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

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

«На правах рукопису»
УДК 004.043

«До захисту допущено»

Завідувач кафедри
_____ І.Р. Пархомей
(підпис)

“___”_____2018 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: ____Система продажу та купівлі рекламних повідомлень в Telegram

Виконав: студент другого курсу, групи ІТ-74мп
(шифр групи)

Муренко Владислав Володимирович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник ст. викладач, к.т.н. Дьяков С.О. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент к.ф.-м.н., доц., доц. каф. АСОІУ Гавриленко О.В. ____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

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

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Рівень вищої освіти – другий (магістерський)

Спеціальність 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ І.Р. Пархомей
(підпис)

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Муренку Владиславу Володимировичу
(прізвище, ім'я, по батькові)

1. Тема дисертації «Система продажу та купівлі рекламних повідомлень в Telegram», _____
науковий керівник дисертації ст. викладач, к.т.н. Дьяков С.О., _____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «07»листопада 2018 р. №4112-с

2. Термін подання студентом дисертації 10.11.2018 р.

3. Об'єкт дослідження – процес проектування та розробки CRM-системи, її взаємодії з API "Telegram" та платіжної системи.

4. Предмет дослідження – процес безпечної купівлі та продажу рекламних повідомлень в Telegram.

5. Перелік завдань, які потрібно розробити – оглянути API Telegram, описати архітектуру додатку, обрати програмні рішення та середовище, розробити front-end та back-end частини системи, розробити API, розробити статистичну та аналітичну частину системи, перенести проект на тестовий сервер.

6. Орієнтовний перелік ілюстративного матеріалу – шість плакатів

7. Орієнтовний перелік публікацій – дві публікації

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз предметної області	15.09.2018 р.	
2	Постановка задачі	17.09.2018 р.	
3	Аналіз інформаційного забезпечення	22.09.2018 р.	
5	Аналіз алгоритмічного забезпечення	27.09.2018 р.	
6	Розробка алгоритмічного забезпечення	17.10.2018 р.	
7	Розробка програмного забезпечення	03.11.2018 р.	
8	Маркетинговий аналіз стартап-проекту	12.11.2018 р.	
9	Висновки	17.11.2018 р.	

Студент _____ Муренко В.В.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Дьяков С.О.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

У роботі розглянуто проблему в області купівлі та продажу рекламних повідомлень у сучасних мессенджерах, зокрема, “Telegram”, показано основні особливості існуючих рішень проблеми, їх переваги та недоліки.

Розроблено систему, яка дозволяє повністю автоматизувати процеси продажу та купівлі рекламних повідомлень. Дана система може бути використана бізнесом для реклами своїх товарів та послуг. Дозволяє підвищити швидкість продажу та купівлі рекламних повідомлень і зекономити ресурси.

Ключові слова: crm, система, реклама, telegram.

Розмір пояснювальної записки – 91 аркушів, містить 21 ілюстрацій, 22 таблиць, 6 додатків.

ABSTRACT

Examines the problem in field of buying and selling promotional messages in modern messengers, in particular, Telegram, the main features of the existing solutions of the problem, their advantages and disadvantages are shown.

Developed a system that allows you to fully automate the sales and purchase of promotional messages. This system can be used by businesses to advertise their products and services. Allows you to increase the speed of sales process and purchase of advertising messages and save resources.

Keywords: crm, system, ad, telegram.

Explanatory note size – 91 pages, contains 21 illustrations, 22 tables, 6 applications.

Пояснювальна записка

до магістерської дисертації

на тему: *Система продажу та купівлі рекламних повідомлень в Telegram*

Київ – 2018 року

ЗМІСТ

ВСТУП ……5
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ... 7
1.1 Типова CRM-система та її складові 7
1.2 Призначення, структура та задачі які вирішує система 11
1.3 RESTful API, як архітектурний підхід до створення системи ……. 12
1.3.1 Елементи даних в REST…… 13
1.3.2 Методи REST …… 15
1.4 Опис проблеми, її актуальність та постановка задачі..... 15
Висновки до розділу ……..……..... 17
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ КУПІВЛІ ТА ПРОДАЖУ РЕКЛАМНИХ ПОВІДОМЛЕНЬ……..18
2.1 Проектування системи..... 18
2.1.1 Архітектура додатка..... 18
2.1.3 ODM “Mongoose”..... 20
2.1.4 Модуль регістрації/авторизації “Passport” ……..21
2.1.5 Frontend частина, Angular 6 …….……22
2.2 Моделі додатка ……..... 24
2.3 API-запити ……..... 26
Висновки до розділу…….... 27
РОЗДІЛ 3. РОЗРОБКА СИСТЕМИ КУПІВЛІ ТА ПРОДАЖУ РЕКЛАМНИХ ПОВІДОМЛЕНЬ ТА ІНТЕРФЕЙСУ КОРИСТУВАЧА…….28
3.1 MEANSTACK та приклади його використання ……..... 28
3.2 MongoDB або MySQL, чому Mongo?..... 29
3.3 Розробка backend частини……..... 34
3.3.1 Створення серверу……..… 34
3.3.2 Створення структури додатка…… 36
3.3.3 Створення роутів авторизації, каталогу, замовлення……38
3.3.4 Підключення БД “MongoDB”……..40

3.4 Формування API системи……41
3.5 Розробка frontend частини……... 42
3.5.1 Angular CLI та початок розробки…… 42
3.5.2 Розробка UI сторінки логіну …….. 43
3.6. Суть інтерфейсу користувача……..…44
3.6.1 Проблематика інтерфейсу користувача……..45
3.6.2 Методологія проектування інтерфейсних рішень…….46
3.6.3 Підхід дружнього інтерфейсу……..46
3.6.4 Принцип web-інтерфейсу користувача……...48
3.6.5 Принципи людського фактору в ІК…….49
3.6.6 Створення ІК системи…….…..50
3.7 Підготовка до взаємодії з телеграм-ботом…….…55
3.8 Опис роботи системи, керівництво користувача …….….56
Висновки до розділу …….….65
РОЗДІЛ 4. МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЕКТУ……..…67
4.1 Опис ідеї проекту……...….67
4.2 Технологічний аудит ідеї проекту……..68
4.3 Аналіз ринкових можливостей запуску стартап-проетку……68
4.4 Розроблення ринкової стратегії проекту……...74
4.5 Розроблення маркетингової програми стартап-проекту……..76
Висновки до розділу……..78
ВИСНОВКИ……..….80
ПЕРЕЛІК ПОСИЛАНЬ……..……82
ДОДАТКИ…….....84

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

API (application programming interface) – програмний інтерфейс;

Id – унікальний ідентифікаційний номер;

JSON (JavaScript Object Notation) – текстовий формат обміну даними;

ПЗ – програмне забезпечення;

Node.js (Nodejs) – платформа з відкритим кодом;

JavaScript (JS) – динамічна, об'єктно-орієнтована мова програмування;

Header – заголовки в HTTP - запитах;

Token – ключ, призначений для ідентифікації користувача;

jQuery – популярна JavaScript-бібліотека;

NPM – менеджер пакетів, що входить до складу Node.js;

Callback – асинхронний еквівалент функції;

ВСТУП

Технології миттєвих повідомлень та мессенджерів стрімко розвиваються та стають невіднятою частиною життя людини. На ринку мессенджерів доступно багато рішень, які тісно конкурують між собою, але один з них виділяється своєю простотою та функціональністю, це – “Telegram”.

Даний продукт з’явився на ринку нещодавно і є цілковито безкоштовним. Багато людей використовують його на своїх портативних пристроях із-за можливості обміну миттєвими повідомленнями. Але вони зовсім не знають про інші можливості.

Зокрема мова піде про можливість створення приватних і публічних каналів, які можна використовувати зовсім по-різному. Часто ці канали використовуються бізнесом для акумулювання постійної аудиторії, яка користується послугами або продуктами цього бізнесу. Багато відомих людей також створюють свої телеграм-канали з метою акумулювання бази своїх фанатів. Таких людей називають лідерами думок.

Нерідко так стається, що інтереси аудиторії лідера можуть пересікатися з інтересами бізнесу. Так, наприклад, продукт, який виготовляється бізнесом може бути прорекламований у телеграм-каналі зірки. Звісно, що такі послуги не є безкоштовними. Із-за цього виникає потреба в купівлі та продажу рекламних повідомлень (постів).

Із особистого досвіду, можу сказати, що на даний момент процес купівлі-продажу рекламних повідомлень не є зручним навіть для людини з досвідом. Так, в процесі купівлі рекламних повідомлень Ви можете натрапити на такі підводні камені, як: продаж по завищеній ціні, видалення рекламного повідомлення раніше за зазначений термін або повне небажання публікації повідомлення адміністратором каналу після отримання грошей.

Звісно можна знайти менеджера або іншу людину, яка буде займатися закупкою реклами для вашого бізнесу вручну, але для бізнесу – це додаткові витрати, які не завжди є правильним вкладенням коштів.

Метою даного дипломного проекту є створення CRM-системи, яка буде інтегрована з API “Telegram” і допоможе користувачам купувати та продавати рекламні повідомлення в мессенджері зручно, швидко, за графіком та в обхід вказаних вище проблемних моментів.

Актуальність розробки даної системи полягає в ефективності вирішення поставленої бізнесом задачі зручним шляхом та в зменшенні кількості часу, який потрібен користувачу для купівлі або продажу рекламного повідомлення в “Telegram”.

Об’єктом дослідження є процес проектування та створення CRM-системи, її взаємодії з API “Telegram” та платіжної системи, підготовка системи до підключення до спеціально розробленого telegram-боту (буде розроблений пізніше) для швидкої роботи з системою без використання комп’ютера та браузера, все що потрібно клієнту – додаток “Telegram” встановлений на портативному пристрої.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Типова CRM-система та її складові

CRM-система (Customer Relationship Management System) [1] – це поняття, яке часто використовується бізнесом для управління взаємовідносинами між компанією і клієнтами, які користуються її послугами або продуктом.

Така система часто може збирати, сортувати та аналізувати будь-яку інформацію про клієнта, яку вона отримує під час взаємодії з ним. Існують 3 основні принципи, які часто використовуються при створенні таких систем:

- єдине сховище інформації, в якому можна знайти будь-яку інформацію про взаємодію з клієнтом;
- синхронізація управління каналами взаємодії;
- постійний аналіз зібранної про клієнта інформації та прийняття рішень залежно від неї.

Щодо підходів роботи сучасні CRM-системи поділяються на такі:

- Оперативний – автоматизація споживчих бізнес-процесів і допомога існуючому персоналу у швидкому виконанні своїх функцій.
- Співробітницький – під час використання цього підходу CRM-система самостійно взаємодіє з користвачем без використання людини. У цьому разі будь-яка інформація, котра є результатом взаємодії завжди залишається у системі і далі може бути використана у власних цілях бізнесу.
- Аналітичний – доволі популярний на сьогоднішній день метод роботи CRM-систем. При використанні цього підходу у систему нерідко завантажується база даних з інформацією про клієнтів компанії, яка згодом сортується, аналізується з різноманітними цілями, які будуть корисними для бізнесу.

На наступних сторінках буде розглянуто призначення та структуру системи, розроблюваної у даному дипломному проекті. Виходячі з цієї

інформації, можна зробити висновок, що під час розробки будуть використовуватися оперативний та аналітичній підходи створення CRM.

Розібравшись з визначенням та описом типової CRM-системи у користувача може виникнути питання, чи не легше замість розробки складної системи використовувати працю професіоналів, які, напевно, виконають ті ж самі операції значно продуктивніше. Тому варто розглянуто основні переваги CRM-системи для бізнесу:

1. Оперативність обслуговування та проведення різних операцій;
2. Мультизадачність при роботі з потенційними клієнтами;
3. Відсутність помилок у роботі;
4. Зменшення витрат на заробітну плату (потрібно оплачувати тільки тих спеціалістів, котрі будуть слідкувати за роботою системи);
5. Безперервна робота та доступ 24/7;

Із цих переваг виникає ряд типових можливостей сучасних CRM-систем:

1. Швидке отримання усієї необхідної інформації про клієнта;
2. Оперативність у роботі з клієнтами та у виконанні операцій у системі;
3. Збереження документів та організація зручного документообігу;
4. Швидкий та постійний доступ до аналітики;
5. Контроль роботи персоналу, який займається розробкою, підтримує роботу системи, або просто спілкується з клієнтом;
6. Створення певного плану роботи для працівників та робочих підрозділів;
7. Автоматизація усіх послідовних операцій, які виконуються працівниками організації;
8. Система дозволяє виконувати надзвичайно швидкий пошук по контрагентам організації або клієнта: зазвичай усі дані про переговори, угоди, листування та зустрічі можна знайти у системі. Із-за наявності такої інформації підвищується потенційна працездатність працівників, так як система може автоматично

нагадувати їм про необхідність зробити дзвінок, провести зустріч, або підписати той, чи інший договір.

9. CRM-система дозволяє спланувати увесь цикл продажу компанії, за допомогою історії продажів можна зробити якісну аналітику, яка допоможе позбутися явних проблем у подальшому розвитку компанії. Всі необхідні звітні документи з продажів легко знаходяться працівниками та замовниками, можна проводити контроль виконання роботи по факту.
10. За допомогою системи можна завжди мати актуальну інформацію про залишки продукції на складах і виходячи з цього – робити актуальні закупівлі нового товару; також з використанням системи можна легко організовувати швидку доставку до користувача та бачити завантаженість кур’єрів або транспортних компаній.
11. Система з легкістю може допомагати користувачам робити постійні розсилки клієнтам про актуальні акції та спеціальні пропозиції, також можна відслідковувати результативність таких маркетингових розсилок та корегувати свою стратегію відповідно до результатів;
12. Система може містити у собі актуальні шаблони документів компанії, що дозволить користувачам лише заповнювати необхідні поля і відразу друкувати їх. За допомогою взаємодії з системою банку можна отримувати виписки про сплату товарів чи послуг, а також автоматично, без витрати зайвого часу перевіряти сплату;
13. Зазвичай типова CRM-система володіє можливістю підключення до іншої подібної системи;
14. Також така система може отримувати актуальні дані з баз контрагентів компанії.

Як бачимо, можливості сучасних CRM-систем є практично необмеженими і вони постійно розширюються. Щодо складу типової CRM-системи, то найчастіше використовується такий розподіл на процеси:

1. Маркетинг.
2. Обробка заявок.
3. Обробка побажань.
4. Продажі.
5. Сервісне обслуговування.

Також є і окремі складові, які зазвичай поділяють на:

- call-центри — зазвичай сюди включають усі канали взаємодії з клієнтом: телефонні дзвінки, онлайн-чати, різноманітні боти-помічники.
- модулі обробки інформації:

✓ **оперативна функція** — ця функція пов’язана з швидкодією системи, яка зазвичай дозволяє користувачу швидко зареєструватися у системі для того щоб отримати доступ до всієї інформації, яка розміщена у БД;

✓ **аналітична функція** — ця функція дозволяє користувачам системи отримати усі необхідні звіти і аналітику, яка необхідна для покращення роботи з системою; нерідко застосовується аналітика інформації у розрізі;

✓ **кооперативна функція** — ця функція відноситься до функцій безпосередньої взаємодії з клієнтами (опитування, тести), які можуть допомогти розробникам CRM-системи покращити роботу своєї розробки та зробити її більш функціональною, простою та зручною у використанні. Ця функція може реалізуватися за допомогою контакту персоналу з клієнтом або методом автоматичної роботи системи.

1.2 Призначення, структура та задачі які вирішує система

Призначення системи продажу та купівлі рекламних повідомлень в “Telegram” полягає у створенні та розвитку середовища, яке б задовольняло всі потреби клієнтів, которі виступають, як покупцями, так і продавцями рекламних повідомлень в телеграм-каналах.

Основні принципи розроблюваної системи:

- зручність у використанні;
- висока швидкодія;
- захищеність статистики та інформації системи;
- безпечна взаємодія з платіжними системами;
- вирішення проблем, які часто трапляються під час закупівлі реклами у ручному режимі;

Структура альфа-версії системи:

- Система реєстрації та логіну;
- Перелік актуальних категорій для бази телеграм-каналів;
- Каталог каналів для купівлі рекламних повідомлень;
- Аналітика;
- Інтеграція з актуальною платіжною системою;
- API.

Задачі, які повинна вирішувати розроблювана система:

- Відображення каталогу телеграм-каналів;
- Купівля та продаж рекламних повідомлень у цих каналах;
- Надання актуальної детальної статистики для користувачів системи;

1.3 RESTful API, як архітектурний підхід до створення системи

REST – (Representational State Transfer) – підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів. У разі використання цього підходу дані у системі повинні передаватися у вигляді невеликих кількості стандартних форматів (у нашому випадку - json) [2].

Компоненти REST системи (рис.1.1) спілкуються, передаючи один одному представлення ресурсу в форматі, що обирається з оновлюваного набору стандартних форматів даних. Формат обирається динамічно відповідно до бажань компонента-клієнта і можливостей сервера. Чи представлення має той самий формат, що й сам ресурс, чи є результатом якогось перетворення — це деталь реалізації, яка ховається за інтерфейсом.

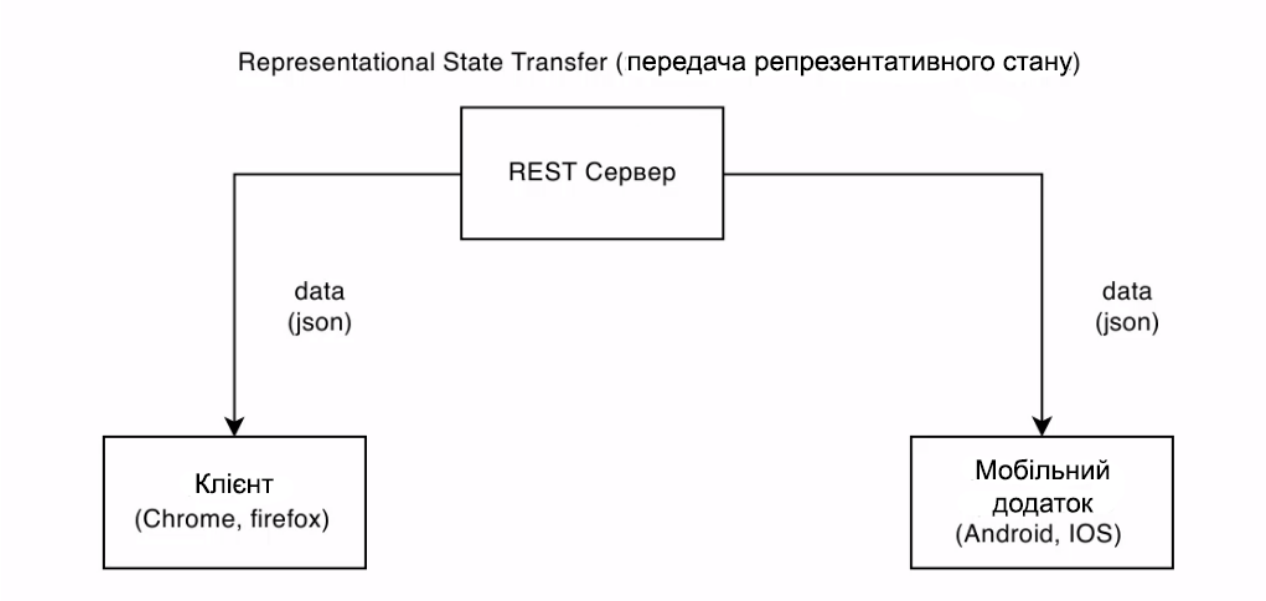


Рисунок 1.1. Принцип роботи RESTful API

Якщо розглядати стандартну на даний час архітектуру створення web-сайтів, зображену на рис.1.2, то можна зрозуміти, що використання архітектури REST для своєї CRM-системи дає наступні переваги:

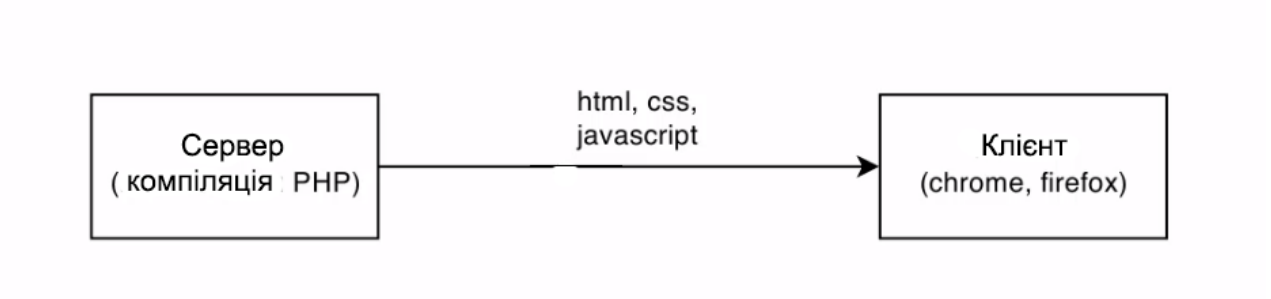


Рисунок 1.2. Стандартна архітектура web-сайтів

- Можливість передачі даних у форматі json.
- Використання меншої кількості трафіку від клієнта та сервера;
- Можливість створення API та використання його для роботи з зовнішніми платіжними системами;
- Можливість розробки телеграм-бота, який буде взаємодіяти з системою;
- Швидка робота з існуючим API “Telegram”;
- Перспектива розробки мобільного додатку без внесення змін у функціональну структуру системи.

1.3.1 Елементи даних в REST

Ресурс — це ключовий елемент даних в REST. Ресурсом може бути що завгодно що можна назвати: якийсь документ (наприклад зображення), динамічне значення (наприклад погода у Львові), щось з реального світу (наприклад працівник компанії). Але якщо точніше, то ресурс R — це функція приналежності $M_R(t)$ що відображає моменти в часі на множину однотипних сутностей чи значень. Множина може бути порожньою, тобто REST дозволяє посилання на якийсь об'єкт якого ще не існує

Ресурс може бути динамічним, наприклад каталог телеграм-каналів час від часу буде оновлювати свій вміст, а може бути статичним, і після появи ніколи не змінювати свого значення, наприклад, допоміжні статичні сторінки.

У REST такі два ресурси вважаються різними, хоча в певний момент часу вони можуть вказувати на одну й ту ж сутність. Єдине що важливо — семантика відображення імені ресурсу на його вміст.

Ідентифікатор ресурсу

Для того, щоб посилатись на ресурси, використовуються ідентифікатори ресурсів. Компонент, який надав ресурсу ідентифікатор і дозволяє звертатись до нього за цим ідентифікатором, відповідає за збереження функції приналежності незмінною [3]. Якість ідентифікатора залежить від якості компонента, який цей ідентифікатор надає, тому деякі ідентифікатори стають «мертвими посиланнями», коли інформацію переміщують або знищують. Приклади ідентифікаторів ресурсу: URL, URN.

Представлення (англ. representation) — це послідовність байтів та метадані представлення, для того щоб описати ці байти. Часто, представлення називають документом, файлом, повідомленням HTTP тощо.

Приклади представлення: фотографія JPEG, документ HTML. Приклад метаданих представлення — тип медіа, час останньої зміни.

Контрольні дані в представленні описують ціль повідомлення між компонентами, наприклад прохання про дію (створити, змінити видалити ресурс), або значення відповіді (наприклад поточний стан ресурсу, чи значення якогось іншого ресурсу, наприклад опис помилки).

Також, контрольні дані, включені в запити чи відповіді, можуть керувати поведінкою кеша. Прикладом таких контрольних даних можуть бути заголовки HTTP if-modified-since та cache-control.

Формат даних представлення називають типом медіа (англ. media type). Одні типи медіа краще підходять для автоматичної обробки, інші — для того щоб бути показаними користувачу. Композитні типи медіа можуть використовуватись для того, аби поєднати кілька видів представлення в одному.

Від формату даних дуже залежить латентність застосунку, яку сприймає користувач. Наприклад, браузер може почати показувати сторінку ще до того, як завантажиться ввесь HTML, це збільшує видиму швидкість роботи.

1.3.2 Методи REST

Ресурс у разі використання методу REST може бути будь-яким, але дії, які можна виконувати над таким ресурсом, визначаються повідомленнями, які базуються на стандартному протоколі. В системі Інтернету цим протоколом можна назвати HTTP. Так як основною з особливостей розроблюваної системи є її захищеність, то треба говорити про HTTPS.

Стандарт HTTPS визначає такі типи повідомлення.

Найчастіше використовують 4 з них:

GET – отримання ресурсу;

DELETE – видалення ресурсу;

POST – знищення сторогу ресурсу та вставлення на його місце нового за допомогою переданого представлення;

PUT – заміна стану поточного ресурсу на інший, який може бути переданим разом з новим представленням;

У розроблюваній CRM-системі також буде використовуватися і п’ятий метод.

PATCH – зміна лише частини даного ресурсу. Такий метод використовується для зменшення кількості інформації, котру необхідно передавати.

Методи *GET, PUT* та *DELETE* — ідемпотентні, це означає, що результат не залежить від кількості виконаних операцій. Звісно, при виконанні таких операцій декілька разів код відповіді буде змінюватися, але результат буде тим самим. Так, якщо ви видалите ресурс один раз, або 10 разів поспіль – його не буде, що після одного, що після 10 разів. Метод *POST* не є ідемпотентним, це означає, що кожне відправлення повідомлення таким методом призведе до отримання всіх цих повідомлень, а не одного схожого на перше.

1.4 Опис проблеми, її актуальність та постановка задачі

Виходячи з описаної раніше в даній роботі проблематики можна сказати про те, що купівля та продаж рекламних повідомлень у месенджері “Telegram”

у ручному режимі має дуже багато мінусів, а деякі з них можуть викликати додаткові нерентабельні витрати бізнесу.

Тим більше, купівля та продаж таких повідомлень у ручному режимі змушує власників бізнесу та каналів використовувати додаткові людські ресурси, витратами на які при користуванні системою можна просто знехтувати.

На момент квітня 2018 року об’єм рекламного ринку СНГ сегменту месенджеру “Telegram” складав 24 млн. грн. [8] і цей показник поступово зростає. Аудиторія російських та українських користувачів у цей період року складала близько 15 млн. чоловік. Офіційний реліз рекламних компаній командою месенджеру призначений на кінець 2019 року.

Я вважаю, що дана проблема є достатньо актуальною, якщо брати до уваги той факт, що на сьогоднішній день кількість аудиторії у месенджері постійно збільшується. Разом з цим збільшується і кількість актуальних телеграм-каналів для розміщення реклами. А власники бізнесів кожен день, натикаючись на новий цікавий канал, зі схожою із їх діяльністю тематикою бажають розповісти про свій продукт або послугу та перетворити купівлю рекламного повідомлення у додатковий прибуток для себе.

Об’єктом дослідження є процес проектування та створення CRM-системи купівлі та продажу рекламних повідомлень, її взаємодії з API “Telegram” та платіжних систем, підготовка системи до підключення до телеграм-боту (буде розроблений пізніше) для швидкої роботи з системою без використання комп’ютера та браузера.

Внаслідок проведеного аналізу області, було поставлено наступні задачі:

1. Провести аналітичний огляд API месенджеру “Telegram”.
2. Описати архітектуру додатка.
3. Обрати програмні рішення та середовище, яке допоможе максимально розгорнуто виконати поставлену задачу.
4. Розробити back-end частину системи.
5. Розробити front-end частину системи.

6. Розробити API.
7. Розробити статистичну та аналітичну частину системи, яка дозволить користувачам отримувати якісні дані щодо своїх закупівель.
8. Підготувати систему до підключення телеграм-боту, який дозволить користуватися системою без використання комп’ютера та браузера.

Висновки до розділу

В даному розділі була розглянута проблематика купівлі та продажу рекламних повідомлень у месенджері “Telegram” у ручному режимі. Було описано призначення, структуру та задачі, які буде вирішувати створювана CRM-система.

Метою даної роботи є проектування та створення системи купівлі та продажу рекламних повідомлень. На основі аналізу створюваної системи було поставлено десять задач.

Об’єктом даного дослідження є процес проектування та створення CRM-системи купівлі та продажу рекламних повідомлень, її взаємодії з API “Telegram” та платіжної системи, підготовка системи до підключення телеграм-боту (буде розроблений пізніше) для швидкої роботи з системою без використання комп’ютера та браузера.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ КУПІВЛІ ТА ПРОДАЖУ РЕКЛАМНИХ ПОВІДОМЛЕНЬ

2.1 Проектування системи

2.1.1 Архітектура додатка

У попередньому розділі даного дипломного проекту було узгоджено, що для розробки даної системи буде використано архітектурний підхід RESTful api. У зв’язку з цим архітектура такої системи буде виглядати так, як зображено у додатку А.

На схемі зображено 3 найголовніших сутності проекту: база даних (БД), сервер (backend) та клієнт (frontend).

Зі схеми видно, що сервер проекту буде використовувати фреймворк Express.js, а клієнт Angular 6; у якості бази даних MongoDB.

Клієнт буде контактувати з сервером через REST api, але знаходитися вони будуть окремо один від одного. За потреби клієнт буде відправляти запити до сервера, а той відповідатиме на них даними, які було запитано з використанням інформації, яка у той момент буде знаходитись у БД.

2.1.2 Обґрунтування вибору основного набору технологій для розробки

Мова програмування

В якості мови програмування для розробки скріпту був обраний JavaScript з наступних причин:

- Поширеність і документація.
- JSON - нативний формат JavaScript і використовується для обміну даними між клієнтською частиною та API серверної частини.
- Використання однієї мови JavaScript і на клієнті, і на сервері, дозволить спростити і прискорити процес розробки, використовувати одні і ті ж бібліотеки.

JavaScript - прототипна-орієнтована сценарна мова програмування. Іншими словами вона є обмеженою об'єктно-орієнтованою, коли немає поняття класу, успадкування. В цьому випадку принципи об'єктно-орієнтованого програмування імітуються, наприклад спадкування - це клонування основного

об'єкта і додавання до нього необхідної поведінки або властивостей. Ставлення будується як "прототип - спадкоємець" без обов'язкового збереження структури прототипу. Таким чином, JavaScript дозволяє імітувати об'єктно-орієнтований підхід до розробки.

Програмна платформа

Для того щоб виконати код JavaScript, він повинен бути інтерпретований в машинний код. Для чого був зроблений вибір на користь програмної платформи Node.js (далі Nodejs), як середовища виконання JavaScript, так як:

- Працює згідно асинхронної моделі.
- Перспективний і добре документований.
- Має в наявності часто використовувані бібліотеки і модулі, написані на JavaScript.

Nodejs - платформа для створення додатків, орієнтованих на високу інтенсивність введення-виведення і не високу інтенсивність обчислень. Дозволяє розробляти на мові JavaScript і надає можливість взаємодіяти з пристроями введення-виведення через API Nodejs. Nodejs використовує розроблений транслятор від Google, який дозволяє код JavaScript безпосередньо перетворити в асемблер цільового процесора, що забезпечує високу продуктивність [4].

Модулі, розроблені під Nodejs здатні створювати нові процеси. Наприклад, Nodejs має модуль веб-сервера для обробки HTTP-запитів. Наявність популярних модулів fs і request дозволить розробнику легко роботи запити до API сторонніх ресурсів і працювати з файлами за принципом «запит-відповідь» (Додаток Б).

Ці особливості платформи дають змогу виконати даний дипломний проект у повному обсязі.

Основною особливістю Nodejs, крім повної асинхронності, є його однопоточна модель. Іншими словами, всі операції виконуються в одному і тому ж потоці ОС, навіть якщо у сервера тисяча одночасних користувачів. Правда, є створення дочірніх процесів і низькорівневе управління виконанням

скріптів (завантаження, компіляція, робота з асемблерним кодом, виконання). Для реалізації багатопоточності і задіяння всіх ядер сучасних процесорів рекомендується просто завантажувати декілька копій програми, але можна взяти на озброєння WebWorker зі стандарту HTML5 і розподілити роботу програми по кількох дочірнім процесам.

Веб-додаток робить корисну роботу дуже швидко, а більшу частину часу просто чекає чогось (даних від бази, від memcached'a або NoSQL-бази), або просто тримає в пам'яті відкриті з'єднання для Comet'a, тому в одному потоці можна обробити і десяток тисяч, не вдаючись до кластеризації.

Другою особливістю архітектури Node є наповненість. Майже кожна операція має callback, генерує подію, а користувачеві доступний об'єкт EventEmiter, через який можна буквально одним рядком генерувати свої події (це нескладно, адже подія - це просто рядок з назвою, а також список параметрів, які передаються в обробник).

Сам по собі Node побудований навколо EventLoop - глобального циклу обробки подій, який на кожному тику перевіряє, чи готові дані для будь-якого з визначених користувачем callback. Якщо дані є, починається виконання коду. Якщо не залишилося більше коду - очікуємо наступного виклику. Цикл виконується поза JS, а в самому ядрі, написаному на C, внаслідок чого це відбувається дуже і дуже швидко (близько сотень тисяч разів в секунду). Це щось типу нескінченного циклу. На додаток до цього в сервер вбудований дуже ефективний збирач сміття (GC), тому навіть тисячі підключень не викликають переповнення пам'яті і падіння сервера. Nodejs має вбудовану рідну систему роботи з подіями.

2.1.3 ODM “Mongoose”

Mongoose - це ODM (* Object Document Mapper - об'єктно-документний відображувач). Це означає, що Mongoose дозволяє вам визначати об'єкти зі строго-типизированной схемою, яка відповідає документу MongoDB [5].

Mongoose надає величезний набір функціональних можливостей для створення і роботи зі схемами. На даний момент Mongoose містить вісім

SchemaTypes (*типи даних схеми), які можуть мати властивість, яке зберігається в MongoDB. Це такі типи даних:

- String;
- Number;
- Date;
- Buffer;
- Boolean;
- Mixed;
- ObjectId (* унікальний ідентифікатор об’єкта, первинний ключ, _id);
- Array.

2.1.4 Модуль регістрації/авторизації “Passport”

Якщо б backend частина нашої системи була б написана на мові програмування PHP, то при кожному переході по елементам сторінки у браузері користувача оновлювалася б сесія.

Але маючи на увазі те, що буде використано архітектурний принцип REST, зрозуміло, що оновлень сторінки у цьому разі не буде. Усі необхідні дані і елементи будуть динамічно підгружатися за допомогою JS. Тому немає сенсу використовувати сессії для авторизації користувачів.

На рис.2.2 зображено спрощену схему авторизації з використанням модуля “Passport” і Passport JWT. Процес авторизації виглядає наступним чином:

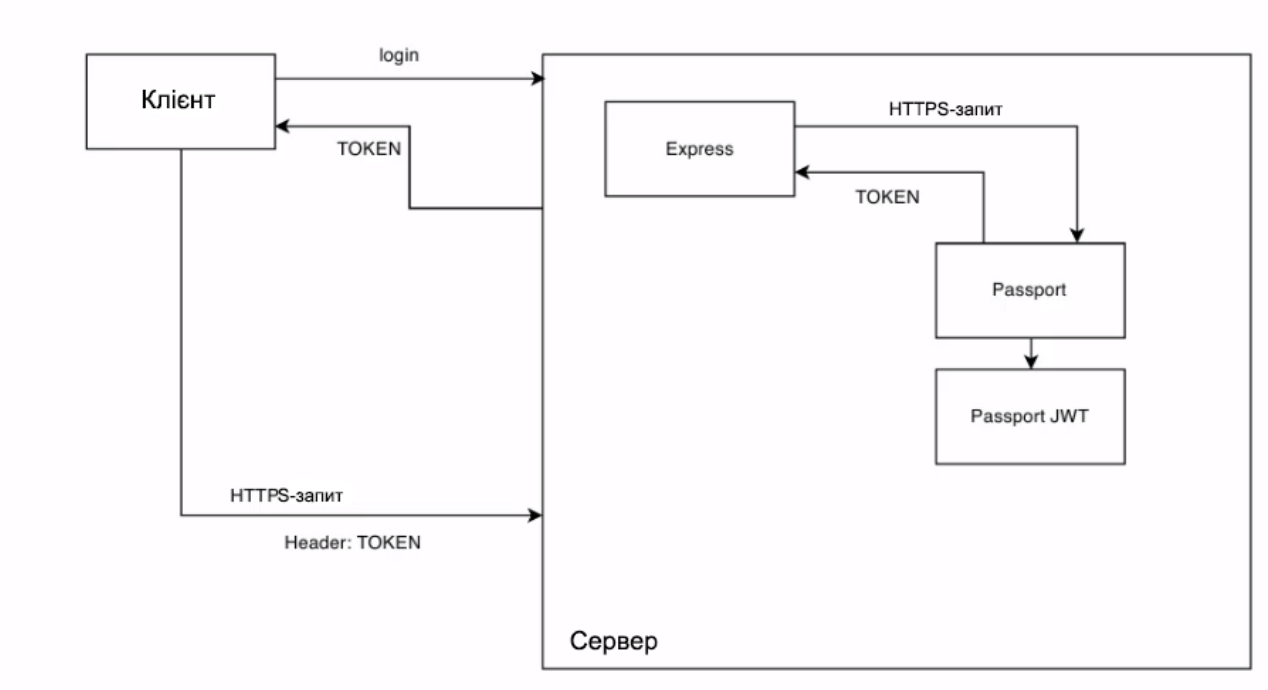


Рисунок 2.2. Спрощена схема авторизації у системі

Є два основні блоки клієнт і сервер (великий блок справа).

1. Клієнт надсилає запит до сервера Express.js.
2. Сервер обробляє цей запит і відправляє його до Passport.js.
3. Passport JWT перевіряє коректність логіна/пароля користувача через БД Mongo.
4. У разі якщо дані для логіну коректні – генерується спеціальний токен, який відправляється на клієнт через сервер.
5. Клієнт авторизується у системі.

Важливо розуміти, що у токені, який буде відправлятися сервером на клієнт знаходиться інформація, по якій можна зрозуміти, через який час цей токен стане недійсним.

Це гарний аналог сессій PHP, який допоможе системі бути більш захищеною.

2.1.5 Frontend частина, Angular 6

Angular 6 RC 2 – не остання версія популярного фреймворку, який буде використаний у якості frontend частини проетку, але він має певні особливості, які значно спростять процес розробки і допоможуть у ньому.

Оновлення ng - тепер у нас є команда автоматичного оновлення залежностей Angular додатків CLI. Усі наші @ angular / * залежності будуть оновлені до останньої стабільної версії, яка включає в себе всі основні пакети в наших залежностях і devDependencies, такі як rxjs, zone, typescript і т.п., а також сам CLI. Ця функція допоможе заощадити час при переході на наступну стабільну версію, оскільки розробнику не потрібно вручну ідентифікувати і оновлювати версії залежностей тимчасових вузлів [6].

Основні відмінності між Angular та AngularJS

- У новій версії Angular було додано CLI – систему, за допомогою якої розробник може розпочати створення нової фронт-енд частини додатка за допомогою лише однієї команди ng new [app name].
- Нова версія фреймворку застосовує ієрархію компонентів, повністю ігноруючи концепцію «області видимості» або контролерів.
- Для білдингу властивостей тепер застосовують "[ ]", а для біндінгу даних івентів - "( )".
- Тепер основний функціонал розроблюваних додатків міститься у модулях.
- Angular рекомендує та застосовує розроблену Microsoft мову — TypeScript, що містить:
    - Класи, а отже ООП;
    - Систему типізації;
- TypeScript — охоплююча множина ECMAScript 6 (ES6), і вона є зворотно сумісною зі стандартом ECMAScript 5 (тобто JavaScript). Angular також має такі ES6-можливості, як:
    - Функції, які самостійно запускаються (анонімні);
    - Ітератори;
    - Цикли типу For/Of;
    - Python-подібні генератори.

2.2. Моделі додатка

Так як у ході виконання даного дипломного проекту буде розроблятися складна CRM-система з достатньо великим функціоналом, треба виділити основні сутності розроблюваної системи, а також зазначити поля, котрі будуть використовуватися для їх опису та відображення у системі.

1. Категорії
    - Назва.
    - Зображення.
    - Користувач (адміністратор).

Сутність «категорії» буде використовуватися для сортування каналів у каталогу за смаком користувача, для відображення на даному етапі розробки необхідна лише назва.

2. Канал
    - Ім’я.
    - Категорія.
    - Кількість підписників.
    - Зображення.
    - Опис.
    - Користувач, котрий створив канал.
    - Ціна.
    - Актуальний час розміщення.
    - Актуальний час розміщення 2.
    - Актуальний час розміщення 3.
    - Варіант розміщення 1.
    - Варіант розміщення 2.
    - Запропонований користувачем варіант розміщення.
    -

Сутність «канал» буде використовуватися для відображення інформації про канали та безпосередньої покупки рекламного посту.

3. Користувач
    - E-mail.
    - Пароль.
    - Баланс.
    - Аватар.
    - Канали.
    - Рекламні повідомлення.
    - Налаштування.
    - Телеграм-акаунт.

Сутність «користувач» буде використовуватися для реєстрації та логіну користувачів у системі, а також для іншого функціоналу, який буде описано нижче.

4. Замовлення
    - Дата.
    - Дата створення замовлення.
    - Обраний користувачем актуальний час розміщення.
    - Канал.
    - Номер замовлення.
    - Ім’я каналу.
    - Ціна.
    - Рекламне зображення.
    - Замовник (користувач).
    - Тривалість повідомлення.
    - Автовидалення.
    - Автопублікація.

Сутність «замовлення» буде використовуватися для безпосередньої купівлі та продажу рекламних повідомлень та їх публікації згідно заданих параметрів.

2.3. API-запити

Так як у системі було вирішено використовувати REST архітектуру, то необхідно реалізувати заданий перелік адресацій для API для того щоб користувач міг отримати повний перелік даних, які йому потрібні. Перелік указано нижче, в дужках зображено тип запиту, який потрібно буде використовувати.

1. Авторизація
    - /api/login/ (POST)
    - /api/register/ (POST)
2. Каталог
    - /api/category/ (GET)
    - /api/category/:id/ (GET), (DELETE), (PATCH)
    - /api/category/ (POST)
    - /api/channel/:category (GET)
    - /api/channel/ (POST)
    - /api/channel/:id (PATCH), (DELETE)
    - /api/channel/:sub (PATCH), (GET)
    - /api/channel/:price (POST), (GET), (PATCH)
    - /api/channel/:desc (POST), (GET), (PATCH), (DELETE)
    - /api/channel/:time (POST), (GET), (PATCH)
3. Замовлення
    - /api/order (GET)
    - /api/order (POST)
    - /api/order/:id (PATCH), (DELETE)
    - /api/order/:date/ (POST), (GET), (PATCH), (DELETE)
    - /api/order/:channel/ (POST), (GET)
    - /api/order/:price/ (GET)
    - /api/order/:user/ (POST)
    - /api/order/:duration/ (GET), (POST)
    - /api/order/:deletion/ (POST)

- /api/order/:autopublish/ (POST)

Висновки до розділу

В даному розділі було проведено проектування функціоналу системи купівлі та продажу рекламних повідомлень. А саме – була обрана архітектура додатка та технологічний стек, було виконано проектування моделей та API-запитів.

Нажаль базові засоби мови програмування javascript не дозволяють виконати поставлене завдання у повному обсязі.

Надалі буде використовуватися Node.js, Express.js та Angular 7, як основна програмна платформа для вирішення поставленого завдання.

РОЗДІЛ 3. РОЗРОБКА СИСТЕМИ КУПІВЛІ ТА ПРОДАЖУ РЕКЛАМНИХ ПОВІДОМЛЕНЬ ТА ІНТЕРФЕЙСУ КОРИСТУВАЧА

3.1 MEANSTACK

Стек MEAN, в якому поєднуються кілька вільних фреймворків JavaScript. Обговоримо, як вони працюють і як створити з їхньою допомогою односторінковий додаток. Технології JavaScript в даному випадку задіюються не тільки на фронтенді, але і на бекенд [9].

Як зрозуміло з назви, MEAN - це абревіатура, під якою об'єднані три фреймворка JavaScript і документоорієнтована NoSQL-база даних.

Отже,

M - це MongoDB

MongoDB - це вільна документоорієнтована база даних, організована таким чином, щоб забезпечити як масштабованість, так і гнучкість при розробці. Дані в MongoDB записуються не в таблицях і шпальтах, як в реляційної базі даних. Замість цього в MongoDB зберігаються JSON-подібні документи з динамічними схемами.

E - це ExpressJS

Express.js - це фреймворк для сервера додатків Node.js, призначений для створення односторінкових, багатосторінкових і гібридних веб-додатків. Де-факто він є стандартним серверним фреймворком для node.js.

A - це AngularJS

AngularJS - це структурний фреймворк для динамічних веб-додатків. У ньому можна використовувати HTML в якості мови шаблонів, а також розширювати синтаксис HTML для чіткого і лаконічного опису компонентів вашого застосунку. Зв'язування даних і впровадження залежностей в Angular дозволяють позбутися від маси коду, який довелося б писати в іншому випадку.

N - це NodeJS

Node.js - це вільне кроссплатформене виконуюче середовище для розробки серверних веб-додатків. Додатки Node.js пишуться на мові JavaScript і можуть запускатися в виконавчому середовищі Node.js в операційних системах

OS X, Microsoft Windows, Linux, FreeBSD, NonStop, IBM AIX, IBM System z і IBM.

На рис.3.1 зображено стандартну взаємодію між компонентами стеку MEAN.

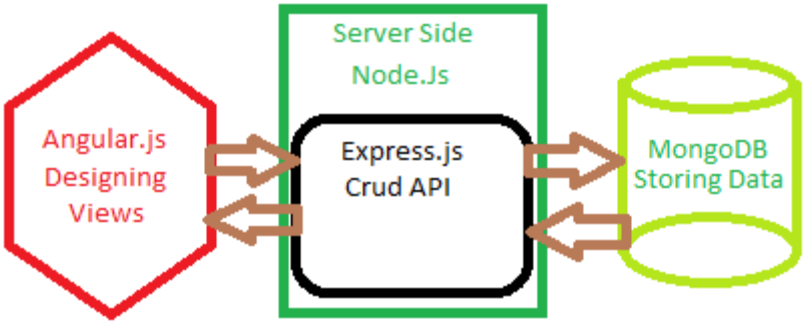


Рисунок 3.1. Взаємодія між компонентами стеку MEAN

Термін «стек MEAN» означає набір технологій на базі JavaScript, призначених для розробки веб-додатків. MEAN - це скорочення від MongoDB, ExpressJS, AngularJS і Node.js. На рівні клієнта, сервера і бази даних весь стек MEAN написаний на JavaScript.

Як вони взаємодіють? Angular використовується для розробки клієнтської частини. Я буду розробляти уявлення за допомогою Angular 7 так, щоб всі вони відображалися на одній сторінці. На серверній стороні я застосую Node.js, тобто, скористаюся фреймворком Express.js. За допомогою Express я напишу API для комунікації з базою даних. Нарешті, я скористаюся MongoDB для зберігання даних. [10]

3.2 MongoDB або MySQL?

Якщо подивитися такий відомий ресурс, як DB-Engines Ranking (рис.3.2.), то можна побачити, що в перебігу багатьох років популярність open source баз даних зростає, а комерційних - поступово знижується.

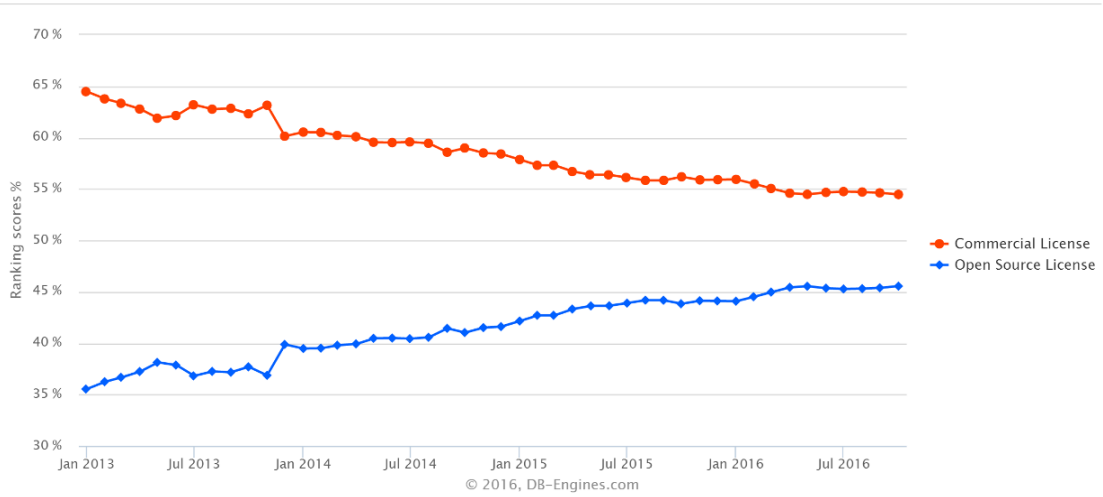


Рисунок 3.2. Тренд популяризації open source БД

Ми бачимо, що для багатьох додатків використовують кілька баз даних для того, щоб задіяти їх сильні сторони. Жодна база даних не оптимізована для всіх кейсів. Навіть якщо це PostgreSQL.

З одного боку, це хороший вибір, з іншого - потрібно намагатися знайти баланс, тому що чим у нас більше різних технологій, тим складніше їх підтримувати, особливо, якщо компанія не дуже велика.

Часто що бачимо, що люди приходять на такі конференції, слухають Facebook або «Яндекс» і кажуть: «Ух ти! Скільки ось люди роблять цікавого. У них технологій різних використовується штук 20, і ще штук 10 вони написали самі». А потім вони той же самий підхід намагаються використовувати в своєму стартапі з 10 осіб, що працює, зрозуміло, не дуже добре. Це як раз той випадок, де розмір має значення.

Дуже часто ми бачимо, що використовується основне операційне сховище і якісь додаткові сервіси. Наприклад, для кешування або повнотекстового пошуку.

Інший підхід до архітектури з використанням різних баз даних - це мікросервіси, у кожного з яких може бути своя база даних, яка краще оптимізована для задач саме цього сервісу. Як приклад: основне сховище може бути на MySQL, Redis і Memcache - для кешування, Elastic Search або рідний Sphinx - для пошуку. І щось на зразок Kafka - щоб передавати дані в систему аналітики, яка часто робилася на чомусь схожому на Hadoop.

Якщо ми говоримо про основне операційне сховище, напевно, у нас є два варіанти. З одного боку, ми можемо вибрати реляційні бази даних, з мовою SQL. З іншого боку – щось з нереляційних, а далі вже дивитися на підвиди, які доступні в даному випадку.

Якщо говорити про NoSQL-моделі даних, то їх теж досить багато. Найбільш типові - це або key value, або document, або wide column бази даних. Приклади: Memcache, MongoDB, Cassandra, відповідно.

Чому в даному випадку я порівнюємо саме MySQL і MongoDB? Насправді причин декілька. Якщо подивитися на Ranking баз даних, то ми бачимо, що MySQL, згідно з цим рейтингом, - найбільш популярна реляційна база даних, а MongoDB - найбільш популярна нереляційних база даних. Тому їх розумно порівнювати.

Ще одна причина: обидві технології з самого початку орієнтовані на розробників простих додатків. Для тих людей, для яких PostgreSQL - це занадто складно.

Компанія MongoDB спочатку дуже активно фокусувалася на користувачах MySQL. Тому дуже часто у людей є досвід використання і вибір між цими двома технологіями.

Якщо говорити про MySQL - це перевірена технологія. Зрозуміло, що MySQL використовується великими компаніями більш 15 років. Так як він використовує стандарт SQL, є можливість досить простої міграції на інші SQL-бази даних, якщо захочеться. Є можливість транзакцій. Підтримуються складні запити, включаючи аналітику. І так далі.

З точки зору MongoDB, тут перевага те, що у нас гнучкий JSON-формат документів. Для деяких завдань і якимось розробникам це зручніше, ніж мучитися з додаванням колонок в SQL-базах даних. Не потрібно вчити SQL - для деяких це складно. Прості запити рідше створюють проблеми. Якщо подивитися на проблеми продуктивності, в основному вони виникають, коли люди пишуть складні запити з JOIN в купу таблиць і GROUP BY. Якщо такої

функціональності в системі немає, то створити складний запит виходить складніше.

У MongoDB вбудована досить проста масштабованість з використанням технології шардінга. Складні запити якщо виникають, ми їх зазвичай вирішуємо на стороні додатку. Тобто, якщо нам потрібно зробити щось на зразок JOIN, ми можемо вибрати дані, потім вибрати дані по посиланнях і потім їх обробити на стороні додатку. Для людей, які знають мову SQL, це виглядає якось ненатурально. Але насправді для багатьох розробка application-серверів таке куди простіше, ніж розбиратися з JOIN.

Що таке масштабованість в даному контексті? Те, наскільки легко нам взяти наш маленький додаток і масштабувати його на багато мільйонів, може бути, навіть на мільярди користувачів.

Масштабованість буває різна. Вона буває середня, в рамках однієї машини, коли ми хочемо підтримувати додатки середнього розміру, або масштабованість на кластері, коли у нас додатки вже дуже великі, коли зрозуміло, що навіть одна найпотужніша машина не впорається.

Також має сенс говорити про те, що ми масштабуємо - читання, запис або обсяг даних. У різних додатках їх пріоритети можуть відрізнятися, але в цілому, якщо додаток дуже великий, зазвичай вони повинні працювати з усіма з цих речей.

В MySQL в нових версіях вельми гарна масштабованість в рамках одного вузла для LTP-навантажень. Якщо у нас маленькі транзакції, є якесь залізо, в якому 64 процесора, то масштабується досить добре. Аналітика або складні запити масштабуються погано, тому що MySQL може використовувати для одного запиту тільки один потік, що погано.

Традиційно читання в MySQL масштабується з реплікацією, запис і розмір даних - через шардінг. Якщо дивитися на все більші компанії - Facebook, Twitter - вони всі використовують шардінг. Традиційно шардінг в MySQL використовується вручну. Є деякі фреймворки для цього. Наприклад, Vitess - це фреймворк, який Google використовує для scaling сервісу YouTube, вони його

випустили в open source. До цього був framework Jetpants. Стандартного рішення для шардінга MySQL не пропонує, часто перехід на шардінг вимагає уваги від розробників.

У MongoDB фокус спочатку був в масштабованості на багатьох вузлах. Навіть у випадках з маленьким додатком багатьом рекомендується використовувати шардінг з самого початку. Може, всього через пару replica set, ви будете рости разом зі своїм додатком.

У шардінзі MongoDB є деякі обмеження: не всі оператори з ним працюють, наприклад, є isolated-варіант для забезпечення консистентності. Вона не працює якщо використовувати шардінг. Але при цьому багато основних операцій добре працюють з шардінгом, тому людям дозволяється scale'ти додатки значно краще. На мій погляд, шардінг і взагалі реплікація в MongoDB зроблені куди краще, ніж MySQL, значно простіше у використанні для користувача.

Адміністрування - це все ті речі, про які не думають розробники. Принаймні в першу чергу. Адміністрування - це те, що нам додаток доведеться бекапити, оновлювати версії, моніторити, відновлювати при збоях і так далі.

MySQL досить гнучкий, у нього є багато різних підходів. Є хороші open source реалізації всього, але ця безліч варіантів породжує складність.

У MongoDB все більше орієнтовано на те, що воно працює якимось одним стандартним чином - є мінімізація адміністрування. Але зрозуміло, що це відбувається при втраті гнучкості. Ком'юніті open source рішень для MongoDB значно менше. Багато речей в MongoDB з точки зору рекомендацій досить жорстко прив'язані до Ops Manager - комерційної розробки MongoDB.

Типовий приклад, де використовується MySQL-рішення - це сайт електронної комерції. Коли у нас йде питання про гроші, часто ми хочемо повноцінні транзакції і консистентність. Для таких речей добре підходить реляційна структура, яка була опрацьована, і commerce на реляційних базах даних вже робиться багато десятиліть. Так що можна взяти один з готових підходів до структури даних і використовувати його.

Зазвичай з точки зору e-commerce обсяг даних у нас не такий великий, так що навіть досить великі магазини можуть довго працювати без шардінга. Додатки у нас постійно розробляються і вдосконалюється протягом багатьох років. І у цієї програми багато компонентів, які працюють з одними і тими ж даними: хтось розраховує, де ціни поміняти, хтось ще щось робить.

MongoDB часто задіюється як бекенд великих онлайн-ігор. Electronic Arts для дуже багатьох ігор використовує MongoDB. Чому? Тому що масштабованість важлива. Якщо якась гра добре вистрілить, її доводиться масштабувати значно більше, ніж передбачалося.

З іншого боку, якщо не вистрілить, нам хотілося б, щоб інфраструктуру можна було б зменшити. У багатьох іграх це йде так: ми запустили гру, у неї є якийсь пік, доводиться робити великий кластер. Потім гра вже виходить з популярності, для неї бекенд потрібно стискати, зберігати і використовувати. В даному випадку є один додаток (гра), база даних, з одного боку, нескладна, з іншого - сильно прив'язана до додатка, в якому зберігаються всі важливі для гри параметри.

Часто консистентність бази даних на рівні об'єктів тут достатня, тому що багато питань консистентості вирішуються на рівні додатку. Наприклад, дані одного гравця зберігає тільки один application service [11].

3.3 Розробка backend частини

3.3.1 Створення серверу

Express.js – програмний каркас розробки веб-застосунків для Node.js. Реалізований для створення веб-застосунків та API.

Перед тим як почати створення необхідного для реалізації системи серверу потрібно встановити пакетний менеджер npm, або оновити його до останньої версії.

Після оновлення або встановлення npm треба створити файл package.json, який буде відповідати за усі залежності проекту. Для генерації файлу використовуємо команду npm init та заповнюємо усі подальші запитання консолі.

В корені проекту створюємо файл з назвою index.js. Для встановлення Express.js та включення його до проекту слід використати команду npm install express.

Для підключення модулю Express.js до проекту використовуємо наступну структуру коду:

```
const express = require('express');
const app = express();
```

Далі переходимо до створення простого серверу, який буде запускатися на вказаному користувачем порту. Для цього використовуємо наступну команду:

```
app.listen(5000, () => console.log('Server start');
```

Така команда запускає сервер на 5000 порту, але для того, щоб перевірити його працездатність необхідно задати дефолтний роут. Виконуємо це наступним чином:

```
app.get('/',  (req, res) => {
        res.status(200).json({
                message: 'Working'});
});
```

Відтепер можна перевірити роботу серверу через браузер (рис.3.1). Для цього потрібно запустити файл на виконання за допомогою команди node index.js, відкрити браузер та перейти на localhost:5000.

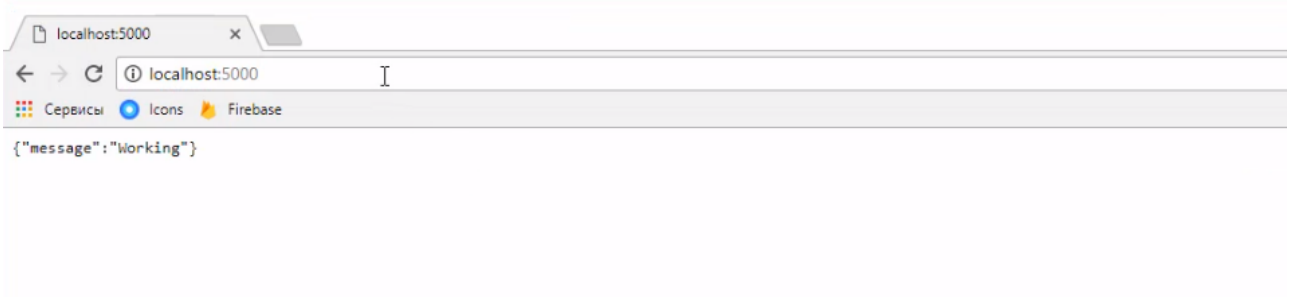


Рисунок 3.3. Демонстрація роботи серверу

Як бачимо, сервер відповів з кодом 200 та вивів задане повідомлення у json-форматі.

3.3.2 Створення структури додатка

Для того щоб почати будувати структуру додатку потрібно створити у корені проекту файл з назвою app.js. У ньому буде знаходитися весь код додатку.

Файл index.js віднині буде віповідати тільки за запуск серверу.

```
const express = require('express');
const app = express();


module.exports = app;
```

Так буде виглядати вміст файлу app.js, а для його підключення у інші файли проекту треба використовувати конструкцію require, при цьому треба пам'ятати, що два цих файла повинні знаходитися на одинаковому рівні вкладеності. У іншному разі потрібно використовувати відносний шлях до файлу app.js.

Далі потрібно підготувати проект до роботи, треба створити автоматизований запуск серверу разом із запуском проекту. Для цього можна використовувати спеціальну директиву scripts, яка знаходиться у раніше створенному файлі package.json.

На даний момент вміст цього файлу виглядає наступним чином:

```
{
    "name": "telegram-crm",
    "version": "1.0.0",
    "description": "crm system for telegram ads" ,
    "main": "index.js",
    "scripts": {
        "test": "echo \"Error: no test specified" && exit 1"
    },
    "keywords": [
```

```
        “crm”,
        “telegram”,
        “ad”
    ]
```

Як видно із вмісту файлу, скріпти запуску додатку вимкнені за замовчуванням. Для того щоб увести їх в експлуатацію необхідно замінити строку “test”: “echo \”Error: no test specified” && exit 1”.

Замінюємо стандартну команду тест на команду start, яка буде виглядати наступним чином – “start”: “node index”. Тепер у ході використання команди npm run start автоматично буде виконуватися і запуск серверу, але цього недостатньо для повноцінної розробки проекту.

Так, при оновленні тих або інших скріптів проекту або їх частин користувач повинен буде кожен раз перезапускати і заново білдити проект для того щоб їх побачити. Це не є рентабельним з точки зору затрачуваного часу. Тому додатково для роботи необхідно встановити пакет nodemon.

Це можна зробити за допомогою стандартного пакетного менеджера npm.

Nodemon – додаткова утиліта, котра буде моніторити вихідний код на будь-які зміни внесені користувачем та у разі виявлення таких – автоматично рестартувати сервер для того щоб користувач міг побачити їх.

Особливості пакету:

- автоматичний рестарт додатку;
- виявляє дефолтне розширення файлу для моніторингу;
- підтримка node.js;
- ігнорування змін у службових файлах або директоріях;
- працює з серверними додатками або одноразовими утилітами;
- пакет з відкритим вихідним кодом.

У файлі package.json в розділі скріптів додаємо команду “server”: “nodemon index”.

Тепер можна вважати структуру додатку закінченою з точки зору backend-розробки, а середовище проекту – підготовленим до роботи.

3.3.3 Створення роутів авторизації, каталогу, замовлення

Виходячи з інформації з пункта 2.2 про моделі додатка треба створити декілька роутів, які пізніше будуть використані для створення API системи.

Для того щоб почати роботу по створенню роутів необхідно створити у корені проекту папку з назвою routes та додати у неї окремо усі необхідні файли. Так, у цьому пукнті буде розглянуто створення роутів авторизації, каталогу та замовлення.

Для створення роуту авторизації створюємо файл з назвою auth.js, який повинен містити наступні роути:

Авторизація

- /api/login/ (POST)
- /api/register/ (POST)

Алгоритм роботи роуту авторизації зображено на блок-схемі додатку Е.

Як бачимо, треба використати 2 метода POST, які будуть використані для логіну та регістрації користувачів у системі.

Перед тим як почати створення ознайомимось з основами маршрутизації за допомогою методу route класу express.

Маршрутизація визначає, як додаток буде відповідати на клієнтський запит до якоїсь конкретної адреси (кінцевої точки), якою є URI (або шлях), і визначеному методу запиту HTTP (GET або POST, чи інші).

Кожен маршрут може мати одну або декілька функцій обробки, які будуть виконуватися в період зіставлення маршруту.

Маршрут має наступну структуру:

app.METHOD(PATH, HANDLER),

де:

- app – це екземпляр класу express;
- METHOD – метод запуску HTTP;
- PATH – шлях на сервері;
- HANDLER – функція, яка буде виконуватися під час зіставлення маршруту.

Для того щоб під’єднати метод route до файлу роуту треба використати наступну команду:

```
const router = express.Router();
```

Далі потрібно виконати створення роуту авторизації та логіну у цьому файлі, виконується це за допомогою наступних команд:

```
router.post(‘/login’, controller.login);
router.post(‘/register’, controller.register);
```

Як бачимо, для того щоб створені роути працювали треба далі буде створити папку з контролерами та описати їх функціонал.

Роути каталогу та замовлення створюються додатковими файлами по аналогії з прикладом вище.

Також потрібно відразу створити роути для аналітики щоб потім не витрачати на це свій час.

Далі потрібно прописати функціонал всіх необхідних контролерів, котрі були зазначені та використані у роутах. Для цього потрібно у корені проекту створити папку controllers та додати туди усі необхідні файли.

Так як роути аналітики були додані останніми, то й розглянемо створення контролерів саме для неї.

Для цього створюємо файл analytics.js у папці controllers. Для аналітики проекту були прописані наступні роути:

Аналітика

- /api/analytics/overview/ (GET)
- /api/analytics/analytics/ (GET)
- /api/analytics/channelAnalytics (GET)

Виходячи з цього, створюємо дві функції у файлі контролеру аналітики. Робиться це за допомогою наступного коду:

```
module.exports.overview = function (req, res) {
}
module.exports.analytics = function (req, res) {
```

}

Заповнення цих функцій буде виконано у пункті 3.4 даної роботи під час формування API системи.

3.3.4 Підключення БД “MongoDB”

Як було оголошено раніше для роботи з даними у даній CRM-системі буде використану БД Mongo з ODM “Mongoose”.

Але для проектування і розробки БД локальної версії Mongo встановлюватися не буде. Це питання можна вирішити більш просто за допомогою спеціального сервісу під назвою “mLab”.

mLab – це повністю керована служба хмарної бази даних, у котрій можна зберігати самі бази і виконувати різноманітні операції з їх вмістом. mLab працює на хмарних потужностях таких компаній як Amazon, Google і Microsoft Azure, тому можна вважати даний сервіс гарною альтернативою локальній версії Mongo.

Серед мінусів використання такого сервісу є час відгуку від БД. Так як проект буде знаходитися на власному сервері, а БД – на віддаленому, то потрібно не забувати й про той час, який буде необхідним для їх взаємодії між собою.

Після реєстрації, налаштування та створення БД за допомогою сервісу треба створити користувача, який буде використовуватися для внесення будь-яких змін до БД та інсталювати ODM “mongoose”.

Для того щоб використовувати даний пакет у проекті необхідно підключити його до app.js за допомогою спеціальної команди –

```
const mongoose = require(‘mongoose’);
```

При зверненні до “mongoose” слід використовувати метод connect(). У цьому методі слід вказати url адресу бази даних, за якою слід звертатися до БД через сервіс mLab.

Ім’я та пароль користувача БД вказується у цій адресі, наприклад:

```
mongodb://admin:asd123@ds1111mlab.com:1111/telegram
```

Для перевірки підключення до БД слід використовувати проміс .then(), через який можна передати спеціальне повідомлення до консолі, або проміс .catch() для виявлення тих або інших помилок при підключенні до БД. Схема БД систем изображена у додатку В.

3.4 Формування API системи

Раніше у пункті 3.3.3 даної роботи було створено роути авторизації, каталогу, замовлення та аналітики. Для того щоб клієнт міг користуватися ними був створений цілий набір контролерів, які містять у собі певний перелік функцій.

На даному етапі розробки необхідно описати ці функції у файлах контролерів.

Розпочнемо зі створення функціоналу каталогу, а саме з отримання списку каналів за тими чи іншими критеріями. У нашому випадку найпростішим з критеріїв на даний момент буде id категорії.

Як було оголошено спочатку – суть у тому, що користувач може відсортувати необхідні йому канали за категорією.

Для створення та тестування API каталогу будемо використовувати методи try та catch, які допоможуть у виявленні помилок.

Так, конструкція функції буде виглядати наступним чином:

```
try {
} catch (e) {
        errorHandler(res, e)
  }
}
```

Як видно із коду, для відслідковування помилок відразу мною був створений метод errorHandler.

Для отримання списку каналів за відповідним id категорії створюємо змінну channels, код якої буде виглядати наступним чином:

```
const channels =  await channels.find ({
        category: req.params.categoryId,
```

```
        user.req.user.id,
        date.req.date,
        time.req.time
    })
    res.status(200).json(channels);
```

Як можна побачити, у цій змінній використовується модель channels, яка вже була створена мною та знаходиться у папці models проекту.

Разом зі списком каналів за id певної категорії ми отримуємо юзернейм користувача, який додав цей канал до каталогу, дату та час, на які можна запланувати публікацію рекламного повідомлення у цих каналах.

Останній функціонал контроллерів створюється по аналогії з прикладом. Принцип роботи системи та API розглянуто у додатку Г.

3.5 Розробка frontend частини

3.5.1 Angular CLI та початок розробки

Як було сказано раніше, для розробки фронтенд частини системи я буду використовувати фреймворк Angular 7 версії.

Для того щоб швидко під’єднати його до проекту та почати розробку слід використати Angular CLI. Його встановлення можна виконати через пакетний менеджер npm за допомогою команди –

```
npm install –g @angular/cli
```

Після цього за допомогою команди ng new client можемо додати фреймворк Angular до вже існуючого беку. Фреймворк буде розміщуватися у папці client проекту.

Після додавання фреймворку треба провести деякі заходи по налаштуванню і можна переходити безпосередньо до розробки. Angular CLI удаляти непотрібно, так як його можна також використовувати і для формування сторінок за допомогою команди –

```
ng g c login-page –-spec false
```

Так, наприклад, можна сгенерувати компонент сторінки логіну, з якою ми далі будемо працювати. Після генерації компоненту отримаємо стандартні для фреймворку Angular файли з розширенням .html, .css і .ts.

3.5.2 Розробка UI сторінки логіну

Перед тим як починати розробку UI сторінки логіну, треба сказати про те, що вона буде схожа на сторінку реєстрації. Також треба звернути увагу на те, що макет цих сторінок буде значно відрізнятися від макету сторінок, котрі будуть використовуватися всередині системи. Виходячи з цього, відповідно і треба будувати структуру системи.

Для початку розробки макету потрібно сгенерувати новий компонент за допомогою Angular CLI, команда:

ng g c shared/layouts/auth-layout –-spec false

Після виконання команди у структурі проекту з’явиться відповідна директорія з .html-, .css- і .ts-файлами. Ця директорія буде призначена для компоненту авторизації, а для інших компонентів – треба буде повторити виконання відповідної команди, змінивши назву компоненту.

Після створення всіх необхідних макетів компонентів треба створити відповідні роути у файлі роутів. Після створення роутів необхідно підготовану html сторінку логіну перенести у відповідний .html файл, при цьому для надання їй відповідного зовнішнього вигляду слід також використати .css-стилі.

На момент переносу макету сторінки у вікні браузера буде постійно відображатися тільки сторінка логіну, перейти на будь-які інші сторінки буде неможливо, так як у системі доступний лише один головний макет – логіну.

Перед тим як зробити доступною зміну шаблонів зі зміною роутів необхідно визначитися, де у всіх шаблонах знаходиться незмінна (статична) частина, а де – та, яка відрізняє їх між собою.

Після цього потрібно зробити активними посилання на сторінки логіну та реєстрації, а також бажано повторити це і для останніх посилань на сайті за допомогою атрибуту routerLink=”назва компоненту для посилання”. Також треба використовувати атрибут routerLinkActive=”active” для того щоб

візуально користувач міг розуміти, по якому посиланню він зараз може перейти, а по якому – ні.

Підсумовуючи, після цих пунктів треба переходити до створення форми логіну і сервісу авторизації. Приблизно подібні кроки треба виконати для створення UI і інших компонентів системи. Вибір такого підходу до створення UI описано в наступних пунктах. На рис.3.4 зображено, як буде виглядати UI сторінки логіну та реєстрації в кінці кінців.

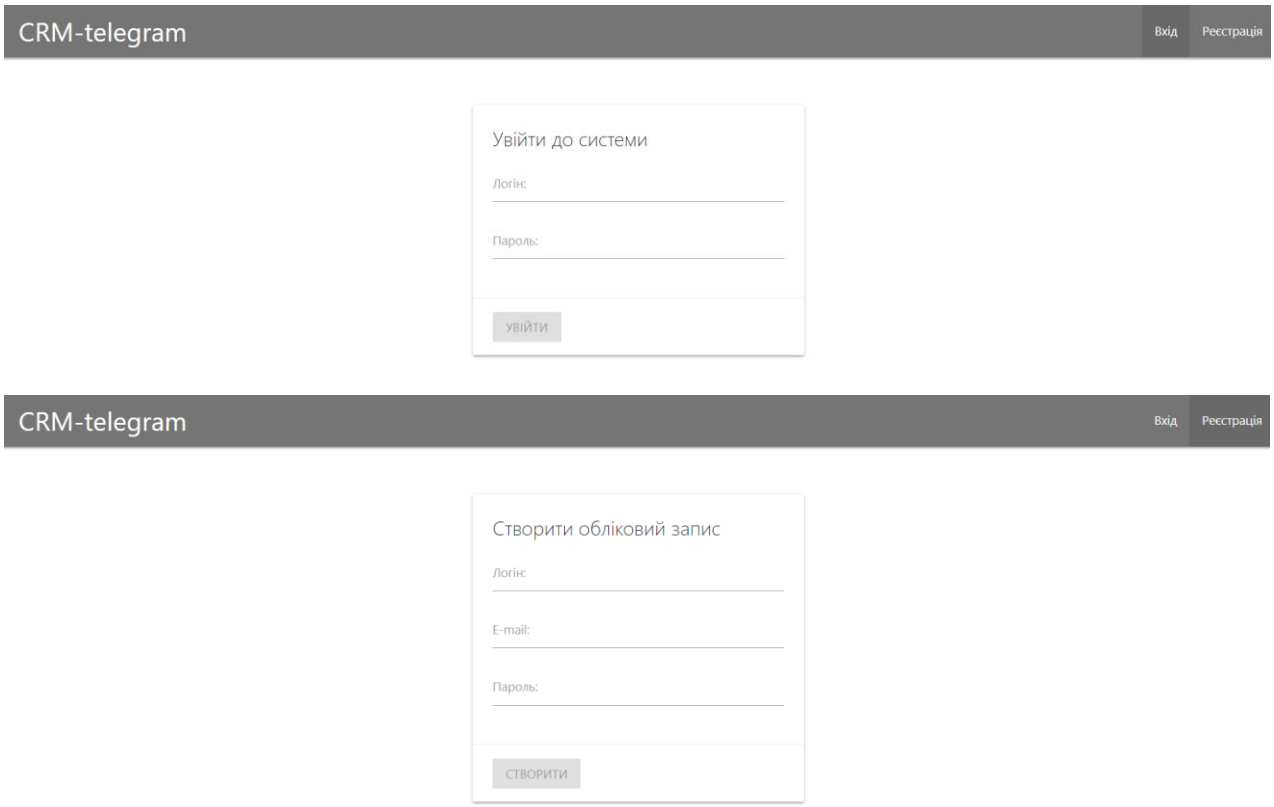


Рис. 3.4. UI сторінок логіну та реєстрації

3.6 Суть інтерфейсу користувача

Інтерфейс користувача (ІК) – це набір засобів, які дозволяють користувачу взаємодіяти з додатком.

Процес проектування ІК є складним і нелінійним, його складність полягає у невизначеностях у процесі розробки, які значним чином впливають на нього.

3.6.1 Проблематика проектування ІК

Процес проектування інтерфейсу користувача складається з таких задач [12]:

- зниження витрат;
- скорочення термінів;
- підвищення якості;
- розробка простого в освоєнні рішення;
- використання нових технологій та засобів;
- створення конкурентоспроможного рішення.

Загальна задоволеність користувача під час використання ПЗ найчастіше визначається задоволеністю від користування інтерфейсом. Пересічного користувача зазвичай не цікавить той стек технологій, який використовував розробник, для нього найбільше значення має те, що він бачить перед собою.

Задоволеність від користування ІК визначається такими особливостями:

- можливості ІК;
- практичність;
- час відгуку;
- надійність;
- підтримка розробником, супроводження та модернізація.

Можливості інтерфейсу користувача повинні повністю відображати увесь функціонал додатку: його інтерфейс має бути комфортним, а час відгуку – мінімальним. Останній параметр є досить важливим, так як кожен розробник повинен вміти дати своїм користувачам усе, що вони бажають за мінімальні проміжки часу.

Розроблюваний додаток повинен бути надійним та зберігати усі параметри користувача, які потрібні для роботи протягом усього періоду взаємодії користувача з додатком. Будь-який додаток, який розроблюється для масового споживання повинен постійно підтримуватися та оновлюватися, щоб робити конкуренцію вже існуючим рішенням.

До проблематики проектування ІК також можна віднести той факт, що деякі користувачі є дуже категоричними щодо наявності певних елементів чи функцій. Не завжди користувач може висловити свої побажання щодо проектування інтерфейсу, тому часто розробник повинен робити все на свій розсуд і вже потім консультуватися з користувачем, або навіть проводити тестування свого інтерфейсу серед декількох користувачів.

3.6.2 Методологія проектування інтерфейсних рішень

Для чіткого представлення інформації існує багато методів. В даному дипломному проекті було обрано основні методи, що використовуються у створені інтерфейсних застосунків :

- Підхід дружнього інтерфейсу.
- Принцип web інтерфейсу користувача.
- Принципи людського фактору в ІК.

3.6.3 Підхід дружнього інтерфейсу

Дружній інтерфейс користувача (UI) – це шаблон, який може забезпечити природню взаємодію користувача з розроблюваною системою. Така «природність» досягається найрізноманітнішими шляхами, багато з яких постійно оновлюються. «Дружність» інтерфейсу почала хвилювати проектувальників та розробників програмних продуктів не так давно, причиною цього став загальний розвиток дизайну у цілому, поява таких професій, як UI/UX-дизайнер та тестувальник.

Будь-який досвідчений користувач може надзвичайно швидко оцінити якісність і перевагу одного інтерфейсу користувача над іншим. Як показує практика, якісний інтерфейс має базуватися на 5 принципах, які скорочено позначаються абревіатурою SAPCO (рис.3.5).

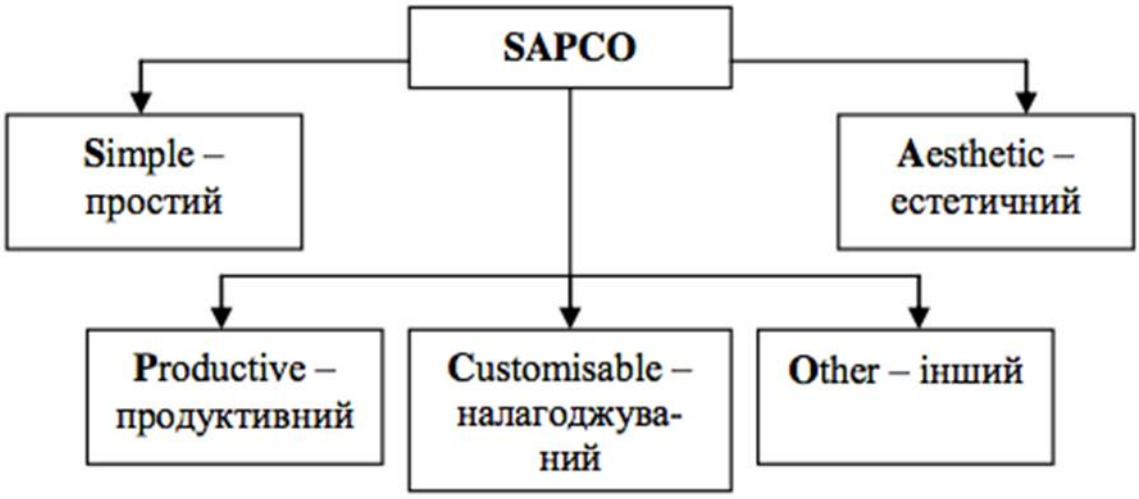


Рис. 3.5. Принципи SAPCO проектування якісного ІК

Простота. Будь-яке розширення функціоналу або внесення додаткових можливостей до програмного забезпечення не повинно вносити додаткові зміни до інтерфейсу користувача. Якісно розроблений інтерфейс є простим та зрозумілим для користувача, для використання такого інтерфейсу не потребується наявності додаткового довідника або цілої системи, яка буде робити підказки. Якісним прикладним ІК вважається інтерфейс, використання якого можливе на інтуїтивному рівні (підказки даються лише для того, щоб користувач міг зрозуміти, якого кінцевого результату від нього очікує система).

Естетичність. Програмний продукт повинен бути гарним та мати ергономічну привабливість для користувача. Зрозуміло, що поняття естетичності для кожного користувача є різним, то ж краще використовувати вже розроблені напрацювання дизайнерів, які розраховані на широку масу користувачів і відповідно будуть подобатися їм. Це правило може не працювати у тому випадку, якщо розробник розроблює вузько-спрямовану систему і її кінцевий споживач має нетипові естетичні вподобання. У такому випадку краще порадитися з ним та зробити дизайн, виходячи з побажань майбутнього клієнта. Якісний інтерфейс не повинен викликати будь-яких негативних емоцій у користувача та вміти максимально візуалізувати інформацію, яку треба донести до нього.

Продуктивність. Продуктивність інтерфейсу користувача полягає у тому, що користувач повинен докласти мінімум зусиль для того щоб виконати поставлену системою задачу. Ідеальним з точки зору продуктивності вважається інтерфейс, у якому відсутня велика кількість вікон та екранів. Також у такому інтерфейсі повинен бути відсутній тиск на користувача з метою виконання зайвих дій.

Налагоджуваність. Важливо розуміти те, що у сучасному світі кожна людина є індивідом і має певні свої погляди на програмний продукт. Саме тому, можливість налагоджувати інтерфейс користувача під свої бажання – основний признак гарно реалізованого продукту.

3.6.4 Принцип web-інтерфейсу користувача

Основний принцип сучасних web-інтерфейсів – виконання взаємодії системи з користувачем повинно відбуватися у єдиному вікні. Так як розроблювана у даній дисертації система є single page application – то можна з впевненістю сказати, що використання цього принципу повністю збережено.

Навігація у такому додатку повинна виконуватися з використанням гіперпосилань (вони можуть бути текстовими або візуальними). Основна навігація між сторінками такого додатку реалізується за допомогою навігаційного меню, яке зазвичай розміщується вверху або збоку сторінки. У такому меню повинні знаходитися посилання абсолютно на всі сторінки, які доступні у системі. Якщо кількість таких сторінок перевищує розміри меню, то повинно використовуватися скорочення меню (елементи меню розбиваються на декілька груп, вибирається по одному головному елементу, а останні – приховуються у ньому). При цьому подібні скорочення меню повинні бути зрозумілими для користувача.

Додатки повинні використовувати графіку та анімацію в естетичних та навігаційних цілях, але треба пам’ятати про те, що в цілому, додаток не повинен бути перевантажений такими елементами, так як це може позначитися

на їх швидкодії, особливо якщо користувач має слабкий ПК.

У той же час браузер повинен забезпечувати повне віключення графічної анімації у разі необхідності.

3.6.5. Принципи людського фактору в ІК

На сьогоднішній день створення зручного та комфортного середовища для продуктивної роботи користувача – основна задача будь-якого розробника та його системи. Проводиться маса досліджень, які орієнтуються на вивчення поведінки людини у тому чи іншому інтерфейсі користувача. Зазвичай будь-які зміни до вже готового продукту робляться на основні аналізу поведінки користувачів.

Проектування інтерфейсу – це спільна діяльність розробника та користувача. Треба пам’ятати, що кожен розробник – це також користувач інших програмних продуктів. Серед цих продуктів можуть бути як розробки конкурентів, так і щоденне програмне забезпечення, яке використовується з різною метою – від отримання інформації і до розважання.

Зрозуміло, що характеристичні особливості персональних комп’ютерів кінцевих користувачів істотно відрізняються між собою, але треба пам’ятати, що те, як користувач вводить, сприймає та виводить свою інформацію із розроблюваної системи теж грає велику роль. Кожен розробник повинен розуміти, що користувач може мати певні обмеження та слабкості, які набуваються останнім із-за віку або стану здоров’я. Саме проектування ІК з врахування слабких сторін потенційних користувачів є ознакою розробки ефективного програмного забезпечення.

Недостатність уваги до таких людських факторів в результаті буде призводити до створення непродуктивного програмного забезпечення, яке буде важко вивчити та використовувати. Для вимірювання такої характеристики як продуктивність користувача можуть використовуватися різноманітні метрики

(наприклад, час, який необхідний для користувача для того щоб він безпомилково додав свій товар у корзину і т.п.). Комфортність у використанні програмного забезпечення для будь-якого користувача полягає у задоволеності останнього та його думок з приводу легкості використання програмного забезпечення.

При проектуванні інтерфейсу користувача слід брати до уваги і нефізичні особливості кожної людини та враховувати її психологічні характеристики. Врахування цих та інших особливостей людини-кінцевого користувача дозволяє розробнику створити максимально гнучке та практичне програмне рішення з комфортним інтерфейсом, яке зможе конкурувати з вже існуючими на ринку продуктами.

3.6.6 Створення ІК системи

Для створення інтерфейсу користувача було вирішено використовувати вже готовий фреймворк Material від “Google”. Він є легким, швидким та зрозумілим для користувача, що задовольняє поставлені задачі щодо інтерфейсу користувача.

Матеріальний дизайн (англ. Material design) [14] — це спеціальні правила дизайну, які вперше були створені компанією “Google” та представлені на однойменній конференції у 2014 році. Актуальну реалізацію такого дизайну можна побачити у операційній системі від “Google” – “Andorid”, а також – майже у всіх продуктах компанії, які постійно оновляються.

Сама компанія каже про те, що матеріальний дизайн – є спеціальною дизайн мовою, яка повинна бути властива усім програмним продуктам, розроблюваним у 21 столітті. Ідея такого дизайну полягає у використанні інтерфейсів, які за схожістю нагадують паперові картки.

Особливості. Як кажуть дизайнери компанії “Google” – «було зроблене дослідження, яке допомогло повністю змінити підхід до візуалізації інтерфейсу

та його частин». Використання матеріального дизайну у своєму продукті робить взаємодію користувача із додатком більш реальнішою, що призводить до отримання людиною задоволення під час роботи.

- Використання тіней у дизайні допомагає створити так званий простір між самим додатком і екраном користувача, що дозволяє зрозуміти останньому те, що він працює з чимось реальним.
- Використання спеціально-розробленої UI анімації дозволяє користувачу візуально оцінити стан системи та її елементів і на основі аналізу цих рухів – система дозволяє користувачу здогадатися, що від нього очікують, або що відбувається у даний момент часу.
- Використання інтерактивних іконок дозволяє системі викликати акцент користувача на певній операції у системі.
- Застосування так званої об’єктної хореографії дозволяє одним елементам системи впливати на інші, що натякає користувачу на поетапність роботи з системою.

Базові елементи матеріального дизайну

Кожен елемент, з якого може складатися сайт, створений в стилі матеріал, докладно описаний в документації. Починаючи від того, як його створити і на що він повинен бути схожий, до його місця, способу і моменту появи на екрані користувача. Загалом, важко придумати щось таке, що творці втратили або ж ні.

Список включає 19 основних структурних компонентів, нижче перерахую деякі з них [13]:

- Нижній екран (нижні шари дизайну)
- Кнопки
- Карти

- Діалоги
- Роздільники
- Сітки
- Списки
- Меню
- Смуги прогресу і активності
- Слайдери
- Підзаголовки
- Перемикачі
- Текстові поля
- Спливаючі вікна

Іконки і інші деталі йдуть в парі зі стилями і підлаштовуються під обраний шаблон.

Основні принципи матеріального дизайну

У документації з матеріального дизайну велику увагу зосереджено саме на доступності, зручності та призначеному для користувача досвіді, що вельми важливо. У той час як багато-які з естетичних властивостей матеріального дизайну можуть здатися досить примітивними для досвідчених дизайнерів, деякі з понять користувацького досвіду і взаємодії реалізовані на найвищому рівні.

Особливо корисний розділ про моделі взаємодії. Він розкриває ідеї, спрямовані на те, щоб зробити деякі елементи дизайну універсальними по відношенню до всіх можливих веб-проектів. Наприклад, формат зазначення дати і часу, або робота пошуку.

Вони можуть відрізнятися на різних сайтах, але відмінності незначні, і в основному ці елементи досить універсальні. Якщо відвідувач бачить лупу - він розуміє що перед ним пошук, яким можна скористатися, навіть якщо поруч з

нею немає текстового покажчика. Саме тому матеріальний дизайн містить деякі з найпростіших інструментів, які користувачі очікують побачити на будь-якому сайті, і з якими вони звикли працювати.

Доступність і зрозумілість дизайну - ще один аспект, широко розкритий в документації з матеріального дизайну. Матеріальний дизайн повинен враховувати користувачів, які можуть взаємодіяти з ним не тільки за допомогою кольорів і форм, але і за допомогою звуків і голосового пошуку.

Також, важливо продумати можливість перегляду дизайнів на пристроях з високою контрастністю, великим екраном, без видимого екрану, тільки за допомогою голосового управління, або з огляду на комбінацію всіх цих елементів.

Приклад реалізації веб-сайту та мобільних додатків за допомогою матеріального дизайну можна побачити на рис.3.6.

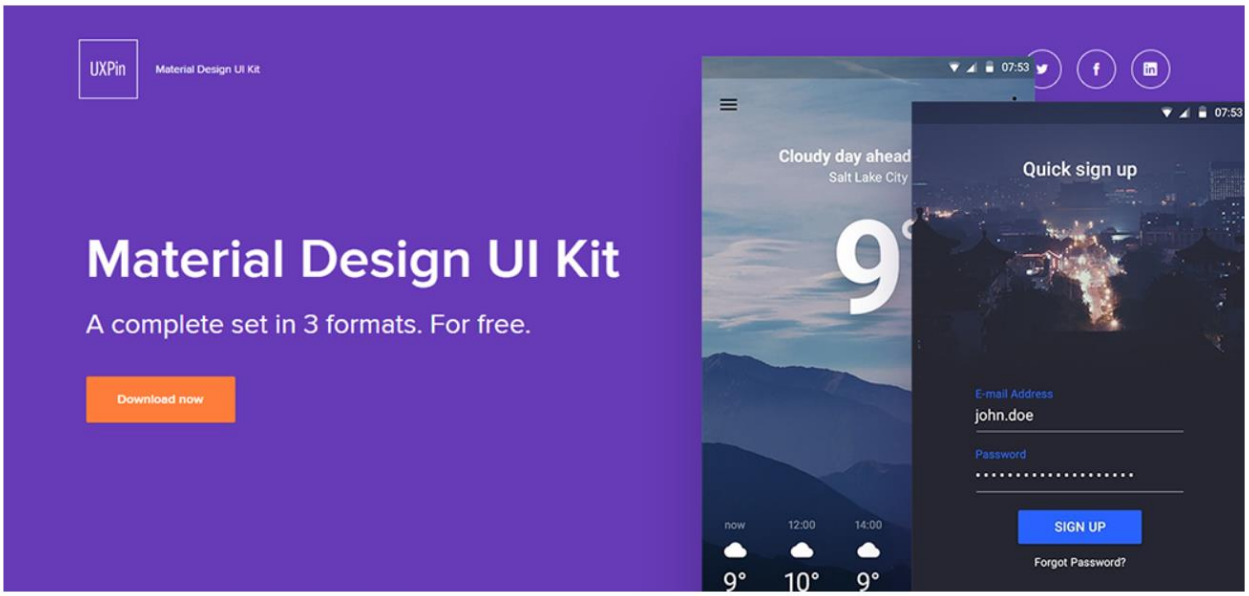


Рис. 3.6. Приклад реалізації веб-сайту та мобільних додатків від “UXPin”

Під час розробки системи купівлі та продажу рекламних повідомлень у “Telegram” було вирішено користуватися актуальними напрацюваннями матеріального дизайну та створити веб-інтерфейс з використанням наступних елементів дизайну:

1. Вертикальне навігаційне меню.
2. Тап-таргет (іконка) – допоміжна кнопка, яка викликає вікно з поясненням роботи системи або певної сторінки.
3. Кольорове підсвічування тексту у разі зменшення або збільшення певного статистичного показника системи.
4. Зрозумілі для користувача іконки.
5. Діалогові вікна попередження та підтвердження дій від користувача.
6. Графіки, котрі будуються у режимі реального часу під час роботи користувача з системою.
7. Помітні та зрозумілі для користувача кнопки (деякі з них з зображенням).
8. Календарі для вибору дати.
9. Модальні вікна, котрі зменшують кількість переходів користувача між сторінками у середині системи.
10. Спливаючі підказки.
11. Спливаючі повідомлення про результат виконання дії у системі.
12. Випадаючі списки.
13. Кольорові, зрозумілі для користувача повідомлення про помилки з різними варіантами тексту помилок.
14. Фільтри для більш зручного завантаження даних.
15. Маленькі зображення для миттєвого представлення інстанцій системи.

Приклади реалізації інтерфейсу користувача та використання всіх описаних вище елементів можна побачиту у додатку Г.

3.7 Підготовка системи до взаємодії з телеграм-ботом

Як було сказано раніше, головне завдання системи – позбутися усіх проблем, які можуть виникнути у користувача під час купівлі або продажу рекламних повідомлень у месенджері “Telegram” у ручному режимі.

Для того щоб досягти цього можна пійти стандартним підходом – використати працю людини, яка б постійно слідкувала за розміщенням рекламних повідомлень у каналах, видаляла б та слідкувала за тим, щоб адміністратори каналів не видаляли рекламні повідомлення раніше зазначеного сроку. Але робити це немає сенсу, так як використання будь-якої праці людини – це додаткові затрати на етапі запуску проекту.

Більш того, навіщо використовувати працю людини, якщо можна скористатися більш інноваційним підходом: організація роботи системи з спеціально ствоворюваним у майбутньому телеграм-ботом.

Важко уявити собі, як людина справлялася б з тисячами каналів у рекламному каталозі системи, а для боту – це звичайне завдання. Більш того такий бот зможе виступати повноцінною заміною веб-інтерфейсу системи, тобто користувач зможе використовувати увесь важливий для нього функціонал напряму з додатку “Telegram”, який може бути встановлений на будь-якому портативному пристрою [15].

Використання подібного боту дозволить не тільки використовувати стандартний функціонал системи, але й дасть можливість запровадити нові функції, зокрема серед них:

1. Автоматична публікація рекламних повідомлень у стрічці каналу.
2. Перевірка наявності публікації у каналах каталогу.
3. Автоматичне видалення публікації після заданого терміну.
4. Блокування каналів, які порушили правила та видалили рекламне повідомлення, або видалили бота з адміністраторів каналу.
5. Отримання списку останніх публікацій та виведення їх у систему.
6. Отримання кількості публікацій, кількості їх переглядів, кількості підписників каналу раз у день.
7. Ведення аналітики кожного каналу, яка буде вираховуватися з статистичних даних.
8. Запровадження унікальних метрик, які допоможуть користувачу відрізнити накручений канал від нормального.

9. Введення показників, які допоможуть користувачу обрати кращий канал для розміщення реклами в подальшому.
10. Поповнення рахунку системи через бота.
11. Організація спеціальних чатів з оголошеннями стосовно обміну рекламними повідомленнями на певний час.
12. Обмін безкоштовними рекламними повідомленнями.
13. Можливість розповсюджувати рекламу серед користувачів боту.

Як можна зрозуміти, спектр можливостей, які дає для системи бот – достатньо великий.

На стадії розробки даної системи був продуманий та реалізований певний функціонал, котрий допоможе системі взаємодіяти з ботом, зокрема це:

- захищене API системи;
- функції авто видалення та авто публікації рекламних замовлень у системі.

3.8 Опис роботи системи, керівництво користувача

Початок роботи з розроблюваною системою починається з реєстрації та логіну користувача. Ці можливості доступні на головній сторінці системи. Після авторизації у системі користувач може знаходитися там безперервно протягом 30 хвилин, після цього система запропонує заново залогінитися.

Після авторизації у системі користувач потрапляє на головну сторінку (рис.3.7):

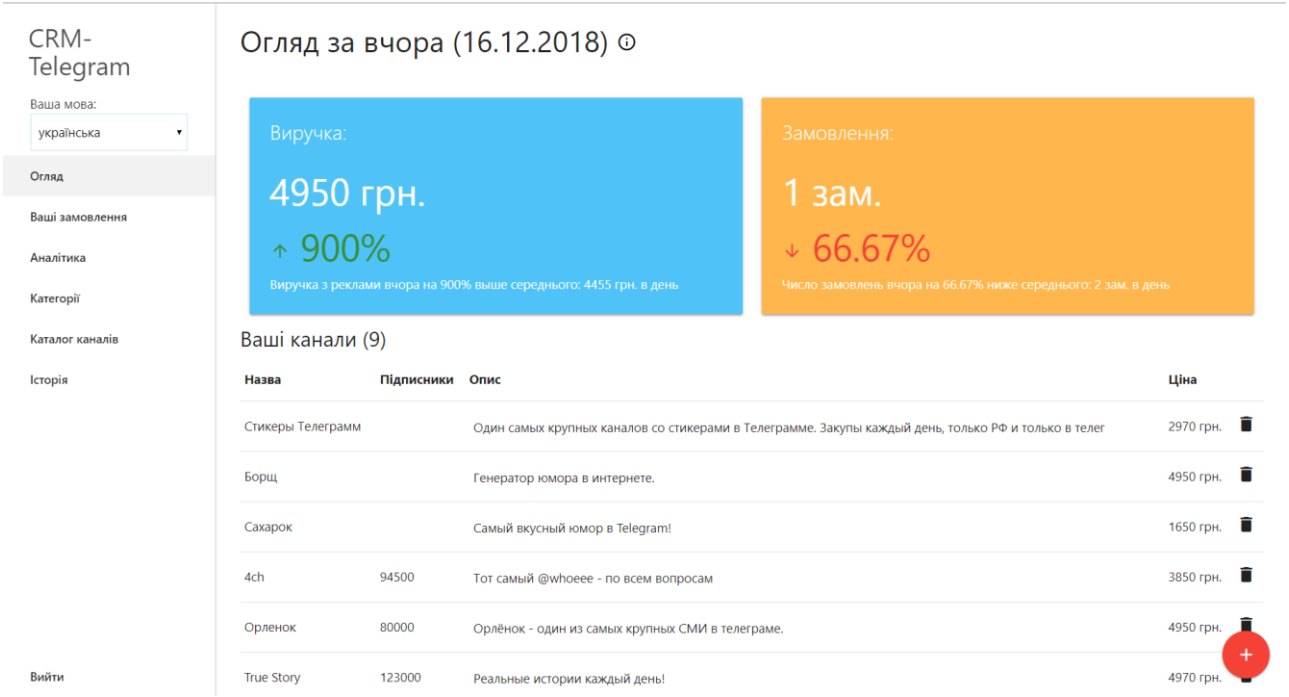


Рисунок 3.7. Головна сторінка системи

Як можна побачити з рисунку на цій сторінці можна знайти деяку статистичну інформацію про вчорашній робочий день. Також тут знаходиться список особисто-доданих користувачем каналів. Зліва знаходиться навігаційне меню системи, у якому також можна обрати потрібну для користувача мову роботи з системою: на даний момент часу на вибір доступні дві мови: українська та російська. При зміні мови усі оповіщення у системі також перекладаються.

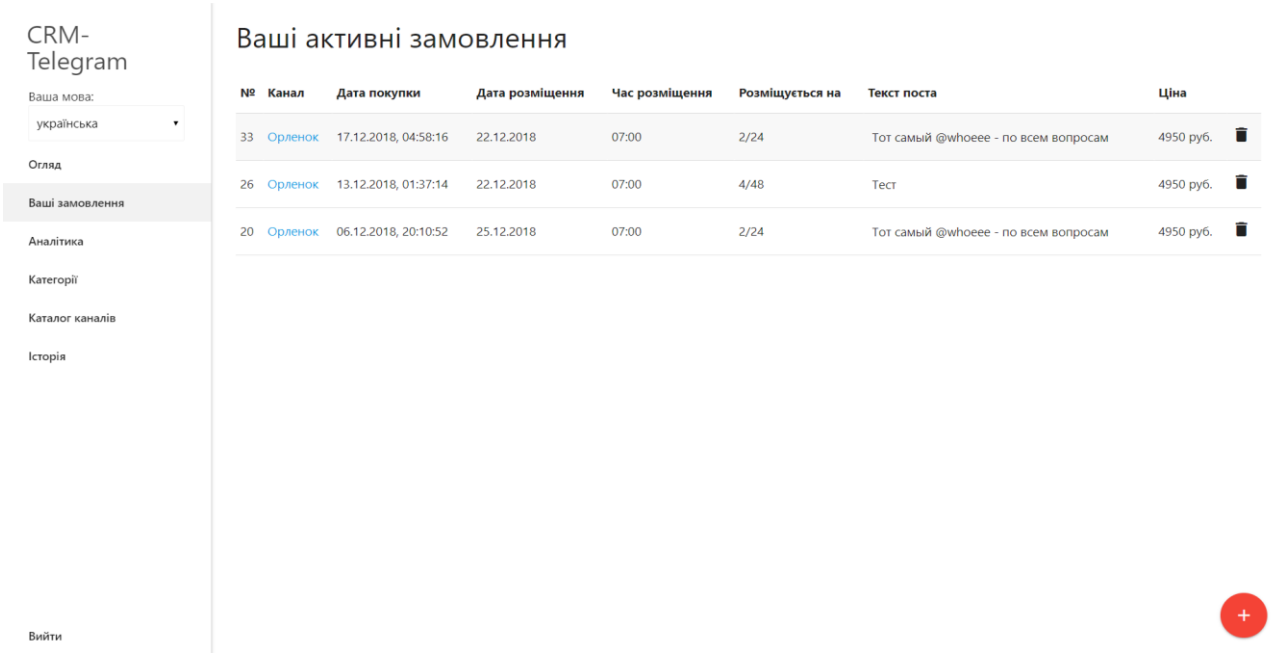


Рисунок 3.8. Активні замовлення користувача

Після цього користувач може побачити свої активні замовлення (рис.3.8).

На цій сторінці людина має можливість відмінити публікацію придбаного рекламного повідомлення. Тут можна побачити тільки актуальні замовлення, дата розміщення яких ще не прийшла. При натисканні можна побачити склад замовлення (рис.3.9), або перейти у канал, в якому було придбано рекламне повідомлення.

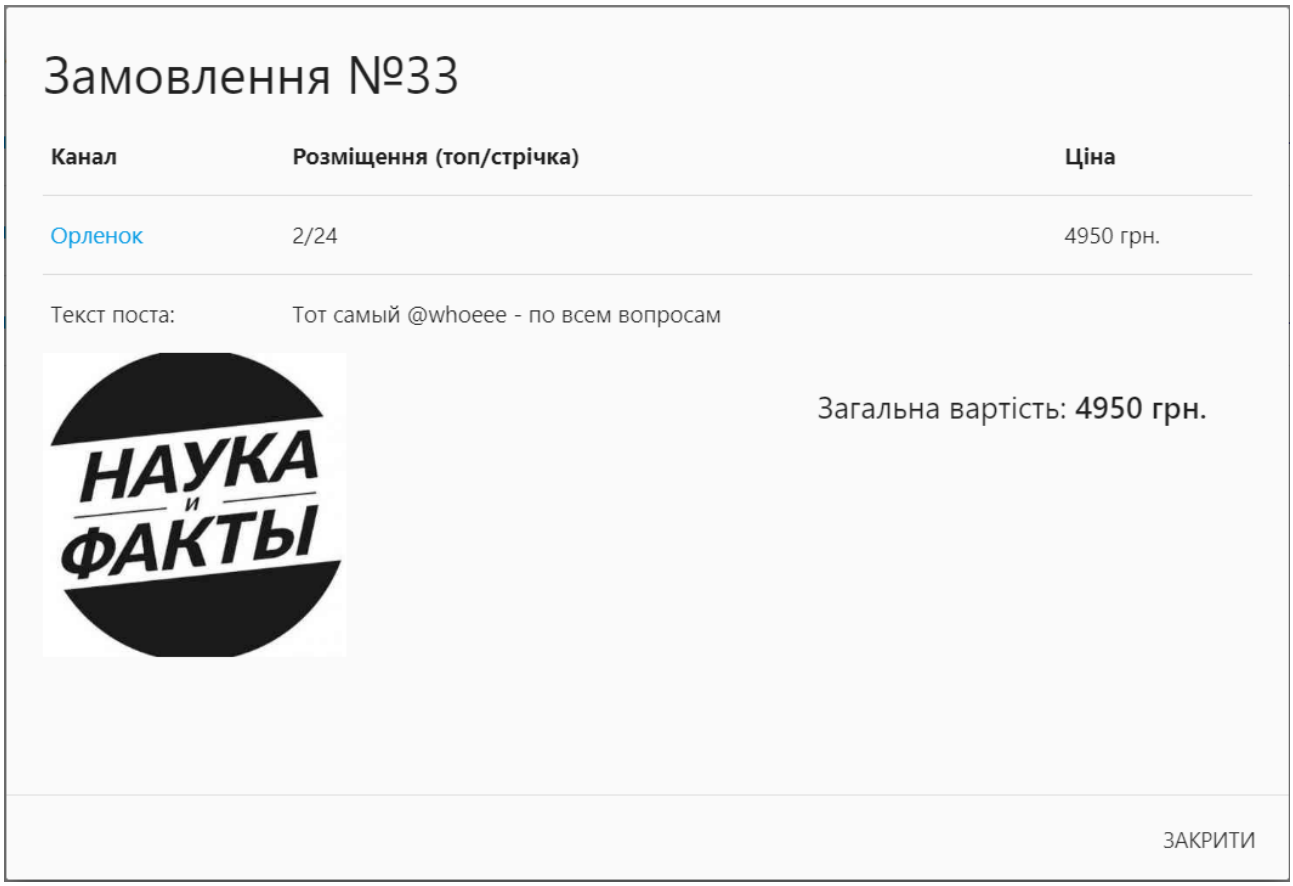


Рисунок 3.9. Склад замовлення

Після цього блоку знаходиться загальний блок системи під назвою «Аналітика» (рис.3.10):

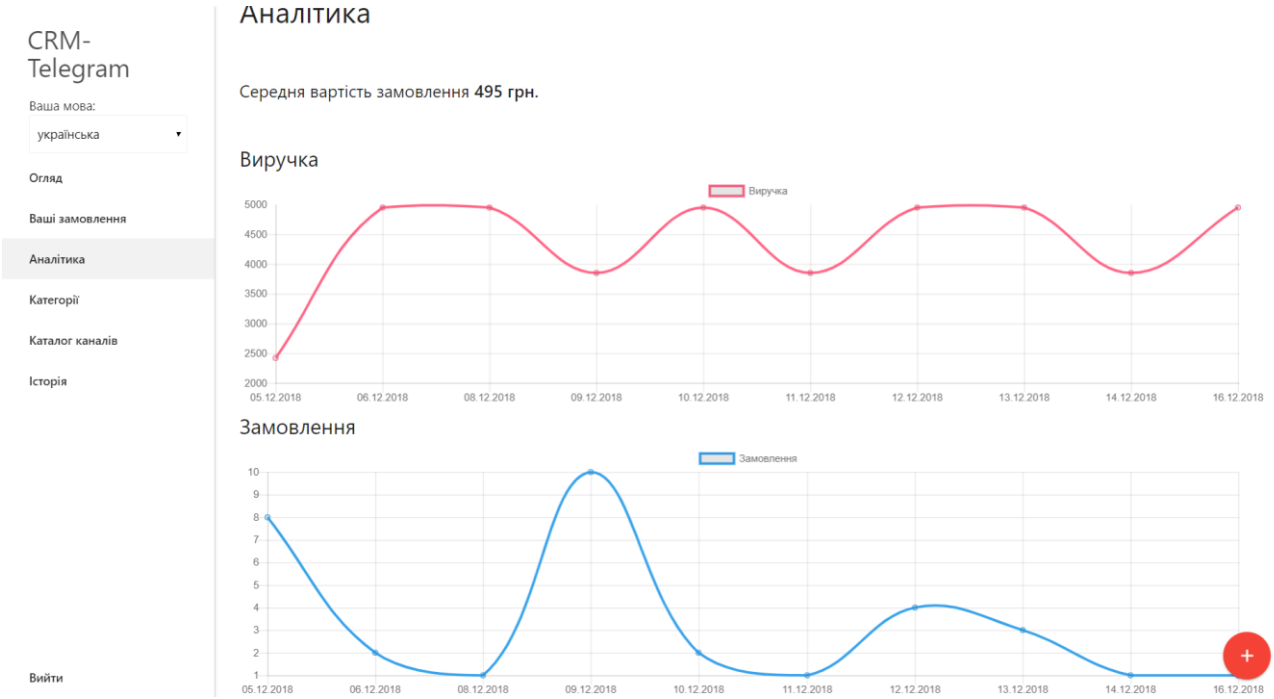


Рисунок 3.10. Сторінка аналітики

На цій сторінці можна проаналізувати дані, котрі відсортовані по усім доданим даним користувачем каналам. Тут можна знайти середню вартість замовлення і наглядні графіки. Якщо користувач є замовником рекламних повідомлень, то будуть виведені дані про актуальні витрати цього користувача у системі.

Далі (рис.3.11) знаходиться сторінка з категоріями системи:

На цій сторінці користувач має можливість додати нову категорію або відредагувати вже існуючі. Даний функціонал може використовувати лише адміністратор системи. У разі необхідності адміністратор системи може завантажити зображення до кожної з категорій.

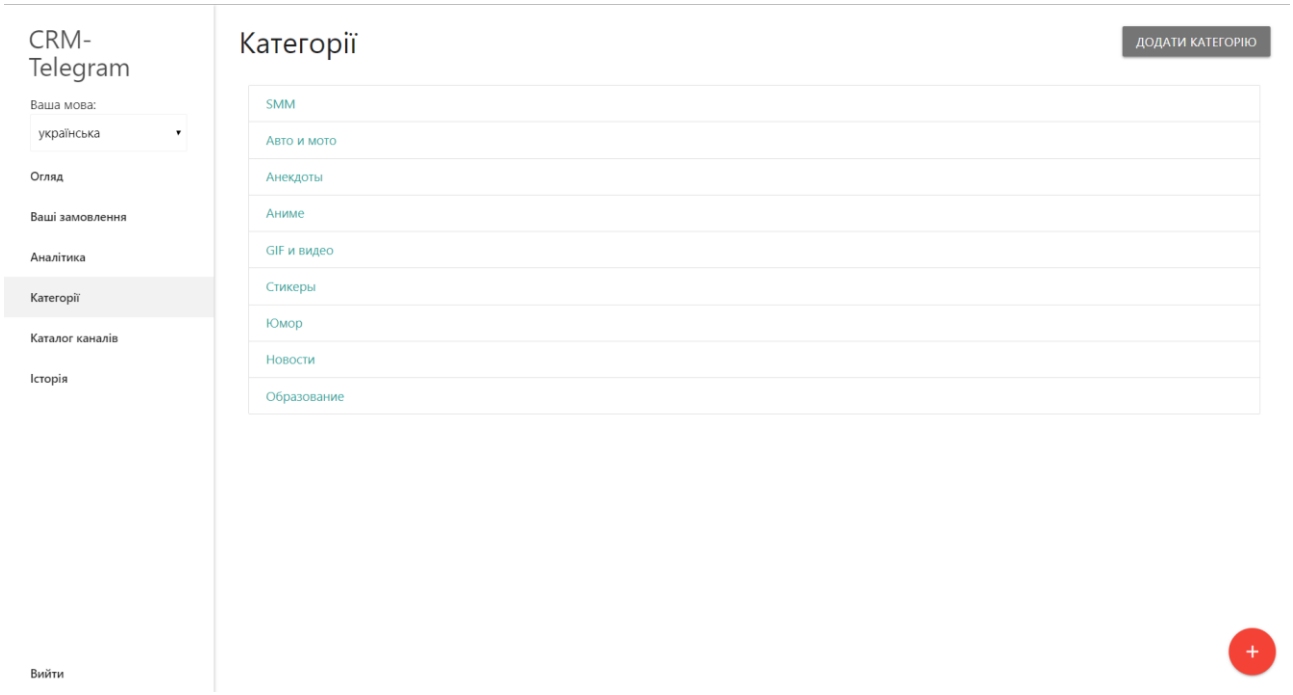


Рисунок 3.11. Сторінка категорій

Сторінка редагування категорій (рис.3.12) виглядає наступним чином:

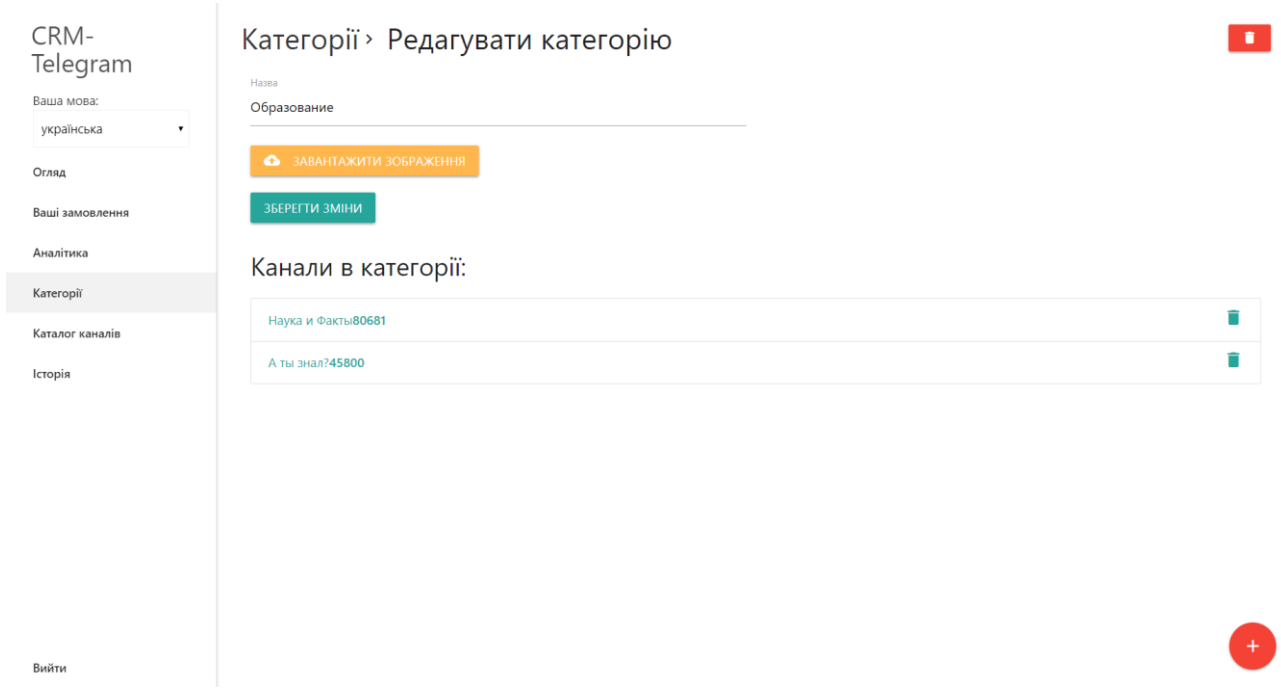


Рисунок 3.12. Сторінка редагування категорії

На цій сторінці користувач-адміністратор має можливість відредагувати

назву та зображення обраної категорії, видалити її (кнопка видалення знаходиться справа зверху, передбачено повідомлення, котре отримає користувач перед видаленням, для того щоб випадково не видалити потрібну йому категорію). Також для зручності на цій сторінкі виводиться список усіх каналів даної категорії. Видалення каналу у категорії призведе до повного видалення його із каталогу.

Далі знаходиться одна з найважливіших частин системи – каталог актуальних каналів для розміщення рекламних повідомлень (рис.3.13).

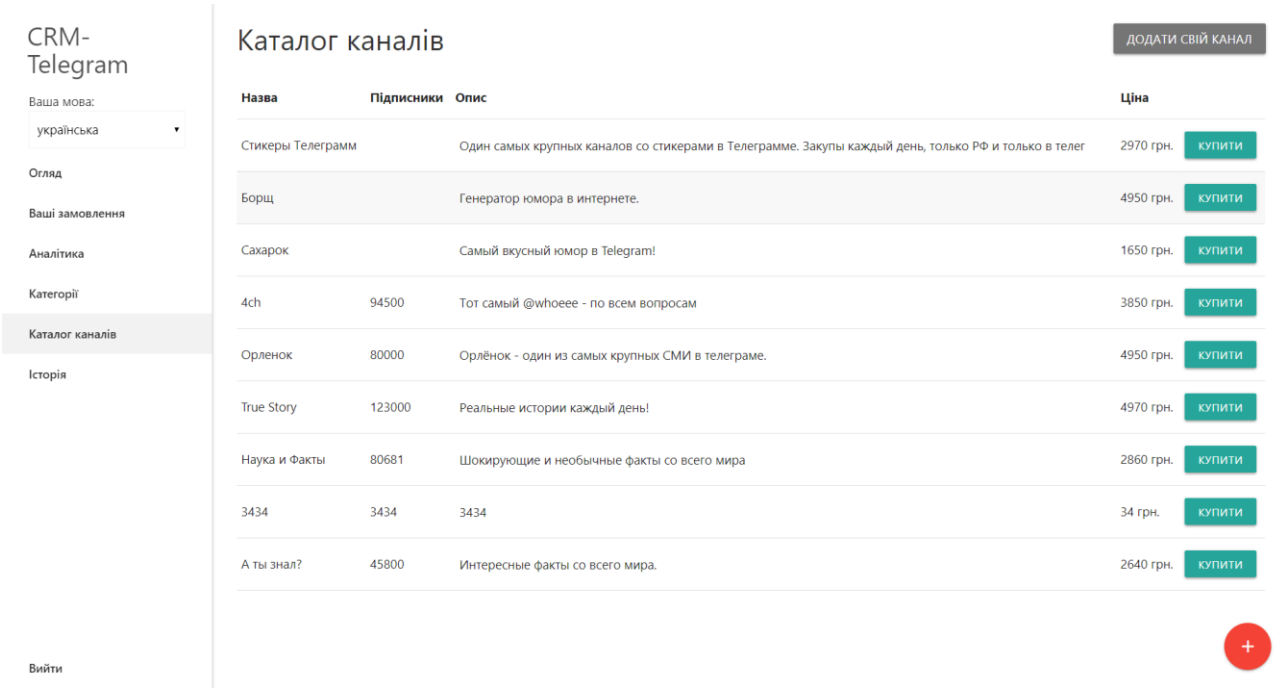


Рисунок 3.13. Каталог актуальних каналів

На даний момент у каталозі не передбачено сортування каналів по різноманітним критеріям і категоріям. Також у майбутньому буде реалізована можливість закріплення певних каналів у каталозі та опція вибору «улюблених» категорій для користувачів системи.

На даній сторінці користувач має можливість обрати необхідний канал для придбання реклами або додати свій канал у каталог. Додавання нового каналу робиться за допомогою модального вікна (рис.3.14):

Рисунок 3.14. Модальне вікно «Додати канал»

Після додавання канал можна знайти у каталозі або на сторінці «Огляд».

Якщо користувач забажає придбати рекламне повідомлення, то він зможе скористуватися кнопкою «Купити» і потрапить на сторінку обраного каналу (рис.3.15), де він зможе знайти інформацію про канал та умови розміщення, деякі статистичні дані та кнопку «Розмістити рекламу».

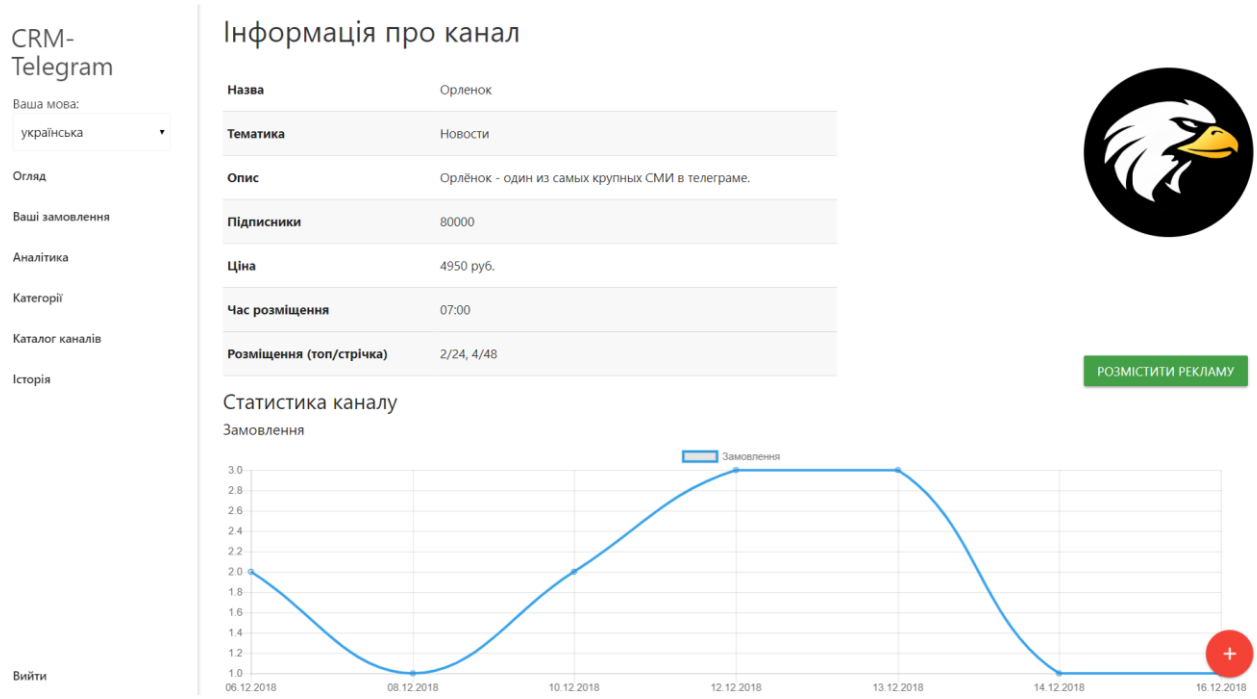


Рисунок 3.15. Сторінка каналу

Розміщення рекламного повідомлення можна здійснити за допомогою цієї кнопки, розміщення робиться через модальне вікно (рис.3.16):

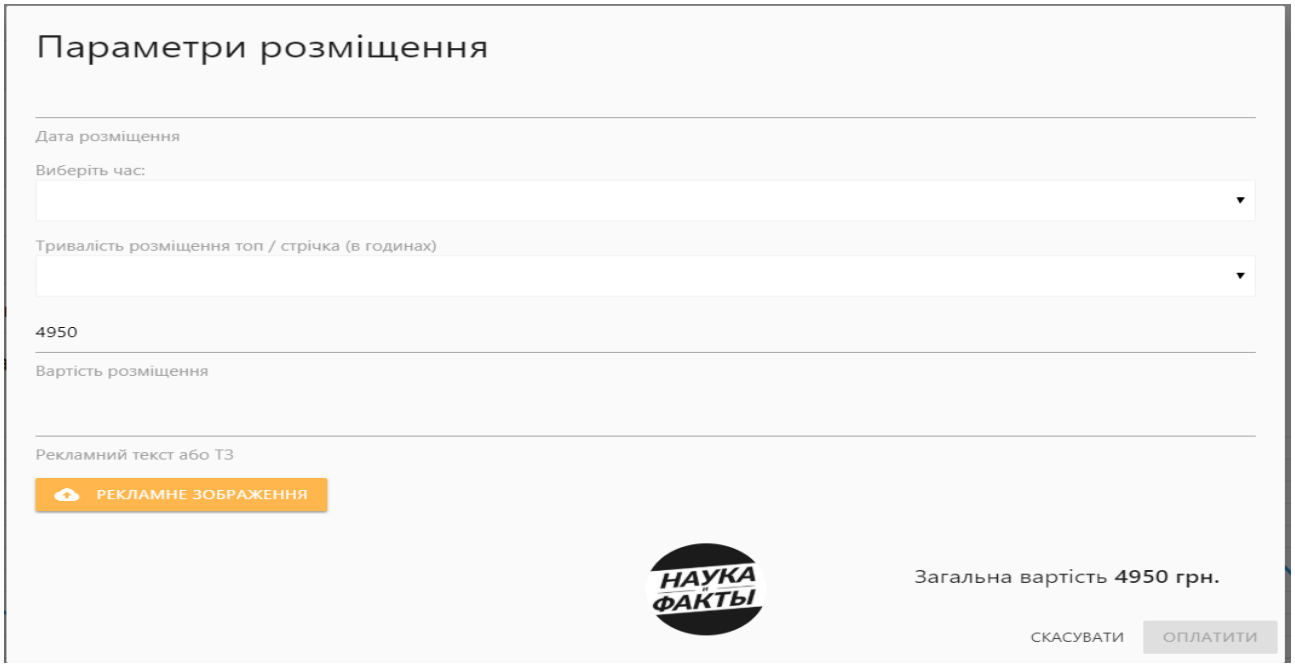


Рисунок 3.16. Створення замовлення

Після створення замовлення власник телеграм-каналу отримає повідомлення на E-mail, а також у майбутньому буде реалізована можливість оповіщення користувача через телеграм-бота.

На сторінці «Історія» (рис. 3.17) користувач зможе швидко знайти усю історію замовлень по його аккаунту:

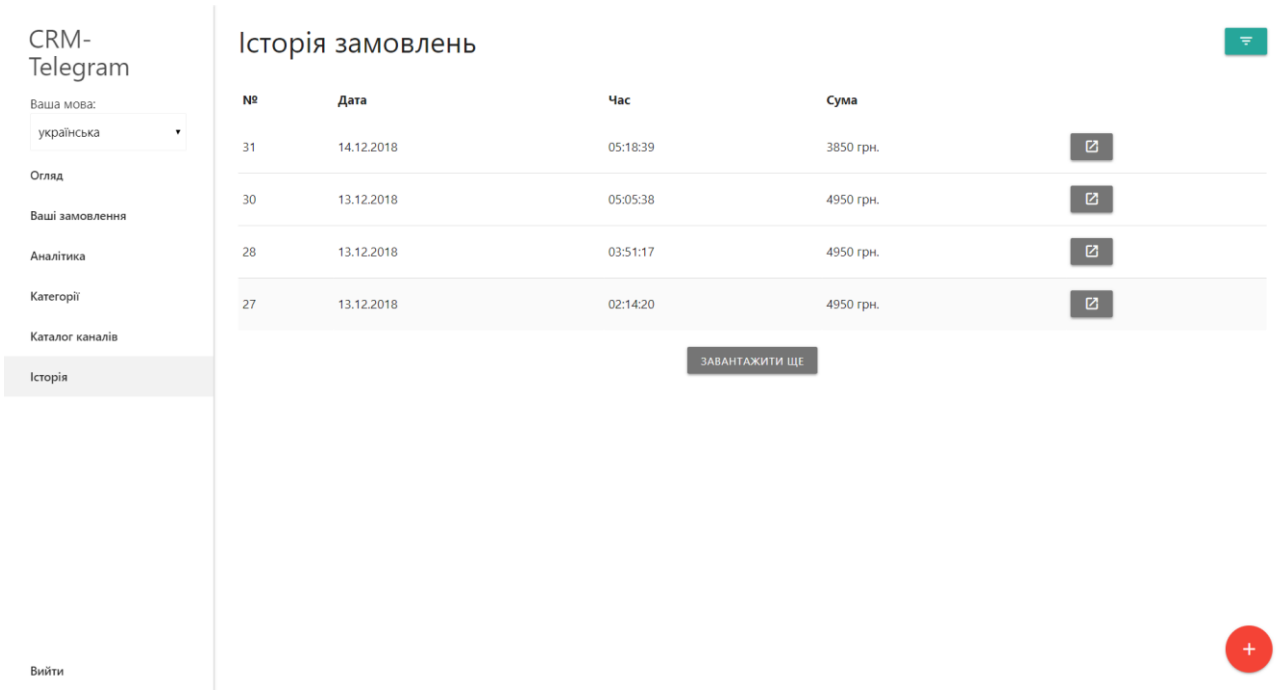


Рисунок 3.17. Сторінка історії

На дану сторінку не виводяться активні замовлення (дата публікація яких ще не прийшла), щоб не плутати користувача, за допомогою зручного фільтру (рис.3.18) можна швидко знайти потрібне замовлення.

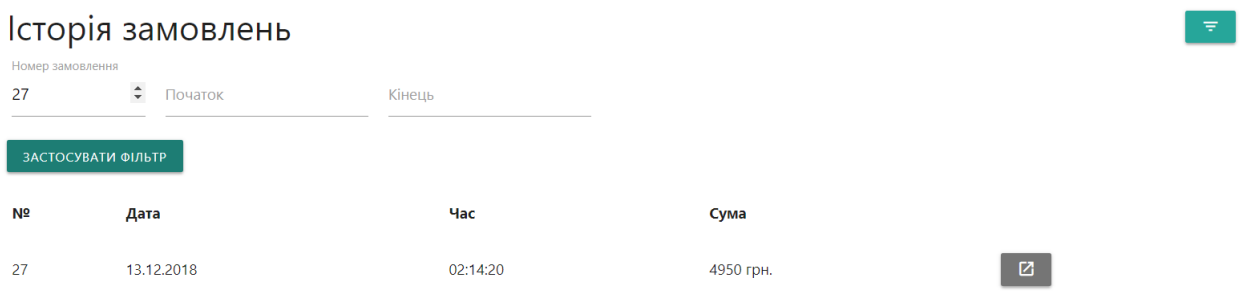


Рисунок 3.18. Зручний фільтр пошуку замовлень

Актуальна версія розробленої системи знаходиться за адресою: https://morning-shelf-31930.herokuapp.com/ Дані для входу: логін – admin1, пароль – 123456.

Висновки до розділу

Було розроблено функціонал та ІК системи купівлі та продажу рекламних повідомлень у месенджері “Telegram”. Система має покроковий інтерфейс і є легким і зручним у використанні для користувача.

Для його реалізації було використано наступні технології:

- Nodejs;
- Модулі Express, Mongoose, Passport.js, Materialize, Moment.js, Chart.js;
- Angular 7;
- JavaScript, TypeScript;
- jQuery;

Для проектування інтерфейсу користувача було розглянуто такі принципи:

- Підхід дружнього інтерфейсу.
- Принцип web інтерфейсу користувача.
- Принципи людського фактору в ІК.

Користувач має можливість за допомогою ІК реєструватися у системі, виконувати логін у свій аккаунт, ознайомлюватися з каталогом доступних каналів для оформлення замовлення.

Користувач може замовити рекламне повідомлення, оплатити його за допомогою встановленої платіжної системи (LiqPay) та очікувати на публікацію.

Публікація рекламного повідомлення у каналі буде виконуватися спеціально-розробленим для проекту телеграм-ботом, що допомогло вирішити основні проблеми, котрі виникали при купівлі повідомлень у ручному режимі.

Реалізована система та ІК повністю відповідає об’єктній моделі та меті роботи даної системи.

РОЗДІЛ 4. МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї проекту

Результати даного дослідження можна використати для створення CRM-системи купівлі та продажу рекламних повідомлень у мессенджері “Telegram” на комерційній основі.

Таблиця 4.1. Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
CRM-система купівлі та продажу рекламних повідомлень у мессенджері “Telegram”	1. Маркетинг	1. Ефективне рішення для безпечної купівлі та продажу рекламних повідомлень. 2. Зменшення витрат часу на дослідження та аналіз ринку, а також на пошук ефективних місць для розміщення рекламних повідомлень.
	2. Розробка CRM-систем	1. Ефективне рішення у сфері автоматизації бізнесу та CRM-систем. 2. Зменшення витрат часу на пошук, дослідження та тестування існуючих на ринку рішень.

Висновок: в таблиці приведені основні напрямки застосування CRM-системи купівлі та продажу рекламних повідомлень у мессенджері “Telegram”. Споживачами є розробники прикладного програмного забезпечення, власники бізнесу, власники телеграм-каналів, блогери, менеджери, маркетологи та аналітики.

Таблиця 4.2. Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W	N	S
		Мій проект	buzz.im	sea-media.ru	socialeg.ru			
1	Ефективність	висока	середня	низька	висока			+
2	Функціональність	висока	висока	середня	не є безкоштовною			+
3	Вартість	безкоштовна	відсоток, середня	відсоток, середня	індивідуальна, висока		+	
4	Багатоплатформність	десктоп, мобайл, telegram	десктоп, мобайл, telegram	десктоп, мобайл	десктоп		+	

Закінчення таблиці 4.2

5	Простота у використанні	присутня	присутня	відсутня	індивідуальна (є помічники-менеджери)		+	
6	Безпечність	висока	висока	середня	висока			+
7	Наявність служби технічної підтримки	присутня	присутня	присутня	присутня		+	
8	Задоволеність покупця	висока	висока	середня	індивідуальна			+

У порівнянні із головними конкурентами програмне рішення має ряд переваг – ефективність, функціональність, безпечність та задоволеність користувача.

4.2 Технологічний аудит ідеї проекту

Таблиця 4.3. Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Багатоплатформність	Багатоплатформний фреймворк “Angular” в основі	Наявні	Доступні
		Залежні від платформи реалізації	Наявні	Доступні
2	Ефективна система оплати	Високотехнологічна система “Unitpay” з різними способами оплати	Наявні	Доступні
		API для введення та виведення коштів із системи	Наявні	Доступні
Обрана технологія реалізації ідеї проекту: 1, 2.				

Висновок: технологічна реалізація продукту – можлива, вибрані технології № 1 та 2.

4.3 Аналіз ринкових можливостей запуску стартап-проетку

Таблиця 4.4. Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	4
2	Загальний обсяг продаж, грн/ум.од	24 000 000

Закінчення таблиці 4.4

3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	45%

За результатами аналізу ринкових можливостей запуску стартап-проекту можна зробити висновок про привабливість ринку для входження за попереднім оцінюванням. Середня кількість гравців свідчить про високий поріг входу та малу конкуренцію при виборі правильного вектору розвитку.

Таблиця 4.5. Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Безпечна купівля та продаж рекламних повідомлень в мессенджері “Telegram”	1.Власники бізнесу. 2.Власники телеграм-каналів. 3.Блогери. 4.Маркетологи. 5.Менеджери.	Використання розроблювано го програмного продукту у різних цілях – купівля, продаж, реклама.	1. Висока безпечність системи. 2. Повна автоматизація процесу. 3. Швидкодія. 4. Задоволеність результатами.

Таблиця 4.6. Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Виникнення більш функціонального та ефективного програмного рішення	Випуск нових версій програмного продукту з розширеним функціоналом
2	Якість	Низька якість програмного продукту	Відмова від продукту
3	Функціонал	Недостача наявного функціоналу для задоволення потреб клієнтів	Розширення функціоналу

Висновки: головними факторами загроз є конкуренція та якість програмного продукту. Вже існуючі рішення на ринку мають певну репутацію і підтримку у споживачів. Конкуренти здатні демпінгувати ціни для отримання нових клієнтів для свого продукту.

Таблиця 4.7. Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Отримання необхідних інвестицій	Додатковий до початкового капітал, який допоможе долучити більше розробників до розробки системи, що дасть можливість реалізувати нові функції більш швидше	Розробка життєздатного продукту з максимальним з доступного набором функціоналу
2	Освоєння нових сфер	Використання системи у інших соціальних мережах	Створення спеціальної групи розробників та маркетологів, которі будуть займатися розробкою під обрані соціальні мережі
3	Розширення аналітичних можливостей	Використання інформації, яка міститься у системі з метою отримання нових знань про сферу та аналіз отриманих даних	Масштабування функціоналу системи
4	Успішна маркетингова політика	В результаті проведеної маркетингової політики отримана висока зацікавленість користувачів	Підтримання стабільної роботи та розвитку системи. Введення платних преміум-можливостей для користувачів

Закінчення таблиці 4.7

5	Вихід аналогу	Надати продукт з певними характеристиками та можливостями що відсутні у компаній конкурентів	Аналіз ринку та користувачів задля задоволення їх потреб та надання функціональності у найкоротші строки за ціну, котра є дешевшою ніж у продуктів-замінників
6	Зворотній зв'язок від користувачів	Можливість отримання необхідної інформації для вдосконалення продукту	Наявність вхідних даних та реакція на них з боку команди розробників задля задоволення потреб та бажань кінцевих користувачів системи кешування даних.
7	Грошова винагорода за рекламу	При достатньому попиту на систему можлива комерціалізація продукту на основі реклами задля отримання грошової винагороди для подальшого розвиту продукту та оплати заробітної плати працівникам	Точкова комерціалізація продукту; Введення реклами; Ведення додаткових коштів у проект задля його подальшого розвитку.

Висновки: сфера швидко розвивається, ринок клієнтів постійно зростає. Збільшення зацікавленості в продукті призведе до різкого збільшення об'ємів купівлі та продажу, що дасть поштовх до розробки нового функціоналу. Це досягається шляхом рекламування та освоєння нових сфер.

Таблиця 4.8. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції: Олігополія	Незначна кількість конкурентів Велика ринкова сила Схожість вик. технологій	Інформування ринку щодо появи нової системи Співпраця із провідними конкурентами

Закінчення таблиці 4.8

2. За рівнем конкурентної боротьби: Галузева	Загроза появи нових конкурентів Ринкова влада споживачів Висока потреба у товарі	Інформування ринку щодо якості використовуваної новаторської технології Пропозиція гнучких цін
3. За галузевою ознакою: Міжгалузева	Діяльність в декількох галузях економіки Надання продуктів одного типу	Зменшення вартості сервісу Примноження каналів розподілу
4. Конкуренція за видами товарів: Товарно-видова	Надання різних продуктів одного виду	Маркетингова політика
5. За характером конкурентних переваг: Цінова	Використання цін для покращення економічних умов збуту	Зменшення вартості сервісу Використання нових каналів розподілу
6. За інтенсивністю: Марочна	Пропозиція схожого сервісу Спільна цільова аудиторія	Інформування ринку щодо якості використовуваної новаторської технології

Таблиця 4.9. Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товаризамінники
	Buzz.im, sea-media.ru, socialeg.ru	Середні бар'єри входження на ринок	Адміністратори телеграм-каналів	Рівень чутливості до цін, продуктова диференціація: якість, спосіб отримання продукту, швидкість роботи з системою	Копіювання функціоналу, зменшення цін
Висновки:	CR4 = 70% Індекс ХерфіндаляХіршмана (HHI) = 5878 Значення показників вказує на високу концентрацію (монополізацію) даного ринку	Можливості виходу на ринок середні, потенційні конкуренти на даний момент відсутні	Самостійно встановлюють ціни на рекламу, що призводить до великого впливу даного учаснику ринку	Клієнти диктують умови гнучкості цінової політики, високої і довгострокової якості підтримки продукту та наявність кооперації із сервісами та продуктами суміжних компаній, що вони використовують	Пропонування вигідних умов, забезпечення захисту інтелектуальної власності, гнучкість цінової політики

Висновки: проаналізувавши можливості роботи на ринку з огляду на конкурентну ситуацію можна зробити висновок: оскільки кожний з існуючих продуктів не впливає у великій мірі на поточну ситуацію на ринку в цілому, кожний з існуючих продуктів має свої позитивні та негативні сторони щодо рішення певних типів задач, то робота та вихід на даний ринок є можливою і реалізованою задачею.

Для виходу на ринок продукт повинен мати функціонал що відсутній у продуктів-аналогів, повинен задовольняти потреби користувачів, мати необхідний та достатній функціонал з конфігурування, підтримку зі сторони розробників.

Таблиця 4.10. Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Унікальність системи	Система забезпечує високу ефективність та безпечність в період роботи користувача з нею
2	Функціональність	Висока функціональність, яка дозволить потенційному користувачу виконати всі необхідні завдання у мінімальний проміжок часу
3	Цінова політика	Отримання прибутку здійснюється за рахунок користувачів або отримання відсотків з їх прибутку. Даний підхід дозволить обійти цінову конкуренцію на ринку цільової аудиторії

Висновки: оцінено основні фактори конкурентної спроможності. Система забезпечує високу ефективність та функціональність завдяки інноваційним технологіям та особливостям. Простота використання, гнучкість реалізації та цінова політика робить продукт більш привабливим для клієнта.

Таблиця 4.11. Порівняльний аналіз сильних та слабких сторін «системи купівлі та продажу рекламних повідомлень у “Telegram”

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим						
			-3	-2	-1	0	1	2	3
1	Унікальність системи								+
2	Функціональність								+
3	Цінова політика						+		

Висновки: спираючись на фактори конкурентоспроможності та підсумовуючи рейтинг товару відносно головного конкурента, запропонований продукт має більший рейтинг відносно прямих конкурентів. Дана таблиця показує якими саме особливостями розроблений продукт відрізняється від аналогів та в яку саме сторону.

Таблиця 4.12. SWOT- аналіз стартап-проекту

Сильні сторони:	Слабкі сторони:
Ефективність Задоволеність користувача Безпечність Висока функціональність	Недостача стартових капіталовкладень Бізнес модель залежить від політики окремих бізнесів
Можливості:	Загрози:
Інвестиції Реалізація бізнес-моделей Висока зацікавленість цільової аудиторії	Крадіжка інтелектуальної власності Відмова суміжних компаній у співпрац

Таблиця 4.13. Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Купівля готової CRM-системи з необхідним функціоналом для впровадження	Мало ймовірне	-
2	Маркетингова компанія для приваблювання користувачів	Ймовірне	6 місяців
3	Пропонування безкоштовного продукту з платними можливостями	Дуже ймовірне	3 місяці
4	Пошук клієнтів у інших галузях	Дуже ймовірне	4 місяці

4.4 Розроблення ринкової стратегії проекту

Таблиця 4.14. Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Власники бізнесу	Середня	Середній	Висока	Середня
2	Власники телеграм-каналів	Висока	Високий	Висока	Низька
3	Маркетологи	Низька	Низький	Низька	Висока
4	Менеджери	Висока	Високий	Висока	Низька
5	Блогери	Висока	Високий	Середня	Низька
6	Аналітики	Низька	Низький	Низька	Висока
Які цільові групи обрано: власники бізнесу, власники телеграм-каналів, блогери.					

Відповідно до проведеного аналізу можна зробити висновок, що підходящою цільовою групою для розповсюдження даного програмного продукту є власники бізнесів та телеграм-каналів, а також блогери та менеджери. Відповідно до стратегії охоплення ринку збуту товару обрано стратегію масового маркетингу, оскільки для бізнесу та інших потенційних споживачів у цілому надається стандартизований продукт з можливістю розширення функціональності за домовленістю (платно).

Таблиця 4.15. Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Надання функціональності що відсутня у товарів-замінників, підтримка клієнтів	Проведення реклами, освітлення унікальної функціональності через інтернет ресурси та інші канали, контакт напряму з споживачами; формування лояльності і прихильності споживачів	Здатність протистояти прямим конкурентам Низькі витрати Ефективна співпраця посередників Прихильність клієнтів Відмітні властивості продукту Відмітні характеристики продукту	Стратегія диференціації

Таблиця 4.16. Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Ні, оскільки є товари-заміннники, але дані товари замінники не мають деякого необхідного функціоналу	Так, ціль компанії знайти нових споживачів та, частково, забрати існуючих у	Компанія частково копіює характеристики товару конкурента, основна ціль	Стратегія заняття конкурентної ніші

		конкурентів задля задоволення потреб останніх	компанії розробка нового унікального функціоналу, з підтримкою основного функціоналу конкурентів	

Таблиця 4.17. Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартаппроекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Високі показники ефективності та функціоналу	Стратегія диференціації	Формування регулярного попиту Виявлення нових груп споживачів Нові напрями застосування існуючого продукту	Захист авторського права Інноваційність технології Простота використання

Відповідно до проведеного аналізу можна зробити висновок, що стартап-компанія вибирає як базову стратегію розвитку – стратегію диференціації, як базову стратегію конкурентної поведінки – стратегію заняття конкурентної ніші.

4.5 Розроблення маркетингової програми стартап-проекту

Таблиця 4.18. Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Високі показники ефективності	Підвищення показників продуктивності за допомогою використання системи	Якість продукту Інноваційність підходу Цінова перевага
2	Легкість інтеграції	Розробка можливості швидкого використання	Інноваційність підходу Простота реалізації

		системи у схожих соціальних мережах для продажу та купівлі рекламних повідомлень	

Таблиця 4.19. Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Система купівлі та продажу рекламних повідомлень у мессенджері “Telegram”		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Комплексні метрики ефективності	М	Е
	Якість: стандарти, рівень оптимізації написаного коду, коректність використаних технологій, звіти з тестування програмного забезпечення		
	User Manual		
	Марка: CRM Telegram		
III. Товар із підкріпленням	До продажу: наявна повна документація, акції на придбання додаткових функцій, знижки для певних сегментів на покупку		
	Після продажу: додаткова підтримка спеціалістів налаштування, підтримка з боку розробника		
За рахунок чого потенційний товар буде захищено від копіювання: за рахунок технік обфускації програмного коду товар буде захищеним від копіювання			

Висновки: основними засобами захисту від копіювання є використання технік обфускації програмного рішення. Закладені характеристики на другому та третьому рівнях товару робить його унікальним серед конкурентів.

Таблиця 4.20. Визначення меж встановлення ціни

№ п/п	Рівень цін на товаризамінники	Рівень цін на товарианалоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	-	$ 1500	$ 2000-5000	$1000 /10000

Таблиця 4.21. Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Закупівля здійснюється через довірені джерела	Інформування користувачів Доступ користування сервісом	Канал одного рівня	Селектив на з використа нням комбінованого каналу збуту

Таблиця 4.22. Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Ведення бізнесу Ведення телеграм-каналу Блоггінг Менеджмент	Прямі, неофіційні	Послідовність в реалізації обраної позиції Доступність та об’єктивність інформації про фірму і товар Унікальність послуг	Формування у цільової аудиторії обізнаності про появу нового продукту Інформування користувачів про властивості та переваги продукту Інформування користувачів про нові способи використання відомого продукту	Раціоналістична стратегія реклами

Як результат було створено ринкову (маркетингову) програму, що включає в себе визначення ключових переваг концепції потенційного товару, опис моделі товару, визначення меж встановлення ціни, формування системи збуту та концепцію маркетингових комунікацій.

Висновки по розділу

В четвертому розділі описано стратегії та підходи з розроблення стартап-проекту, визначено наявність попиту, динаміку та рентабельність роботи ринку, як висновок було вказано що існує можливість ринкової комерціалізації проекту. Розглянувши потенційні групи клієнтів, бар’єри входження, стан конкуренції та конкурентоспроможність проекту було встановлено що проект є перспективним. Розглянуто та вибрано альтернативу впровадження стартап-проекту та доведено доцільність подальшої імплементації проекту.

ВИСНОВКИ

У ході виконання магістерської дисертації було розглянуто питання пов’язані з необхідністю та актуальністю створення системи продажу та купівлі рекламних повідомлень у мессенджері “Telegram”. Наведено характеристику предметного середовища та обґрунтовано причину розробки системи. Описано вимогу до додатку та технології створення CRM-системи.

На основі даних, отриманих в процесі аналізу було сформульовано задачу створення системи продажу та купівлі на основі платформи Node.js.

Для розробки системи було використано мову програмування Javascript та Typescript. Для створення frontend частини проекту було використано фреймворк “Angular” 7 версії, для backend частини – Express.js. У якості бази даних було обрано “MongoDB” та спеціальний модуль ODM “mongoose”, який допоміг значно спростити та прискорити процес розробки. База розміщена на віддаленому хмарному-сховищі сервісу “mLab”. Все перераховане є широко вживаними та безкоштовними інструментами, що дозволить легко розширити функціонал системи у разі виникнення відповідної потреби.

Розроблена та наведена інструкція користувача по експлуатації системи продажу та купівлі рекламних повідомлень. Наведено приклади використання створеного додатку.

Проведений маркетинговий аналіз стартап-проекту. Проведено опис ідеї проекту, технологічний аудит ідеї проекту, аналіз ринкових можливостей запуску стартап-проекту. Розроблено ринкову стратегію проекту та маркетингову програму.

Результати роботи над магістерською дисертацією опубліковані у одній статті.

Наукова новизна одержаниї результатів магістерської дисертації полягає у наступному:

вперше:

- проаналізовано стандартний ручний підхід до купівлі та продажу рекламних повідомлень у мессенджері “Telegram”.
- надана можливість повністю автоматизувати процеси купівлі та продажу рекламних повідомлень та зробити їх без втручання людини.

удосконалено:

- для зменшення кількості пам’яті, що використовує система для її реалізації було обрано легкий фреймворк.
- усі subscribe-оператори у системі завершують свою діяльність після перемикання користувачем сторінок.
- було додано більш зручну та детальну методику додавання каналів у каталог та рекламних повідомлень для них (допоможе просунутим користувачам системи).

здобуло подальший розвиток:

- реалізація спеціального телеграм-боту за допомогою якого користувач зможе користуватися системою напряму через одноіменний мобільний додаток.

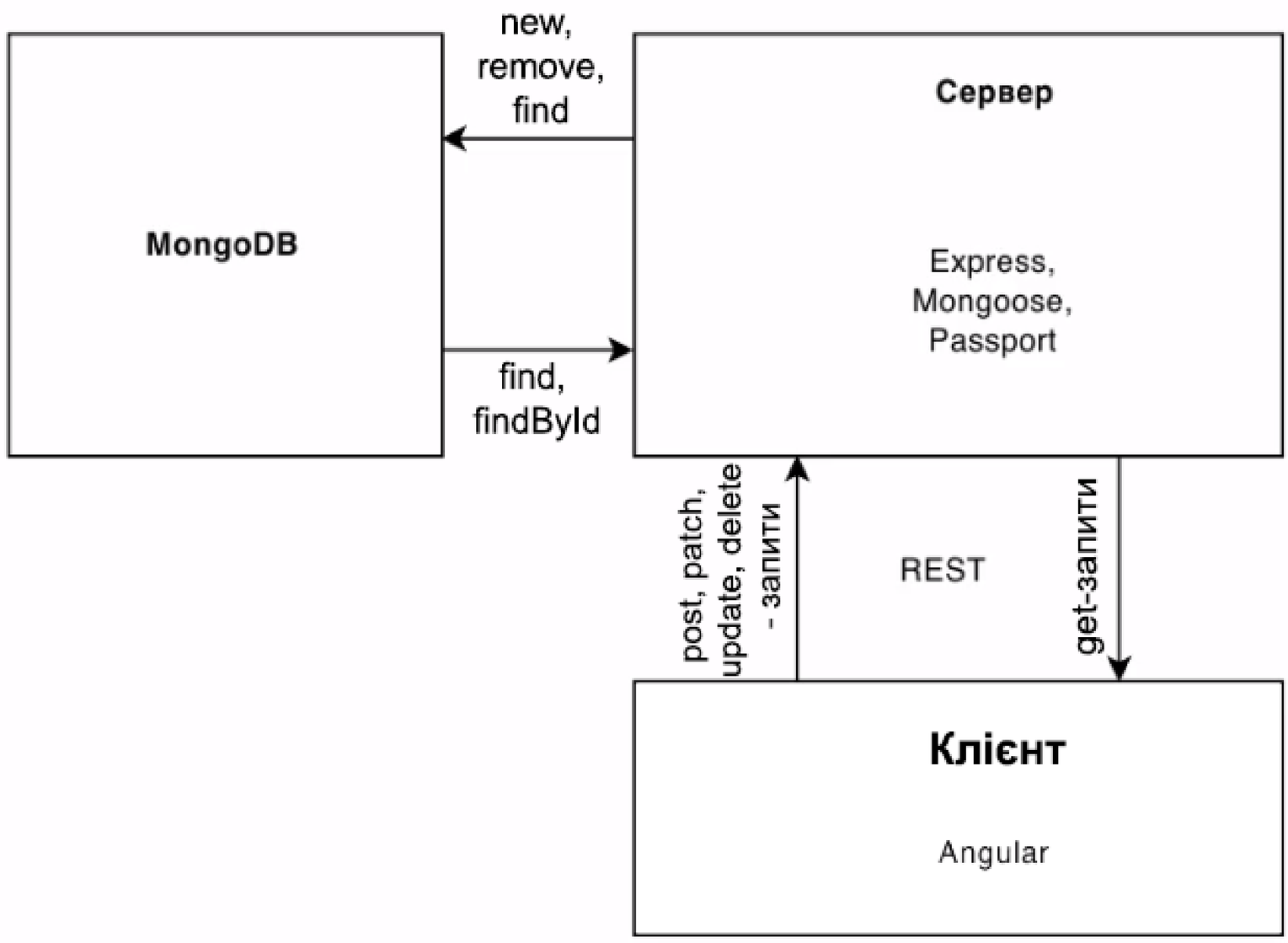
ПЕРЕЛІК ПОСИЛАНЬ

1. Управління відносинами з клієнтами [Електронний ресурс] // Режим доступу: https://uk.wikipedia.org/wiki/Управління_відносинами_з_клієнтами
2. Архітектура REST [Електронний ресурс] // Режим доступу: https://habr.com/post/38730/
3. Елементи даних в REST [Електронний ресурс] // Режим доступу: https://uk.wikipedia.org/wiki/REST
4. Ethan Brown. Web Development with Node and Express: Leveraging the JavaScript Stack. 2014. 332c.
5. Введение в Mongoose [Електронний ресурс] // Режим доступу: https://code.tutsplus.com/ru/articles/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527
6. Документація фреймворку “Angular” [Електронний ресурс] // Режим доступу: https://angular.io/docs
7. Документація “Materialize” [Електронний ресурс] // Режим доступу: https://materializecss.com/
8. Рынок рекламы в Telegram после блокировки сократится на треть [Електронний ресурс] // Режим доступу: https://thebell.io/rynok-reklamy-v-telegram-posle-blokirovki-sokratitsya-na-tret/
9. MEAN (software bundle) [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/MEAN_(software_bundle)
10. Стек MEAN. Пример использования [Електронний ресурс] // Режим доступу: https://habr.com/company/piter/blog/279237/
11. MySQL и MongoDB — когда и что лучше использовать [Електронний ресурс] // Режим доступу: https://habr.com/post/322532/
12. Поморова О.В. Проектування інтерфейсів користувачів. 76с.
13. Материальный дизайн [Електронний ресурс] // Режим доступу: https://www.motocms.com/blog/ru/materialny-dizayn/

14. Material Design [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/Material_Design
15. Инструкция: Как создавать ботов в Telegram [Електронний ресурс] // Режим доступу: https://habr.com/post/262247/
16. Customer loyalty through customer satisfaction and CRM [Електронний ресурс] // Режим доступу: https://www.cas-crm.com/crm-its-benefits/crm-glossary/customer-loyalty.html
17. “Management Study Guide”. Increasing Customer Loyalty [Електронний ресурс] // Режим доступу: https://www.managementstudyguide.com/increasing-customer-loyalty.htm
18. Раянов Р. Как создать собственную CRM. 2015. 10 с.

ДОДАТКИ

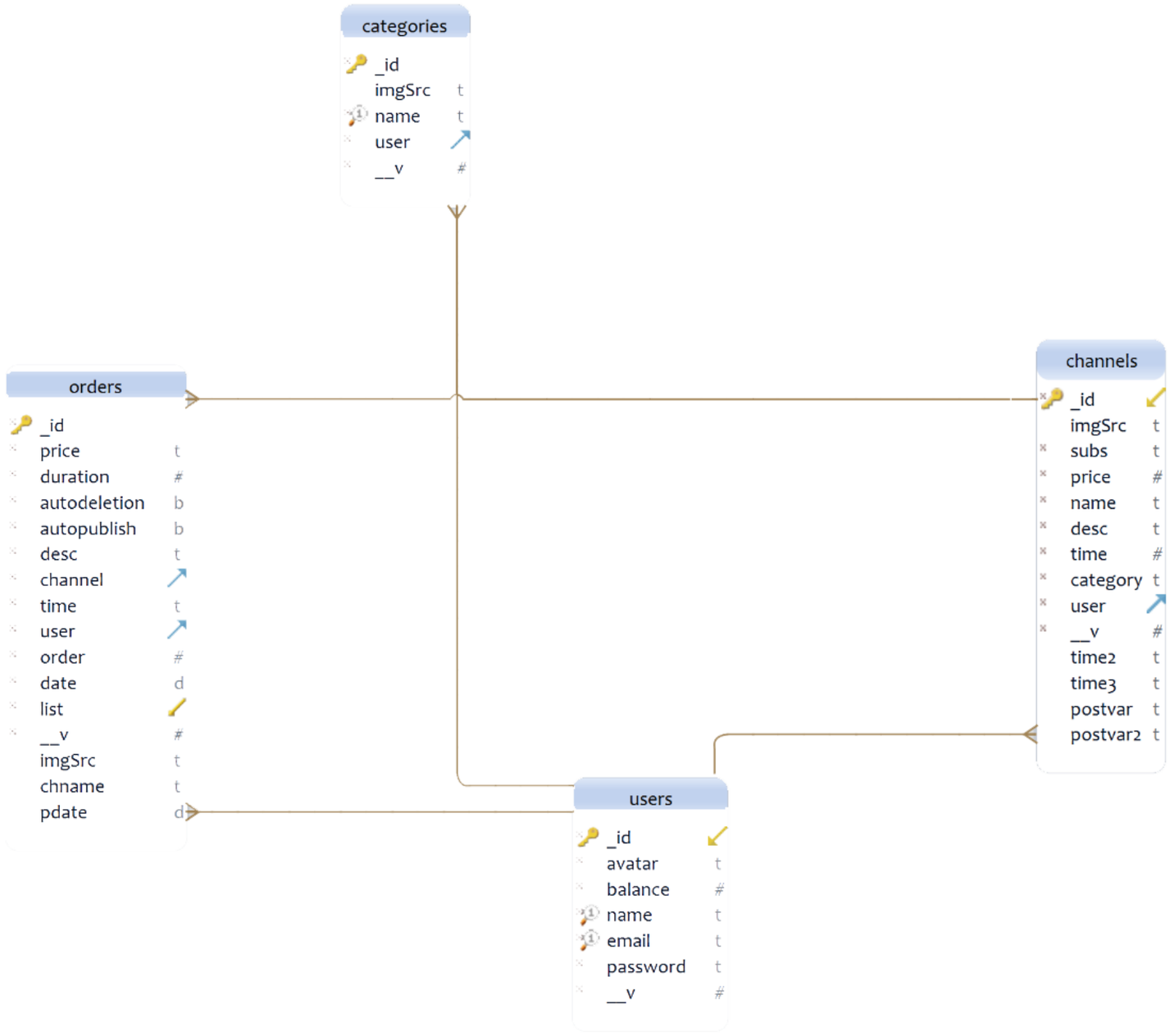
Архітектура додатка



Демонстраційний плакат №1
до магістерської дисертації на тему
„Система продажу та купівлі рекламних повідомлень в Telegram”

Розробив: Муренко В.В.
Прийняв: Дьяков С.О.

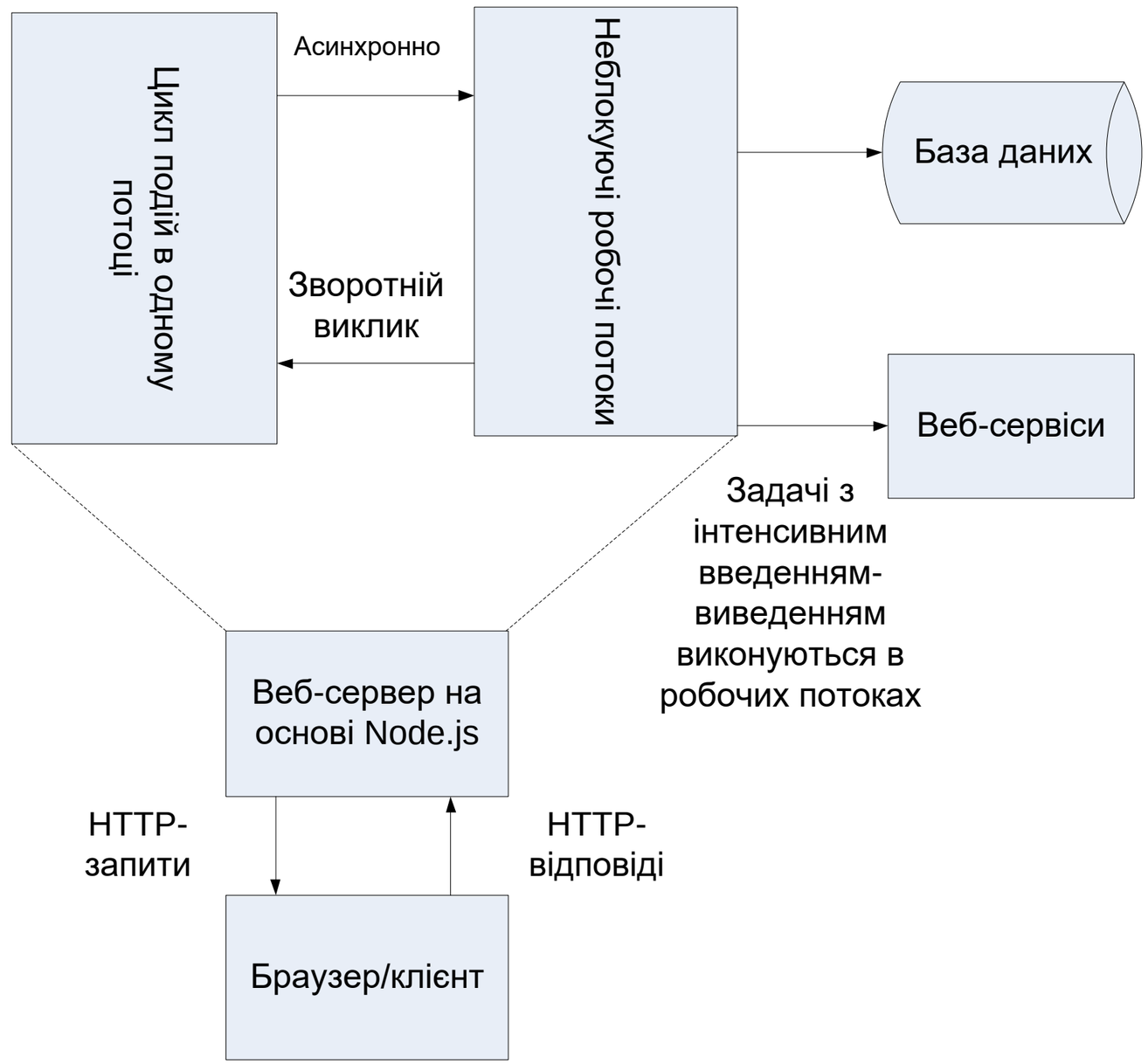
Схема БД системи



Демонстраційний плакат №2
до магістерської дисертації на тему
„Система продажу та купівлі рекламних повідомлень в Telegram”

Розробив: Муренко В.В.
Прийняв: Дьяков С.О.

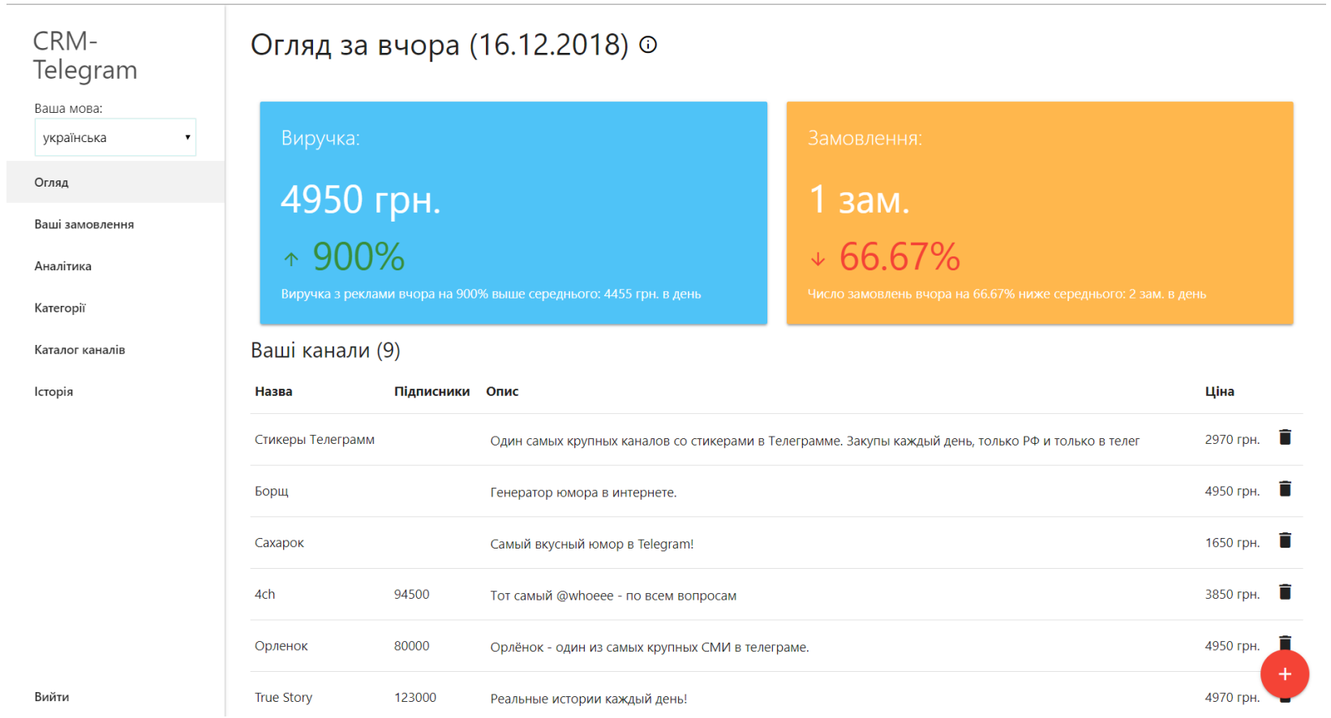
Схема управління «запит-відповідь» в Node.js



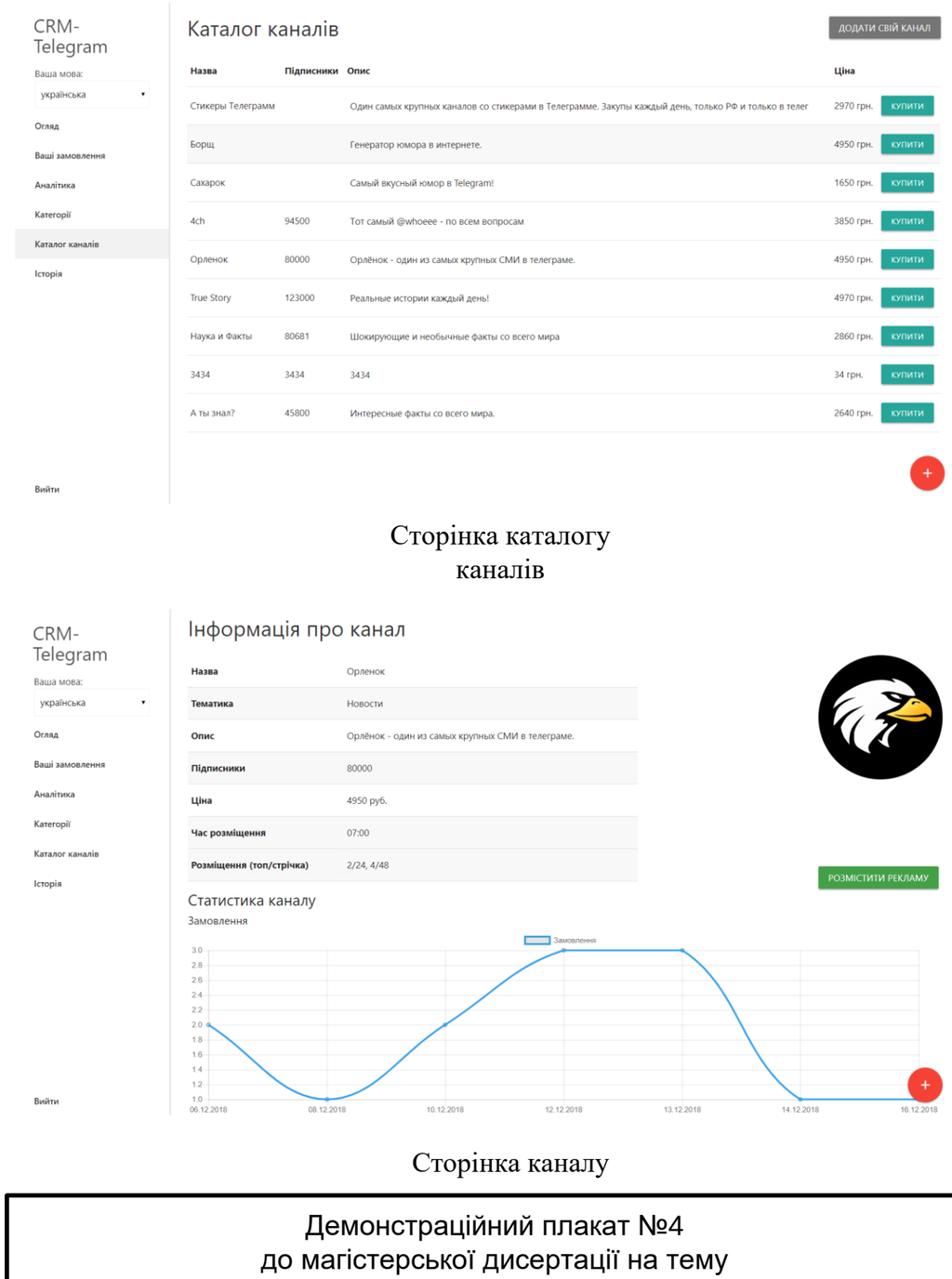
Демонстраційний плакат №3
до магістерської дисертації на тему
„Система продажу та купівлі рекламних повідомлень в Telegram”

Розробив: Муренко В.В.
Прийняв: Дьяков С.О.

Інтерфейс системи



Головна сторінка системи



Сторінка каталогу каналів

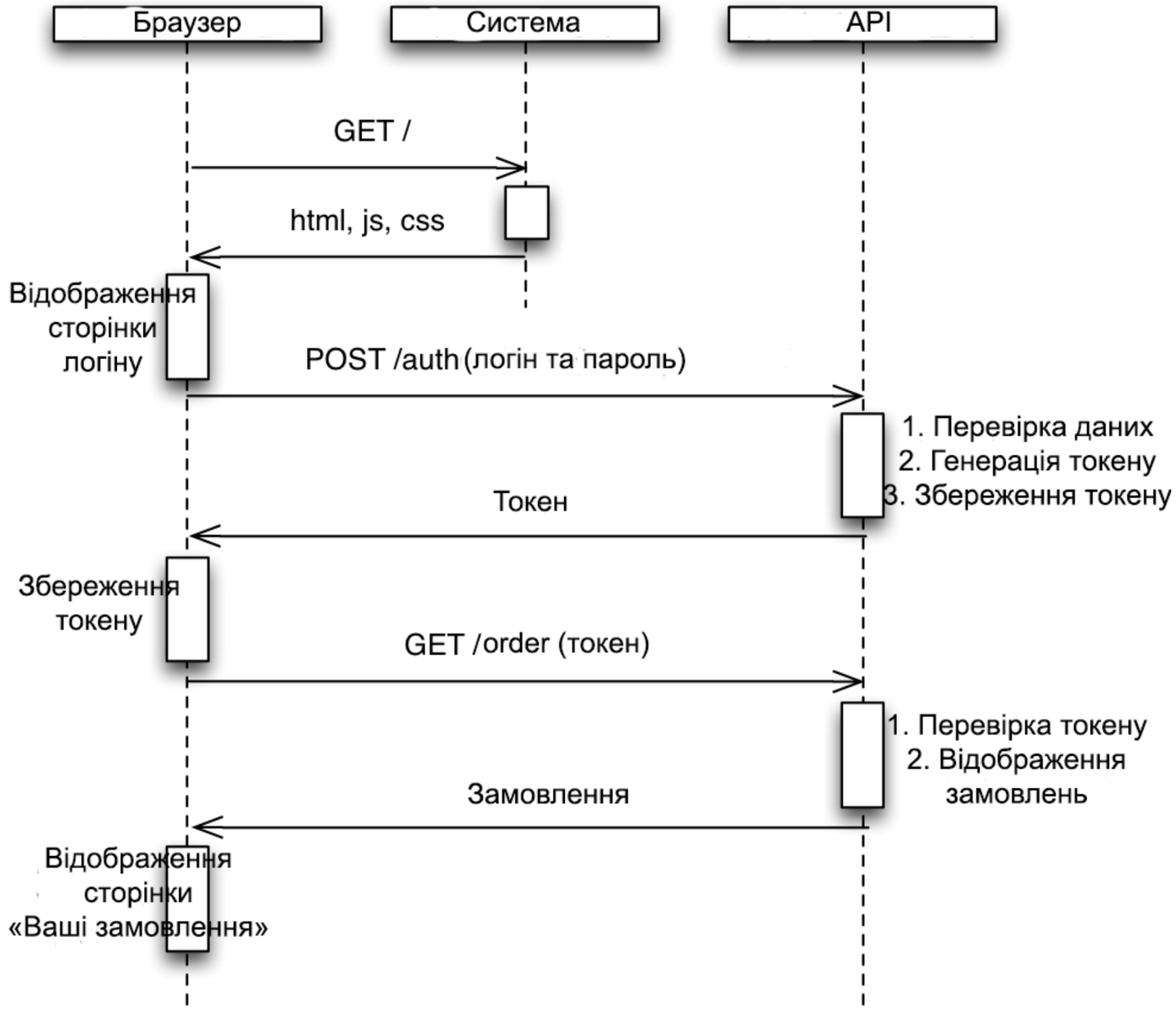
Сторінка аналітики

Сторінка каналу

Демонстраційний плакат №4
до магістерської дисертації на тему
„Система продажу та купівлі рекламних повідомлень в Telegram”

Розробив: Муренко В.В.
Прийняв: Дьяков С.О.

Принцип роботи системи та API



Демонстраційний плакат №5
до магістерської дисертації на тему
„Система продажу та купівлі рекламних повідомлень в Telegram”

Розробив: Муренко В.В.
Прийняв: Дьяков С.О.

Блок-схема реєстрації у системі

Демонстраційний плакат №6
до магістерської дисертації на тему
„Система продажу та купівлі рекламних повідомлень в Telegram”

Розробив: Муренко В.В.
Прийняв: Дьяков С.О.