Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та
комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету(відділення))

Кафедра інформаційних технологій
(повна назва кафедри (предметної циклової комісії))

Пояснювальна записка

до дипломної роботи

другий (магістерський)

(рівень вищої освіти)

на тему: «Розроблення маркетингового порталу засобами Laravel»

Виконав студент 6 курсу, групи КН(м)
спеціальності:
122 „Комп'ютерні науки"
(шифр і назва напрямку підготовки спеціальності)

Папроцький Р.М.
(прізвище, ініціали)

Керівник: Крошний І.М.
(прізвище, ініціали)

Рецензент:_____
(прізвище, ініціали)

Львів-2021

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ деревооброблювальних та комп'ютерних технологій і дизайну

Кафедра Інформаційних технологій

Рівень вищої освіти другий (магістерський)

Галузь знань 122 „Інформаційні технології"

Спеціальність 122 *„Комп'ютерні науки"*

ЗАТВЕРДЖУЮ:

Завідувач кафедри ІТ

_____Крошний І.М.

„___" _____ 2021.

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Папроцький Роман Мирославович

(прізвище, ім'я, по батькові)

1.Тема магістерської роботи: Розроблення маркетингового порталу засобами Laravel

керівник роботи Крошний І.М. , к.т.н., доцент ,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "31" грудня 2020 року, № С-593

2.Термін подання студентом проекту(роботи) 10 грудня 2021р

3. Вихідні дані до проекту (роботи) Розробити веб-орієнтоване програмне забезпечення для маркетингової розкрутки та продажу товарів різної необхідності. Реалізувати зручний та простий інтерфейс для використання розроблених функцій проекту. Створити інтуїтивно зрозумілу навігацію посистемі.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Стан проблемної області

Інформаційне забезпечення

Математичне забезпечення

Програмне забезпечення

Розроблення стартап проекту

Висновки

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

Підготовка матеріалу до доповіді .

6. Консультанти розділів проекту (роботи)

7. Дата видачі завдання 18 грудня 2020р.

КАЛЕНДАРНИЙ ПЛАН

№, з/п	Етапи бакалаврської роботи	Термін виконання етапів роботи	Примітка
1.	Огляд літератури згідно досліджуваної теми. Збір необхідних матеріалів.	09.03.21-28.03.21	
2.	Постановка задачі і її формалізація	28.03.21-20.04.21	
3.	Виконання вхідного етапу технології	20.04.21-24.05.21	
4.	Реалізація головних алгоритмів проекту	24.05.21-10.06.21	
5.	Виконання етапу відлагодження проєкту	10.06.21-25.06.21	
6.	Виконання етапу впровадження та випуску бета-версії.	25.06.21-29.10.21	
7.	Оформлення записки до дипломного проекту.	01.11.21-10.12.21	

Студент _____ Папроцький Р.М.

(підпис) (прізвище та ініціали)

Керівник роботи _____ Крошний І.М.

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота містить 55 сторінки пояснювальної записки, 45 рисунків, 1 додаток, 5 джерел.

Дипломна робота присвячена розробці та створенню веб-маркетингової системи для продажу товарів в мережі інтернет. У проекті обгрунтовано актуальність вибраного проекту та реалізоване програмне забезпечення засобами фреймворку Laravel та CMS October. Інтерфейс програмного забезпечення інтуїтивно зрозумілий та готовий для збирання лідів.

Ключові слова: CMS October, Laravel, Програмне забезпечення.

ABSTRACT

Thesis contains 55 pages of explanatory note, 19 figures, 5 tables, 1 appendix, 12 sources.

Thesis is devoted to the development and creation of a web marketing system for the sale of goods on the Internet. The project substantiates the relevance of the selected project and implemented software using the Laravel and CMS October framework. The software interface is intuitive and ready to collect leads.

Keywords: CMS October, Laravel, Software.

ТЕХНІЧНЕ ЗАВДАННЯ

У відповідності з вимогами технічного завдання, необхідно розробити інтелектуальну веб-платформу з використанням фреймворку Laravel та CMS October. Головною задачою є збільшення продаж товарів та збір статистичних даних по продажах. Також потрібно врахувати створення реферальної системи для користувачів платформ.

ПЕРЕЛІК СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ОС – Базовий комплекс програм, що виконує управління апаратною складовою комп'ютера або віртуальної машини; забезпечує керування обчислювальним процесом і організовує взаємодію з користувачем.

ПЗ – Програмне забезпечення.

UI – Інтерфейс.

ІС – Комплекс програмних, лінгвістичних і логіко-математичних засобів для реалізації основного завдання: здійснення підтримки діяльності людини і пошуку інформації в режимі розширеного діалогу на природній мові.

ЗМІСТ

ВСТУП

Задача контент-маркетингу – перетворювати відвідувачів сайту на ліди. До речі, якщо порівняти з традиційною рекламою, то контент генерує в три рази більше клієнтів та коштує на 62% менше.

- Контент приваблює нових користувачів, які раніше не знали про вас;
- публікації, відео або статті викликають зацікавлення, користувач підписується та стає постійним читачем/глядачем;
- згодом він починає довіряти вам та робить першу покупку;
- якщо ви пропонуєте якісний товар та ще й відмінне обслуговування, користувач здійснює ще одну покупку та стає постійним клієнтом;

Продовжуючи отримувати корисний контент та якісний продукт, постійний клієнт стає адвокатом бренду. Він залишає гарні відгуки та рекомендує вас знайомим.

Об’єктом дослідження є маркетингова веб-платформа для продажі товарів, через мережу інтернет.

Метою роботи є розробка системи, яка збільшить продажі в будь якій галузі.

Предметом дослідження є налагодження продуктивного процесу продаж та реклами у мережі інтернет.

Практичне значення роботи базується на використанні оптимально простих у практиці методів, які дають користувачам програмний інструмент готовий до вирішення проблемної задачі у найкоротший відрізок часу.

Наукова новизна – це розроблення швидкої веб-системи для продаж з підрахунком та збору статистичних даних. Впроваджено алгоритми інтелектуального аналізу даних для збільшення продаж та налаштування реферальної системи.

Актуальність програмного забезпечення полягає створенні швидкої платформи продаж товарів з врахування реферальної системи.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Огляд проблемної області.

Сучасний інтернет-маркетинг - це не тільки просування в інтернеті за допомогою контекстної реклами або пошукової оптимізації. З'являються нові інструменти, і відмінності між існуючими іноді стираються. Раніше контекстна і медійна реклама була незалежною і запускалася окремо, тепер вони інтегровані в один інтерфейс рекламного кабінету. Раніше seo, контент-маркетинг і PR були розділені, тепер вони працюють разом і скоріше доповнюють один одного. Сьогодні всі інструменти інтернет-маркетингу об'єднуються, щоб привести рекламодавця до запланованих цілей: знаходити і залучати користувачів, підвищувати впізнаваність бренду, підвищувати продажі. Підходити до інтернет-маркетингу цілісно. Початок впровадження інтернет-маркетингу

Позиціонування. Інтернет-маркетинг починається з визначення поточних позицій. Перед впровадженням проаналізуйте рейтинг сайту або цільової сторінки при видачі, обсяг і джерела трафіку, конверсії і зручності використання сайту. Інформацію про трафік і конверсії, дані про відвідувачів візьміть з аналітичної системи Google.Analytics. Дізнайтеся позиції за допомогою сервісу Topvisor або MegaIndex. Оцініть зручність використання експертним методом. Окремо проаналізуйте конкурентів, наприклад, в сервісі Similarweb, і порівняйте показники. Подивіться на їх веб-сайтита стратегії онлайн-просування: чи є якісь корисні ідеї для вашого бізнесу. Дізнайтеся, з якими підрядниками вони працюють. Якщо ви хочете обігнати їх, вам доведеться рухатися краще.

Стратегія. Визначте основну мету. рухати бізнес вгору, хоча дохід збільшується. Ви можете встановити цілі у вигляді обсягу трафіку або розмістити в пошукових видачах. Якщо у вас є інформаційний ресурс, ви зберете велику аудиторію. Однак інтернет-магазинам,наприклад, потрібно пам'ятати, що велика кількість відвідувачів не завжди приносять великі продажі. Розробити стратегію просування. Розставте завдання і виберіть інструменти інтернет-маркетингу.

Визначте проміжні терміни і показники, яких плануєте досягти до цього часу. Складіть стратегію у вигляді документа. Щоб зробити її реалістичною, узгодити її з співробітниками, які будуть брати участь в її реалізації.

Реклама. В інтернеті є контекстна, медійна, цільова, банерна і RTB реклама. Межі між ними розмиті, тому є сенс ділити рекламу не за типом, а за каналами: пошуковими системами,исоціальними мережами, мобільними додатками, відеоканалами і т.д. Запустіть рекламну кампанію комплексно. Для кожного креативу використовуйте відповідну цільову сторінку. Якщо ви рекламуєте певний продукт або послугу,перейміть відвідувача на цільову сторінку з описом і формою замовлення. При просуванні конкурсу відвідувач повинен бути доставлений на сторінку конкурсу або пост з умовами в соціальних мережах. Переконайтеся, що виклики в оголошенні та на цільовій сторінці збігаються. Це збільшить кількість конверсій. Реклама одного продукту для різних цільових аудиторій, створення мульти-посадок. Заклик до дії на них змінюється в залежності від тексту оголошення, згідно з яким приходить користувач інтернету.

SEO. Просування сайту включає в себе великий спектр робіт з внутрішньої і зовнішньої оптимізації. Спочатку усувайте технічні помилки, працюйте над структурою сайту,додавайте необхідну інформацію. Потім просувайте сайт з урахуванням вимог Яндекса і Google. Пошукові системи регулярно оновлюють алгоритми, тому SEO роблять безперервно, щоб не втратити отримані результати.

SMM (Маркетинг в соціальних мережах). Інструмент призначений для роботи в соціальних мережах.Він допомагає побудувати лояльність аудиторії, збільшити продажі, надати підтримку клієнтів, отримати зворотний зв'язок. Перед створенням каналів у SMM визначте бізнес-завдання та оцініть ресурси. Не починайте канали соціальних мереж просто тому, що вони є у кожного. Запущений, але покинутий рахунок - сумне видовище, яке скоріше зіпсує репутацію компанії і маркетолога, ніж принесе бажані продажі.

Проведіть опитування серед потенційних клієнтів і дізнайтеся, якими соціальними мережами вони користуються і чим там користуються – купуйте, підписуйтесь на розсилки, відпочивайте і веселіться. Вивчайте соціальні мережі конкурентів. Залежно від цього вибирайте канал і підготуйте контент. Формат контенту буде залежати від того, що найбільше подобається вашій аудиторії. Проведіть A/B-тестування, виміряйте результати та визначте, на що найкраще реагує ваша аудиторія. Виходячи з завдань і завдань компанії,стане зрозуміло, чи достатньо найняти штатного фахівця для ведення соціальних мереж або краще звернутися в SMM-агентство. Головне не зобов'язувати співробітника без досвіду проводити SMM як додаткове навантаження.

Управління репутацією в Інтернеті. Це комплекс заходів для відстеження згадок бренду, моніторингу тону публікацій, роботи з негативом і формування позитивного іміджу. Ефективним інструментом моніторингу є сервіс Kribrum.

Слідкуйте за показниками в email-маркетингу: доставка, відсоток відкриттів і відсоток показань, кількість відписаних. Використовуйте спеціальні поштові сервіси.

1.2. Актуальність розроблення програмного забезпечення.

Контент-маркетинг. Цей інструмент допомагає просувати мережу через публікації. Вважається, що перевагами контент-маркетингу в інтернеті є дешеві ліди. Однак це справедливо тільки для компаній, які вже мають велику аудиторію. В цьому випадку дійсно можна перетворити відвідувачів в покупців за допомогою корисного і цікавого контенту. Але потрібно пам'ятати, що контент-маркетинг - це інструмент, який розрахований на довгострокову перспективу. Створення і поширення контенту за ціною не поступається SEO і контекстній рекламі, тому, якщо трафіку ще немає, контент-маркетинг стане дорогим і трудомістким. Існує кілька варіантів організації контент-маркетингу: наймання редакторів і письменників в штаті, звернення в агентство або створення текстів всередині компанії. Якщо у вас немає фінансових ресурсів для вмісту, ви можете знайти

співробітників у вашій організації. Вони повинні бути експертами і мати бажання писати регулярно. Це один з найбільш ефективних методів отримання експертом унікальних і відносно недорогих матеріалів.

1.3. Висновки до розділу.

З врахуванням попередньо описаної інформації можна зробити висновок про актуальність розвитку розділу «Контент-маркетинг» та потрібність гнучкої веб-платформи для поставлених цілей.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. October CMS

October CMS позиціонується як система управління вмістом нового покоління. В October реалізовані найкращі моменти двох систем Modx і Wordpress. Основа October CMS – це популярний на ринку PHP фреймворк Laravel. Якщо ви раніше працювали з Laravel, то розібратися, як працює October, буде дуже просто.

Одна з плюшок October CMS – це Composer із коробки. За грамотного підходу можна красиво оформити в адмінці будь-який модуль. Обслуговувати сайт зможе навіть дитина. До інтерфейсу адмінки розробники October CMS підійшли дуже серйозно. Все дуже зручно.

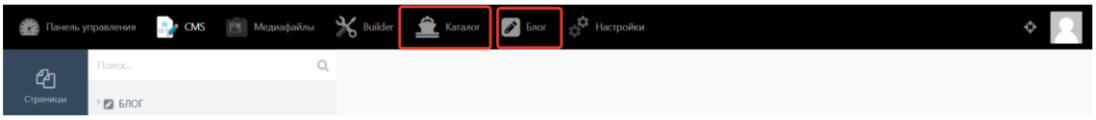

Рис 1. Інтерфейс адмін частини

З Modx October CMS взяв концепцію шаблонів. Ви побачите:

- сторінки (modx-ресурси),
- шаблони, фрагменти (modx-чанки),
- вміст(просто текст, в modx цього немає),
- ресурси (файли теми),
- фрагменти (modx-сніпети)

Насправді реалізація зовсім інша. Наприклад фрагменти vs сніпети може і схожі, але їх ніяк не можна повною мірою порівнювати, тому що принцип роботи трохи інший. Але і той, і цей служать для обробки або виведення будь-кої інформації.

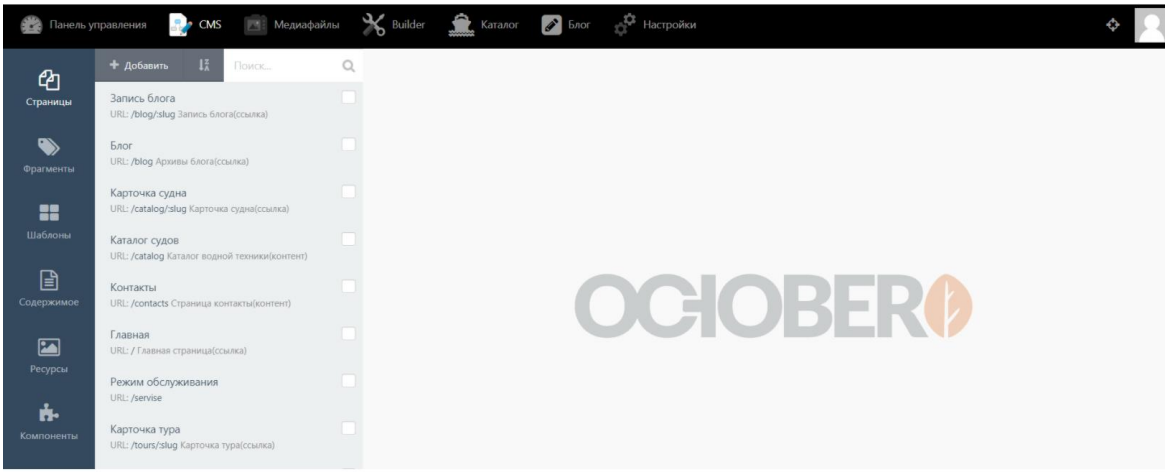

Рис 2. Риси Modx та Wordpress

Єдина схожа риса з Modx – це підключення чанків, тільки замість самописного шаблонизатора у October зашили Twig. Найпомітніша різниця між October та Modx криється у реалізації системи шаблонів. Modx елементи шаблону лежать у базі даних і кешуються, в October всі файли залишаються файлами.

Від Wordpress October CMS взяв лише концепцію встановлення шаблонів та плагінів. Наприклад Modx шаблон інтегрується вручну. October може встановити через репозиторій як Wordpress. Також варто відзначити дуже класний медіаменеджер October. Я не пригадаю нічого кращого, що мені вдалося бачити раніше. Все красиво як у Wordpress, тільки ви можете працювати з папками, тепер все дуже просто можна розіпхати по категоріях і не шукати серед сотень зображень потрібний медіафайл.

Незважаючи на схожість концепції, реалізація абсолютно інша. По факту ви працюєте з ООП за моделлю MVC, що вже стала стандартом. У Wordpress ви писали гору коду у function.php або підключали додаткові файли, іноді ставили Composer і підтягували необхідні пакети з наступним ручним підключенням.

October CMS взяв найкраще з перерахованих вище систем і переосмислив все. По можливості October більше фреймворк ніж CMS. Відповідно, і складність розробки інша. Якщо ви припустимо не знаєте PHP, а саме ООП, то поки не варто робити реальний проект.

Але спробувати варто однозначно. Для October написали чудовий модуль Builder, який дозволить швидко накидати каркас плагіна і в подальшому його допрацьовувати, наприклад, змінити поля введення в адмінці. Звичайно багато доведеться написати самостійно, але в будь-якому випадку це полегшить розробку. Як офіційно заявляє команда October CMS, у системі вбудований Ajax фреймворк.

Data attributes API – варіант для лінивих, принцип у тому, що для елементів ви вказуєте потрібний атрибут data. Наприклад ми натиснемо кнопку Delete, в наш контролер прилетить переданий id і запуститься метод onDelete. Наприклад, цей метод видалить запис з id = 7. JavaScript API - Просунутіший метод.

2.2. Висновки до розділу.

October CMS – це гнучкий інструмент для виконання будь-якого веб-завдання, який базується на фреймворку Laravel.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Дисперсійний аналіз

Більш складним є перевірка впливу декількох факторів на певну ознаку. Так, нехай необхідно дослідити наявність впливу факторів A і B на ознаку O. Для цього доцільно застосувати двофакторний дисперсійний аналіз. Результати вимірювань і проміжних обчислень двофакторного аналізу зручно здійснювати в таблиці, кожний рядок якої відповідає сталому рівню фактора A, а кожний стовпець – рівню фактора B. Вимірювання здійснюються l разів для кожного з рівнів факторів A і B при фіксованих рівнях факторів A і B, відповідно. Двофакторна модель має вигляд:

$$x_{ijk} = \bar{x} + u_{A_i} + v_{B_j} + w_{A_iB_j} + \varepsilon_{ijk},$$

де x_{ijk} – конкретне значення ознаки O, якого вона набуває в k-му вимірюванні ($k = \overline{1,n}$) при i-му рівні фактора A ($i = \overline{1,p}$) та j-му рівні фактора B ($j = \overline{1,q}$);

$\bar{x}$ – вибіркова середня, тобто загальна середня за всім обсягом вибіркової сукупності;

u_{A_i} – доданок, що обумовлений впливом i-го рівня фактора A;

v_{B_j} – доданок, що обумовлений впливом j-го рівня фактора B;

$w_{A_iB_j}$ – доданок, що обумовлений спільним впливом i-го рівня фактора A та j-го рівня фактора B; ε_{ijk} – помилка моделі, яка обумовлена варіацією

випадкової величини в межах блоку; вважається, що помилки розподілені за нормальним законом N $(0;\sigma^2)$. Метою вимірювань є дослідження впливу кожного з факторів A і B окремо, а також їх одночасний вплив на ознаку O. Розв'язання цієї задачі передбачає виконання певного обсягу обчислень, які проводяться у наступній послідовності.

1. Для кожної пари значень факторів A і B обчислюють блочні середні $\overline{x_{ij}} = \frac{1}{n}\sum_{k=1}^{n} x_{ijk}$ ($i = \overline{1,p}$, $j = \overline{1,q}$), а також визначають середні значення ознаки X за стовпцями $\overline{x_j} = \frac{1}{n \cdot p}\sum_{i=1}^{p}\sum_{k=1}^{n} x_{ijk}$ ($j = \overline{1,q}$) та за рядками $\overline{x_i} = \frac{1}{n \cdot q}\sum_{j=1}^{q}\sum_{k=1}^{n} x_{ijk}$ ($i = \overline{1,p}$). Ці значення порівнюють з вибірковою середньою ознаки X, що обчислюється за формулою:

$$\overline{x} = \frac{1}{npq}\sum_{i=1}^{p}\sum_{j=1}^{q}\sum_{k=1}^{n} x_{ijk}.$$

2. За всіма цими середніми обчислюють виправлені дисперсії. Відповідно отримують:

$S_1^2 = \frac{nq}{p-1}\sum_{i=1}^{p}\left(\overline{x_i} - \overline{x}\right)^2 = \frac{SS_1}{p-1}$, яка є виправленою дисперсією ознаки O, що зумовлена впливом на неї лише фактора A;

$S_2^2 = \frac{np}{q-1}\sum_{j=1}^{q}\left(\overline{x_j} - \overline{x}\right)^2 = \frac{SS_2}{q-1}$, яка є виправленою дисперсією, що зумовлена впливом на ознаку O лише фактора B;

$S_3^2 = \frac{n}{(p-1)(q-1)} \sum_{i=1}^{p} \sum_{j=1}^{q} \left(\overline{x_{ij}} - \overline{x_i} - \overline{x_j} + \overline{x}\right)^2 = \frac{SS_3}{(p-1)(q-1)}$, яка є виправленою дисперсією, що зумовлена одночасним впливом на ознаку $\tilde{O}$ і фактора A, і фактора B;

$S_4^2 = \frac{1}{N-pq} \sum_{i=1}^{p} \sum_{j=1}^{q} \sum_{k=1}^{n} \left(x_{ijk} - \overline{x_{ij}}\right)^2 = \frac{SS_4}{N-pq}$, яка є виправленою дисперсією, що обумовлена дією на ознаку $\tilde{O}$ факторів, вплив яких не розглядається в межах даної моделі.

Для цих дисперсій визначається емпіричні значення критерію Фішера: $F_A = \frac{S_1^2}{S_4^2}$; $F_B = \frac{S_2^2}{S_4^2}$; $F_{AB} = \frac{S_3^2}{S_4^2}$, які при рівні значущості α порівнюють з критичними точками статистики Фішера – Снедекора: $F_\alpha(k_1; k_4)$; $F_\alpha(k_2; k_4)$ та $F_\alpha(k_3; k_4)$, відповідно. Якщо $F_A > F_\alpha(k_1; k_4)$, то нульова гіпотеза про відсутність впливу на ознаку X фактора A відхиляється. Якщо $F_B > F_\alpha(k_2; k_4)$, то нульова гіпотеза про відсутність впливу фактора *B* відхиляється. Якщо $F_{AB} > F_\alpha(k_3; k_4)$, то нульова гіпотеза про відсутність спільного впливу факторів *A* і B відхиляється.

3.2. Висновки до розділу.

Любий статистичний аналіз та прогнозування відбувається при алгоритмах багатофакторного дисперсійного аналізу. Тому дисперсійний аналіз є математичнною складовою любої маркетингової розкрутки.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Проектування бази даних

Для того, щоб розробити логіку програми потрібно спочатку створити таблиці БД, в яких будуть зберігатися дані. Конкретно це:

- таблиця User – у ній будуть зберігатись дані маркетологів при реєстрації;
- таблиця Offers – інформація по продуктах для яких потрібна рекламна кампанія;
- таблиця Threads – налаштування потоків по конкретному офферу, а також в них буде зберігатись статистика відвідування для кожного з них;
- таблиця Lids – тут будуть збережені усі заявки які були надіслані користувачами що бажали придбати товар;

Створивши файли по даним таблицям [див. рис. Файли міграцій таблиць]

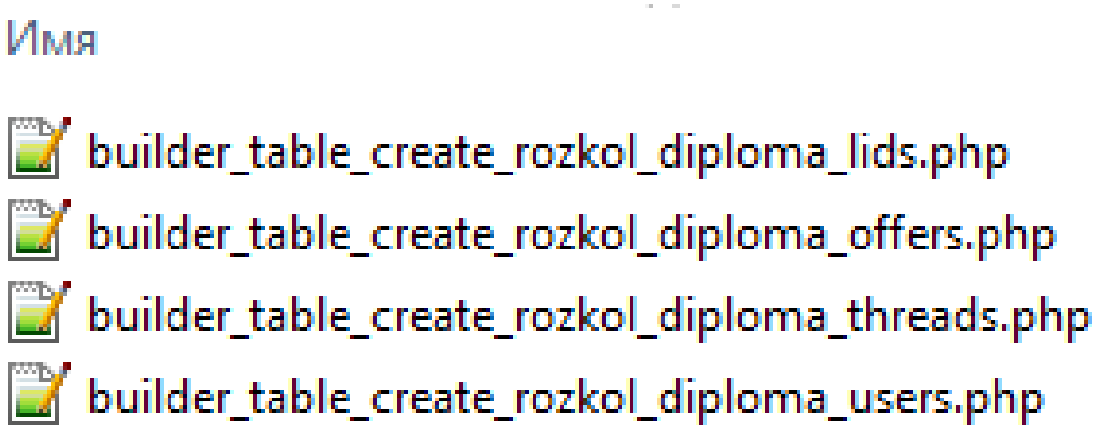

Рис 3. Файли міграцій таблиць

вписуємо для кожного з них стовпці, які мають бути у даних таблицях. Це:

- User (Манеджери)
    1. id
    2. name
    3. email
    4. password

- Offers (Продукція) [див. рис Offers]
    1. id
    2. short_name (коротка назва)
    3. long_name (повна назва)
    4. description (опис)
    5. compilation (додаткова інформація)
    6. price (ціна)
    7. warehouse (кількість на складі)
    8. lend (силки на сайт Лендінг)
    9. pre_lend (силки на сайт Пре лендінг)

- Threads (Потоки) [див. рис Threads]
    1. id
    2. name (назва потоку)
    3. slug (унікальний ключ потоку)
    4. lend (силка на сайт Лендінг)
    5. pre_lend (силка на сайт Пре лендінг)
    6. utms (спеціальні мітки для реклами)
    7. offer_id (ключ продукції для якої створений потік)
    8. user_id (ключ менеджера який створив цей потік)
    9. counter_lend (загальна кількість переходів на Лендінг)
    10. counter_lend_person(унікальна кількість переходів на Лендінг)
    11. counter_pre_lend (загальна на Пре лендінг)
    12. counter_pre_lend_person (унікальна на Пре лендінг)

- Lids (Замовлення клієнтів) [див. рис Lids]

1. id
2. name (ім’я)
3. phone (телефон)
4. status (статус)
5. sum (сума замовлення)
6. comments (коментар, якщо він є)
7. offer_id (ключ продукції яку замовив клієнт)
8. user_id (ключ менеджера від якого прийшов клієнт)
9. created_at (дата створення замовлення)
10. updated_at (дата оновлення замовлення)

```php
<?php namespace RainLab\User\Updates;

use Schema;
use October\Rain\Database\Updates\Migration;

class CreateUsersTable extends Migration
{

    public function up()
    {
        Schema::create('users', function($table)
        {
            $table->engine = 'InnoDB';
            $table->increments('id');
            $table->string('name')->nullable();
            $table->string('email')->unique();
            $table->string('password');
            $table->string('activation_code')->nullable()->index();
            $table->string('persist_code')->nullable();
            $table->string('reset_password_code')->nullable()->index();
            $table->text('permissions')->nullable();
            $table->boolean('is_activated')->default(0);
            $table->timestamp('activated_at')->nullable();
            $table->timestamp('last_login')->nullable();
            $table->timestamps();
        });
    }
```

Рис 4. Users

```php
<?php namespace Rozkol\SellSystem\Updates;

use Schema;
use October\Rain\Database\Updates\Migration;

class BuilderTableCreateRozkolSellsystemOffers extends Migration
{
    public function up()
    {
        Schema::create('rozkol_sellsystem_offers', function($table)
        {
            $table->engine = 'InnoDB';
            $table->increments('id')->unsigned();
            $table->string('short_name');
            $table->string('long_name');
            $table->text('description');
            $table->text('compilation');
            $table->text('price');
            $table->integer('warehouse');
            $table->text('lend');
            $table->text('pre_lend');
            $table->timestamp('created_at')->nullable();
            $table->timestamp('updated_at')->nullable();
        });
    }

    public function down()
    {
        Schema::dropIfExists('rozkol_sellsystem_offers');
    }
```

Рис 5. Offers

```php
<?php namespace Rozkol\SellSystem\Updates;

use Schema;
use October\Rain\Database\Updates\Migration;

class BuilderTableCreateRozkolSellsystemThreads extends Migration
{
    public function up()
    {
        Schema::create('rozkol_sellsystem_thread', function($table)
        {
            $table->engine = 'InnoDB';
            $table->increments('id')->unsigned();
            $table->string('name');
            $table->string('slug');
            $table->string('lend');
            $table->string('pre_lend');
            $table->text('utms');
            $table->integer('offer_id');
            $table->integer('user_id');
            $table->integer('counter_lend');
            $table->integer('counter_lend_person');
            $table->integer('counter_pre_lend');
            $table->integer('counter_pre_lend_person');
            $table->timestamp('created_at')->nullable();
            $table->timestamp('updated_at')->nullable();
        });
    }
```

Рис 6. Threads

```php
<?php namespace Rozkol\SellSystem\Updates;

use Schema;
use October\Rain\Database\Updates\Migration;

class BuilderTableCreateRozkolSellsystemLids extends Migration
{
    public function up()
    {
        Schema::create('rozkol_sellsystem_lids', function($table)
        {
            $table->engine = 'InnoDB';
            $table->increments('id')->unsigned();
            $table->string('name');
            $table->string('phone');
            $table->string('status');
            $table->integer('sum');
            $table->text('comments')->nullable();
            $table->integer('offer_id');
            $table->integer('user_id');
            $table->timestamp('created_at')->nullable();
            $table->timestamp('updated_at')->nullable();
        });
    }

```

Рис 7. Lids

У командному рядку пишемо команду:

```
php artisan migrate
```

Дана команда знаходить усі файли міграцій, які є у проекті, і керуючись налаштуваннями, які в них знаходяться, генерує відповідні таблиці та стовпці у базі даних MySql.

Після створення таблиць в бд створимо моделі для кожної з таблиць в яких опишемо відносини між таблицями, які будуть знаходитись в папці «models».

```php
<?php namespace Rozkol\SellSystem\Models;

use Model;
/**
 * Model
 */
class Offer extends Model
{
    use \October\Rain\Database\Traits\Validation;

    /**
     * @var string The database table used by the model.
     */
    public $table = 'rozkol_sellsystem_offers';

    /**
     * @var array Validation rules
     */
    public $rules = [
    ];

    protected $jsonable = ['price', 'promo', 'pre_lend'];

    /* Relations image */

    public $attachOne = [
        'image' => 'System\Models\File'
    ];

    public $hasMany = [
        'lids' => 'Rozkol\SellSystem\Models\Lid'
    ];

    public $belongsToMany = [
        'categories' => [
            'Rozkol\SellSystem\Models\Category',
            'table' => 'rozkol_sellsystem_offers_categories'
        ]
    ];
}
```

Рис 8. Модель «Offer»

```php
<?php namespace Rozkol\SellSystem\Models;

use Model;
/**
 * Model
 */
class Thread extends Model
{
    use \October\Rain\Database\Traits\Validation;

    /**
     * @var string The database table used by the model.
     */
    public $table = 'rozkol_sellsystem_threads';

    /**
     * @var array Validation rules
     */
    public $rules = [
    ];

    protected $fillable = [
        'name',
        'url',
        'site',
        'utms',
        'offer_id',
        'user_id',
        'counter',
        'counter_person',
        'counter_lids'
    ];

    protected $jsonable = ['utms'];

    /* Relations */
    public $belongsTo = [
        'offer' => 'Rozkol\SellSystem\Models\Offer',
        'user' => 'RainLab\User\Models\User'
    ];
}
```

Рис 9. Модель «Thread»

```php
<?php namespace Rozkol\SellSystem\Models;

use Model;
/**
 * Model
 */
class Lid extends Model
{
    use \October\Rain\Database\Traits\Validation;

    /**
     * @var string The database table used by the model.
     */
    public $table = 'rozkol_sellsystem_lids';

    /**
     * @var array Validation rules
     */
    public $rules = [
    ];

    protected $fillable = [
        'name',
        'phone',
        'status',
        'sum',
        'comments',
        'offer_id',
        'user_id',
        'operator_user_id',
        'slug'
    ];

    /* Relations */
    public $belongsTo = [
        'offer' => 'Rozkol\SellSystem\Models\Offer',
        'user' => 'RainLab\User\Models\User',
        'operator_user' => ['Backend\Models\User', 'key' => 'operator_user_id']
    ];
}
```

Рис 10. Модель «Lids»

4.2. Розробка web-додатку

Почнемо з верстки (графічний інтерфейс) і розпочнемо розробку програми у вигляді домашньої сторінки. Дизайн буде максимально простим та інтуїтивно зрозумілим.

Для цього створю навігаційне меню, яке відображатиме доступні нам сторінки. Для різних груп користувачів можна використовувати різні сторінки.

Наприклад, домашня сторінка доступна лише для користувачів, які не ввійшли в систему. Зареєстрований користувач матиме доступ до свого особистого кабінету, сторінки продукції та статистики але головна сторінка їм буде недоступна.

На домашній сторінці створю дві форми для «авторизації» та «реєстрації» [рис. Головна сторінка] цього буде цілком достатньо тому що основний функціонал за логікою дозволений тільки авторизованим користувачам

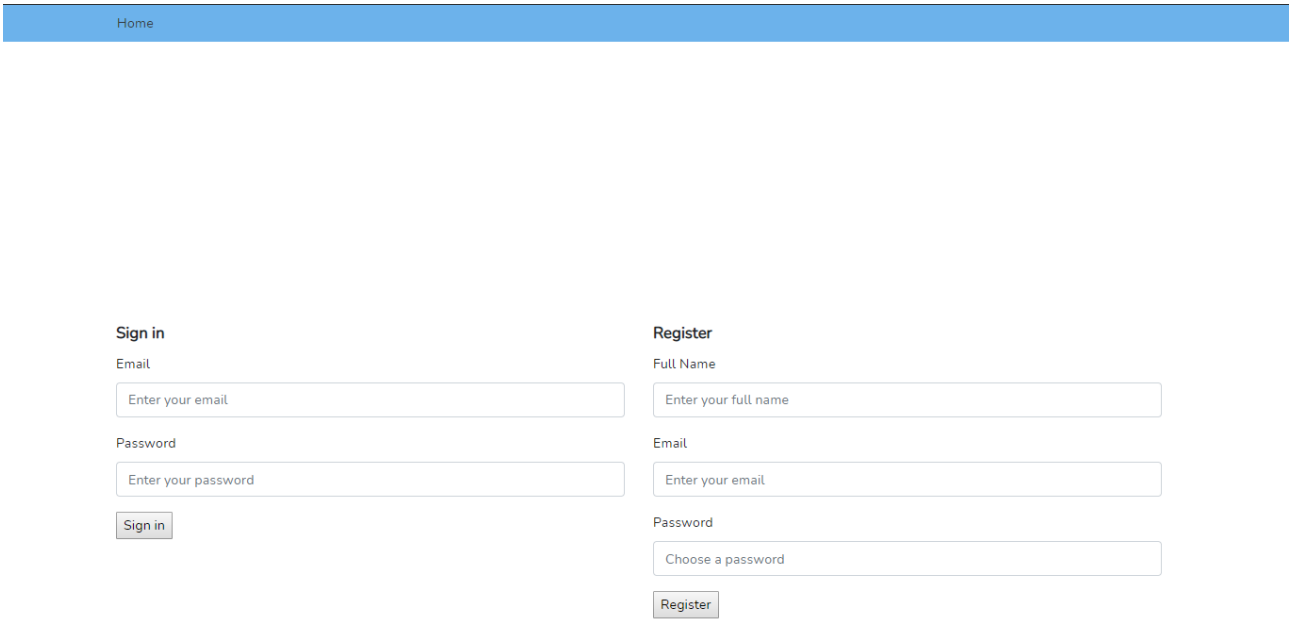

Рис 11. Головна сторінка

Для того щоб перевірити чи зайшов користувач у систему я буду використовувати глобальний об’єкт «Auth», в якому знаходиться інформація про користувача, якщо обєкт пустий значить користувач не авторизований тому відкриєм сторінку для авторизації, якщо ж інформація присутня то перенаправимо його на сторінку «Профілю» [рис. Провірка користувача].

```
function onInit()
{
    $user = Auth::user();

    if($user){
        return Redirect::to('/profile');
    }

}
```

Рис 12. Провірка користувача

Отже після реєстрації або авторизації користувача «Laravel» автоматично зберігає дані в сесію браузера, і тепер коли ми знаємо що користувач увійшов ми можемо перейти до головного функціоналу програми.

Так як «Laravel» використовує компонентний підхід для уникнення повторення коду на сторінках, створимо головний файл (Шаблон) проекту в який помістимо все те що буде повторюватись для кожної з сторінок, а це «Каскадні таблиці стилів», «JavaScript», меню навігації та підвал сайту якщо він потрібний [рис. Головний шаблон]

```
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <title>{{ this.page.title }}</title>
        <meta name="description" content="{{ this.page.meta_description }}">
        <meta name="title" content="{{ this.page.meta_title }}">
        <!-- Styles -->
        <link href="{{ 'assets/css/app.css'|theme }}?version=1.0.0" rel="stylesheet">
        {% styles %}
    </head>
    <body>
    <!-- Main wrapper -->
    <div class="wrapper" id="wrapper">
        <header class="bg-dark">
            <div class="container">
                <div class="row">
                    <div class="col p-0 d-flex justify-content-between">
                        <ul class="nav">
                          <li class="nav-item">
                            <a class="nav-link" href="/profile">Профіль</a>
                          </li>
                          <li class="nav-item">
                            <a class="nav-link" href="/offers">Offers</a>
                          </li>
                          <li class="nav-item">
                            <a class="nav-link" href="/statistic">Статистика</a>
                          </li>
                        </ul>

                        <ul class="nav">
                          <li class="nav-item text-light">
                            <p class="nav-link m-0">{{ user.name }}</p>
                          </li>
                        </ul>
                    </div>
                </div>
            </div>
        </header>

        <!-- Content -->
        <main>
            {% page %}
        </main>
        <!-- /Content -->

        <footer class="bg-dark d-flex align-items-center justify-content-between">
        </footer>
    </div>

    <!-- //Main wrapper -->
    <script src="{{ 'assets/js/app.js'|theme }}?version=1.0.0"></script>

    {% framework extras %}
    {% scripts %}
    </body>
```

Рис 13. Головний шаблон

Отже для всіх сторінок ми будемо відображати головне меню в якому буде:

1. Профіль – де користувач може редагувати свій профіль
2. Offers – сторінка наякі будуть відображатись усі доступні товари

3. Статистика – на ній будуть відображатись аналітика відвідування та замовлень

Також в головному меню я вивів назву користувача, щоб меню наздавалось таким пустим. Далі йде головний блок в якому ми будемо виводити вже інформацію для кожної сторінки свою, і вже в самому кінці завантажуємо скрипти якщо вони потрібні.

З головним шаблоном розібрались перейдемо до сторінок, почнемо з сторінки «Профілю», зробимо її максимально простою виключно для зміни даних якщо в цьому виникне необхідність [рис. Код сторінка профілю].

```
{{ form_ajax('onUpdate') }}

    <div class="form-group">
        <label for="accountName">Full Name</label>
        <input name="name" type="text" class="form-control" id="accountName" value="{{ user.name }}">
    </div>

    <div class="form-group">
        <label for="accountEmail">Email</label>
        <input name="email" type="email" class="form-control" id="accountEmail" value="{{ user.email }}">
    </div>

    <div class="form-group">
        <label for="accountPassword">New Password</label>
        <input name="password" type="password" class="form-control" id="accountPassword">
    </div>

    <div class="form-group">
        <label for="accountPasswordConfirm">Confirm New Password</label>
        <input name="password_confirmation" type="password" class="form-control" id="accountPasswordConfirm">
    </div>

    {% if updateRequiresPassword %}
        <p>To change these details, please confirm your current password.</p>
        <div class="form-group">
            <label for="accountPasswordCurrent">Current Password <small class="text-danger">* required</small></label>
            <input name="password_current" type="password" class="form-control" id="accountPasswordCurrent">
        </div>
    {% endif %}

    <button type="submit" class="btn btn-default">Save</button>

{{ form_close() }}
```

Рис 14. Код сторінка профілю

У фреємворка Laravel вже є стандартна форма оновлення даних тож використаємо її так як вона цілком нам підходить, тож сторінка профілю виглядає наступним чином [див. рис. Сторінка профілю]

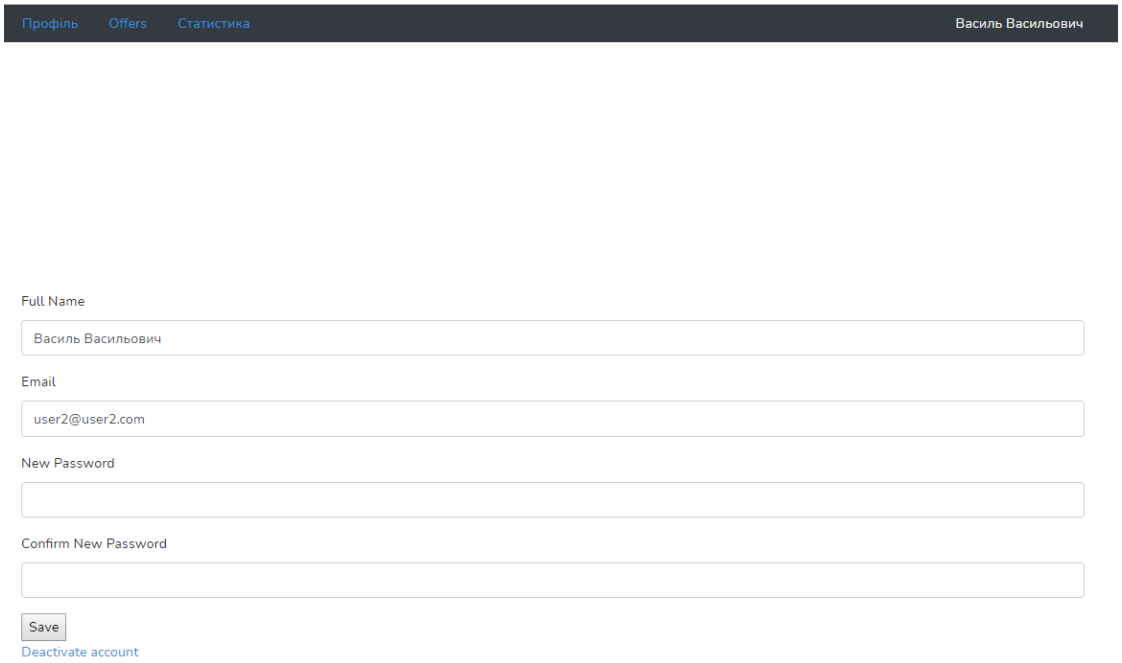

Рис 15. Сторінка профілю

Щоб продовжити розробку двох наступних сторінок, спочатку потрібно розробити інтерфейс адміністратора який зможе додавати товари в базу даних. Для цього використаємо спеціальну мову розмітки «YAML» який орієнтований на зручний «ввід-вивід» структурованих даних [див. рис. Yaml вводу виводу товарів]. Для того щоб розмітка інтерпретувалась фреємворком у стандартну html форму потрібно створити «контролер» для неї де вкажемо даний файл який створили [див. рис. Контролер товарів]

```
use Backend\Classes\Controller;
use BackendMenu;

class Offer extends Controller
{
    public $implement = [        'Backend\Behaviors\ListController',        'Backend\Behaviors\FormController'    ];

    public $listConfig = 'config_list.yaml';
    public $formConfig = 'config_form.yaml';

    public function __construct()
    {
        parent::__construct();

    }
}
```

Рис 16. Контролер товарів

Окрім файлу з формою було також створено список усіх товарі за аналогією з формою товарів

```yaml
fields:
    short_name:
        label: 'Short Name'
        span: auto
        type: text
    long_name:
        label: 'Long Name'
        span: auto
        type: text
    description:
        label: Description
        size: ''
        span: full
        type: textarea
    compilation:
        label: Compilation
        size: ''
        span: full
        type: richeditor
    price:
        label: Price
        prompt: 'Add new item'
        span: full
        type: repeater
        form:
            fields:
                country:
                    label: Country
                    span: full
                    type: text
                value:
                    label: Value
                    span: auto
                    type: number
                type:
                    label: Type
                    span: auto
                    type: text
                reward:
                    label: Rewards
                    span: full
                    type: text
    warehouse:
        label: Warehouse
        span: auto
        type: number
    image:
        label: Image
        mode: image
        imageWidth: '100'
        imageHeight: '100'
        useCaption: true
        thumbOptions:
            mode: crop
            extension: auto
        span: auto
        type: fileupload
```

Рис 17. Yaml вводу виводу товарів

В результаті вийшов стандартний інтерфейс для додавання, редагування та видалення товарів

Рис 18. Список товарів

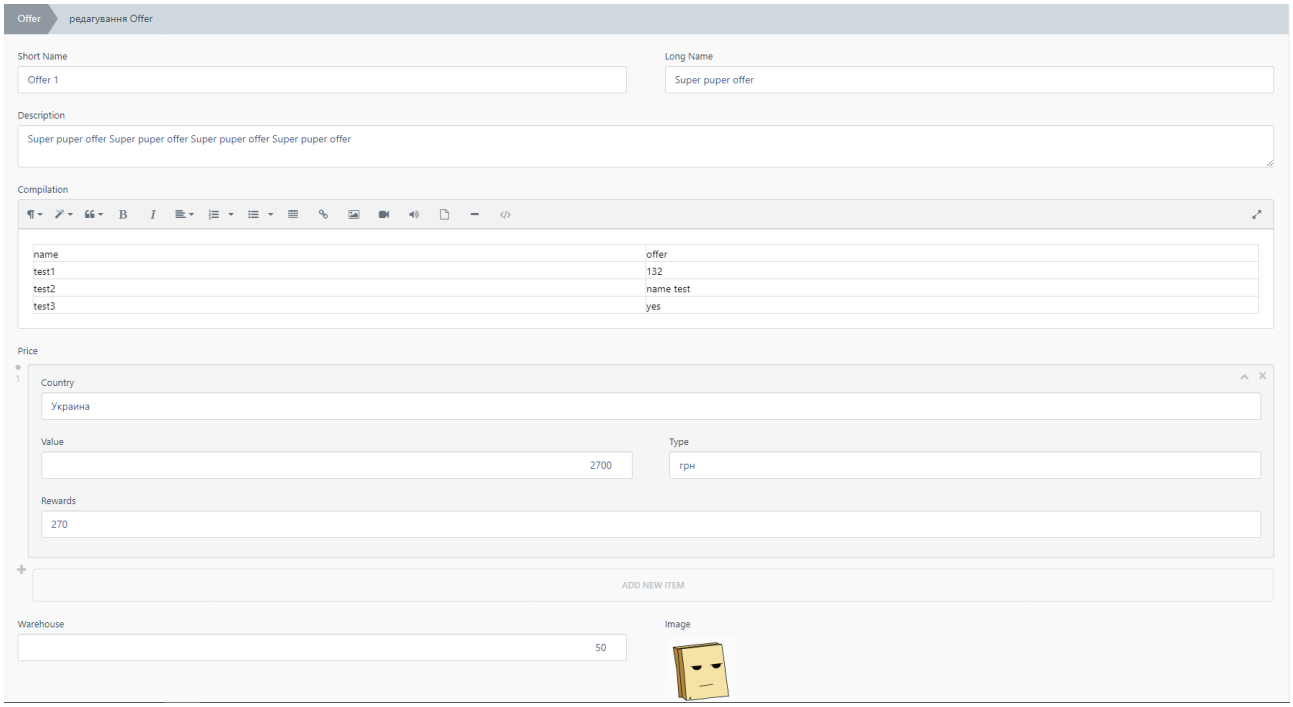

Рис 19. Форма товарів

Тож після створення та додавання декількох товарів в бд, перейдемо до їх відображення на сторінці Offers для наших користувачів (Маркетологів)

Створимо сторінку аналогічну назві і перейдемо до її логічної частини, вданій сторінці створимо функцію «onStart()» яка запуститься одразу після завантаження даної сторінки, в даній функції напишемо логіку для виводу всіх товарів з бд через модель яку ми створювали на початку [див. рис. Func вивід товарів]

```php
use Rozkol\SellSystem\Models\Offer;

function onStart(){
    $offers = Offer::all();

    $this['records'] = $offers;
}
```

Рис 20. Func вивід товарів

Після того як ми передали масив з даними на дану сторінку за ключем «records» перейдемо тепер до верстки виводу даних товарів

```twig
<section class="section-offers my-4">
    <div class="container">
        <div class="row row-cols-lg-2">
            {% for record in records %}
                <article class="col">
                    {# Use spaceless tag to remove spaces inside the A tag. #}
                    {% spaceless %}

                        <div class="row article-offer bg-dark text-light h-100 py-2 m-0">
                            <div class="col-lg-5">
                                <img class="w-100" src="{{ record.image.thumb(200,200) }}" alt="">
                            </div>
                            <div class="col-lg-7">
                                {% if detailsPage %}
                                    <a href="{{ detailsPage|page({ (detailsUrlParameter): attribute(record, detailsKeyColumn) }) }}" class="text-light">
                                {% endif %}

                                <h3>{{ attribute(record, displayColumn) }}</h3>

                                {% if detailsPage %}
                                    </a>
                                {% endif %}

                                <p>{{ record.long_name }}</p>
                                <span class="offers-price-block">
                                    {% for price in record.price %}
                                        <p class="w-100">{{ price.country }} - {{ price.value }} {{ price.type }} - {{ price.reward }} {{ price.type }}</p>
                                    {% endfor %}
                                </span>
                                <p>На складі - {{ record.warehouse }}</p>
                                <a href="/offer/{{ record.id }}/thread/add">+підключити</a>

                                {% if user.threads|length > 0 %}
                                    <div class="offers-thread-block my-2">
                                        {% for thread in user.threads %}
                                            {% if thread.offer.id == record.id %}
                                                <span>
                                                    <a href="/offer/{{ record.id }}/thread/{{ thread.id }}">{{ thread.name }}</a>
                                                    {% component 'deleteThread' delete_id=thread.id %}
                                                </span>
                                            {% endif %}
                                        {% endfor %}
                                    </div>
                                {% endif %}
                            </div>
                        </div>

                    {% endspaceless %}
                </article>
            {% else %}
                <li class="no-data"> Товарів немає</li>
            {% endfor %}
        </div>
    </div>
</section>
```

Рис 21. Верстка сторінки товарів

Получивши дані та відобразивши її на сторінці в нас получився наступний вигляд

Рис 22. Сторінка товарів

В даному прикладі ми вивели картинку товару її коротку назву, довгу назву, загальну ціну товару та процент який отримає маркетолог за успішну продану одиницю товару, та виведено кількість яка залишилась на продажу, а також кнопку «+ Підключити» ця кнопка буде відкривати сторінку для створення унікального потоку за допомогою якого користувач зможе згенерувати власне посилання на товар з додатковими настройками.

За допомогою даного каналу ми в подальшому зможемо реалізувати статистику в якій буде відображатись кількість загальних та унікальних переходів поданому з генерованому посиланні, також можна буде переглянути кількість відправлених замовлень які були зроблені через даний сайт, ну і в кінцевому результаті кількість коштів які були перераховані за успішне підтвердження даних замовлень.

Перед тим як створити даний функціонал по створенню потоків, перше зробимо сторінку детальної інформації товару. Логіка аналогічна виводу усіх товарів з поправкою на те що ми будемо виводити конкретний товар за ключем товару а точніше через його «id» посилання на дану сторінку буде мати наступний вигляд :

site_name/offer/{id},

де {id} – буде ключем товару

Тож створимо логіку для виводу даних з бд [див. рис Func вивід товару]

```php
use Rozkol\SellSystem\Models\Offer;

function onStart(){
    $offer = Offer::find($this->param('id'));

    $this['record'] = $offer;
}
```

Рис 23. Func вивід товару

Далі по аналогії розробимо верстку даної сторінки де виведемо всі дані по даному товару, а також поставимо вкінці кнопку «+ Підключити» щоб не повертатись на попередню сторінку, якщо даний товар зацікавить користувача і він захоче створити по ньому канал.

```twig
<section class="my-3">
    <div class="container">
        <div class="row">
            <div class="col">
                {% if record %}
                 <div class="row">
                    <div class="col-lg-4">
                        <img class="w-100" src="{{ record.image.path }}" alt="">
                    </div>
                    <div class="col-lg-8">
                        <h1>{{ record.short_name }}</h1>
                        <h3>{{ record.long_name }}</h3>
                        <p>{{ record.description }}</p>
                        <h3>Compilation</h3>
                        <div class="offer-compilation my-4">
                            {{ record.compilation|raw }}
                        </div>
                        <div class="offer-price-block">
                            {% for price in record.price %}
                                <p>{{ price.value }} {{ price.type }}</p>
                            {% endfor %}
                        </div>
                        <p>На складі - {{ record.warehouse }}</p>

                        <b><a href="/offer/{{ record.id }}/thread/add" class="text-dark">+підключити</a></b>
                    </div>
                </div>
                {% else %}
                    Товару немає
                {% endif %}
            </div>
        </div>
    </div>
</section>
```

Рис 24. Код сторінки товару

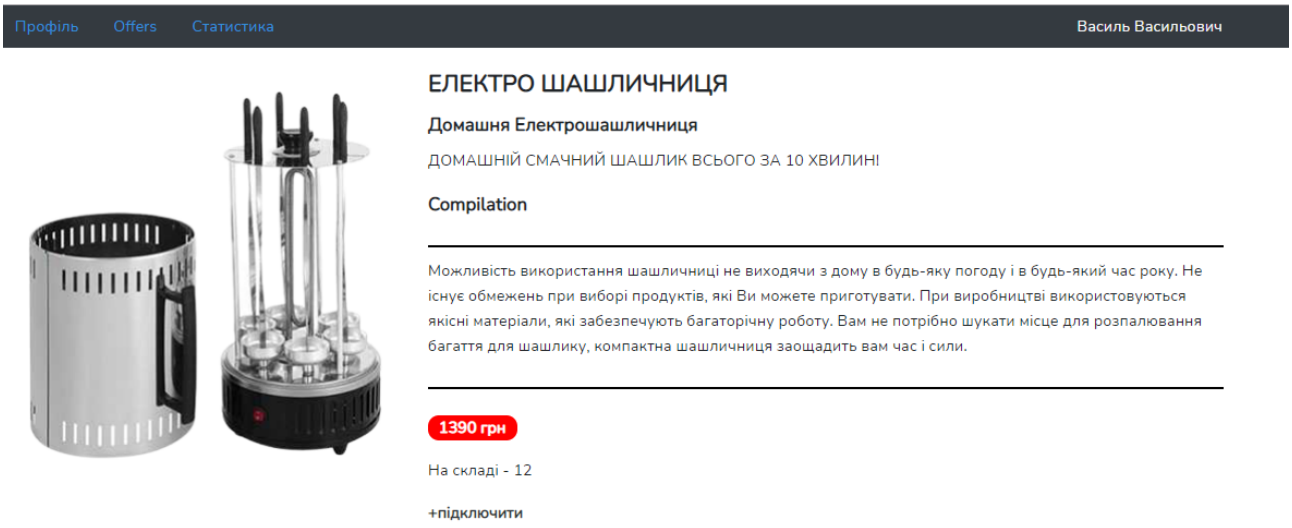

Рис 25. Вигляд сторінки товару

Тепер розробимо сторінку для створення унікального каналу (потоку), почнемо формування адреси на дану сторінку:

/offer/{offer_id}/thread/{thread_id}

де, offer_id – буде ключем товару,

thread_id – буде ключем потоку

По даним параметрам ми зможемо передавати в адресі ключ товару для якого потрібно створити потік, а також ключ самого потоку і якщо цей ключ буде «add» це буде створення нового каналу, інакше це буде код потоку для його редагування.

Тож почнемо з написання логіки для даної сторінки

```php
public function onStart() {

    if($this->property('thread_id') == 'add'){

        $result = ['offer' => Offer::find($this->property('offer_id')), 'thread' => null];

    }else{

        $model = Thread::find($this->property('thread_id'));
        $result = ['offer' => $model->offer, 'thread' => $model];

    }

    return $result;

}
```

Рис 26. Перевірка для створення потоку

Тепер зверстаємо сторінку з формою створення нового потоку або редагування вже існуючого

```twig
{% if this.param.thread_id == 'add' %}
    <h1>Створення потоку</h1>
{% else %}
    <h1>Редагування потоку [ {{ result.thread.slug }} ] {{ result.thread.name }}</h1>
{% endif %}

<form data-request="formThread::onSend" data-request-validate data-request-flash>
    <div class="form-group">
        <label for="url_lend"><b>Лендінг:</b></label>
        <input id="url_lend" class="w-100 p-2" disabled value="{{ result.thread.site }}?link={{ result.thread.slug }}">
    </div>
    <div class="form-group">
        <label for="url_pre_lend"><b>Пре лендінг:</b></label>
        <input id="url_pre_lend" class="w-100 p-2" disabled value="{{ result.thread.pre_lend }}?link={{ result.thread.slug }}">
    </div>
    <div class="form-group">
        <label for="name">Назва</label>
        <input type="text" class="form-control" name="name" id="name" value="{{ result.thread.name }}" placeholder="Назва Потока">
        <span data-validate-for="name"></span>
    </div>
    <div class="form-group">
        <label for="site">Лендінг</label>
        <input class="form-control" name="site" id="site_thread" list="site" value="{{ result.thread.site }}" placeholder="Силка на лендінг">
        <datalist id="site">
            {% for promo in result.offer.promo %}
                <option value="{{ promo.url }}">{{ promo.name }}</option>
            {% endfor %}
        </datalist>
        <span data-validate-for="site"></span>
    </div>
    <div class="form-group">
        <label for="site">Пре лендінг</label>
        <input class="form-control" name="pre_lend" id="pre_lend_thread" list="pre_lend" value="{{ result.thread.pre_lend }}" placeholder="Силка на пре лендінг">
        <datalist id="pre_lend">
            {% for pre_lend in result.offer.pre_lend %}
                <option value="{{ pre_lend.url }}">{{ pre_lend.name }}</option>
            {% endfor %}
        </datalist>
        <span data-validate-for="site"></span>
    </div>
    <div class="panel panel-default">
        <div class="panel-heading">
            <h3 class="panel-title">Utm мітки</h3>
        </div>
        <div class="panel-body">...</div>
    </div>
    <button id="btnSend" data-attach-loading>
        Зберегти
    </button>
</form>
```

Рис 27. Верстка сторінки потоку

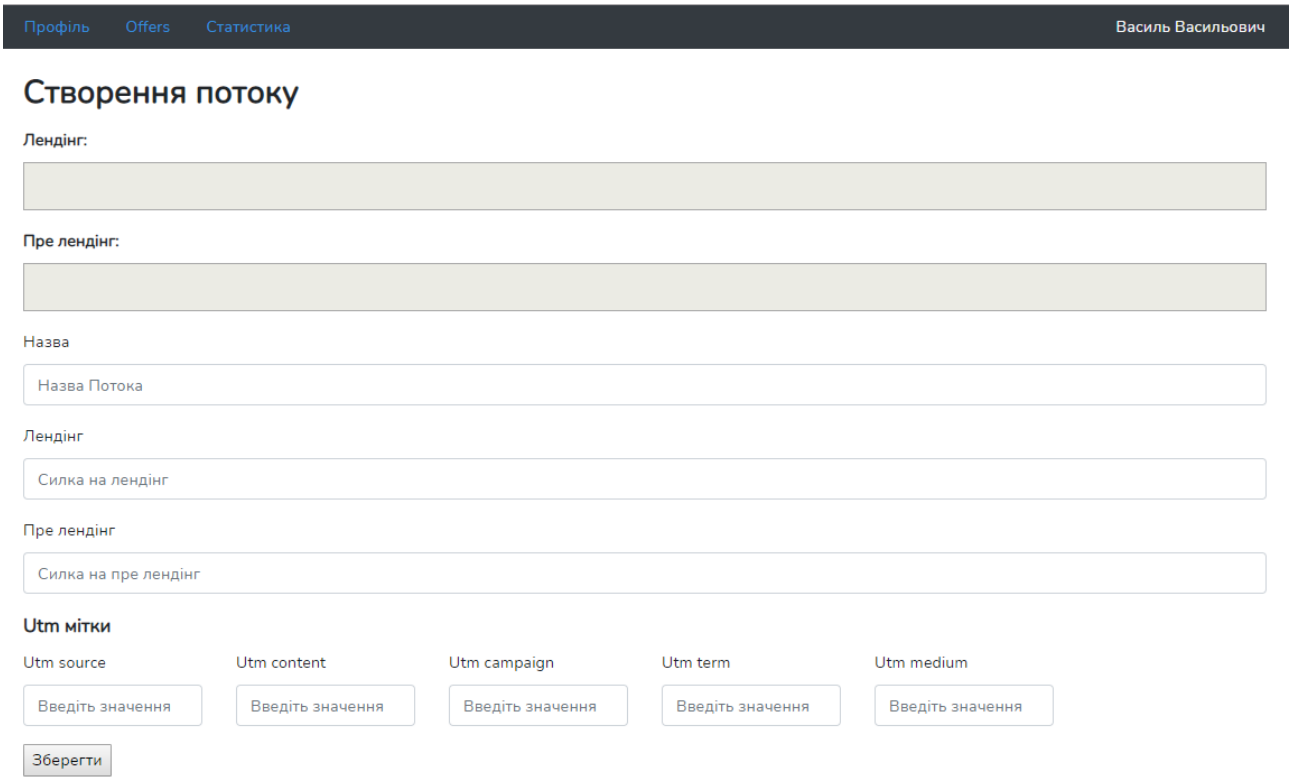

Рис 29. Сторінка створення потоку

Створимо метод «onSend()» для обробки та збереження відправлених даних на сервері. В даному методі створимо перевірку де в залежності від ключа « thread_id » буду створювати новий потік або редагувати старий. Також розробимо валідацію отриманих даних щоб не зберігати порожні дані в БД.

```php
public function onSend() {

    $data = post();

    $rules = [
        'name' => 'required',
        'site' => 'required',
    ];

    $validation = Validator::make($data, $rules);

    if ($validation->fails()) {

        throw new ValidationException($validation);

    }else{

        if ($this->property('thread_id') == 'add') {

            $thread = new Thread();

            $this->saveThread($data, $thread, base_convert(sha1(uniqid(mt_rand(), true)), 10, 36));

            return Redirect::to('/offer/'.$this->property('offer_id').'/thread/'.$thread->id);
        }
        else{

            $thread = Thread::find($this->property('thread_id'));

            $this->saveThread($data, $thread, $thread->slug);

            Flash::success('Save');

        }

    }
}
```

Рис 30. Обробка даних для створення або редагування потоку

Як можна спостерігати з даного рисунка даний метод приймає дані через глобальний метод « post() », дані створена перемінна « $rules » в якому вказаний масив з обов’язковими полями для замовлення, після чого викликаємо глобальний класс «Validator» в який передаємо дані які отримали, а також масив з параметрами валідації.

Після чого пишемо перевірку якщо валідація не проходить ми виводимо помилку поданим полям які її не пройшли.

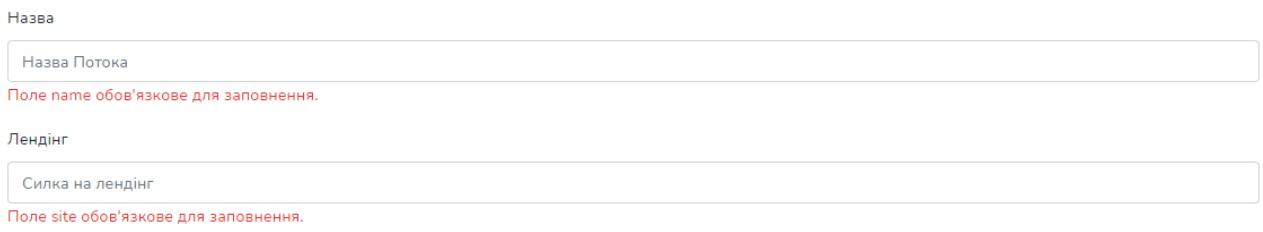

Рис 31. Вивід повідомлення про помилку

Якщо ж валідація пройдена переждемо до перевірки ключа «thread_id» якщо він дорівнює «add», то ми розуміємо що потрібно створити новий потік, інакше редагуємо старий потік по унікальному ідентифікатору.

Для створення нового потоку ми ініціалізуємо класс «Thread()» та викличемо метод saveThread в який передаємо [див. рис. Метод saveThread()]

saveThread($data, $thread, base_convert(sha1(uniqid(mt_rand(), true)), 10, 36));

де, data – це отримані дані,

thread – це наш ініціалізований класс,

base_convert – метод для генерації унікального ключа для потоку, так як в даному випадку буде доцільніше використати для ключа символи а не числа, так як дана назва буде фігурувати у силках.

```
public function saveThread($data, $thread, $slug) {
    $thread->slug = $slug;
    $thread->name = $data['name'];
    $thread->site = $data['site'];
    $thread->pre_lend = $data['pre_lend'];
    $thread->utms = [$data['utms']];
    $thread->offer_id = $this->property('offer_id');
    $thread->user_id = Auth::getUser()->id;
    $thread->save();
}
```

Рис 32. Метод saveThread()

Якщо ж ми редагуємо потік то замість «base_convert» ми передамо вже створений ключ, який отримаємо з самого запису.

Як видно з рисунка [Метод saveThread()] даний метод приймає дані та присвоює їх агалогічним полям які є у БД, після чого викликається метод «save()» який вже зберігає нові або редагує старі дані.

Отже після створення нового потоку нас переадресовує ту саму сторінку з поправкою на то що тепер ключ «thread_id» не «add», а індентифікатор створеного потоку.

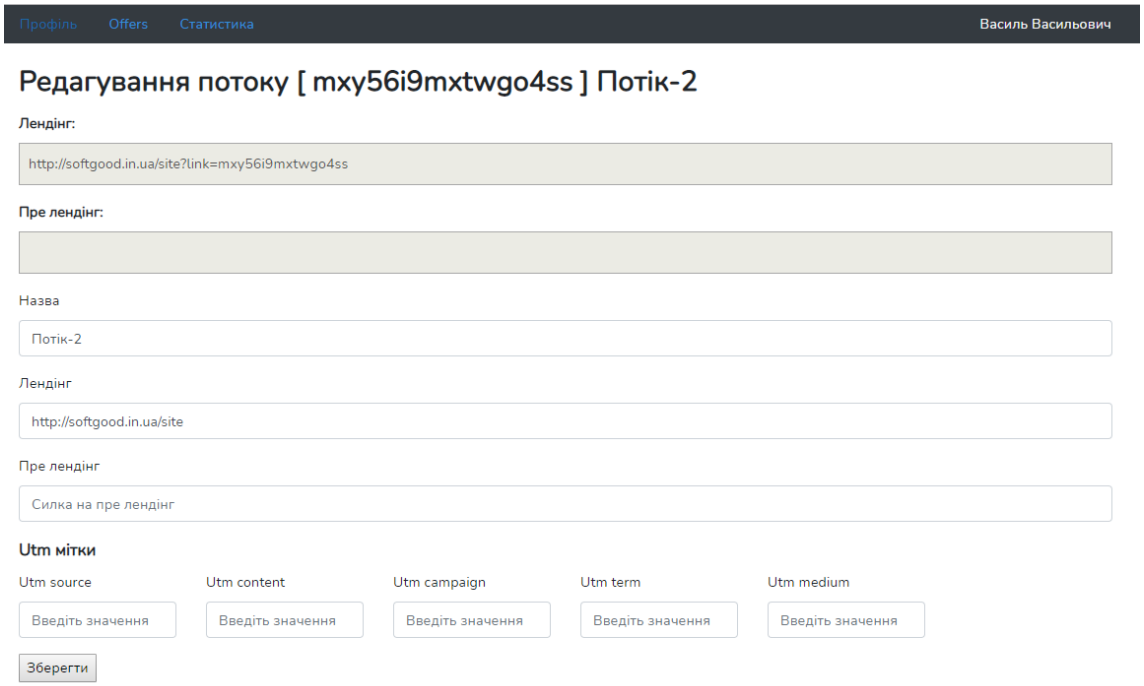

Рис 33. Сторінка редагування потоку

Єдине що тепер залишилось зробити для даної сторінки, це написати невеликий JavaScript код який буде генерувати силки для полів «Лендінг» і «Пре лендінг» який буде складатись з доменного імені, унікального ключа потоку який ми генерували при створені та utm – міток, дані мітки призначені для маркетологічних компаній в які передаються різноманітні дані для зручності самого маркетолога.

```
<script>
    let btn = document.querySelector('#btnSend'),
        url_lend = document.querySelector('#url_lend'),
        url_pre_lend = document.querySelector('#url_pre_lend'),
        inputSite = document.querySelector('#site_thread'),
        inputPre_lend = document.querySelector('#pre_lend_thread'),
        inputUtm = document.querySelectorAll('.form-control.utm_'),
        urlLend,
        urlPre_lend,
        i;

        btn.addEventListener("click", function() {
            urlLend = inputSite.value+'?link={{ result.thread.slug }}';
            urlPre_lend = inputPre_lend.value+'?link={{ result.thread.slug }}';
            for (i = 0; i < inputUtm.length; i++) {
                if (inputUtm[i].value) {
                    urlLend += '&'+inputUtm[i].id+'='+inputUtm[i].value;
                    urlPre_lend += '&'+inputUtm[i].id+'='+inputUtm[i].value;
                }
            }
            url_lend.value = urlLend;
            url_pre_lend.value = urlPre_lend;
            urlIF();
        });

        urlIF();

        function urlIF(){
            if(!inputPre_lend.value){
                url_pre_lend.value = '';
            }
            if(!inputSite.value){
                url_lend.value = '';
            }
        }

</script>
```

Рис 34. JS код для генерації силок по каналам

Дані маніпуляції потрібні так як дані доменна, ключа потоку і мітки, знаходяться в різних комірках запису, і для того щоб ці дані були єдиним посиланням і потрібна дана логіка, яка викликається одразу після загрузки сторінки [див. рис. Генерація посилання на потік]

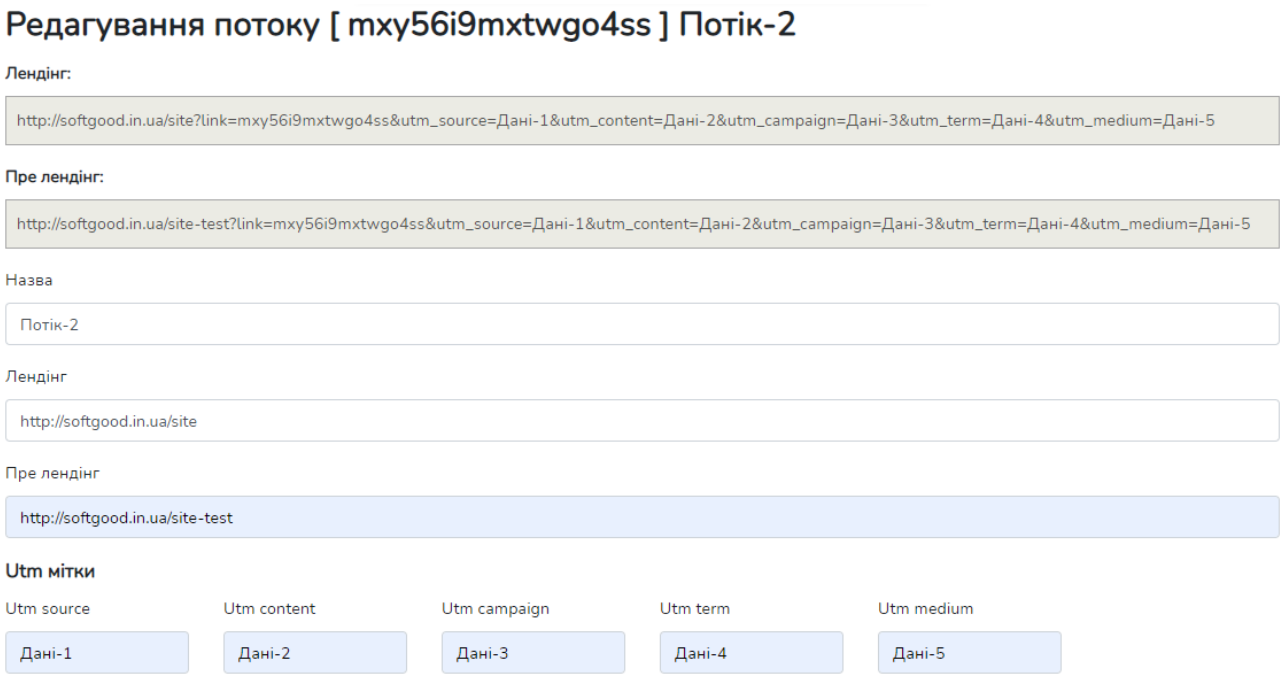

Рис 35. Генерація посилання на потік

Тепер перейшовши на сторінку з продуктами, під кожним з продуктів нам виводиться потоки які були створенні для даного оффера [див рис. Сторінка товарів з відображенням потоків]

Рис 36. Сторінка товарів з відображенням потоків

Наступним кроком буде розробка Лендінгової сторінки для замовлення товару, наприклад «ЕЛЕКТРО ШАШЛИЧНИЦЯ» в якому зробимо два блоки, це блок презентації [див. рис. Блок презентації товару], та блок з формою замовлення [див. рис. Форма замовлення товару]

Рис 37. Блок презентації товару

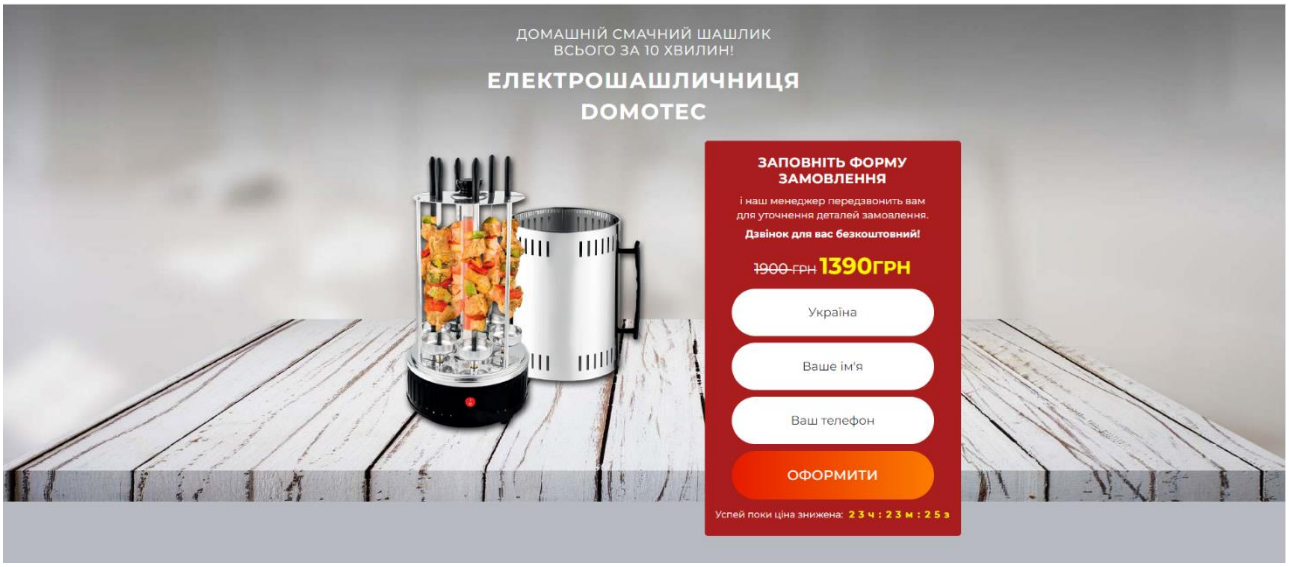

Рис 38. Форма замовлення товару

В даному «Лендінгу» напишемо логіку яка буде провіряти GET-параметри посилання та знаходити в ній ID-код потоку який був з генерований маркетологом на нашій платформі, після чого будемо передавати статистику відвідуваності по даному потоку на наш сервер див. рис Логіка Лендінга

```php
public function onStart() {

    if($_GET){

        $data = $this->setCurl('thread/'.$_GET['link']);
        if ($data){

            setcookie("Test", 'test');

            if (!isset($_COOKIE["Test"]))
            {
                $this->setCurl('thread/counter_lend/'.$_GET['link'].'/true');
            }else{
                $this->setCurl('thread/counter_lend/'.$_GET['link'].'/false');
            }

            $this['data'] = $data;
        }

    }

}

public function setCurl($url) {
    $curl = curl_init('http://softgood.in.ua/api/'.$url);

    curl_setopt($curl, CURLOPT_CUSTOMREQUEST, "GET");
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);

    $result = curl_exec($curl);
    $http_status = curl_getinfo($curl, CURLINFO_RESPONSE_CODE);

    curl_close($curl);

    return json_decode($result);
}
```

Рис 39. Логіка Лендінга

Як видно на рисунку логіка розділена на два метода, в першому ми провіряємо чи існують GET-параметри в нашому посиланні якщо так то в змінну “data” ми присвоюємо потік який передається в посиланні по ключу. Присвоєння відбувається через другий метод в якому ми відправляємо API дані на наш сервер який всою чергу передає там всю інформацію про даний потік якщо він існує в БД, в іншому випадку він поверне помилку що даного потоку не існує.

Далі ми перевіряємо чи є інформація про даний потік, якщо все ок то зберігаємо мітку у «Куки», після чого будемо провіряти його, якщо її ще не має то будемо відправляти дані на сервер з відміткою “true” в іншому випадку “false”. Це потрібно для того щоб записувати кількість унікальних переходів на сайт, тобто перший раз, всі наступні відвідування не будуть записуватись. Робимо

ми це для того щоб у статистиці відобразити загальну кількість людей які перейшли за посиланням, а також будемо виводити усю кількість і унікальних і повторних переходів.

Чудово наш «Лендінг» готовий тепер при переході наданий сайт він відправляє дані на сервер, тепер нам потрібно створити даний API на сервері який буде приймати запроси провіряти чи існує такий потік по даному запиту, та записувати у нього статистику у разі істини див. рис. Контролер Потоків

```php
<?php namespace Rozkol\SellSystem\Api;

use Rozkol\SellSystem\Models\Thread;

use Backend\Classes\Controller;

class ThreadController extends Controller
{



    public function getThread($slug) {

        $data = Thread::where('slug', $slug)->first();

        return $data;
    }

    public function counterLend($slug, $person) {

        $data =  $this->getThread($slug);
        $data->counter_lend += 1;
        if($person == 'true'){
            $data->counter_lend_person += 1;
        }

        $data->save();

        return 'ok';

    }

    public function counterPreLend($slug, $person) {

        $data =  $this->getThread($slug);
        $data->counter_pre_lend += 1;
        if($person == 'true'){
            $data->counter_pre_lend_person += 1;
        }
        $data->save();

        return 'ok';

    }
```

Рис 40. Контролер Потоків

У контролері який зображений на рисунку вище, перший метод провіряє на чи існує даний потік в базі, другий записує кількість відвідувань, третій роби теж саме що й другий тільки для «Пре легдінгу».

Створимо ще один контролер тільки вже для зберігання «Лідів» в якому будемо записувати інформацію вразі замовлення продукту див. рис. Контролер Лідів

```php
<?php namespace Rozkol\SellSystem\Api;

use Redirect;

use Rozkol\SellSystem\Models\Lid;

use Backend\Classes\Controller;

class LidsController extends Controller
{

    public function add() {

        $data = post();

        $lid = new Lid();

        $lid->name = $data['name'];
        $lid->phone = $data['phone'];
        $lid->status = 'Нова';
        $lid->sum = $data['sum'];
        $lid->offer_id = $data['offer'];
        $lid->user_id = $data['user'];
        $lid->save();

        return Redirect::back();

    }
}
```

Рис 41. Контролер Лідів

В контролері записуємо:

- Ім’я
- Телефон
- Статус
- Суму

- ID товару
- ID маркетолога

Тепер потрібно створити API роутер який буде в провіряти запити і запускати відповідні методи для них див. рис. API routers

```php
<?php

Route::group(['prefix' => 'api'], function () {

    Route::post('lids/add', 'Rozkol\SellSystem\Api\LidsController@add');

    Route::get('thread/{slug}', 'Rozkol\SellSystem\Api\ThreadController@getThread');

    Route::get('thread/counter_lend/{slug}/{person}', 'Rozkol\SellSystem\Api\ThreadController@counterLend');

    Route::get('thread/counter_pre_lend/{slug}/{person}', 'Rozkol\SellSystem\Api\ThreadController@counterPreLend');

});
```

Рис 42. API routers

Перейдемо до фінальної сторінки для Маркетолога, це сторінка статистики по факту най головніша сторінка для чого і розроблювалась дана система В ній ми будемо відображати:

- дату створення потоку
- ключ потоку
- посилання на продукт по якому створювався даний потік
- трафік (в ньому буде відображатись кількість відвідувань)
- ліди (інформація про замовлення та їх статус)
- фінанси (інформація про нарахування процентів від замовлення)

Всі ці дані ми збираємо з нашої бази даних яку формували на початку та прив’язували до наших користувачів див. рис. Сторінка статистики

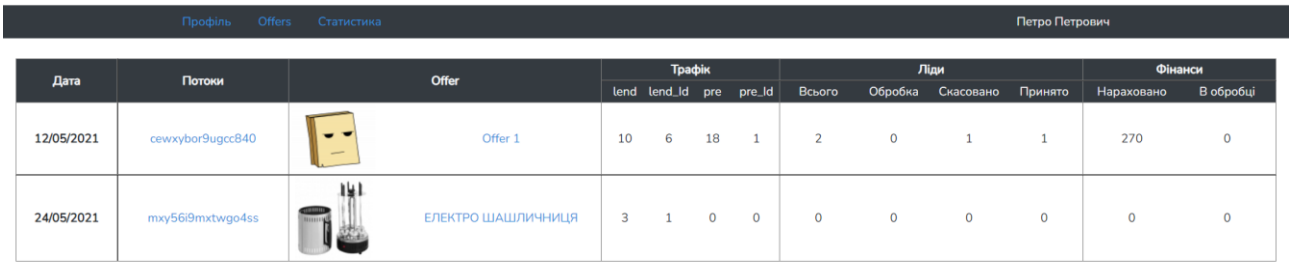

Рис 43. Сторінка статистики

Кошти за замовлення нараховуються коли після прийнятого замовлення, приймаються дані замовлення менеджерами які опрацьовують їх і вказують статуси. Тому для них ми створимо окрему систему в панелі адміністрації, як це було зроблено з товарами, виведемо їм список усіх замовлень [див. рис. Список замовлень] та форму обробки конкретного замовлення [див. рис. Форма обробки замовлення]

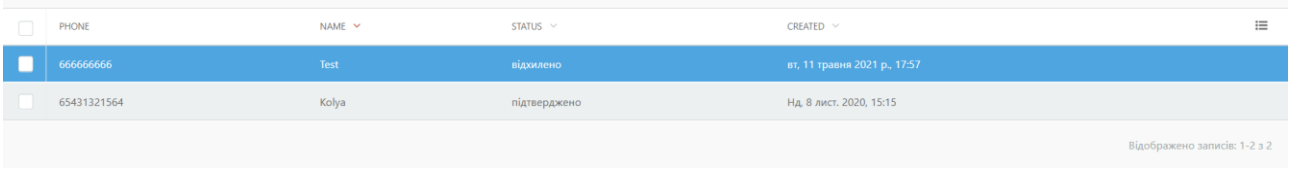

Рис 44. Список замовлень

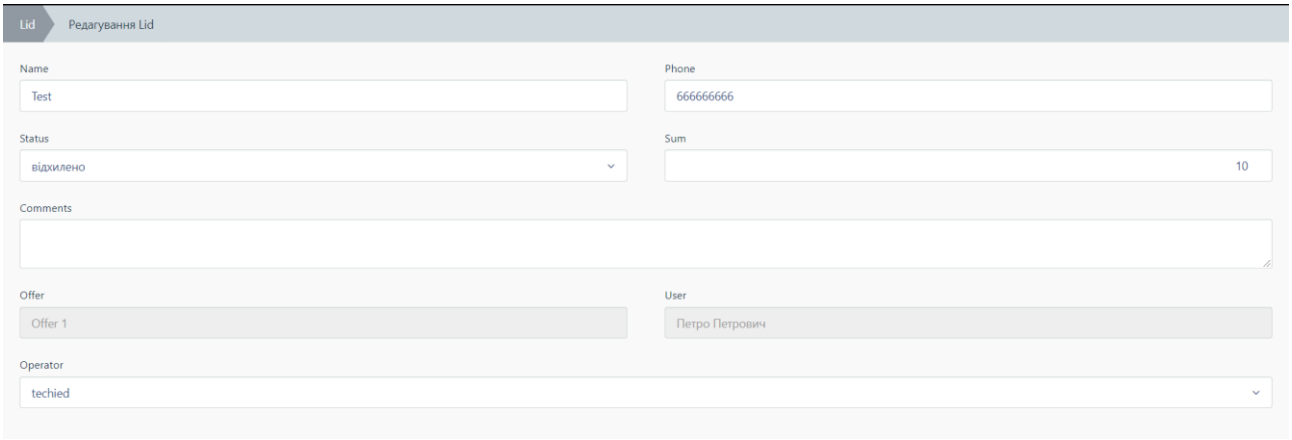

Рис 45. Форма обробки замовлення

4.3. Висновки до розділу.

October CMS – чудове вирішення для маркетингової розкрутки та продажу товарів. Крім того фреймворк laravel є досить швидким та оптимізованим. Тому веб-завдання великої складності є досить швидкими, в порівнянні із CMS Wordpress.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ

5.1. Опис ідеї проекту

На даний час більшість людей у світі, які займаються бізнесом працюють в мережі інтернет. Тому автоматизація контенту є досить актуальною темою і існує стабільний пошук спеціалістів у цій сфері, адже матеріали, які опубліковані на веб-сторінці набагато більше продаються. Це є економічно вигідніше, ніж випускати рекламну фізичну продукцію, яка є неабгрунтованою розкісшю.

Пошукова оптимізація, або SEO, - це процес, за допомогою якого веб-сайт оптимізований для кращої роботи та вищого рангу у відповідних пошуках. Контент-маркетинг, з іншого боку, є підмножиною вхідного маркетингу, яка включає в себе переміщення відвідувача веб-сайту через воронку продажів, пропонуючи їм корисні, цікаві або переконливі фрагменти вмісту.

Хоча ці два можуть здатися ніччю і днем, можна створити стратегію контент-маркетингу для SEO, тобто стратегію контент-маркетингу, яка допомагає вашому веб-сайту краще ранжувати, для більшої кількості ключових слів і охопити більше клієнтів.

Оскільки алгоритми Google розвивалися і змінювалися, вони почали підтримувати багато елементів хорошого плану контент-маркетингу , наприклад, свіжий контент певної довжини, з хорошим націлюванням на ключові слова і на конкретних сторінках. Насправді, тверда стратегія SEO тепер включає в себе створення контенту так само, як вона включає в себе класичні, більш технічні фактори.

5.2. Переваги SEO контент-маркетингу

Давайте подивимося трохи ближче на деякі переваги SEO контент-маркетингу, і дізнатися конкретні способи, якими план контент-маркетингу може дати вам перевагу над вашими конкурентами в рейтингу пошукових систем.

Більше вмісту = більше потенційних ключових слів

Пошукові системи не можуть "бачити" веб-сайти, як ви і я можу. Хоча ми, як люди, можемо дивитися на веб-сайт з фотографіями товарів для домашніх тварин і розуміти, що це сайт електронної комерції, що продає повідки, миски для собак та іграшки для собак, Google не має можливості знати, яка мета веб-сайту, якщо ці слова десь не написані.

Контент-маркетинг дозволяє вийти за рамки невеликих абзаців домашньої копії та текстових описів на фотографіях, де ваш простір для ключових слів може бути обмежений. За допомогою статей, публікацій у блозі, посібників та іншого письмового вмісту ви можете використовувати більше ключових слів, що мають відношення до того, що стосується вашого веб-сайту та того, що люди шукають.

Наприклад, якщо ваш веб-сайт продає товари для домашніх тварин, ви можете добре оцінити ключові слова, пов'язані з нашийниками для собак. Замість того, щоб намагатися втиснути всі слова і фрази, пов'язані з коміром, які ви можете придумати в одному невеликому просторі або сторінці, ви можете написати дуже природний посібник, такий як "Як вибрати правильний комір розміру для вашої собаки", або створити блог про найновіші кольори та стилі комірів у вашому магазині. Цей контент не просто корисний і цікавий. Це також чудова можливість використовувати ключові слова та добре ранжувати в пошукових запитах!

5.3. Вплив контент-маркетингу на пошукові системи та користувачів

Чи завжди контент краще? Не обов'язково. Зрештою, якщо ваша стаття, блог, або продукція не міняється, не роблячи точки відправлення або пропонуючи будь-яку істотну інформацію, відвідувачі можуть втомитися від нього і піти, і це може фактично зашкодити вашим рейтингам. Однак істотний, інформативний і детальний контент, як правило, довший, тому має сенс, що він буде корелювати з найвищими рейтингами.

Подумайте, коли ви шукали відповідь або підручник в Інтернеті. Скільки часу вам знадобилося, щоб знайти сторінку, достатньо детальну, щоб повністю відповісти на ваше запитання або допомогти вам вирішити вашу проблему? Швидше за все, ви відразу знайшли цю сторінку.

Як SEO розвивалася, він змінився, щоб включити інші фактори, ніж щільність ключових слів (тобто, кількість разів, коли ви використовували ключове слово, яке ви хочете ранжувати). Таким чином, створюючи значний контент, який відвідувачі дійсно цінують, у вас є більше шансів зайняти перше місце, ніж той, хто намагається отримати їх використання ключових слів досконалим!

Скажімо, хтось приходить на ваш сайт з пошуку, не може знайти те, що вони шукають, і йде відразу (не відвідуючи будь-які інші сторінки). Це називається відскоком, і відсоток людей, які роблять це на кожній сторінці на вашому сайті, враховується у вашій відмовостійкій кількості.

Google може враховувати такі показники, як показник відмов, у загальному профілі SEO вашого сайту. Якщо сторінка має дуже високий показник відмов, це, як правило, ознака того, що з нею щось не так - і Google, звичайно, не хоче продовжувати високо ранжувати її, якщо вона відправляє всіх своїх відвідувачів назад у результати пошуку. Іноді вміст може допомогти вам боротися з високим відскоком і вирішувати такі проблеми, особливо якщо це зроблено вдумливо.

Наприклад, скажімо, показник відмов на вашій домашній сторінці дуже високий, і ви думаєте, що це тому, що відвідувачі не знають, з чого почати дізнаватися про ваші продукти або послуги. Ви можете додати певний вміст, щоб представити вашу компанію, запропонувати, де вони можуть почати роботу, або навіть посилання на деякі сторінки. Це може тримати їх на сторінці довше, відправляти їх на інші сторінки, і зменшити плутанину - так що це насправді рішення трьох проблем, а не тільки одна!

Якщо ваш вміст починає дуже швидко набридати людям, це також може завдати шкоди вашій відмовостійкості. Так що довжина важлива, але не витрачайте час вашого відвідувача, або!

Нарешті, є одна річ, яку SEO і контент-маркетинг часто мають спільне: вони зосереджені на пошуку того, що не було зроблено раніше, і намагаються високо оцінити це.

Об'єднавши дослідження ключових слів - один з найважливіших навичок, які ви повинні навчитися вдосконалювати свій SEO- і вашу програму контент-маркетингу, ви можете зосередитися не тільки на створенні стратегії контент-маркетингу для отримання SEO, але і на створенні унікального, ніколи раніше не бачених контенту, який піддає ваш бізнес або веб-сайт відвідувачам, які шукають ключові слова "long-tail", де у вас є набагато більше шансів ранжувати на перше місце.

Наприклад, якщо ви продаєте вишукані шоколадки в Інтернеті, ваша негайна реакція може по-заснути на створення статті про те, чому шоколад так добре смакує, або навіть чому він робить хороший подарунок. Але це, безумовно, було зроблено раніше! Замість цього, про роблячи невелике дослідження, ви можете виявити, що кілька десятків людей шукають щомісяця докладну статтю про користь для здоров'я темного шоколаду, але там немає хороших частин на цю тему.

Це ідеальна тема для вашої програми контент-маркетингу, тому що ви можете авторитетно писати про це, і мати знання та ресурси, необхідні для цього. Крім того, оскільки ви продаєте шоколад, ви можете закінчити статтю ніжним закликом до дії за рядком "натисніть тут, щоб побачити наш широкий вибір темних цукерок, відправлених безкоштовно до вашого будинку або коханої людини".

До тих пір, поки вміст, який ви пишете, є унікальним і актуальним для вашої компанії, ці унікальні частини можуть допомогти вам зайняти додаткові місця в

пошукових видачах, оцінити ще більше ключових слів і охопити більш широку аудиторію.

Хоча ви можете абсолютно поліпшити SEO з вмістом на вашому сайті, переваги контент-маркетингу виходять далеко за рамки підвищених рейтингів і видимості пошукової системи. Насправді, ваша головна мета зі створенням програми контент-маркетингу не обов'язково повинна покращити seo, тому що вміст вашого сайту не обов'язково може мати істотний вплив на нього.

Контент-маркетинг має ряд інших переваг для ваших потенційних клієнтів і потенційних клієнтів. Це заповнює прогалини у вашій воронці продажів, піддає ваш бізнес додатковій аудиторії та створює почуття доброї волі серед тих, хто відвідує ваш сайт.

Ви можете дізнатися більше про переваги контент-маркетингу, прочитавши третій розділ нашого посібника для початківців з контент-маркетингу

5.4. Маркетингова програма стартап-проекту

Стратегію виходу на ринок можна розбити на 2 етапи. Першим – є розробка та випуск бета версії вес-системи. В другові етапі заховано зразу кілька кроків. Першим буде знаходження реального партнера для тестування продаж та збору статистики. Після цього переходимо на другий етап, в якому буде підключена контекстна реклама для збільшення кола клієнтів.

5.5. Висновки до розділу.

Підсумувавши все вище сказане, можна зробити висновок, що дана веб-система заслуговує на займання певної ринкової ніші та, враховуючи стратегії, буде працювати на всіх провідних платформах через перших успішної торгівлі.

ВИСНОВКИ

У відповідності з вимогами технічного завдання, розроблено інтелектуальну веб-платформу з використанням фреймворку Laravel та CMS October. Після запуску бета-версії та збору статистичних даних збільшено продажу товарів. Створено реферальну систему для користувачів платформи.

Зручний інтерфейс - дозволяє максимально в короткий час створити свій кабінет, де появляється можливість додати товари для продажі. Створення своєї сітки продавців та ефективна реферальна система дозволить не тільки збільшити продажі, а і створити структуру, яка буде приносити прибуток і в подальшому часі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rufus Stewart. Laravel The Ultimate Beginner's Guide to Learn Laravel Step by Step , 2nd Edition – Oct 27, 2020.
2. Oluwafemi Alofe. Beginning PHP Laravel Step to step approach to building an Inventory App. – Mar 19, 2020.
3. Patrick Sherry. October CMS Up and Running: The Complete Guide To Starting An October CMS Site Quickly – April 17, 2017
4. https://www.youtube.com/watch?v=3SmnExVW0Ag&list=PLUBR53Dw-Ef-X-_A1KLy41r2QArCBy4rM&ab_channel=WatchandLearn
5. https://habr.com/ru/post/354036/

ДОДАТКИ

```
public function onStart() {
    if($_GET){
        $data = $this->setCurl('thread/'.$_GET['link']);
        if ($data){
            setcookie("Test", 'test');
            if (!isset($_COOKIE["Test"]))
            {
                $this->setCurl('thread/counter_lend/'.$_GET['link'].'/true');
            }else{
                $this->setCurl('thread/counter_lend/'.$_GET['link'].'/false');
            }
            $this['data'] = $data;
        }     }}
public function setCurl($url) {
    $curl = curl_init('http://softgood.in.ua/api/'.$url);
    curl_setopt($curl, CURLOPT_CUSTOMREQUEST, "GET");
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
        $result = curl_exec($curl);
    $http_status = curl_getinfo($curl, CURLINFO_RESPONSE_CODE);
    curl_close($curl);
    return json_decode($result);}
class LidsController extends Controller{
public function add()
{ $data = post(); $lid = new Lid();  $lid->name = $data['name'];
 $lid->phone = $data['phone'];
$lid->status = 'PSPsPIP°CЏ';
$lid->sum = $data['sum'];
$lid->offer_id = $data['offer'];
$lid->user_id = $data['user'];
$lid->save();
return Redirect::back();
```

```
    //return $data;    } }
<?phpRoute::group(['prefix' => 'api'],
 function () {
 Route::post('lids/add','Rozkol\SellSystem\Api\LidsController@add');
 Route::get('thread/{slug}',
Rozkol\SellSystem\Api\ThreadController@getThread');
Route::get('thread/counter_lend/{slug}/{person}',
'Rozkol\SellSystem\Api\ThreadController@counterLend');
Route::get('thread/counter_pre_lend/{slug}/{person}',
'Rozkol\SellSystem\Api\ThreadController@counterPreLend');});
<?php namespace Rozkol\SellSystem\Api;
use Redirect;
use Rozkol\SellSystem\Models\Lid;
use Backend\Classes\Controller;
class LidsController extends Controller
{       public function add()
{          $data = post();              $lid = new Lid();
$lid->name = $data['name'];
$lid->phone = $data['phone'];
$lid->status = ';
$lid->sum = $data['sum'];
$lid->offer_id = $data['offer'];
$lid->user_id = $data['user'];
$lid->save();
 return Redirect::back();              //return $data;    }          }
class FormThread extends ComponentBase
{       public function componentDetails()
{              return [                  'name' => 'Form Thread',
 'description' => 'Thread of offer'                  ];          }
     public function defineProperties()
{                  return [                          'offer_id' => [                          'title' => 'ID Offer',
'description' => ',
'default' => "{{ :offer_id }}",
```

```
'validationPattern' => '^([a-z\:\_])+$',                    'validationMessage' => ' ,'          ],
'thread_id' => [                    'title' => 'ID Thread',
   'description' => ',
    'default' => "{{ :thread_id }}",
   'validationPattern' => '^([a-z\:\_])+$',
    'validationMessage' => 'PўPsP»CЊPєPs C,PµPєCЃC,'          ]       ];          }
 public $result;
 public function onRun() {          $this->result = $this->loadOffer();          }
 public function loadOffer() {
if($this->property('thread_id') == 'add'){
$result = ['offer' => Offer::find($this->property('offer_id')), 'thread' => null];       }
else{                $model = Thread::find($this->property('thread_id'));
    $result = ['offer' => $model->offer, 'thread' => $model];                }
       return $result;          }       public function onSend()
{          $data = post();          $rules = [          'name' => 'required',          'site' => 'required',
];
 $validation = Validator::make($data, $rules);
 if ($validation->fails()) {                throw new ValidationException($validation);                }
else{                if ($this->property('thread_id') == 'add')
{                $thread = new Thread();                $this->saveThread($data, $thread,
base_convert(sha1(uniqid(mt_rand(), true)), 10, 36));                return Redirect::to('/offer/'.$this-
>property('offer_id').'/thread/'.$thread->id);          }
       else{                $thread = Thread::find($this->property('thread_id'));                $this-
>saveThread($data, $thread, $thread->slug);                Flash::success('Save');                }
}          }
 public function saveThread($data, $thread, $slug)
{          $thread->slug = $slug;          $thread->name = $data['name'];          $thread->site =
$data['site'];          $thread->pre_lend = $data['pre_lend'];          $thread->utms = [$data['utms']];
$thread->offer_id = $this->property('offer_id');          $thread->user_id = Auth::getUser()->id;
$thread->save();       }       }
```