами (CRM) та системи управління бізнес-процесами (BPM). BPM Online дозволяє:

- вести клієнтську базу і повну історію взаємин з клієнтами;
- контролювати ефективність роботи співробітників;
- виробляти чітке планування і керувати продажами;
- автоматизувати бізнес-процеси компанії;
- автоматизувати документообіг [30].

Використання системи **Microsoft Dynamics CRM** дозволяє поліпшити і підвищити ефективність взаємодії з клієнтами.

Microsoft Dynamics CRM – підсистема бізнес-додатку Microsoft Dynamics, яка створена корпорацією Microsoft для вирішення завдань комплексної автоматизації діяльності організацій.

Основні функції Microsoft Dynamics CRM:

1. Робота з контактами організацій.
2. Робота з діями і примітками.
3. Робота з інтересами і можливими угодами.
4. Використання маркетингових списків.

5. Робота з діями і відгуками від компаній.

6. Відстеження запитів на обслуговування.

За допомогою використання даної системи можна відстежити такі дії: дзвінки, завдання, факси, листи, повідомлення електронної пошти, зустрічі, сервісу і відгуки від компаній. Надається можливість створювати власні типові завдання і рішення, аналізувати потреби клієнта, грамотний аналіз яких дає можливість отримання угод [31].

Якщо розглядати ринок CRM-систем в Україні, то згідно опитування, яке було проведено компанією Бітрікс24 основними гравцями на ринку є: Бітрікс24, AmoCRM, та Мегаплан [32].

Бітрікс24 – це не тільки CRM-система. CRM є одним з модулів. Бітрікс24 включає в себе корпоративний інтернет з внутрішньої соціальною мережею, планувальник, платформу для відеоконференцій тощо. Він має широкий функціонал, який є як плюсом, так і мінусом. Деякі з численних функцій можуть навіть не використовуватися, але за них доводиться платити. Так як система є досить великою, потрібен час, щоб в ній розібратися [33]. Розділ технічної підтримки має 22 мануали по навчанню роботі з системою, керівництво з графічними зображеннями, відеоматеріалами, які детально описують кожен крок. Інтерфейс Бітрікс24 кожен користувач може налаштувати під себе. Система універсальна, підходить як великим, так і невеликим підприємствам. Система буде корисна таким категоріям бізнесу, як:

- приватні підприємці та малому бізнесу, яким вистачить і безкоштовної версії з мінімальним набором функцій;
- великому бізнесу – мережевим підприємствам, холдингам.

У хмарній та локальній версіях Бітрікс24 майже всі функції вимагають налаштувань. Перевагою Бітрікс24 є те, що безкоштовною версією можуть користуватися до 12 співробітників. У інших CRM, як правило, підприємство здійснює оплату за кожного користувача. Мегаплан – це система, за допомогою якої можна керувати бізнеспроцесами, контролювати співробітників, отримувати звіти про виконану роботу. Даний сервіс має можливість збереження всіх записів, рахунків, дзвінків і спілкування з клієнтами. Мегаплан є найпотужнішою з простих CRM-систем і відмінно підійде для малого і середнього бізнесу. У даної системи є двотижневий пробний період, немає обмежень по числу співробітників ні на одному тарифі, простий і зрозумілий інтерфейс, можливість інтеграції з 1С, сайтом і соцмережами. Для комунікації з покупцями в функціонал вбудована пошта. Працюючи в Мегаплані, неможливо заплутатися, так як є відео-інструкція по всій системі і спливаючі навчальні вікна. У функціоналі системи немає нічого зайвого, є можливість установки розширень. Є мобільний додаток для Android і iOS [34]. **AmoCRM** – це система, яка відмінно підійде для аналізу продажів, доступна в онлайн-режимі, з простим інтерфейсом і мобільним додатком. Особливо підійде тим, хто продає товари і послуги з довгим терміном угоди. Також ця система актуальна для забудовників, так як в AmoCRM є зручний інтерактивний каталог по всіх об’єктах (Profit Base). За допомогою цієї системи можна контролювати абсолютно весь процес продажу, а також створювати багаторівневі воронки продаж під конкретний бізнес. AmoCRM вміє 18 візуалізовувати інформацію – в розділі аналітики багато графіків, діаграм. Налаштування прості, все в системі взаємопов’язане. У AmoCRM є можливість планувати справи по угоді, писати важливі коментарі, прикріплювати документи по клієнту. Головна перевага системи полягає в інтеграції з усіма джерелами лідів і рекламних каналів. Наприклад, відбувається звернення поштою, через форму зворотного зв’язку, діректа – лід відразу потрапляє в CRM-систему. У AmoCRM є примітивна аналітика з продажу та 4 види звіту. AmoCRM не підійде підприємствам, у яких складні бізнес-процеси і нестандартна воронка продажів. А також тим, кому потрібна інтеграція з 1С і повний документообіг [35].

1.3. Інтернет-магазин як різновид Web орієнтованої CRM-системи.

Процес управління взаємодії з клієнтами можна розглядати як цикл процесів з управління маркетингом, продажами і обслуговуванням. Етапи даного циклу взаємопов'язані і взаємозалежні. Керуючи маркетингом, компанія тим самим визначає суб'єкт діяльності в процесі по управлінню продажами, а управління обслуговуванням клієнта дозволяє підвищити лояльність споживача і закріпити позитивний імідж компанії.

Під управлінням маркетингом розуміють аналіз, планування, проведення різних маркетингових заходів, встановлення і підтримання контактів з цільовими покупцями і контроль за результатами, що дозволить отримати додатковий прибуток, збільшить зростання обсягів збуту і частку ринку. Процес управління маркетингом включає в себе етапи представлені на рисунку 1.2.

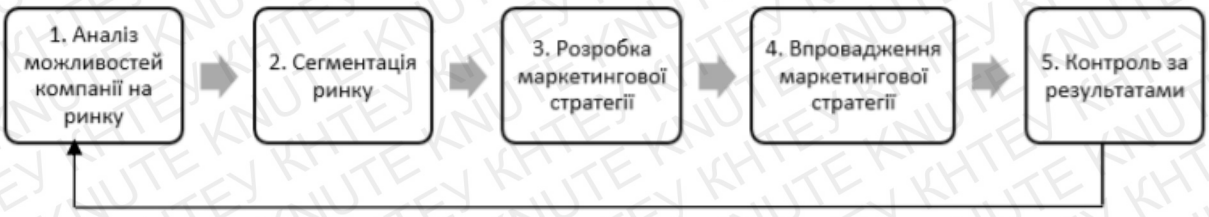

Рисунок 1.2 – Етапи процесу управління маркетингом

На етапі аналізу ринку відбувається здійснення маркетингових досліджень, вивчення ринку споживачів, аналіз зовнішнього середовища компанії. Далі проводиться сегментація ринку, аналіз існуючого попиту та аналіз діяльності конкурентів, розробка і здійснення маркетингової стратегії – плану маркетингових заходів, з допомогою яких компанія розраховує досягти поставлених цілей. Заключним етапом є контроль за результатами.

Метою управління маркетингом є забезпечення максимального збуту продукції, підвищення попиту і, як наслідок, максимізація прибутку. Метою

процесу управління продажами в широкому сенсі є оптимізація організації процесу продажу товарів або послуг, а у вузькому сенсі – здійснення послідовності дій, що призводять до укладення угоди.

Процес прийняття рішення про покупку не завжди зумовлений впорядкованими процесами і діями щодо прийняття рішень щодо доцільності покупки товару або послуги. Кінцеве рішення про придбання будь-якого товару або послуги, як правило, на момент початку побудови взаємин продавцем у споживача ще не сформовано. На результат цих взаємин впливає безліч факторів, як зовнішніх, так і внутрішніх, тому при грамотній побудові контакту з клієнтом можливо не просто схилити його до здійснення покупки, але і переконати в разі якщо спочатку він був налаштований до бренду або продукції бренду негативно.

Однією з основних проблем для інтернет-магазину, є проблема відсутності автоматизованої системи реєстрації та супроводу клієнтів. В результаті відбувається підвищення відтоку клієнтів і як правило зниження їх прибутковості. Вузьким місцем в даному процесі є складність в пошуках інформації про потреби клієнтів [36].

В умовах складних продаж на високо конкурентних ринках застосування автоматизованих систем може забезпечити організації підвищення конкурентоспроможності на ринку, а також стійкого фінансово-економічного стану [37].

Основними етапами управління продажами є:

1. Стадія поінформованості, або першого вступу в контакт.

На цьому етапі відбувається встановлення першого контакту між продавцем і покупцем, і від того, наскільки успішно і ефективно продавець налагодить контакт залежить подальша побудова відносин з клієнтом. На цьому етапі велику роль для продавця відіграє інформація про те, чи знайомий клієнт з брендом, чи має він уявлення про те, що виробляє або продає компанія, зустрічав він рекламу інтернет- магазину, якщо так, то наскільки ефективна вона була.

Для побудови успішного проходження стадії поінформованості велике значення мають рекламні та маркетингові акції, що проводяться компанією.

Причому на даний момент з’являється все більше і більше нових інструментів, що дозволяють підвищувати лояльність споживачів до бренду і залучати нових клієнтів, таких як контент-маркетинг, event-маркетинг, BTL-реклама. Традиційні способи реклами, звичайно ж, не втрачають своєї актуальності, а починають ще більш ефективно впливати на споживача в зв’язці з новими інструментами просування товару, що з’явилися в епоху цифрових технологій.

2. Стадія ознайомлення, або виявлення потреб.

На стадії ознайомлення відбувається безпосередній контакт між клієнтом і продавцем. На цій стадії клієнт вже має уявлення про те, чим займається компані, у нього сформовано ставлення до бренду. Продавцю ж немає необхідності знайомити покупця з галуззю, в якій працює компанія і її основними конкурентними перевагами – можна відразу переходити до етапу конкретизації запиту клієнта.

Ключова ідея даного етапу процесу взаємовідношення з клієнтом для продавця – визначити потреби, співставити запити споживача з пропонованими компанією продуктами або послугами. На цьому етапі починається взаємодія продавця з клієнтом, тому від того, наскільки професійно буде працювати продавець залежить успіх наступних етапів, і, як наслідок, результат угоди.

3. Стадія розширення, або презентації.

На даній стадії продавець вже має розуміння про потреби клієнта, клієнт ж усвідомлює, що з представлених раніше продавцем товарів може його зацікавити. Головним завданням продавця на даному етапі є презентація конкретного продукту або послуги клієнту з раніше представленого асортименту та опис їх переваг. На даній стадії продавцем можуть використовуватися різні способи і методи переконання, техніки продажів.

Основне правило переконання – викликати у клієнта емоції, вплинути як на раціональну, так і на емоційну складову його свідомості, тоді покупка буде здійснена і клієнт залишиться задоволений. В процесі продажі важливо сформувати певний акцент на ключовій характеристиці товару або ефекті від подальшого його споживання, однак найбільш дієвим методом переконання є

акцент на емоції, які отримає клієнт у процесі користування продуктом, так як рішення про покупку велика частина людей сприймає на емоційному рівні.

4. Стадія роботи з запереченнями.

Заперечення – це невід'ємна і обов'язкова частина продажі і процесу переконання в цілому. Заперечення покупця можуть бути як свідомими, так і несвідомими. Свідомі заперечення обумовлені об'єктивними причинами – високою вартістю товару чи послуг, недостатнім рівнем наданої інформації. Тоді як несвідомі заперечення є природною реакцією покупця на втручання продавця, найчастіше вони обумовлені агресивною манерою спілкування і недостатнім рівнем професіоналізму продавця.

Процес роботи з запереченнями, в свою чергу, можна розділити на декілька етапів. Перший з них – формулювання заперечення, визначення конкретних параметрів товару або послуги, які не влаштовують покупця. Далі настає сам етап обробки заперечення – у відповідь на коментарі покупця продавець повинен надати аргументовані відповіді, які зможуть нівелювати виявлені клієнтом негативні параметри товару і знову завоювати його увагу.

5. Стадія завершення угоди.

На даній стадії відбувається безпосереднєзавершення угоди, а саме прийняття клієнтом рішення про покупку. Однак здійснення покупки не є закінченням процесу взаємодії продавця і покупця. Тут важливу роль відіграє таке поняття як «Післяпродажний сервіс», оскільки після закінчення покупки клієнт повинен розуміти, що в будь-який момент він може звернутися до представників компанії з питаннями та коментарями, і його думка буде почута. Правильне спілкування з покупцем після завершення угоди надзвичайно потужний засіб для мотивації подальших покупок і формування лояльності.

У процесі управління обслуговуванням можна виділити дві основні складові: технічний фактор (виконання конкретної роботи по обслуговуванню) і людський фактор (рівень спілкування, ввічливість, доброзичливість, професіоналізм). Глобально, процес управління обслуговуванням спрямований на підвищення лояльності існуючих клієнтів і залучення нових. Поняття управління

обслуговуванням включає такі процеси як управління роботою з клієнтами, управління роботою з замовленнями, управління базами даних, післяпродажне обслуговування, робота з відгуками і претензіями.

Для управління процесом обслуговування клієнтів необхідна детальна інформація про кожного клієнта, його характеристики, історія всіх угод, оскільки в основі управління обслуговуванням клієнтів лежить процес накопичення та аналізу інформації про них.

Сьогодні компанії прагнуть використовувати в управлінні процесом взаємовідносин з клієнтами єдиний клієнт-орієнтований підхід.

Основні фактори, що впливають на процес управління відносинами з клієнтами: 1) Інформація.

Компанія повинна постійно вивчати, накопичувати і систематизувати інформацію про своїх клієнтів. Це є основою системи управління взаємовідносинами з клієнтами, усі подальші процеси взаємодії будуються на основі інформації, що була отримана та проаналізована раніше. Розуміючи, до якого сегменту відносяться клієнти компанії, які їхні основні потреби і очікування, чому вони вибирають певний бренд або певний продукт набагато простіше формувати бізнес модель компанії, розробляти асортиментну і маркетингову політики і реалізовувати продукцію на певному ринку [38].

Якщо раніше інформація про клієнтів накопичувалася на паперових носіях або взагалі не документувалася і містилася тільки в голові у продавця, то в епоху цифрових технологій з’явилися рішення, що дозволяють оптимізувати роботу з отриманою інформацією і будувати новий підхід до роботи з інформацією про клієнтів.

Сьогодні широке застосування отримали CRM-системи (див. рисунок 1.2), що дозволяють накопичувати і зберігати інформацію про угоди і клієнтів в одному місці. CRM дозволяє підтримувати зв’язок з клієнтами, що особливо актуально для великих фірм, у яких є великий список клієнтів [39].

Рисунок 1.2 – Застосування CRM-системи

Завдяки CRM-системам можна не просто визначити статус тієї чи іншої угоди або порахувати ефективність роботи певного каналу продажів, але і визначити тенденції розвитку ринку, проаналізувати динаміку продажів за довгостроковий період і виявити можливі причини зниження попиту.

2) Індивідуальний підхід.

Одним з результатів переходу від продукт-орієнтованого до клієнторієнтованого маркетингу є поява поняття «персоніфікований маркетинг». Такий маркетинг орієнтований в першу чергу на підвищення лояльності шляхом приділення уваги потребам і цілям кожного конкретного клієнта. CRM - система Клієнтська база Управління продажами Аналіз продаж Інформація про компанії

E-mail і SMS розсилки Сервіси розсилок Розмежування прав користувачів Історія роботи з клієнтами Анліз роботи працівників

До основних переваг, які дає компанії стратегія персоніфікованого маркетингу можна віднести:

- підвищення лояльності існуючих клієнтів компанії;
- отримання конкурентної переваги;
- збільшення попиту на продукцію і зростання продажів;
- можливість більш точної сегментації клієнтів, підвищення ефективності спрямованості маркетингових заходів.

3) Рівень професійної підготовки та кваліфікації персоналу.

Одним з найважливіших ресурсів компанії є її співробітники, і від рівня їх професійної підготовки залежить ефективність впровадження нових методів і технологій в бізнес-процеси організації. Від того, наскільки співробітники мотивовані і забезпечені необхідними теоретичними знаннями і практичними вміннями залежить ефективність їх роботи і, як наслідок, ефективність роботи організації. В процесі взаємовідносин з клієнтами ключову роль в кінцевому рахунку відіграють не методи, способи і засоби взаємодії з клієнтом, а особисті навички і професіоналізм представників компанії. Тому крім вкладень в засоби по зберіганню і обробці інформації важливо інвестувати і в навики людей, які використовують дану інформацію в своїй роботі, навчити їх використовувати практичні навички і отримані знання у роботі з клієнтами.

Перераховані вище фактори ефективно впливають на процес взаємовідносин з клієнтами в тому випадку, якщо їх вплив буде комплексний та послідовний. Дійсно, якщо в компанії є програмне забезпечення, що дозволяє

накопичувати і систематизувати інформацію про клієнтів, набагато простіше здійснювати стратегію індивідуального підходу.

Автоматизоване програмне CRM-забезпечення може стати рішенням для реалізації стратегії персоніфікованого маркетингу, що дозволить більш ефективно управляти бізнес-процесами, що протікають в рамках взаємодії співробітників підприємства з клієнтами [40].

Таким чином, наявність систематизованої і доступної до аналізу інформації про клієнтів компанії є основним фактором розвитку процесу управління взаємовідносин з клієнтами.

Висновки до Розділу 1

У даному розділі було розглянуто поняття та історію створення CRM-систем, визначено типи та призначення CRM-систем. Проведено аналіз CRM-систем, які входять в топ самих популярних CRM- систем у світі, серед яких: Salesforce.com, Infor, Oracle Cloud, Bpm' online, Microsoft Dynamics 365, а також топ CRM-систем в Україні – Бітрікс24, AmoCRM, Мегаплан та виділено їхні основні особливості та сильні сторони. Здійснено огляд процесу взаємодії з клієнтами, розкрито його мету та етапи, а також розглянуто основні фактори, що впливають на процес управління відносинами з клієнтами.

РОЗДІЛ 2.

Організація розробки Web орієнтованої CRM-системи

2.1 Аналіз особливостей роботи веб - додатку

Так як розроблювана система управління взаємодією з клієнтами розгортається в мережі Інтернет, вона буде являти собою своєрідний веб - додаток. Програмне забезпечення такого роду функціонує за принципами «клієнт - серверної» архітектури, де в якості клієнта виступає браузер, а сервера – web- сервер. Особливість даного підходу полягає в тому, що сам додаток знаходиться і виконується на сервері – клієнт при цьому отримує тільки результати роботи. Основним завданням останнього є коректне відображення завантаженої з сервера інформації, а також відправка на сервер запитів і даних користувача [6]. Передача запитів і результатів їх обробки відбувається через Інтернет (рис. 2.1).

2.1 – Принцип функціонування web-додатків [7]

За способами розробки веб - додатки розділяють на два основних типи: односторінкові (SPA) та багатосторінкові (MPA). Розглянемо відмінності між ними, а також переваги і недоліки кожного типу. Односторінкові додатки дозволяють імітувати роботу настільних програм, так як вони реагують на дії користувача без затримки.

Концепція SPA полягає в тому, що всі ресурси, необхідні для роботи (елементи CSS, скрипти та ін.), знаходяться на одній сторінці і завантажуються при ініціалізації. При такому підході немає необхідності в перезавантаженні або завантаженні додаткових сторінок – контент оновлюється динамічно за допомогою JavaScript (AJAX - запитів). На рисунку 2.2 наведено принцип роботи односторінкового додатку.

Рисунок 2.2 – Принцип роботи односторінкового додатку [8]

Переваги:

− при роботі завантажуються тільки необхідні дані (без повторюваних елементів і блоків), що дозволяє розвантажити ресурси сервера;

− відсутність повного перезавантаження сторінки підвищує швидкість роботи.

− код серверної частини (бекенду) можна повторно використати для створення власної мобільної програми.

− концепція SPA дозволяє ефективно кешувати будь - які дані. Програма відправляє всього один запит, збирає дані, а після цього може функціонувати в offline - режимі.

Недоліки:

− початкове завантаження триває досить довго, через великий розмір JavaScript коду і допоміжних бібліотек.

− для коректної роботи потрібен JavaScript. Якщо користувач вручну відключить JavaScript в браузері, він не зможе повноцінно використовувати весь функціонал програми.

− через особливості архітектури більшість сторінок просто недоступні для сканування пошуковими роботами, що ускладнює SEO - оптимізацію.

Багатосторінкові додатки працюють за традиційною схемою. Це означає, що при незначній зміні даних чи переході на іншу сторінку браузер робить новий запит до сервера і знову завантажує всі ресурси, навіть ті компоненти, які повторюються на всіх сторінках (заголовок, нижній колонтитул, тощо). Таким

чином, продуктивність витрачається на завантаження тих самих елементів. Принцип роботи даного підходу зображено на рисунку 2.3.

2.3 – Принцип роботи багатосторінкового додатку [8]

Переваги:

− MPA архітектура дозволяє легко оптимізувати web - додаток для пошукових систем. Є можливість додати мета - теги для кожної сторінки окремо;

− як правило, для розробки багатосторінкових додатків потрібен менший стек технологій, таким чином їх вартість виходить дешевшою.

Недоліки:

− розробка як десктопної, так і мобільної версії займає значно більше часу;

− тісний зв'язок між бекендом і фронтендом ускладнює процес додавання нових функціональних можливостей [9].

2.2 Огляд фреймворків для розробки системи

Розробка веб - додатку з нуля вимагає багато роботи. При цьому в більшості випадків доводиться витрачати час на відтворення функцій, що вже були виконані тисячі разів. Фреймворки допомагають обійти цю проблему, надаючи свого роду каркас майбутньої програми та набір додаткових інструментів, покликаних прискорити роботу зі створення продукту.

Фреймворк – це програмне забезпечення, яке спрощує створення і підтримку технічно складних або навантажених проектів. Як правило, вони

містять тільки базові програмні модулі, а всі специфічні для проекту компоненти реалізуються розробником на їх основі. Тим самим досягається не тільки висока швидкість розробки, а й значна продуктивність і надійність рішень [10].

Загальні переваги від використання фреймворків:

− у методологіях фреймворків зазвичай закладено кращі практики програмної інженерії, тож просто дотримуючись цих правил можна уникнути багатьох проблем і помилок в проектуванні.

− екосистеми веб - фреймворків мають велику кількість готових рішень для нестандартних завдань. Також є можливість розширення функціоналу шляхом підключення сторонніх бібліотек або окремих класів;

− побудовані на базі фреймворків додатки мають уніфіковану структуру. Відповідно їх значно простіше супроводжувати і модернізувати, так як стандартизована організація компонентів зрозуміла всім розробникам на цій платформі.

− рішення на фреймворках, як правило, працюють значно швидше, витримують більші навантаження, а також мають вищий рівень безпеки у порівнянні з системами, написаними без їхнього використання[11] .

Виходячи з того, що концепція SPA передбачає чітке розділення відповідальності між зовнішнім представленням (front - end) і внутрішньою реалізацією (back - end) варто окремо потурбуватися про вибір фремворка як для серверної, так і для клієнтської частин. При цьому серверний фреймворк дозволить полегшити вирішення таких завдань, як взаємодія з базою даних, авторизація користувачів і забезпечення захисту від веб - загроз. Фреймворк клієнтської сторони, у свою чергу, допоможе в побудові багатофункціонального інтерфейсу користувача.

2.3 Вибір фреймворку для розробки серверної частини

Проведемо аналіз найпопулярніших серверних фреймворків:

Django – веб - фреймворк на мові Python. Його робота ґрунтується на принципах "DRY" (don't repeat yourself) – механізмі повторного використання, що дозволяє

мінімізувати дублювання коду в ході розробки. Важливою архітектурною відмінністю цього фреймворка є спосіб організації обробників URL, які у випадку Django конфігуруються явно (за допомогою регулярних виразів), а не автоматично задаються із структури контролерів.

Особливості:

− ORM, API доступу до БД з підтримкою транзакцій;

− диспетчер URL на основі регулярних виразів;

− механізм інтернаціоналізації, що забезпечує можливість роботи додатка на декількох мовах;

− спеціалізована архітектура дозволяє встановлювати допоміжні модулі у будь - які Django - проекти;

− вбудований механізм авторизації та можливість підключення зовнішніх модулів аутентифікації: LDAP, OpenID;

− система фільтрів (middleware) для побудови додаткових обробників запитів, як наприклад включені в дистрибутив фільтри для кешування, нормалізації URL та підтримки анонімних сесій;

− бібліотека для роботи з формами (успадкування, побудова форм по існуючій моделі БД) [12];

Laravel – один з найпопулярніших PHP - фреймворків з відкритим вихідним кодом. Дозволяє значно спростити розробку, завдяки вбудованим засобам вирішення загальних завдань, серед яких аутентифікація, маршрутизація, сесії і кешування.

Цей фреймворк є досить гнучким – можна створити власну структуру, яка відповідає вимогам конкретного проекту.

Особливості:

− пакети – дозволяють створювати і підключати модулі у форматі Composer. Багато додаткових можливостей вже доступні у вигляді таких модулів.

− Eloquent ORM – представляє таблиці бази даних у вигляді класів для полегшення доступу і маніпулювання даними.

− зворотня маршрутизація – пов'язує між собою згенеровані додатком посилання і маршрути. При створенні посилань за допомогою іменованих маршрутів Laravel автоматично генерує кінцеві URL.

− REST - контролери – додатковий шар для розділення логіки обробки GET і POST запитів HTTP.

− автозавантаження класів – механізм автоматичного завантаження класів PHP без необхідності підключати файли їх визначень через include.

− міграції – система управління версіями для баз даних. Дозволяє зв'язати зміни в коді додатка зі змінами, що треба внести в структуру БД. Це спрощує процес розгортання і оновлення додатка.

− модульне тестування – дозволяє перевірити окремі модулі вихідного коду програми на предмет регресій (помилок внаслідок оновлення коду або виправлення інших помилок) [13].

Ruby on Rails – багаторівневий фреймворк для побудови веб - додатків, що використовують реляційні і NoSQL бази даних (наприклад, MySQL, MariaDB, PostgeSQL, MongoDB). Фреймворк написаний на мові програмування Ruby. Одним з ключових принципів розробки на Ruby on Rails є "convention over configuration" – при початковій ініціалізації фреймворк використовує угоди по конфігурації, типові для більшості проектів. Це значно спрощує процес створення додатків, оскільки явна специфікація конфігурації потрібна тільки в нестандартних випадках [14].

Особливості:

− бібліотека доступу до БД – реалізує шаблон проектування Active Record, який спрощує роботу з базами даних. Він дозволяє представити таблиці у вигляді класів, а рядки – об'єктів. За рахунок цього є можливість використовувати об'єктно - орієнтований підхід доступу до даних.

− бібліотеки для загальних завдань

– фреймворк містить безліч допоміжних бібліотек, які створюють функціонал для роботи з файлами, відправки електронних листів, валідації форм, підтримки сесій, роботи із зображеннями тощо.

− конфігурація URL – Ruby on Rails має інструменти для гнучкого налаштування URL - адрес.

Встановимо ряд критеріїв для визначення оптимального серверного фреймворка:

- документація – наявність зрозумілої документації для швидкого старту;
- продуктивність – швидкість обробки вхідних запитів;
- безпека – ступінь захисту від веб - загроз;
- поріг освоєння – кількість часу, необхідного для освоєння фреймворка людиною з мінімальним досвідом у веб - розробці;
- гнучкість і масштабування – можливість впровадження допоміжних бібліотек для розширення базових можливостей.

У таблиці 2.1 зазначено відповідність розглянутих фреймворків встановленим критеріям.

Таблиця 2.1 – Аналіз серверних фреймворків

	Django	Laravel	Ruby on Rails
Документація	+	+	–
Продуктивність	+	+	+
Безпека	–	+	–
Поріг освоєння	±	+	–
Гнучкість і масштабування	+	+	+

2.4 Вибір фреймворку для розробки клієнтської частини

Проведемо аналіз найпопулярніших клієнтських фреймворків:

Vue.js – бібліотека для створення інтерактивних веб - інтерфейсів. Метою Vue.js є надання переваг реактивного зв'язування компонентів представлення з API.

Vue.js не являється повноцінним фреймворком, оскільки відповідає тільки за

візуалізацію, тобто працює на рівні представлення. Проте у поєднанні з належними
інструментами та допоміжними бібліотеками, Vue.js чудово підходить для створення веб-додатків будь-якого рівня складності [15].

Особливості:

− прив'язка даних – дозволяє встановити зв'язок між елементами вебсторінки і даними об'єкту Vue.js. Таким чином можна керувати значеннями HTMLатрибутів, змінювати стилі, привласнювати CSS-класи.

− реактивні інтерфейси – при зміни даних автоматично оновлюється їх відображення на сторінці.

− компактність – ядро Vue.js містить лише функціонал для роботи з інтерфейсом. Тому воно компактне і легко інтегрується з іншими технологіями.

React – бібліотека для створення компонентно-орієнтованих інтерфейсів за допомогою JavaScript і (необов'язково) XML. Вона використовує концепцію розділення інтерфейсу користувача на компоненти, самодостатні частини, які просто розширювати і застосовувати повторно. Компоненти можна описати використовуючи JavaScript або XML з HTML-подібним синтаксисом (JSX).

Особливості:

− одностороння передача даних – компонент не може напряму змінювати властивості, що йому передані, а тільки за допомогою callback функції. Такий механізм називають «властивості донизу, події нагору». Він мінімізує ризик виникнення помилок та дозволяє легше відстежувати зміни.

− віртуальний DOM – React створює кеш структуру в пам'яті, що дозволяє визначити різницю між попереднім і поточним станами інтерфейсу для оптимального оновлення елементів сторінки.

− JavaScript XML (JSX) – це розширення синтаксису JavaScript, що дозволяє використовувати схожий на HTML синтаксис для опису структури інтерфейсу [16].

Angular – платформа побудована на TypeScript, що надає розробникам

надійні інструменти для створення клієнтської частини веб-додатків. Архітектура фреймворка ґрунтується на трьох складових: модуль (надає контекст компіляції для компонентів), компонент (визначає набір елементів інтерфейсу користувача, які *Angular* може змінювати відповідно до логіки програми і даних), маршрутизація
(дозволяє зв'язати декілька компонентів один з одним) [17].

Особливості:

- модульність – розділення коду на окремі модулі, що полегшує процес тестування і дозволяє повторно використовувати різні частини програми.
- директиви – дозволяють розширювати базовий функціонал HTML і створювати нові конструкції.

TypeScript – має ряд переваг у порівнянні зі звичайним JavaScript, серед яких: сувора типізація (дозволяє зробити поведінку коду більш передбачуваною і спростити налагодження програми), механізми об'єктно-орієнтованого програмування тощо.

Вибір фреймворка для клієнтської частини буде здійснюватись за наступними критеріями:

− простота – ступінь складності освоєння;

− документація – наявність детально описаного керівництва по використанню;

− швидкість рендерінгу – кількість часу, необхідного фреймворку для відображення на сторінці оновлених даних.

У таблиці 2.2 зазначено відповідність розглянутих фреймворків встановленим критеріям.

Таблиця 2.2 – Аналіз клієнтських фреймворків

	Vue.js	React	Angular
Простота	+	+	–
Документація	+	+	+
Швидкість рендерінгу	+	+	–

Висновки до Розділу 2

Підвівши підсумки, можна зробити висновок, що кожна архітектура має свої переваги для проектів певного типу. Так, варіант з використанням MPA добре підійде для тих випадків, коли потрібно відобразити велику кількість контенту, наприклад, при створенні інтернет - магазину, бізнес сайту, каталогу. Але, так як передбачається, що розроблювана система буде мати інтерфейс з досить високим ступенем інтерактивності, застосування підходу, при якому сторінка перезавантажується після кожної дії користувача, не є доцільним. Виходячи з цих міркувань, для реалізації системи скористаємося концепцією односторінкового додатку.

За результатами проведеного аналізу фреймворків для серверної частини можна зробити висновок, що всі фреймворки мають гарні показники відносно таких важливих характеристик, як продуктивність і гнучкість. Однак в плані безпеки лідирує Laravel, що повязано з наступними можливостями фреймворка:

- захист від SQL - ін'єкцій – вбудована ORM виключає можливість використання “сирих” SQL - запитів, а всі параметри при їх побудові нормалізуються.
- захист від XSS (кроссайтового скриптингу) – фреймворк забезпечує механізм екранування заборонених тегів (зокрема script, php та інших), при цьому код буде представлений у вигляді звичайного тексту без можливості його виконання на сервері.

До його переваг також можна віднести використання PHP – найбільш популярної мови у сфері веб - розробки, що забезпечує її підтримку будь - якими хостинг - провайдерами. Виходячи з цього для написання серверної частини CRM - системи було обрано Laravel.

Виходячи з результатів аналізу фреймворків для клієнтьської частини React і Vue.js випереджають Angular за швидкістю роботи завдяки впровадженню концепції VirtualDOM. Цей підхід дозволяє оптимізувати використання ресурсів,

що позначається на швидкості відображення змін. Ще одним недоліком Angular є складний синтаксис. Також для роботи з цим фреймворком треба додатково вчити TypeScript. Інші фреймворки чудово підходять для реалізації поставленої задачі. Але виходячи з особистого досвіду роботи, а також враховуючи той факт, що обраний раніше серверний фреймворк Laravel забезпечує офіційну підтримку Vue.js, при розробці клієнтської частини будемо використовувати саме його.

РОЗДІЛ 3.

Програмна реалізація CRM-системи управління взаємодією з клієнтами книжкового Інтернет-магазину

3.1. Проектування логічної та фізичної структури сайту

Для комфортного використання можливостей системи та полегшення пошуку потрібної інформації, вона повинна мати чітку і продуману структуру. З позиції розробника структуру умовно можна поділити на два рівні: логічний і фізичний. На логічному рівні вона представляє собою сукупність сторінок, які об'єднані між собою єдиним дизайном, стилем і посиланнями. На рисунку 3.1 наведено логічну структуру системи управління взаємодією з клієнтами книжкового інтернет магазину.

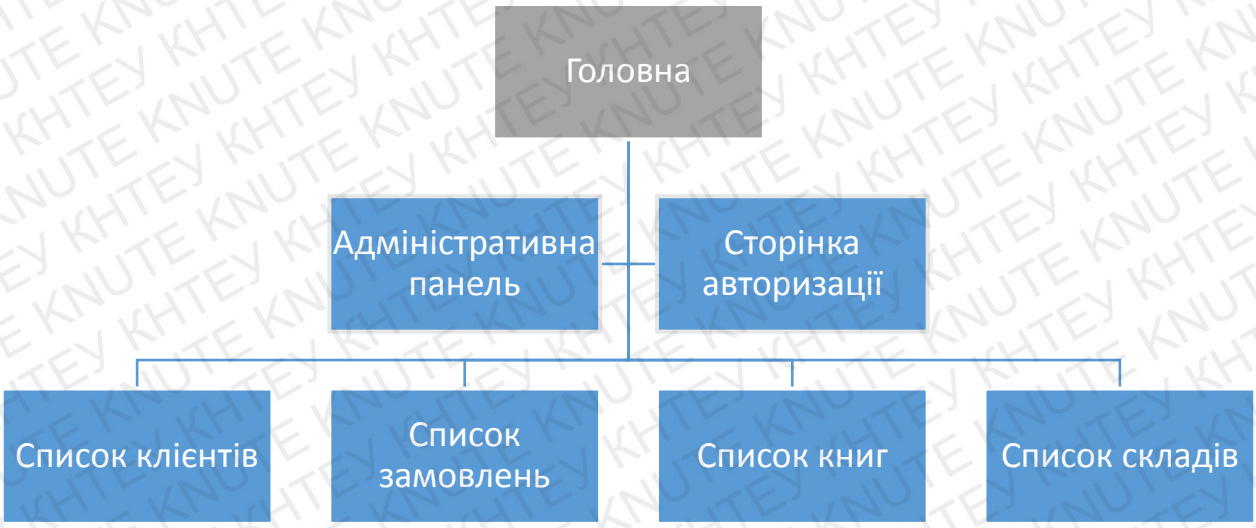

Рисунок 3.1 – Логічна структура

Адміністративна панель забезпечує виконання наступних операцій:
− відображення списку клієнтів;
− пошук клієнта за ініціалами, номером телефону або емейлом;
− внесення до бази інформації про нового клієнта;
− редагування профілю клієнта;
− видалення профілю клієнта;
− відображення списку заказів;
− пошук заказів за назвою;
− внесення інформації в базу про нове замовлення;
− видалення заказу;
− відображення списку книг;
− внесення до бази інформації про нову книгу;
− видалення книги з бази;
− відображення списку складів;
− внесення до бази інформації про новий склад;
− видалення складу з бази;

Фізичний рівень, у свою чергу, відповідає за файлову складову, а саме правильне розміщення елементів різного типу (HTML-сторінки, скрипти, зображення) у відповідних папках та файлах. В ході розробки системи була використана стандартна організація файлів фреймворку Laravel . В ній можна виділити наступні ключові складові:
− каталог css – містить файли, що підключаються до кореневої сторінки та будуть задавати стилі для всього додатку;
− каталог js – містить файли із описаними сценаріями javascript для різних інструментів та сценарії для підтримання динаміки сайту;
− каталог storage (patient-avatars, user-avatars) – містить файли, що завантажують користувачі системи (наприклад зображення профілю);

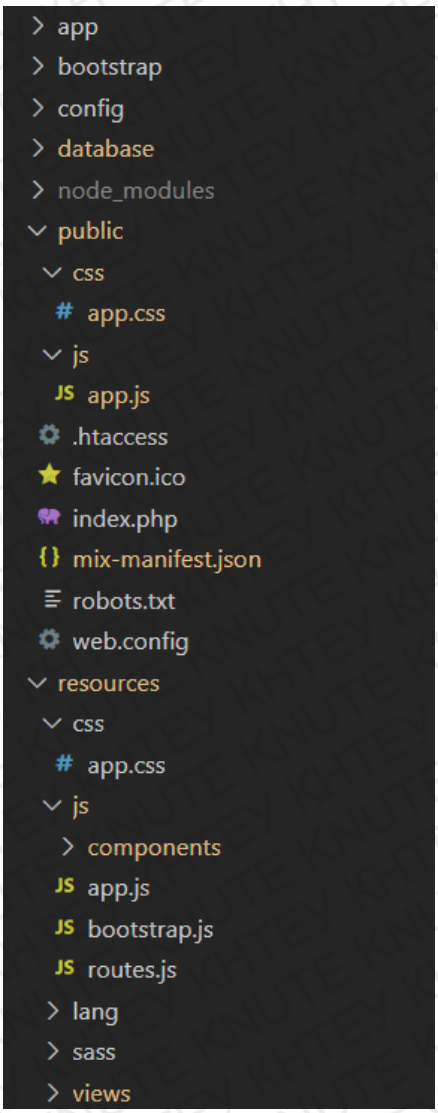

Рисунок 3.2 – Фізична структура

3.2 Проектування API

У веб-додатку, побудованому за архітектурою SPA серверна сторона обмежена бекендом через деякі кінцеві точки входу API (Application Programming Interface). Клієнт відправляє запити на ці кінцеві точки, а сервер повертає відповідь. Проте сервер не піклується про те, як клієнт буде відображати отриману інформацію, що ідеально відповідає принципу розділення відповідальності. Ця архітектура дозволяє розробникам створювати надійні веб-додатки, а також додатки для різних пристроїв [19].

Нижче наведено декілька IDEF0 діаграм, що відображають принцип обробки

вхідних запитів.

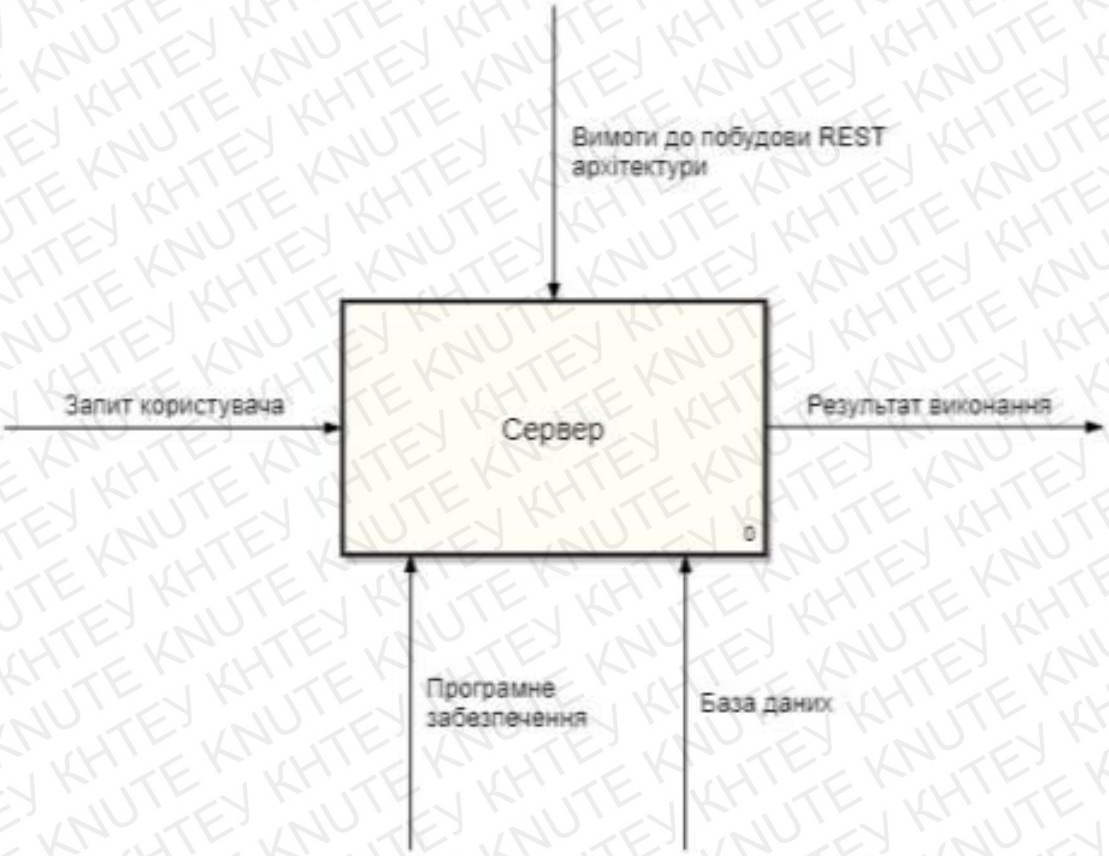

Рисунок 3.3 – Загальна IDEF0 діаграма

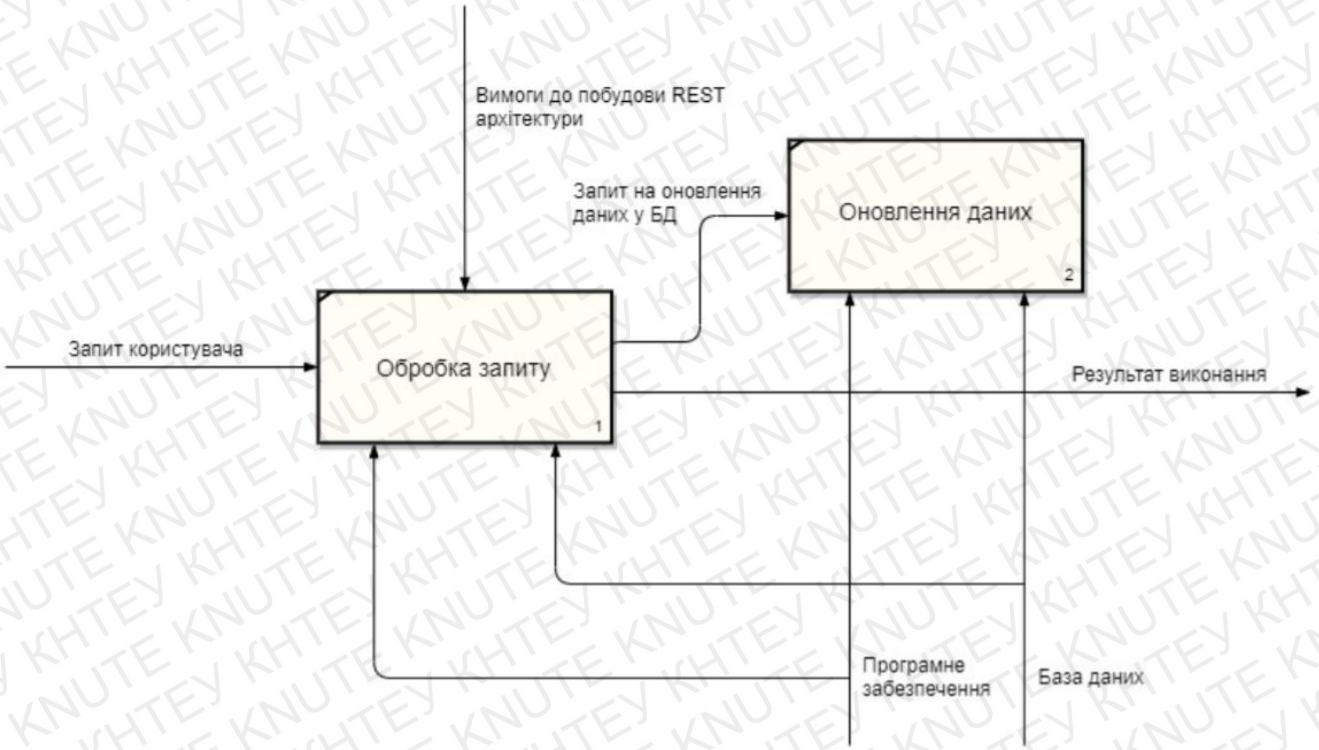

Рисунок 3.4 – Декомпозиція блоку "Сервер"

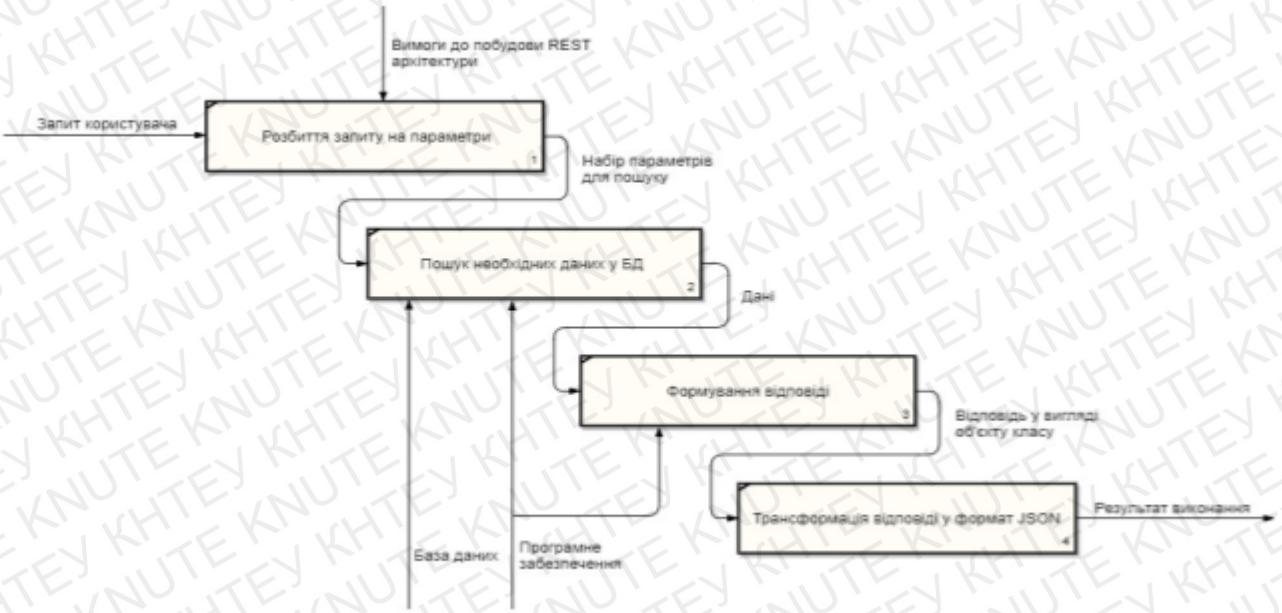

Рисунок 3.5 – Декомпозиція блоку "Обробка запиту"

У цій роботі для створення RESTful API буде використовуватися Laravel версії 8.73.2 Термін "RESTful" вказує на те, що цей інтерфейс буде працювати за принципами REST. У загальному розумінні REST - це архітектурний стиль для проектування розподілених систем. Він не є стандартом, але визначає обмеження, такі як відсутність станів, клієнт-серверний взаємозв'язок і уніфікований інтерфейс. При такому підході тіло запиту і відповіді від сервера представляє собою JSON об'єкт JSON (JavaScript Object Notation) – стандартний текстовий формат для представлення структурованих даних на основі синтаксису об'єкту JavaScript. В якості прикладу розглянемо як виглядають дані клієнта у форматі JSON:


```
[
  {
    "type":"database",
    "name":"crm"
  },
  {
    "type":"table",
    "name":"clients",
    "database":"crm",
    "data":[
      {
        "id":"1",
        "name":"Dr. Viola Aufderhar",
        "email":"kallie11@example.org",
        "phone":"+18149220481",
        "created_at":"2021-11-23 21:00:22",
        "updated_at":"2021-11-23 21:00:22"
      }
    ]
  }
```

Відповідно до архітектурних принципів REST, існує щонайменше 4 HTTP методи, що відповідають основним операціям над сутностями:
завантаження даних (GET), збереження (POST), редагування (PUT/PATCH) і видалення (DELETE). Цей перелік також супроводжується такими операціями, як обробка помилок в запиті, розмежування доступу і валідація вхідних даних. Структура HTTP-методів, що були використані в ході розробки представлена в таблиці 3.2.

Таблиця 3.2 – HTTP-методи, що використовувалися при розробці

Метод	Операція	Тіло запиту	Тіло відповіді
GET	Отримати інформацію	NA	JSON string
POST	Створити новий запис	JSON string	JSON string
PUT/PATCH	Оновити запис	JSON string	JSON string
DELETE	Видалити запис	NA	JSON string

Універсальні відповіді від сервера представлені в таблиці 3.3.

Таблиця 3.3 – Універсальні відповіді від сервера

Помилка	Код відповіді	Опис
Bad request	400	Некоректний запит (універсальна помилка)
Not found	404	Ресурс не знайдено
Unprocessable Entity	422	Неможливо виконати операцію над ресурсом
Not authorized	403	Необхідна авторизація
Server error	500	Помилка сервера

У разі помилки API повертає код статусу HTTP і відомості про її джерело. Крім того, у відповіді міститься інформація відносно того, що призвело до цієї помилки. Об'єкт помилки представлено в таблиці 3.4

Таблиця 3.4 – JSON-об'єкт помилки

Атрибут	Тип	Призначення
http_code	number	Ідентифікатор помилки
message	string	Повідомлення про помилку
data	object	Інформація, що асоціюється з помилкою

Всі інші JSON-об'єкти, якими обмінюються між собою front-end і back-end, відображають свої сутності. Для прикладу розглянемо об'єкт типу «Користувач» (табл. 3.5).

Таблиця 3.5 – JSON-об'єкт користувача

Атрибут	Тип	Призначення
id	number	Ідентифікатор
full_name	string	ПІБ
email	string	Адреса електронної пошти
phone	string	Номер телефону
created_at	string	Дата реєстрації аккаунта
updated_at	string	Дата оновлення аккаунта

Після проходження етапу проектування JSON-об'єктів, необхідно створити URL-адресу для кожної сутності, що дозволить унікально ідентифікувати REST запит. Зазвичай вона складається з пар виду "назва сутності/ідентифікатор", об'єднаних

між собою URL роздільником "/". Ідентифікатор виступає в ролі первинного ключа для визначення об'єкту сутності та є необов’язковим параметром. Представимо структуру кінцевих точок, що використовуються клієнтською стороною для взаємодії з сервером, у вигляді діаграм класів UML.

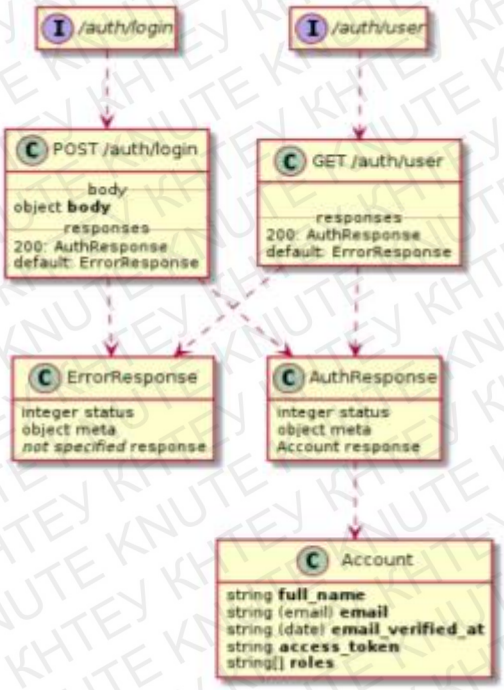

Рисунок 3.6 – Кінцеві точки для авторизації в системі

3.3 Проектування структури інтерфейсу

Структура інтерфейсу використовується при розробці дизайну веб-додатку та показує найбільш важливі елементи, їх положення та взаємозв'язок між сторінками.

При розробці структури інтерфейсу зазвичай будують певний каркас, на якому схематично зображають найбільш важливі елементи, такі як верхній і нижній

колонтитули (header та footer), панель навігації тощо.

Основні переваги розробки структури інтерфейсу перед створенням вебдодатку:

− раціональний розподіл часу і зосередження саме на тому, для чого

призначена кожна сторінка;

− видалення зайвих елементів, що можуть виявитися непотрібними при

роботі з системою;

− отримання чіткого уявлення про те, що потрібно реалізувати;

− зменшення обсягу роботи при розробці дизайну;

Нижче наведено структуру інтерфейсу основних сторінок системи

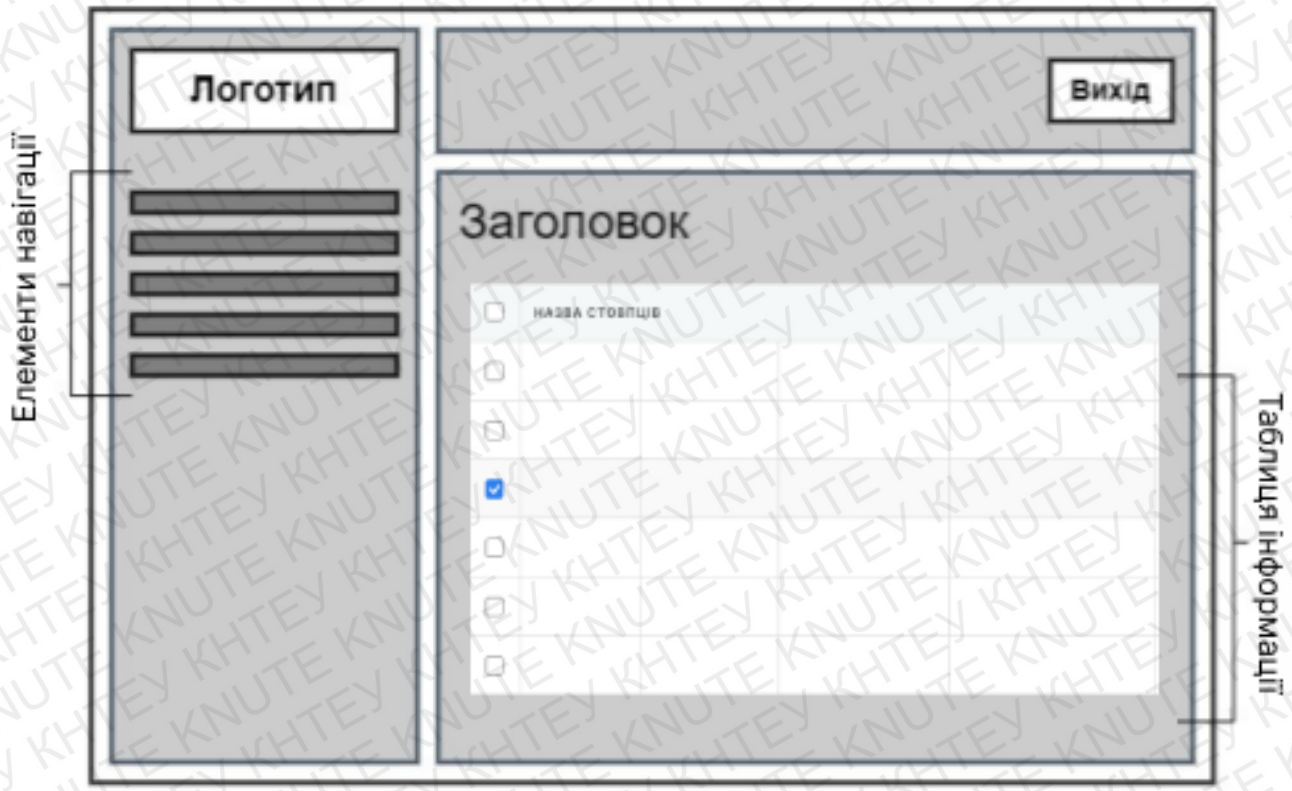

Рисунок 3.7 – Структура сторінки пацієнтів

При розробці інтерфейсу був зроблений акцент на зручність роботи. Таким чином, доступ до основних функцій користувач може отримати з головного меню на будь-якій сторінці системи.

3.4 Проектування бази даних

Ключове місце в системі управління взаємодією з клієнтами займає база даних, на проектування якої слід звернути особливу увагу, оскільки існує потреба в роботі з великими обсягами інформації. Виходячи з цього під час розробки структури БД були використані механізми нормалізації.

Нормалізація полягає в приведенні структури зберігання даних до нормальних форм. Нормальна форма – це ряд вимог, що пред'являється до структури таблиць для забезпечення цілісності бази даних і усунення надмірних функціональних залежностей. В цій системі база даних буде приведена до перших трьох нормальних форм:

− перша нормальна форма: кожна таблиця повинна мати первинний ключ; кожен атрибут має складатись тільки з елементарних (неподільних) значень і не містити групи, що повторюються (атомарність);

− друга нормальна форма: дані, що повторно з'являються в декількох рядках, виносяться в окремі таблиці;

− третя нормальна форма: дані не повинні зберігатися в таблиці, якщо їх можна отримати з не ключових полів [20].

Зв'язки між таблицями бази даних визначаються за допомогою ключів двох типів:

− первинний ключ – значення, що унікально ідентифікує кожний запис;

− зовнішній ключ – значення, що відповідає первинному ключу з іншої таблиці.

Розглянемо існуючі таблиці бази даних системи.[рис.3.8]

Рисунок 3.8 – Модель бази даних

Налаштування підключення до БД

Як було сказано в попередніх розділах за зберігання, організацію і управління доступом до даних відповідає серверна сторона, що в цьому проекті була реалізована за допомогою фреймворку Laravel. Однією з його переваг є наявність спеціалізованого модуля для роботи з базами даних – Eloquent ORM. За допомогою цього інструменту описуються сутності і їх зв'язки, визначається як та чи інша сутність відображається у базі. Він бере на себе значну частину роботи по генерації SQL-запитів, завантаженню даних та кастингу (перетворення типів бази даних в типи цільової мови).

У результаті виходить, що Eloquent ORM ховає основну роботу з базою даних (вимагаючи тільки правильної конфігурації) і самостійно виконує усі необхідні запити. Однак, у складних випадках, можливостей цього модулю може не вистачити. Для вирішення подібних проблем Laravel містить Query Builder – зручний та виразний інтерфейс, який спрощує генерацію власного SQL-коду. Laravel значно спрощує взаємодію з БД завдяки тому, що він працює трьох рівнях: чистий SQL, конструктор запитів (Query Builder), об'єктні моделі Eloquent ORM.

На даний момент фреймворк підтримує роботу з наступними СКБД: MySQL, PostgreSQL, SQLite і SQL Server. Однак, при розробці системи було використано саме MySQL, у зв’язку з її високою швидкістю роботи, надійністю і гнучкістю.

Важливим етапом перед початком роботи з базою даних є налаштування конфігурації. Файл з основними параметрами налаштування БД знаходиться в config/database.php. У ньому можна задати різні варіанти підключення до бази даних, обрати необхідну СКБД, а також визначити підключення за замовчуванням.

Ключові параметри конфігурації наведено нижче:

```
'mysql' => [
'dsn' => 'mysql',
'host' => '127.0.0.1',
'port' => '3306',
```

'database' => 'crm',
'username' =>'root',
'password' => 'root',
'charset' => 'utf8mb4',
]
В даному випадку, dsn містить назву драйвера mysql;
host – доменне ім'я або IP-адресу сервера БД;
database – назву бази даних,
username і password – ім 'я користувача з правами доступу до БД та його пароль;
charset – кодування (використовується для коректного сортування і індексації даних).

Іноді виникає необхідність використати одно з'єднання для виконання запиту
SELECT, а інше для запитів INSERT, UPDATE і DELETE. Laravel максимально спрощує цей процес, причому не важливо, які при цьому використовуються засоби – сирі запити, Query Builder чи Eloquent ORM.
Розглянемо приклад подібної конфігурації:
'mysql' => [
'read' => ['host' => '192.168.1.1'],
'write' => ['host' => '196.168.1.2'],
'dsn' => 'mysql',
'database' => 'database',
'username' => 'root',
'password' => '',
'charset' => 'utf8',
]
З наведеного прикладу можна побачити, що в масиві налаштувань з’явилося два нових ключа: read і write, що дозволяють визначити адресу сервера (host) для виконання відповідних операцій. Таким чином, 192.168.1.1 буде використовуватися

для завантаження інформації, а 192.168.1.2 - для запису.

Для адміністрування сервера MySQL було використано допоміжне програмне забезпечення phpMyAdmin. За допомогою зручного веб-інтерфейсу phpMyAdmin дозволяє через браузер здійснювати управління базами даних MySQL, запускати запити SQL, переглядати та редагувати вміст таблиць. Оскільки цей інструмент в більшості випадків забезпечує можливість обійтися без самостійного введення команд SQL, то робота з базами даних є простою та швидкою.

Розглянемо процес створення нової бази даних за допомогою інтерфейсу phpMyAdmin. Для цього у рядку «Створити БД» необхідно ввести назву бази даних і обрати кодування (рис. 3.9). Для розробленої бази даних було обрано кодування

utf8_general_ci, що дозволить зберігати в таблицях символи кирилиці.

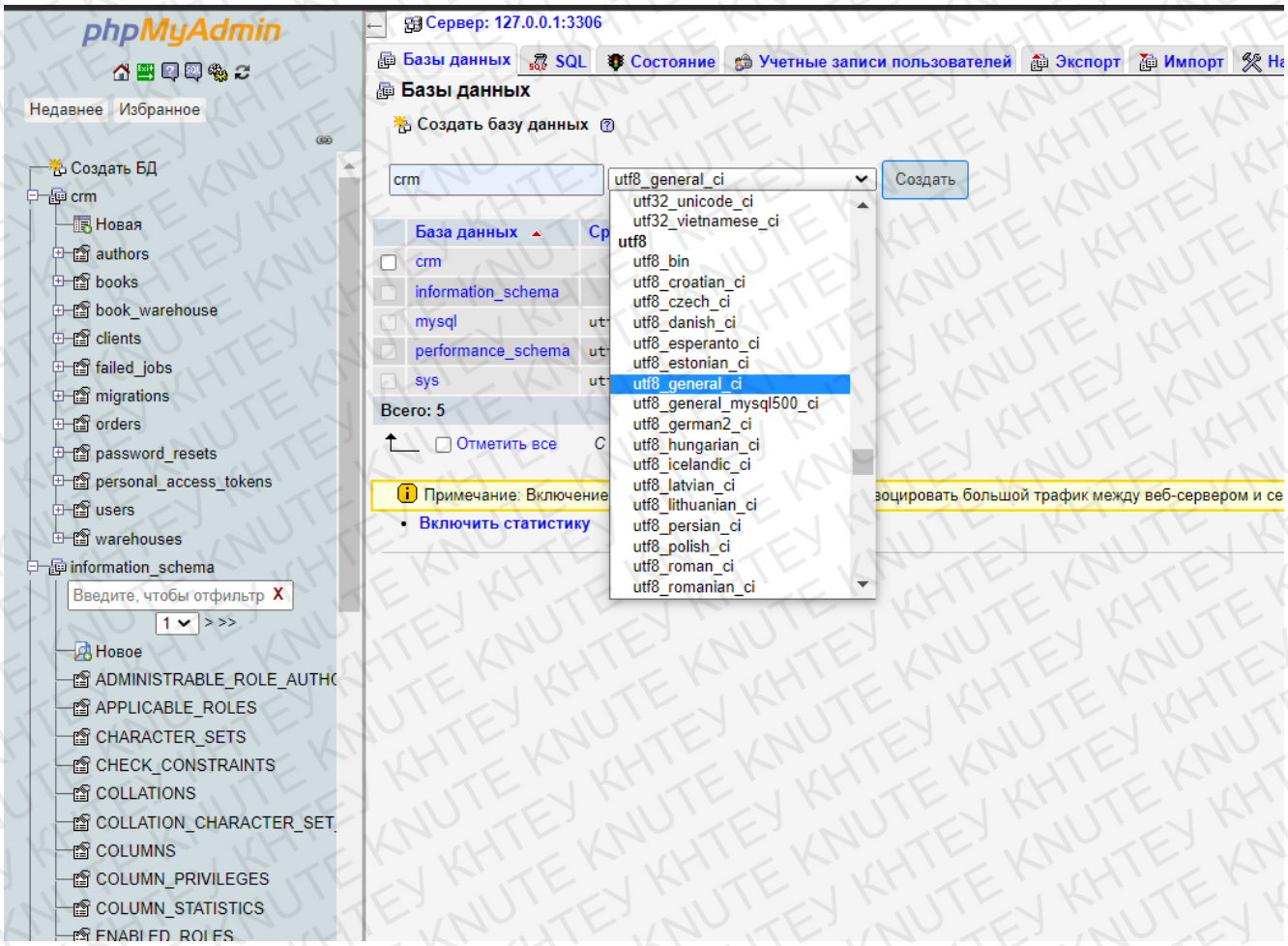

Рисунок 3.9 – Створення БД «book_crm»

Налаштування міграцій бази даних

Після створення нової бази даних необхідно наповнити її відповідними таблицями для успішного функціонування системи. З цією метою було використано механізм міграцій Laravel.

В ході розробки і ведення баз даних структура таблиць розвивається, відповідно до вихідного коду веб-додатку. Наприклад, при розробці додатка може бути виявлено, що для підвищення продуктивності запитів має бути створений певний індекс. Міграції схеми бази даних, у свою чергу, дозволяють виправляти помилки та адаптувати дані до змін. Вони виконуються коли необхідно створити, оновити, або повернути схему бази даних до старішої версії.

Для створення міграції в Laravel використовується консольна команда php artisan make:migration. Нова міграція буде розташована в директорії database/migrations. Також, в її назві буде міститись мітка часу, яка дозволить фреймворку визначати порядок застосування міграцій. В наведеній команді можна також використати параметри --table і --create для вказівки імені таблиці і того факту, що міграція має створити нову таблицю, а не змінювати існуючу.

В структурі класу створеної міграції слід відзначити два основних методи: up() і down(). Метод up() використовується для додавання нових таблиць, стовпців або індексів в БД, а метод down() просто скасовує операції, виконані методом up(). В якості прикладу розглянемо клас для створення таблиці користувачів:

```
class CreateUsersTable extends Migration
{
public function up()
{
Schema::create('users', function (Blueprint $table)
```

```
{ $table->bigIncrements('id');
 $table->string('full_name');
$table->string('phone');
$table->string('email')->unique();
$table->timestamp('email_verified_at')->nullable();
 $table->string('password');
$table->string('hash_file_name')->nullable();
$table->timestamps();
});
}
public function down()
{
Schema::dropIfExists('users');
}
}
```

Для створення нової таблиці БД тут використовується метод create фасаду Schema. Він приймає два аргументи: ім'я таблиці, та функцію Closure, в якій можна визначити структуру нової таблиці. При необхідності редагування існуючої таблиці можна використати метод table фасаду Schema. Як і метод create, він приймає два аргументи: ім'я таблиці і замикання, що отримує екземпляр Blueprint, який можна використати для додавання нових стовпців. Для видалення таблиці в down() застосовується метод dropIfExists.

В розглянутому прикладі для визначення структури таблиці «users» використовуються наступні типи стовпців:

− $table->bigIncrements('id') - інкрементний ID (первинний ключ), що використовує еквівалент "UNSIGNED BIG INTEGER";

− $table->string() - еквівалент VARCHAR;

− $table->timestamp() - еквівалент TIMESTAMP;

Для деяких полів використовується модифікатор ->nullable(), який вказує на те, що поле може бути пустим. Метод $table->timestamps() використовується для

створення двох полів «created_at» та «updated_at» типу TIMESTAMP, що будуть 62 використані для відображення часу і дати створення запису в таблиці, а також її оновлення.

Запуск міграції виконується за допомогою команди php artisan migrate. Результат її виконання можна побачити використавши phpMyAdmin (рис. 3.10).

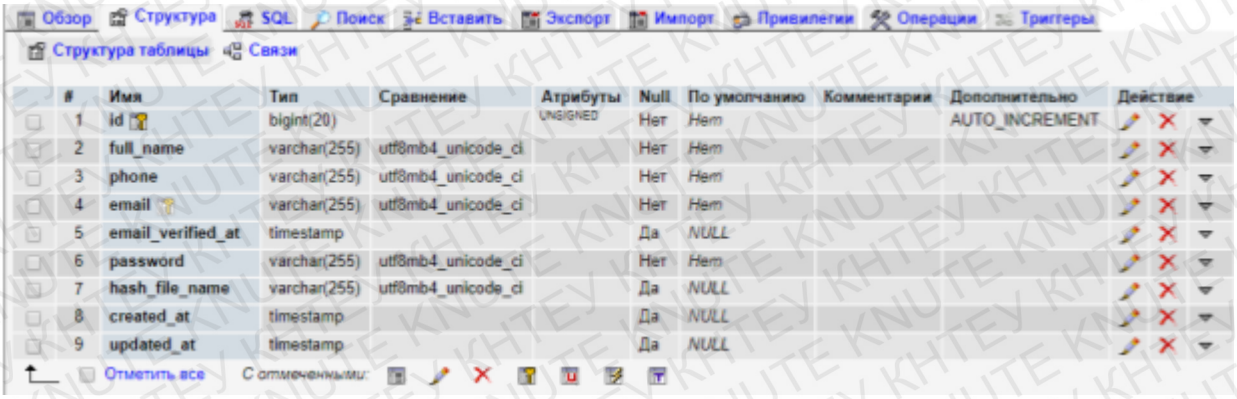

Рисунок 3.10 – Структура таблиці «users»

3.5 Розробка модуля авторизації

Головним завданням модулю авторизації є керування рівнями доступу до ресурсів системи, тобто надання користувачам прав на виконання певних дій в залежності від їх ролі. Передбачається, що в розробленій системі буде наступне розподілення ролей: клієнти, які мають доступ лише до можливості перегляду інформації та запису на прийом, адміністратори, які можуть спостерігати за всім процесом роботи системи та працівники.

Ключовим елементом даного модулю є форма авторизації (вікно для введення логіну та паролю). Код, що відповідає за її відображення наведено нижче:

```
<h1 class="form-title">Авторизація</h1>
<v-card class="form-card elevation-2">
<v-card-text class="card-content">
<v-form>
<v-text-field
```

```
label="Email"
name="email"
prepend-icon="email"
type="email"
v-model="email"
maxlength="255"
:error-messages="emailErrors"
@blur="$v.email.$touch()"
@keyup.native.enter="submit"
></v-text-field>
<v-text-field
label="Password"
name="password"
prepend-icon="lock"
:append-icon="showPassword ? 'mdi-eye' : 'mdi-eye-off'"
@click:append="showPassword = !showPassword"
:type="showPassword ? 'text' : 'password'"
v-model="password"
maxlength="20"
:error-messages="passwordErrors"
@blur="$v.password.$touch()"
@keyup.native.enter="submit"
></v-text-field>
<div class="remember">
<v-checkbox
label="Запам’ятати мене"
color="primary"
hide-details
v-model="remember"
class="mt-2"
```

></v-checkbox>
<router-link :to="/forgot-password">Забули пароль?</router-link>

</v-form>
</v-card-text>
<v-card-actions class="card-actions">
<v-btn class="form-submit" @click.prevent="submit">Login</v-btn>
</v-card-actions>
</v-card>

В розглянутій частині коду можна побачити HTML елементи, що не відповідають жодному зі стандартних тегів, як наприклад <v-card></v-card>. В такий спосіб Vue.js дозволяє впровадити в структуру сторінки власні компоненти – невеликі частини інтерфейсу користувача з ізольованими стилями і скриптами. Кожен з них реалізує певний функціонал, а при необхідності їх можна з легкістю використати знову в іншому місці. Компонент, який в DOM виглядає як тег з нестандартною назвою, може включати не лише HTML-розмітку, але і логіку своєї роботи і використання інших компонентів. Використання компонентного підходу дозволяє розділяти складні сторінки на менші частини, які легше підтримувати. Зовнішній вигляд форми, що є результатом виконання фрагменту коду, наведеного вище, можна побачити на рисунку 3.11.

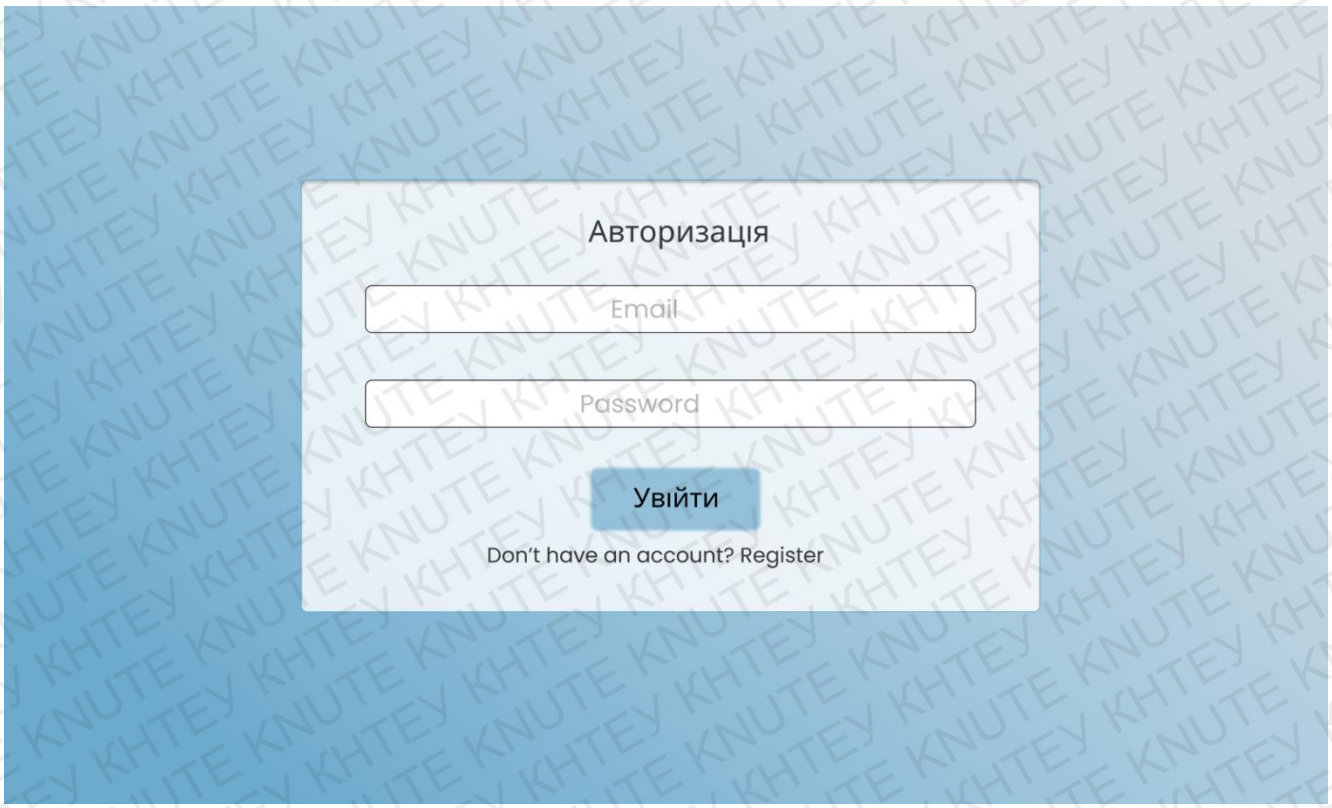

Рисунок 3.11 – Форма авторизації

Одним з головних принципів архітектури REST, яка застосовувалася в ході реалізації методів API (інтерфейсу для взаємодії клієнта з сервером), є stateless – незалежність від стану. Це означає, що сервер не може зберігати інформацію про користувача між запитами. Виходячи з цього, кожен запит з клієнтської сторони повинен містити усі необхідні параметри для його обробки. Такий підхід потребує відправки на сервер секретного токена доступу для ідентифікації користувача і отримання прав на виконання певних операцій.

Розглянемо особливості процесу авторизації з використанням токенів. Після успішного входу в систему на сервері буде згенерована унікальна послідовність символів фіксованої довжини – токен, що ідентифікує поточну сесію. В якості відповіді при виконанні процедури авторизації токен буде переданий на сторону клієнта для подальшого збереження. Надалі його можна буде використовувати в ході відправки запитів для підтвердження особи користувача. Зазвичай, для цього використовується спеціальний http -заголовок Bearer: <token>. Після завершення сеансу роботи (процедура logout), токен стає не дійсним і видаляється на стороні клієнта. При спробі отримати доступ до захищених ресурсів системи без

відповідних параметрів авторизації сервер поверне відповідну помилку – 401 Unauthorized.

Код контролера Laravel, який відповідає за логіку авторизації, наведено нижче:

```
class AuthController extends Controller {
public function login(AuthLogin $request) {
$credentials = $request->only('email', 'password');
if (!auth()->attempt($credentials)) {
return response()->api(null, 422);
}
$user = User::where('email', $request->email)->first();
return $this->respondWithToken($user);
}
public function user() {
$user = auth()->user();
return $this->respondWithToken($user);
}
private function respondWithToken(Authenticatable $user) {
return response()->api([
'full_name' => $user->full_name,
'email' => $user->email,
'email_verified_at' => $user->email_verified_at,
'access_token' => $user->api_token,
'roles' => $user->roles->pluck('name')
]);
}
}
```

У класі AuthController можна виділити такі методи:

− login() – відповідає за обробку облікових даних, відправлених з форми

авторизації. У цьому методі виконується пошук користувача по логіну і паролю, за що відповідає auth() ->attempt ($credentials). Якщо за цими даними у БД немає збігів – з сервера прийде повідомлення про помилку (response () ->api (null, 403)). У разі успіху операції метод поверне токен доступу разом з допоміжною інформацією (список ролей, дата підтвердження електронної адреси).

− user() – завантажує інформацію про користувача на основі переданого токена авторизації (auth () ->user()) і повертає її у вигляді JSON об'єкту

3.6 Розробка адміністративної частини

Адміністративна частина, що призначена тільки для співробітників

має такий спектр можливостей :

− перегляд замовлень, книг, клієнтів, складів;

− додавання нових замовлень, книг, клієнтів, складів;

− видалення замовлень, книг, клієнтів, складів;

− контроль за кількістю продаж ;

− перегляд статистики товарів і клієнтів;

Адміністративна частина є основною складовою CRM-системи, доступ до якої здійснюється за допомогою розпізнавання ролі користувача. Використання механізмів авторизації та розмежування прав доступу дозволяє убезпечити систему від несанкціонованого доступу. При проектуванні адміністративної частини була мета забезпечити повний контроль над роботою книжкового інтернет-магазину, а завдяки простому та інтуїтивно зрозумілому інтерфейсу освоєння її можливостей не відніме багато часу. Для навігації по сторінкам адміністративної частини, використовується меню, зображене на рисунку 3.12.

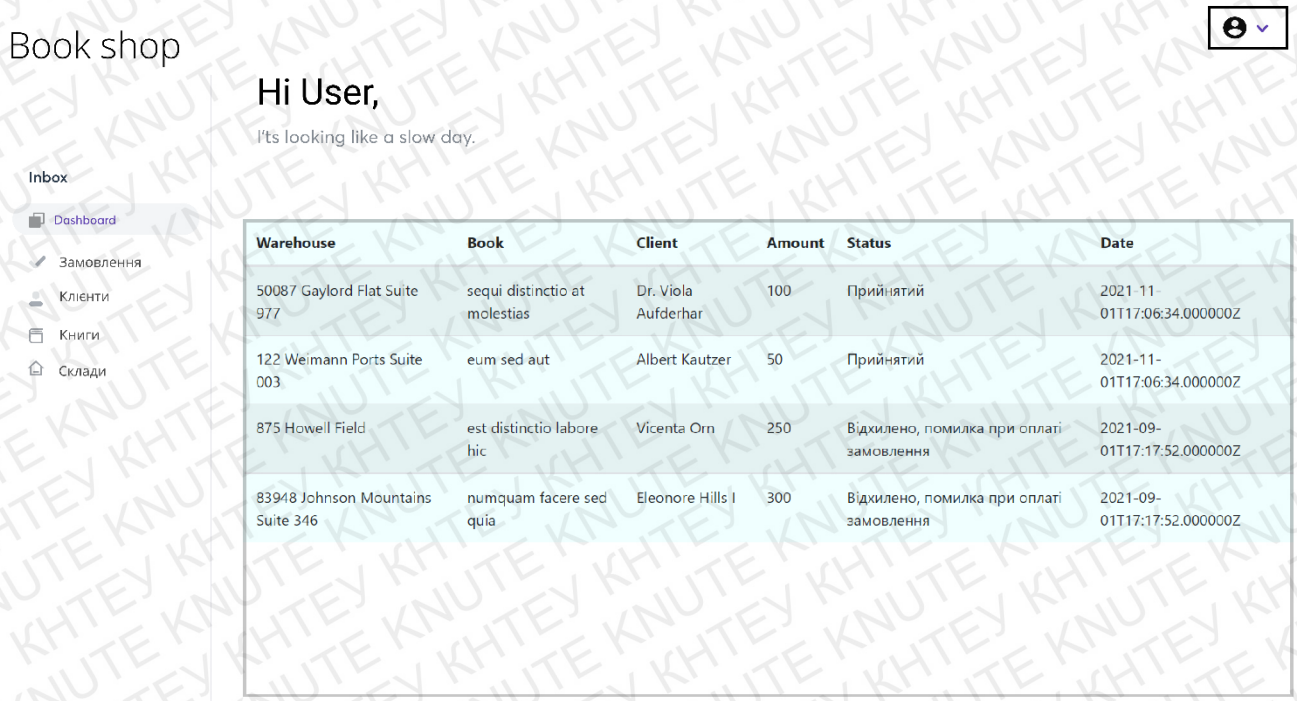

Рисунок 3.12 – Головне меню

Відображення пунктів меню та права доступу до відповідних сторінок залежать від ролі авторизованого користувача. Так, адміністратор системи, на відміну від звичайного користувача, має змогу використовувати можливості модулів аналітики, послуг та користувачів.

Код Vue.js компоненту, що використовується для відображення меню можна побачити нижче:

```
<template>
 <v-navigation-drawer class="app__drawer" :mini-variant.sync="mini" app v-model="drawer"
:width="drawWidth">
 <v-toolbar color="primary darken-1" dark>
 <img :src="logo" width="36" height="36" alt="App logo"/>
 <v-toolbar-title class="ml-0 pl-3">
 <span class="hidden-sm-and-down">Book shop</span>
```

```
</v-toolbar-title>
</v-toolbar>
<v-list dense nav expand class="py-6">
<v-list-item-group color="primary">
<v-list-item
v-for="item in menu"
ripple="ripple"
:to="item.name ? { name: item.name } : null"
:key="item.name"
>
<v-list-item-icon v-if="item.icon">
<v-icon>{{ item.icon }}</v-icon>
</v-list-item-icon>
<v-list-item-content>
<v-list-item-title>{{item.title}}</v-list-item-title>
</v-list-item-content>
</v-list-item>
</v-list-item-group>
</v-list>
</v-navigation-drawer>
</template>
<script>
import menu from "@/js/config/menu";
export default {
props: {
drawWidth: {
type: [Number, String],
default: "260"
},
},
```

```
data() {
return {
mini: false,
scrollSettings: {
maxScrollbarLength: 160
}
}
},
computed: {
logo() {
return "/storage/images/logo.svg";
},
isAdmin() {
return this.$store.getters['auth/isAdmin'];
},
menu() {
return menu.filter(item => item.admin ? this.isAdmin : true);
},
drawer: {
get: function () {
return this.$store.state.app.drawer;
},
set: function (newValue) {
this.$store.dispatch('app/setDrawerState', newValue);
}
},
},
}
</script>
```

Фрагмент, що відповідає за фільтрацію елементів навігації в залежності від

ролі знаходиться в методі menu().

Розробка сторінки “клієнти”

Модуль пацієнтів являє собою автоматизоване робоче місце лікаря, мета якого спростити виконання рутинних завдань.

Головною перевагою цього модуля є зручна робота з клієнтською базою. При переході на головну сторінку даного модуля першим, що побачить користувач системи буде таблиця, яка містить основну інформацію про пацієнтів: ПІБ, контактні дані тощо (рис. 3.5).

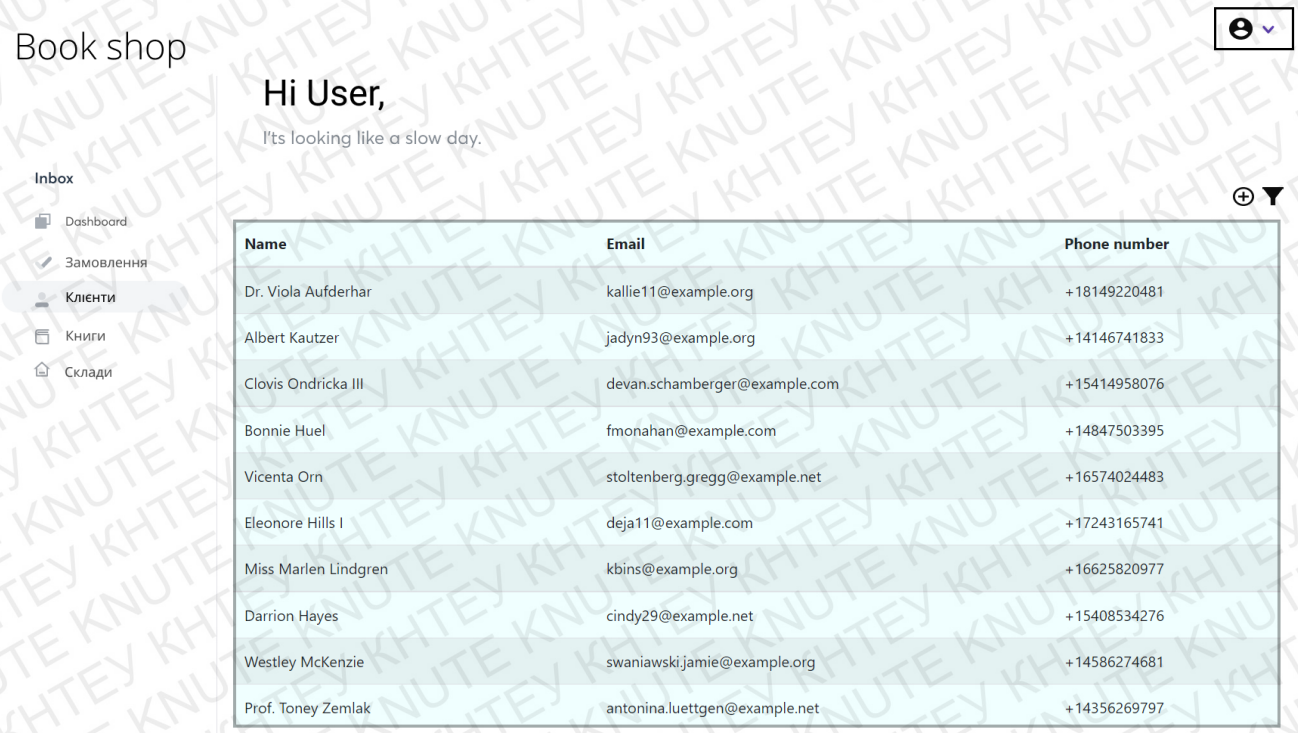

Рисунок 3.12 – сторінка “клієнти”

Відображенням списку клієнтів займається компонент Clients.vue. Серед інших методів цього компоненту можна виділити loadClients(), завданням якого є асинхронне завантаження даних при ініціалізації:

```
loadClients() {
 let params = {
```

```
q: this.search,
page: this.options.page || 1,
limit: this.options.itemsPerPage || 15,
sort_by: this.options.sortBy || [],
sort_desc: this.options.sortDesc || [],
};
this.isLoading = true;
this.$store.dispatch('clients/loadClients', { params })
.finally(() => {
this.isLoading = false;
});
}
```

В наведеній частині коду можна побачити js об’єкт params, що буде відправлений на сервер у вигляді query параметрів для реалізації пошуку (q: this.search), пагінації (page: this.options.page, limit: this.options.itemsPerPage) та сортування (sort_by: this.options.sortBy, sort_desc: this.options.sortDesc). З урахуванням переданих параметрів кінцева url для завантаження списку пацієнтів буде виглядати наступним чином :<base_url>/api/clients?limit=15&page=2&q=test&sort_by=full_name&sort_desc=false. Зі сторони сервера, у свою чергу, було реалізовано контролер для обробки запиту по завантаженню даних клієнтів.

Розглянемо метод, що відповідає за виконання цієї операції:

```
public function index(ClientsIndex $request)
{
$limit = $request->get('limit') ?? 10;
$patients = Client::query()
->when($request->get('q'), function(Builder $query) use (&$request) {
$query->where('id', 'LIKE', '%'. $request->get('q') .'%')
->orWhere('full_name', 'LIKE', '%'. $request->get('q') .'%')
->orWhere('phone', 'LIKE', '%'. $request->get('q') .'%');
```

```
})
->when($request->get('sort_by'), function(Builder $query) use (&$request) {
$direction = $request->get('sort_desc');
$query->orderBy($request->get('sort_by'), (isset($direction)
&& json_decode($direction)) ? 'DESC' : 'ASC');})
->orderBy('created_at', 'DESC')
->paginate($limit);
return response()->api($patients);
}
```

В даному методі використовується засоби Eloquent ORM для завантаження необхідної інформації з БД. При цьому, за допомогою об'єкту request в автоматично згенерований SQL-запит будуть передані вилучені query параметри. Так, 'sort_by' визначає стовпець, за яким буде проведено сортування результатів вибірки:

```
->when($request->get('sort_by'), function(Builder $query) use (&$request) {
$direction = $request->get('sort_desc');
$query->orderBy($request->get('sort_by'), (isset($direction)
&& json_decode($direction)) ? 'DESC' : 'ASC');
})
```

Згенерований SQL-запит:

```
SELECT *
FROM `clients`
WHERE `id` LIKE <query>
 OR `full_name` LIKE <query>
 OR `phone` LIKE <query>
ORDER BY `full_name` ASC, `created_at` DESC
LIMIT 15 offset 0
```

Результатом виконання методу index() стане JSON об'єкт, сформований на основі інформації, отриманої за допомогою наведеного SQL-запиту:

```
{
```

"status":1,
"meta":{
"http_code":200,
"pagination":{
"limit":15,
"current_page":1,
"total":1,
"last_page":1
}
},
"response":[
{
"id":1,
"full_name":"Maribel Reynolds DDS",
"phone":"+38 (052) 795-88-34",
"email":"orion67@example.org",
"user_id":7,
"created_at":"2020-05-01 13:00:50",
"updated_at":"2020-05-01 13:00:50"
},
]
}

По аналогії були створені всі інші сторінки з бокового меню:

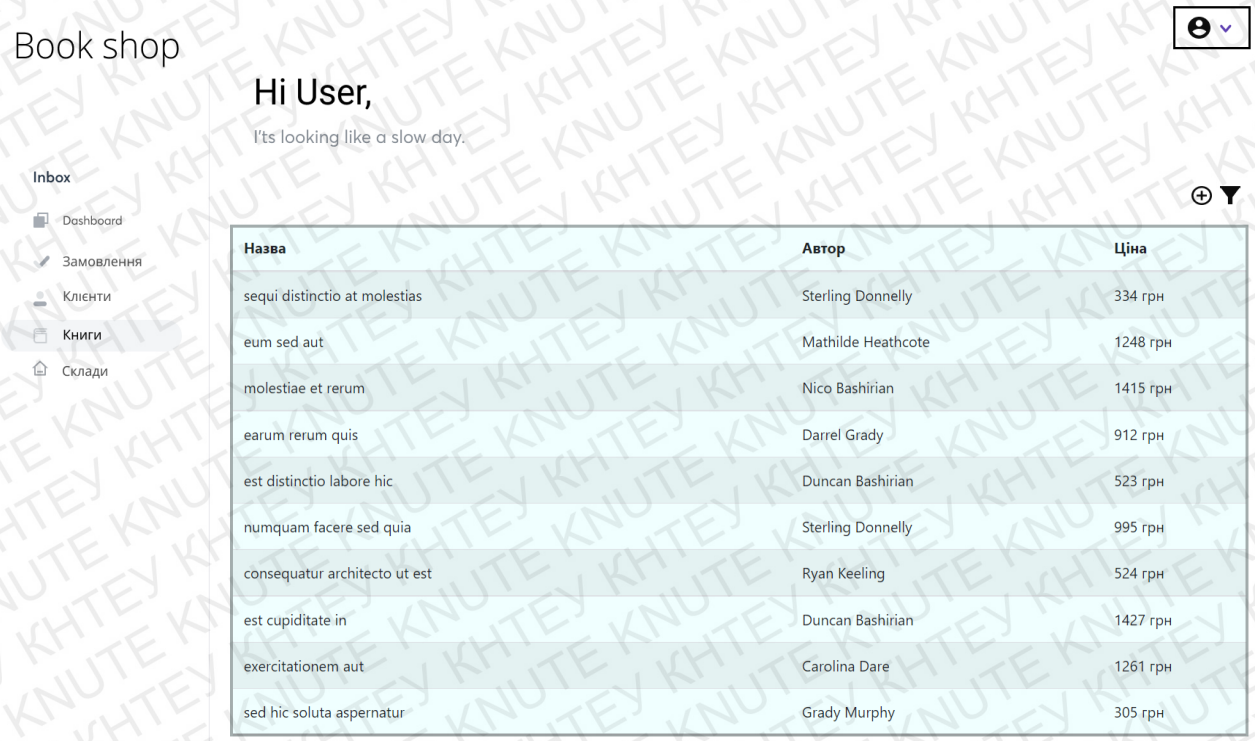

Рисунок 3.13 – сторінка “книги”

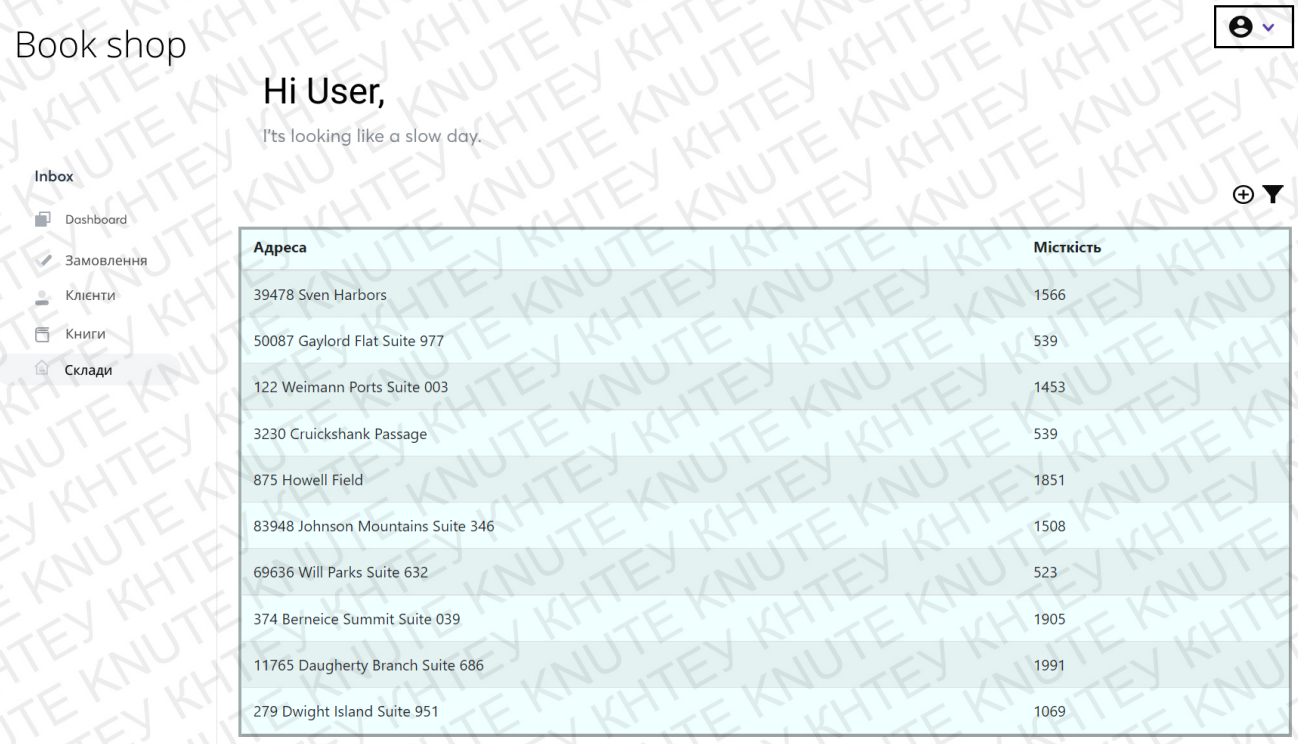

Рисунок 3.14 – сторінка “склади”

Висновки до розділу 3

ВИСНОВКИ ТА РЕЗУЛЬТАТИ

У даній роботі була спроектована система управління взаємодією з клієнтами книжкового інтернет-магазину. Особливий акцент робився на забезпеченні доступу до ресурсу як зі стаціонарних комп'ютерів, так і з мобільних пристроїв, що дало б змогу працівникам отримувати та вносити інформацію безпосередньо на робочих місцях.

Після визначення основної задачі було проведено пошук і порівняльний аналіз готових рішень. Враховуючи всі недоліки та переваги розглянутих аналогів були сформовані вимоги до функціональних можливостей системи.

Наступним етапом був варіантний аналіз існуючих технологій та методів створення веб-додатків і обґрунтування їх вибору для процесу розробки. Так, було прийнято рішення реалізувати систему на основі архітектури SPA, що дозволить побудувати зручний інтерфейс з високим ступенем інтерактивності.

Крім цього були обрані фреймворки і бібліотеки, які б допомогли у вирішенні поставленого завдання.

По завершенні попереднього аналізу було проведено проектування архітектури та програмна реалізація модулів системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Затулов С. Н. Що таке CRM і як його впровадити на підприємстві [Текст] / С.Н. Затулов // Економічний вісник фармації, 2003. – № 9. – С. 31-41.
2. Рожков І. В. Інформаційні системи і технології в маркетингу: монографія [Текст] / І. В. Рожков // 2014. – 199 с.
3. Птащенко О. В. Побудова CRM-системи як основи формування комунікаційної політики між організацією та кінцевим споживачем [Текст] / О. В. Птащенко // Вісник Східноукраїнського національного університетуту ім. Володимира Даля, 2016. – № 6. – С. 108-115.
4. Тарасенко Е. А. Управління клієнтською політикою на основі технологій CRM (на прикладі приватної лікувально-профілактичної установи) [Текст] / Е. А. Тарасенко , Т. Б. Рижкова // Вісник Російського державного гуманітарного університету, 2015. – № 3(146). – С. 89-99.

5. Огляд CRM-систем для стоматології [Електронний ресурс]. – Режим доступу: https://crm-systems.info/dlja-stomatologii/#_CRM.
6. Принципи роботи і структура web-додатків на основі ASP.NET [Електронний ресурс]. – Режим доступу: https://www.sibsau.ru/sveden/edufiles/145507/
7. Методика проектування систем Web 2.0 [Електронний ресурс]. – Режим доступу: https://crm-systems.info/dlja-stomatologii/#_CRM
8. Багатосторінкові та односторінкові додатки, їх переваги та недоліки [Електронний ресурс]. – Режим доступу: https://dou.ua/forums/topic/25444/.
9. Single-page application vs multiple-page application [Електронний ресурс]. – Режим доступу: https://neoteric.eu/blog/single-page-application-vs-multiple-pageapplication/?utm_source=medium.com&utm_medium=social&utm_content=neo&utm_campaign=blog .
10. Горбунов-Посадов М. М. Розширювані програми [Текст] / М. М. Горбунов-Посадов // Москва : Полиптих, 1999. – 336 с.
11. Проценко М. М. Аналіз фреймворків як засобів розробки Web-додатків [Текст] / М. М. Проценко // Міжнародний науковий журнал, 2016. – № 6(1). – С. 86-89.
12. Django [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Django.
13. Laravel [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Laravel.
14. Advantages of Web Development with Ruby on Rails [Електронний ресурс]. – Режим доступу: https://onix-systems.com/blog/advantages-of-web-development-withRuby-on-Rails.
15. Vue.js – Overview [Електронний ресурс]. – Режим доступу: https://www.tutorialspoint.com/vuejs/vuejs_overview.htm.
16. React [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org /wiki/React.
17. Introduction to Angular concepts [Електронний ресурс]. – Режим доступу:

https://angular.io/guide/ architecture.
18. Буч Г. Мова UML. Інструкція користувача [Текст] / Г. Буч, Д. Рамбо, А. Джекобсон // Вид. 2. – СПб. : ДМК Пресс, 2004. – 432 с.
19. Fielding R. Architectural Styles and the Design of Network-based Software Architectures [Електронний ресурс] / R. Fielding. – Режим доступу: https:// www.ics.uci.edu/~fielding/pubs/dissertation/top.htm..
20. Нормалізація реляційних баз даних [Електронний ресурс]. – Режим доступу: https://club.shelek.ru/ viewart.php?id=177 .
21. CRM (Customer Relationship Management) [Електронний ресурс] – Режим доступу до ресурсу: URL: https://www.cfin.ru/itm/crm-review.shtml – Дата доступу 15.07.2020.
22. Управление взаимоотношениями с клиентами (CRM): возможности автоматизированных систем и программные продукты [Електронний ресурс] – Режим доступу до ресурсу: URL: https:////www.kp.ru/guide/ upravlenievzaimootnoshenijami-s-klientami.html
23. Черкашин П.А. Стратегия управления взаимоотношениями с клиентами (CRM) Издательство: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 420 с.
24. Виды CRM-систем [Електронний ресурс] – Режим доступу до ресурсу: URL: https://crm-systems.info/vidy-crm-sistem/
25. Голышева Е., Сорокин М., Кудинов Алексей., CRM: практика эффективного бизнеса / А. Кудинов, М. Сорокин, Е. Голышева. - 1С ФИРМА, 1С- Паблишинг, Манн, Иванов и Фербер,2012. С. 461.
26. Salesforce [Електронний ресурс]: Режим доступу: https://crmsystems.info/salesforce/
27. Infor CRM (ранее SalesLogix) [Електронний ресурс] – Режим доступу до ресурсу: URL: http://www.crmonline.ru/software/foreing/saleslogix/
28. Infor CRM (SalesLogix) [Електронний ресурс] – Режим доступу до ресурсу: URL: https://fbconsult.ru/crmtehnologii
29. Oracle CRM [Електронний ресурс] – Режим доступу до ресурсу:

URL: https://crm-systems.info/oracle/
30. BPM online [Електронний ресурс] – Режим доступу до ресурсу: URL: https://crm-systems.info/bpmonline/
31. Microsoft Dynamics [Електронний ресурс] – Режим доступу до ресурсу: URL: https://crm-systems.info/microsoftdynamics/
32. Бітрікс24: Результати першого дослідження ринку CRM в Україні [Електронний ресурс] – Режим доступу до ресурсу: URL: https://www.bitrix24.ua/crmresearch2018/
33. Рязанцев А. Как внедрить CRM-систему за 50 дней. – М.: Книжкин дом, 2017. – 180 с
34. Цапюк П. #crm_is. – М: Издательские решения, 2017. – 170 с.
35. ТОП-10 лучших CRM-систем для бизнеса – рейтинг 2019 [Електронний ресурс] – Режим доступу до ресурсу: URL: https://bisnesideya.ru/podborki/top-10-crm-sistem-dlya-biznesa.html
36. Наприенко Д. В. Модель управления взаимоотношениями с клиентами в сфере ритейла // Молодой ученый. 2018. №20. С. 159-160.

37. Степанова Т. В., Морсина Е. В. О некоторых аспектах применения CRM-систем для управления сложными продажами // Концепт. 2017. Т. 3. С. 174-178.
38. Романова Е. А. Управление взаимоотношениями с клиентами // Вестник Саратовского государственного социально-экономического университета. 2008. №4 (23). С. 57-60
39. Морозов Е. М. CRM-системы как средство автоматизации взаимодействия с клиентами // Современные научные исследования и инновации. 2015. №3. Ч. 3.
40. Баранова И. В., Мурадов А. А. Формирование стратегии устойчивого развития высокотехнологичного предприятия на основе управления клиентелой // Вопросы инновационной экономики. 2015. Т. 3. №2. С. 20-27.