МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Веб-додаток реалізований мовою програмування Ruby та фреймворком Ruby on Rails»

Завідувач
випускаючої кафедри **Довбиш А. С.**

Керівник роботи **Шаповалов С. П.**

Студентка гр. ІНз-71с **Осмоловська Ю. С.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ

Кафедра комп'ютерних наук

Затверджую

Зав. кафедрою Довбиш А. С.

"_____" _____2021 р.

ЗАВДАННЯ

до випускної роботи

Студентки четвертого курсу, групи ІНз-71с спеціальності «Комп'ютерні науки» заочної форми навчання Осмоловської Юлії Сергіївни.

Тема: «Веб-додаток реалізований мовою програмування Ruby та фреймворком Ruby on Rails»

Затверджено наказом по СумДУ

№_____від_____2021 р.

Зміст пояснювальної записки: 1) аналітичний огляд методів створення веб-додатків, що базуються на архітектурному шаблоні MVC 2) постановка завдання та формування завдань до структури веб-додатку, для просування послуг в мережі інтернет 3) огляд та опис програмної реалізації програмного продукту 4) аналіз результатів та висновки

Дата видачі завдання "_____" _____2021 р.

Керівник випускної роботи _____ Шаповалов С. П.

Завдання прийняла до виконання _____ Осмоловська Ю.

РЕФЕРАТ

Записка: 56 стр, 16 рис, 2 табл, 4 додатки, 14 джерел

Об’єкт дослідження - веб-додаток реалізований мовою програмування Ruby та фреймворком Ruby on Rail

Мета роботи – розробити Ruby on Rails веб-додаток для бізнесу, який буде включати в себе блог, форму комунікації з клієнтами. Веб-додаток повинен бути двомовний, адаптивний, зі зручним функціоналом.

Результати – розроблено, спроєктовано та програмно реалізовано веб-додаток, який містить блог, контакт форму, авторизацію та автентифікацію для адміністратора сайту; відвідувачі сайту мають можливість залишати коментарі та користуватись веб-додатком у двох мовних режимах – українському по замовчуванні, та російському. Відображення сайту спроєктовано для користування у десктопному та мобільному варіанті.

БЛОГ, RUBY ON RAILS, RUBY, ФРЕЙМВОРК, GEM ФАЙЛ,
MVC АРХІТЕКТУРНИЙ ШАБЛОН, POSTGRES

ЗМІСТ

ВСТУП

В сучасному світі говорити про важливість мати веб-сайт не доводиться. Кожен бізнес, не залежно від його розміру, або сфери діяльності повинен бути представлений в мережі інтернет. Присутність бізнесу в інтернеті, незалежно від галузі, може мати значний вплив на його успіх. У наш час деякі компанії досі не усвідомлюють, що більшість їх клієнтів відвідають їх веб-сайт перед тим, як зробити покупку або скористатись послугами. Також одним зі способів, виділитись серед конкурентів це мати сайт, що гарно функціонує, підтримує адаптивність до різних девайсів, має свіжі та актуальні статті, посилається на соцмережі бізнесу. Якісно розроблений продукт, дозволяє виглядати в очах клієнтів більш авторитетно.

Останні світові події, також підштовхують бізнес ставати більш технологічним та доступним цілодобово. Навіть присутність офлайн бізнесу у всесвітній мережі необхідна. Це підвищує довіру до якості та законності послуг і товарів, полегшує пошук інформації про послуги й формує уявлення про бренд, дозволяє збільшувати кількість потенційних клієнтів та підвищувати конверсію, стимулює органічно просуватись в пошукових мережах, заощаджувати час на обслуговування клієнтів тощо.

Кожен бізнес прагне встановити зв’язок і спілкуватись зі своїм клієнтом кількома каналами – соціальні мережі, електронні листи. Однозначно одним з таких місць зіткнення інтересів є бізнес-блог. Якщо на веб-сайті компанії присутній блог, що відноситься до товару чи послуг, це завжди привертає клієнтів. З одного боку, клієнт отримує якісну та вузькоспеціалізовану інформацію від професіоналів, з іншого боку бізнес отримує лояльного клієнта.

Відповівши на питання навіщо бізнесу веб-сайт, логічним наступним питанням буде, який саме сайт і як він повинен працювати. Кількість технологій постійно зростає, тому складність вибору теж є. Створення програмного продукту – це не просто розробка приємного та зручного інтерфейсу; мова також йде про розробку стабільного, безпечного та ремонтопридатного продукту, який не тільки завоює серце клієнта, але й дозволить розширювати бізнес.

Веб-сайт є одночасно інструментом маркетингу. Бізнес хоче, щоб якомога більше людей бачили його сайт. Найпоширеніший спосіб пошуку веб-сайту за допомогою пошукової системи, тому якісно оформлений та постійно актуальний контент, підійматиме веб-сайт вище рейтинг у відповідних пошукових запитах.

Підсумовуючи необхідність сайту для бізнесу, можемо виділити основні переваги, які отримує підприємство:

- цілодобовий віртуальний офіс
- можливість розповісти про свої конкурентні переваги, діяльність компанії
- власний унікальний засіб масової інформації
- недорогий вид реклами
- надавати відповідь на типові питання клієнтів, та розвантажити цим працівників
- за допомогою сайту можливо приваблювати нових клієнтів, та підтримувати зв’язок з вже існуючими

Розробляючи блог для бізнесу, ми повинні бути переконані, що він відповідає вимогам. Ключовою особливістю блогу є те, що він дозволяє відвідувачеві залишити коментар для інших відвідувачів, щоб побачити його та прокоментувати. Як правило, блог містить текст, зображення та інші види медіа-файлів. Текстова інформація або текстовий контент сайту - це основний спосіб передачі відомостей відвідувачеві. Текст, найбільш пластичний вид контенту, він добре піддається seo-оптимізації і з нього можна створити ідеальний інструмент, здатний заощадити бюджет будь-якого проекту. Статті в блозі також можуть містити посилання на веб-сторінки, медіа-файли чи навіть інші блоги. Блог дозволяє використовувати віджети, такі як архівна інформація, останні коментарі та останні повідомлення. Повідомленнями в блозі можна ділитися з користувачами інших сайтів соціальних мереж, таких як Facebook, Twitter тощо.

Отже, мета даної роботи - розробити легкий в користуванні, адаптивний, пристосований для подальшого розширення та додавання нового функціоналу, двомовний сайт для бізнесу зі вбудованим блогом. Також програмний продукт повинен бути конкурентним та вигідно вирізнятись серед інших аналогічних

сайтів. Через те, що все більше і більше представників бізнесу, що реалізовують свої товари та послуги, не лише розширюють свої офлайн послуги. А намагаються головним чином бути представлені в мережі Інтернет. Так як це розширює бізнес і стимулює клієнта бути більш лояльним.

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Веб-фреймворк

Поняття фреймворк – це абстрактний термін, який містить в собі концепцію відокремлення від несуттєвих і неважливих аспектів програми та концентрацію на найбільш значущих з них. Поняття абстракції використовується не тільки в сфері програмної інженерії, а й в нашому повсякденному житті.

Проведемо аналогію. Нам не потрібні знання термодинаміки для того, щоб навчитися керувати автомобілем. Нам досить зрозуміти процеси взаємодії водія з автомобілем, досить складно влаштованими механізмами. Тобто ми сприймаємо автомобіль як абстракцію, за якою приховано весь досвід людства за останні 100 років еволюції індустрії. І все що нам потрібно – просто крутити кермо і натискати на педалі. Все інше відбувається без нашого втручання і не передбачає глибоких пізнань в різних областях фізики. У програмних термінах, можна сказати, що нам необхідно вміти використовувати інтерфейс автомобіля.

Перевага такого підходу в програмній індустрії полягають в тому, що програмісту досить просто включитися в гру, використовуючи ті інструменти, які розроблені колегами раніше. Досить знати, яким чином використовувати той чи інший прикладний програмний інтерфейс (API) та користуватись надбаннями, які вже були написані раніше. В той самий час, нашою роботою, так само можуть користуватись просто ознайомившись з документованим кодом.

Бібліотека – це набір методів або об'єктів, які вирішують конкретне завдання і якимось одним способом. Розробник сам приймає рішення, яка з бібліотек краще підходить для поставленої задачі. Після цього завдання буде вирішуватися тільки обраним способом і ніяк інакше.

Фреймворк, має більш складну структуру. Він надає цілий скелет (каркас) майбутньої програми. Використання фреймворка позбавляє нас від необхідності створювати інфраструктуру на низькому рівні. Це дає можливість зосередитися виключно на бізнес-логіці додатка, що, безумовно, призводить до створення програми більш високого класу. Веб-фреймворк - це програмний інструмент, який забезпечує спосіб створення та запуску веб-додатків. Як результат, не потрібно самостійно писати код і витрачати час на пошуки можливих прорахунків та

помилок. Веб-фреймворки існують, щоб спростити розробнику створення веб-програми, вони допомагають пришвидшити процес розробки та написання коду. [5]

Фреймворк допомагає виконувати одноманітні та постійно повторювані операції швидко, і не переписуючи схожий функціонал, з одного місця в інший.

Це програмне забезпечення, розроблене з метою спрощення процесу веб-розробки та спрощення побудови веб-сайту. Він включає можливості шаблонування, які дозволяють представляти інформацію в браузері, забезпечує середовище для сценаріїв потоків інформації, а також містить безліч інтерфейсів прикладного програмування (API) для отримання доступу до базових ресурсів даних . Більшість фреймворків також надають інструменти для того, щоб веб-розробники побудували систему управління контентом (CMS) для управління інформацією на веб-сайтах та в Інтернеті.

Веб-додаток, створений на основі фреймворку, може розглядатися як заздалегідь побудована структура, яка обробляє більш повторювані процеси та функції, пов'язані з розробкою веб-сайту. Це означає, що веб-розробник витрачатиме більшу частину часу на взаємодію з різними частинами веб-фреймворку за допомогою коду.

Деякі функції, пов'язані з веб-фреймворками, включають веб-кешування, процедури автентифікації та авторизації, відображення та конфігурацію бази даних та відображення URL-адрес. [1]

1.2 Приклади веб-фреймворків

Розглянемо в розрізі функціоналу три найбільш популярні веб-фреймворки, які займаються вирішення аналогічної проблематики та найбільш конкурують між собою.

Django – це безкоштовний веб-фреймворк на основі Python, що відповідає архітектурному шаблону model – template – views (MTV). Він підтримується Django Software Foundation (DSF), американською незалежною організацією, створеною як некомерційна організація.

Laravel – це безкоштовний веб-фреймворк з відкритим кодом PHP, і призначений для розробки веб-додатків за архітектурним зразком модель – вид –

контролер (MVC) і заснований на Symfony. В ньому присутня модульна система пакування зі спеціальним менеджером залежностей, різні способи доступу до реляційних баз даних, утиліти, які допомагають у розгортанні та обслуговуванні додатків.

Ruby on Rails, або Rails, – це серверна програма веб-додатків, написана на Ruby під ліцензією MIT. Rails – це веб-додатків за архітектурним зразком модель – вид – контролер (MVC), що забезпечує структури за замовчуванням для бази даних, веб-служби та веб-сторінок. Це заохочує та полегшує використання веб-стандартів, таких як JSON або XML для передачі даних та HTML, CSS та JavaScript для взаємодії з користувачами. На додаток до MVC, Rails наголошує на використанні інших відомих шаблонів та парадигм програмної інженерії, включаючи домовленість щодо конфігурації (CoC), не повторюваності себе (DRY) та активний шаблон запису.

Перша суттєва різниця між цими фреймворками полягає в тому, що Django написаний на Python, Laravel – на PHP, а Rails – на Ruby. Тобто як мінімум, використання та реалізація веб-додатків з використанням цих, фреймворків, вимагає від нас знання відповідних мов програмувань.

Розглянемо загальні риси вищеописаних фреймворків.

Всі вони є MVC (Django також називається MTV, але архітектурна концепція однакова). Усі фреймворки фокусуються на читабельності та простоті розповсюдження коду та файлів. Всі вони роблять автоматичні запити до баз даних. Тобто не потрібно писати запити до бази даних безпосередньо. І Django, і Laravel, і Ruby on Rails автоматично створюють таблиці в базі даних з моделей. Усі фреймворки мають прості та безпечні системи маршрутизації. Веб-сторінки відображаються динамічно. Ці фреймворки мають власні шаблонні системи, і кожна шаблонна система багата фільтрами та заздалегідь визначеними функціями. Існує лише різниця в синтаксисі. Всі вони гнучкі та портативні з іншими сучасними технологіями.

Тепер розглянемо кожен фреймворк окремо, та виділимо його особливості під час реалізація проектів, в кожному з них. Для порівняння фреймворків за

основні характеристики візьмемо простоту навчання, продуктивності, сильні та слабкі сторони бібліотек та шаблонів, підтримку від ком’юніті.

1. Django:

Django має дуже потужну бібліотеку з такими функціями:

– вбудований розділ адміністратора, декоратори та класи перегляду;

– автоматично згенеровані форми для моделей з валідацією;

– структура кешування, яка може використовувати будь-який із декількох методів кешування;

– підтримка класів проміжного програмного забезпечення, які можуть втручатися на різних етапах обробки запитів та виконувати власні функції;

– внутрішня диспетчерська система, яка дозволяє компонентам програми повідомляти події один одному заздалегідь визначеними сигналами;

– система інтернаціоналізації, що включає переклади власних компонентів Django на різні мови;

– система серіалізації, яка може створювати та читати XML та / або JSON подання екземплярів моделей Django;

– інтерфейс вбудованої модульної тестової основи Python;

– розширювана система автентифікації;

– динамічний адміністративний інтерфейс;

– інструменти для створення каналів RSS та Atom синдикації;

– фреймворк веб-сайту, що дозволяє одній установці Django запускати кілька веб-сайтів, кожен зі своїм вмістом та програмами;

– інструменти для створення файлів Google Sitemap;

– вбудоване пом'якшення для підробки міжсайтових запитів, міжсайтових сценаріїв, ін'єкції SQL, злому паролів та інших типових веб-атак, більшість з яких увімкнено за замовчуванням;

– структура для створення GIS-додатків; [8]

2. Laravel:

Бібліотеки Laravel не такі великі, як Django та Rails, але достатні для створення будь-якого веб-сайту.

– bundles (пакети) та composer (менеджер залежностей) забезпечують зв’язок модульних систем та залежностей;

– routing - найпростіший в управлінні та абстрактний спосіб маршрутизації;

– підтримка ORM – ще одна особливість, яка надається для абстрагування та автоматизації деталей моделі;

– міграції – спосіб набагато вишуканішої версії сценаріїв бази даних, не потрібно тримати всі перевірки щодо міграції;

– управління чергою – щоб упорядкувати неактуальні завдання, поставити їх у чергу та при необхідності набагато швидше відповідати користувачеві;

– внутрішня підтримка Redis (сховище пар ключ/значення з розширеним функціоналом і відкритим вихідним кодом);

– інжекція залежностей – просте тестування та автоматизація навантаження залежностей;

– artisan – створення миттєвих програм командним рядком; [7]

3. Ruby on Rails:

– RoR включає інструменти, що полегшують загальні завдання з розробки "out-of-the-box" (з коробки), наприклад, scaffolding, які можуть автоматично створювати деякі моделі та представлення, необхідні для базового веб-сайту. Також сюди входять WEBrick, простий веб-сервер Ruby, який поширюється разом з Ruby, і Rake, система збірки, що розповсюджується через gem. Разом з Ruby on Rails ці інструменти забезпечують базове середовище для розробки.

– ActiveRecord: він відіграє важливу роль у застосуванні RoR. Інтерфейс такого об'єкта включає функції CRUD, а також поля, що більш чи менш прямо відповідають полям відповідної таблиці в базі даних;

– “shortcts”: розробники, які походять з інших мов програмування або фреймворків, вважають Ruby on Rails фреймворком з великою кількістю ярликів (раціональний спосіб розв’язання проблем, які часто повторюються). Більшість

речей заздалегідь визначені, і потрібно написати лише кілька рядків коду, щоб скласти складну програму;

– автоматична маршрутизація: деякі загальні функції в таблицях баз даних, такі як створення, редагування та показ, визначаються автоматично. Це означає, що не потрібно витрачати час на просту роботу, і є можливість витратити більше зусиль на складні частини проекту;

– командний рядок: багато речей можна зробити за допомогою командного рядка, наприклад, використовуючи rake. Rake – це Ruby Make, автономна утиліта Ruby, яка замінює утиліту Unix "make" і використовує файли "Rakefile" та .rake для створення списку завдань. У Rails Rake використовується для загальних завдань адміністрування, особливо складних, які визнають одне одного;

– модуль ActiveModelHelper містить допоміжні методи для швидкого створення форм з об'єктів, які відповідають умовам Active Model, починаючи з моделей Active Record. [5]

1.3 Переваги та недоліки фреймворка Ruby on Rails

Ruby on Rails, також відомий як RoR and Rails – це веб-фреймворк, який є одним з найпопулярніших інструментів веб-розробки. Ruby on Rails був побудований на основі мови програмування Ruby, яка спочатку повинна була бути технологією для швидкої розробки програмного забезпечення. Ось чому Ruby on Rails часто називають «технологією запуску». Він був створений для швидких запусків продуктів.

Розглянемо спочатку переваги даного фреймворку:

– економічна рентабельність – фреймворк Ruby on Rails є 100% безкоштовним і працює на Linux, який є фреймворком з відкритим кодом. Також з ним легко працювати з точки зору розробника. Доступно безліч gem-ів (плагінів), тому це може заощадити багато часу та зусиль розробника;

– масштабованість – це забезпечує архітектура MVC. Модель містить алгоритми, які працюють над даними програми. Він централізує бізнес-логіку програми та правила маніпулювання даними. Дані користувачам включають формати HTML, PDF, XML, RSS та інші. Контролер взаємодіє з моделями та

видами. Він отримує запит від браузера, працює з моделями для його обробки та вказує, як правильно відображати результат користувачеві;

– простота управління змінами – Ruby on Rails дозволяє легко змінити існуючий код або додати нові функції. Після запуску сайту майбутні модифікації вашого сайту (наприклад, внесення будь-яких значних змін у модель даних) легко та швидко виконуються. Ця структура є найбільш ефективною для довгострокових проектів завдяки своїй стабільності та передбачуваності;

– захищеність – деякі заходи безпеки будуються в фреймворку та вмикаються за замовчуванням. Спільнота Rails активно працює над виявленням та виправленням нових вразливих місць, і система добре задокументована як офіційно, так і неофіційно;

– гнучкість – веб-додатки використовують бекенд і фронтенд можливості Rails. Тому програма швидко та легко комунікує як з користувачем, так і сервером;

– продуктивність – у поєднанні зі сторонніми бібліотеками Rails дозволяє неймовірно швидко розробляти функції. Це одна з найбільш продуктивних мов програмування. Безліч бібліотек упаковані в програми, які можна встановити за допомогою інструменту під назвою Ruby Gems.

– послідовність – розробники дотримуються стандартизованих правил зберігання файлів та програмування, які забезпечують структурованість та читаність проекту. Також це додатково економить багато часу.

– велике ком’юніті. Якщо вам потрібна певна функція, є всі шанси, що хтось інший створив щось подібне раніше або готовий допомогти вам розв’язати будь-які проблеми, які у вас можуть виникнути.

Як і будь-що, RoR теж має свої недоліки.

Однією з проблем є повільна швидкість виконання. Це правда, що інші середовища та фреймворки (Node.js або Django) дещо швидші, ніж RoR.

Наприклад, Twitter намагався підвищити ефективність Ruby on Rails, яка погіршилася після того, як соціальна мережа стала дуже популярною. Хоча Twitter і не повністю відмовився від Ruby on Rails, йому довелося замінити певні компоненти внутрішнього зв'язку та серверні частини. Також швидкість

завантаження проекта займає досить багато в часу в порівняні з аналогічними фреймворками, такими як Django або Laravel.

Проаналізувавши та розібравшись, з найсвіжішими програмними рішеннями для реалізації веб-додатків, вибір зупинився на Ruby on Rails. Важливим аргументом, крім перерахованих вище, стали стабільні відсоткові показники використання розробниками мови програмування Ruby.

2. ПОСТАНОВКА ЗАДАЧІ

2.1 Веб-додаток для просування послуг в мережі інтернет

Одним зі способів демонстрації конкурентної спроможності фреймворку Ruby on Rails, є реалізація веб-додатка, який буде закривати потреби сучасного ринку. Наприклад, сучасний сайт для бізнесу повинен забезпечувати пошук релевантних клієнтів зі сторони бізнесу та швидкий доступ до отриманих послуг, а також надавати гарантію якості споживачу. [3]

Головна задача веб-сайту — ефективна комунікація з користувачами на різних етапах знайомства з бізнесом. А також надалі, підтримувати зв’язок як з поточним користувачем, так і з потенційним клієнтом.

Основні структурні частини веб-сайтів обумовлені потребами бізнесу. В нашому випадку, проаналізувавши запит малого та середнього бізнесу, що спеціалізується на послугах, виділимо наступні частини додатку:

1. блог
2. контакт форма
3. авторизація та автентифікація для адміністратора сайту
4. коментарі для відвідувачів
5. локалізація

Розглянемо кожну бізнес-задачу окремо для обґрунтування та розуміння її необхідності в структурі веб-додатка.

Хоча у більшості підприємств є повноцінні програми контент-маркетингу і блоги, для малого бізнесу або стартапів також важливо створити блог. Сам по собі веб-сайт не завжди пропонують достатньо інформації, щоб по-справжньому виділитися з натовпу. Основна причина, по якій компаніям потрібен блог (рис. 2.1), — це зробити його більш помітним. Простіше кажучи, чим більше контенту в блозі, тим більше можливостей у бізнесу з'явитися в пошукових системах і залучити органічний трафік на свій сайт. Блоги надають ідеальну платформу для посилення стратегії SEO (Search Engine Optimization). Бізнес може підвищити свої шанси генерувати трафік і конвертувати потенційних клієнтів, створюючи нові та добре написані статті, що включають довгі ключові слова, зображення і відео.

Рисунок 2.1. – Відображення списку двох опублікованих постів

Коли на сайт потрапляє новий відвідувач, це створює нову можливість для залучення клієнтів. Наприклад, кнопки CTA (Call-To-Action). CTA — це спосіб направляти відвідувачів вашого веб-сайту, утримувати їх на вашому сайті, направляти їх на шляху від обізнаності до покупки. Вмотивувати відвідувача веб-додатка написати коментар, є одним з прикладів використання CTA. Також можливо використовувати кнопку CTA в блозі, щоб перетворити їх у відповідну сторінку на веб-сайті, яка пояснює послуги.

Контактна форма - стандартний інструмент зворотнього зв'язку (рис 2.2). Через контактну форму можливо отримувати відгуки, скарги та пропозиції відвідувачів. За допомогою контактної форми користувачі ставлять питання, що стосуються конкретних послуг. Це ефективний інструмент формування клієнтської та підписної бази, тому що всі користувачі вказують свою електронну пошту або телефонний номер. З позиції бізнесу контактна форма — це додатковий генератор потенційних клієнтів, де конверсія напряму залежить від кількості взаємодій з веб-додатком.

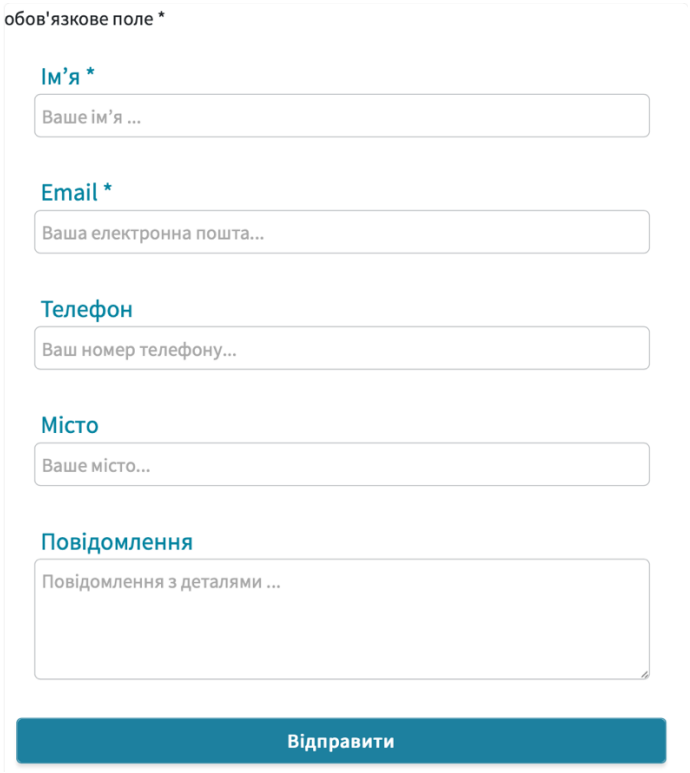

Рисунок 2.2. – Контакт-форма, для зворотного зв'язку

Реєстрація — це спосіб повідомити сайту дані й в обмін отримати доступ до додаткових можливостей (наприклад, додавання чого-небудь в обране) або ресурсів (наприклад, файлів) на сайті, які недоступні гостям (рис 2.3). Більшості відвідувачів сайту реєстрація взагалі не потрібна. Тому за допомогою авторизації та автентифікації ми розділяємо доступ до функціонала веб-сайту.

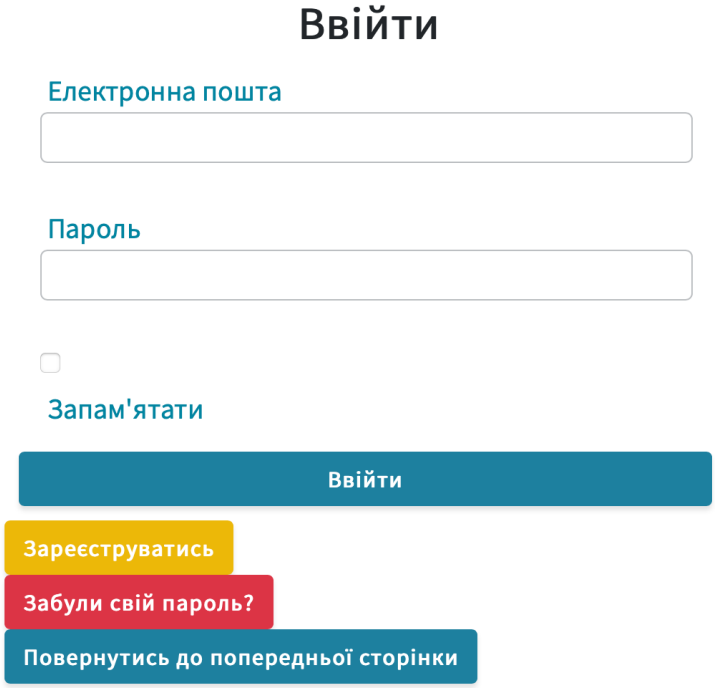

Рисунок 2.3. – Форма для авторизації

Локалізацію часто плутають з перекладом, але насправді ці терміни означають дві різні речі. Згідно Асоціації глобалізації та локалізації, локалізація — це весь процес адаптації продукту або контенту до конкретного місця розташування або ринку. Локалізація також включає адаптацію інших елементів до цільового ринку, в тому числі: зміна графіки та дизайну для правильного зображення перекладеного тексту, зміна вмісту відповідно до своїх уподобань, конвертація в місцеву валюту і одиниці вимірювання, використання правильного форматування для таких елементів, як дати, адреси та номери телефонів, дотримання місцевих норм і вимог законодавства. Тобто локалізація (рис 2.4 – 2.5). надає зовнішній вигляд сайту, який очікує цільова аудиторія.

Рисунок 2.4. – Приклад локального налаштування UK

Рисунок 2.5 – Приклад локального налаштування RU

Також важливими вимогами до будь-якого сайту є можливість подальшої самостійної підтримки його, крос-браузерність, адаптивність на всіх основних типах пристроїв. Тобто це звичайні мінімальні технічні вимоги, які повинен задовольняти будь-який сайт.

3. ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ

3.1 Ruby on Rails, як основний інструмент побудови веб-додатка

Структура Ruby on Rails додатка (рис 2.6), в повній мірі розкриває концепцію Model – View – Controller. В проекті присутнє повне розділення не тільки звичного бекенду та фронтенду, а й на **models,** що представляє дані, **view** що відображає та **controller**, що реагує на дії користувача, та повідомляє модель про необхідність змін.

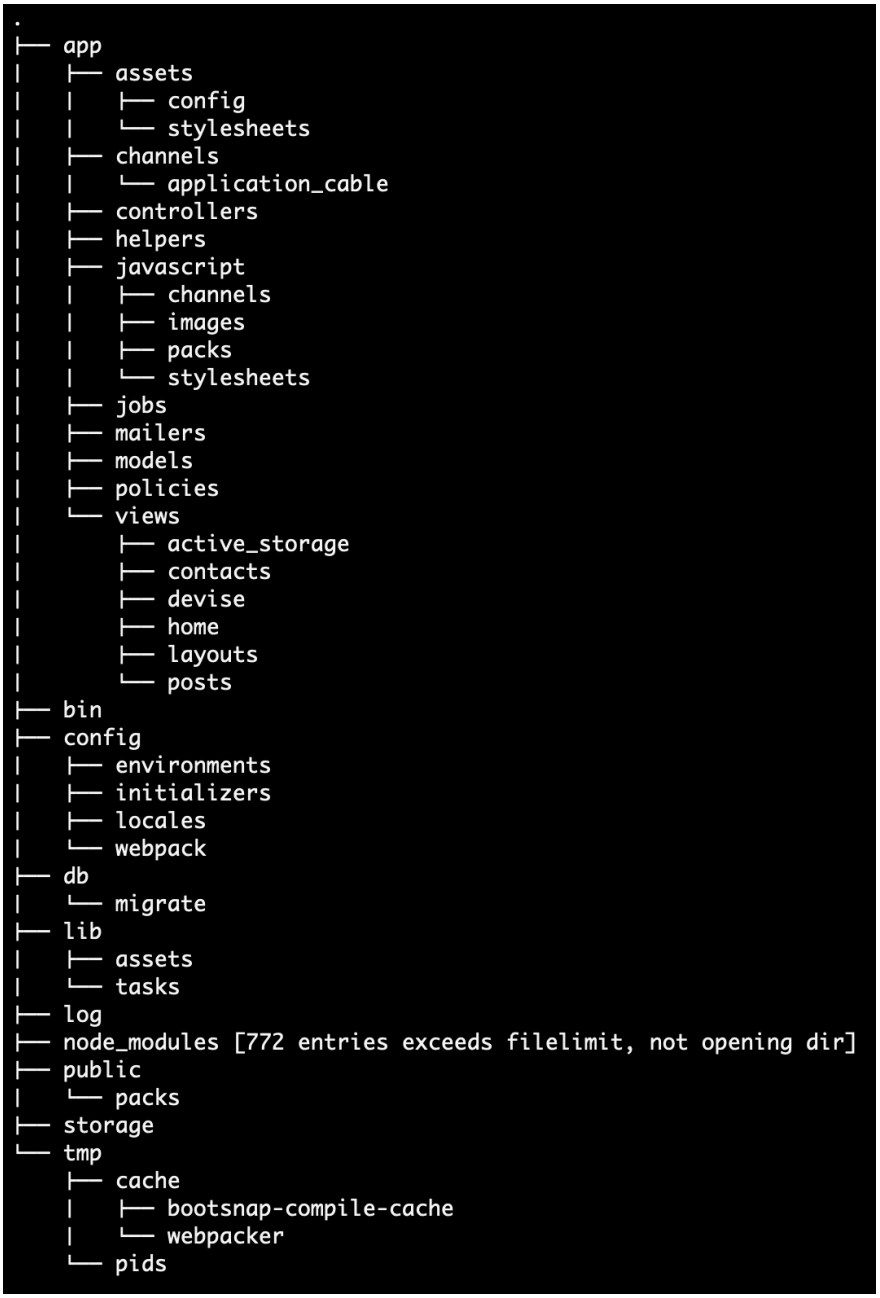

Рисунок 2.6 – Структура Ruby on Rails додатка

Ресурсний роутинг в Ruby on Rails (або маршрутизація) дозволяє швидко оголошувати всі загальні маршрути для заданого ресурсного контролера (рис 2.7). Кожен метод - це запит на виконання операції над ресурсом. Управління ресурсами відбувається в файлі **config/routes.rb**. Ресурсний маршрут відображає ряд пов'язаних запитів на дії в одному контролері.

Routes

Routes match in priority from top to bottom

Helper	HTTP Verb	Path	Controller#Action
Path / Url		Path Match	
root_path	GET	/	home#index
new_user_session_path	GET	/users/sign_in(.:format)	devise/sessions#new
user_session_path	POST	/users/sign_in(.:format)	devise/sessions#create
destroy_user_session_path	DELETE	/users/sign_out(.:format)	devise/sessions#destroy
new_user_password_path	GET	/users/password/new(.:format)	devise/passwords#new
edit_user_password_path	GET	/users/password/edit(.:format)	devise/passwords#edit
user_password_path	PATCH	/users/password(.:format)	devise/passwords#update
	PUT	/users/password(.:format)	devise/passwords#update
	POST	/users/password(.:format)	devise/passwords#create
cancel_user_registration_path	GET	/users/cancel(.:format)	devise/registrations#cancel
new_user_registration_path	GET	/users/sign_up(.:format)	devise/registrations#new
edit_user_registration_path	GET	/users/edit(.:format)	devise/registrations#edit
user_registration_path	PATCH	/users(.:format)	devise/registrations#update

Рисунок 2.7 – Основні марштури веб-додатка, що відповідають за авторизацію адміністратора

За допомогою маршрутів можливо, не тільки створювати посилання в рамках проекту, а й додавати нові методи. Також інтегруючи новий функціонал відразу бачити відповідні зміни.

Та найголовнішим інструментом, для розробки в Ruby on Rails залишається Rubygem. Кожен gem містить ім’я, версію та платформу. Gem працюють лише на Ruby, розробленому для певної платформи на основі архітектури процесора та типу та версії операційної системи.

3.2 Інтернаціоналізація I18n

Інтернаціоналізація (I18n) - процес створення і розробки продукту, який надалі забезпечить безбар'єрну локалізацію ПО. Природні мови відрізняються в багатьох особливостей, тому важко уявити інструменти, які вирішують відразу всі проблеми. Інтернаціоналізований продукт підтримує вимоги місцевих ринків у всьому світі, функціонуючи більш належним чином на основі місцевих норм та краще відповідаючи очікуванням користувачів у країні.

Локалізація додатку на Rails означає визначення переведених значень на бажані мови.

На головній сторінці додатка користувач відразу отримує можливість змінити локалізацію.

Ruby gem I18n розділений на дві частини:

- публічний API фреймворку I18n - модуль Ruby з публічними методами, що визначає як працює бібліотека
- бекенд, який реалізує ці методи

Дотримуючись філософії «Convention over configuration» тобто угода головніше над конфігурацією, Rails надає прийнятні строки перекладів за замовчуванням. При необхідності інших рядків перекладів, вони можуть бути перевизначені.

3.3 Менеджер пакетів для роботи залежностей додатка

Фундаментально важливими для правильної роботи Ruby on Rails додатка є менеджер управління gem-ами Bundler. Ця утиліта дозволяє легко встановлювати необхідні gem-и для застосування, при цьому зовсім не залежати від встановлених в системі. При використанні Rails залежності gem-ів задаються з допомогою **config.gem** та **enviroment.rb**. Bundler розв’язує цю задачу набагато зручніше і простіше. Його включили в Rails 3.0 за замовчуванням і тепер, саме він використовується для управління залежностями gem-ів в даній версії фреймворку. Цю утиліту можна використовувати для будь-якого Ruby фреймворку.

3.4 Об'єктно-реляційна система керування базами даних Postgres

При використанні веб-фреймворку Ruby on Rails програма за замовчуванням налаштована на використання SQLite як бази даних. SQLite - це легка, портативна та зручна реляційна база даних, яка особливо добре працює в середовищах з малою пам’яттю і в багатьох випадках буде добре працювати. Однак для складних додатків, які потребують більш надійної цілісності даних та розширення програмного забезпечення, база даних PostgreSQL стане більш надійним та гнучким вибором.

Опираючись на бізнес-вимоги, схему бази даних можемо умовно розділити на дві частини. Перша частина (рис. 2.8) відповідає за блог, друга частина (рис 2.9) за авторизацію і автентифікацію адміністратора та зворотній зв’язок у вигляді, контакт форми.

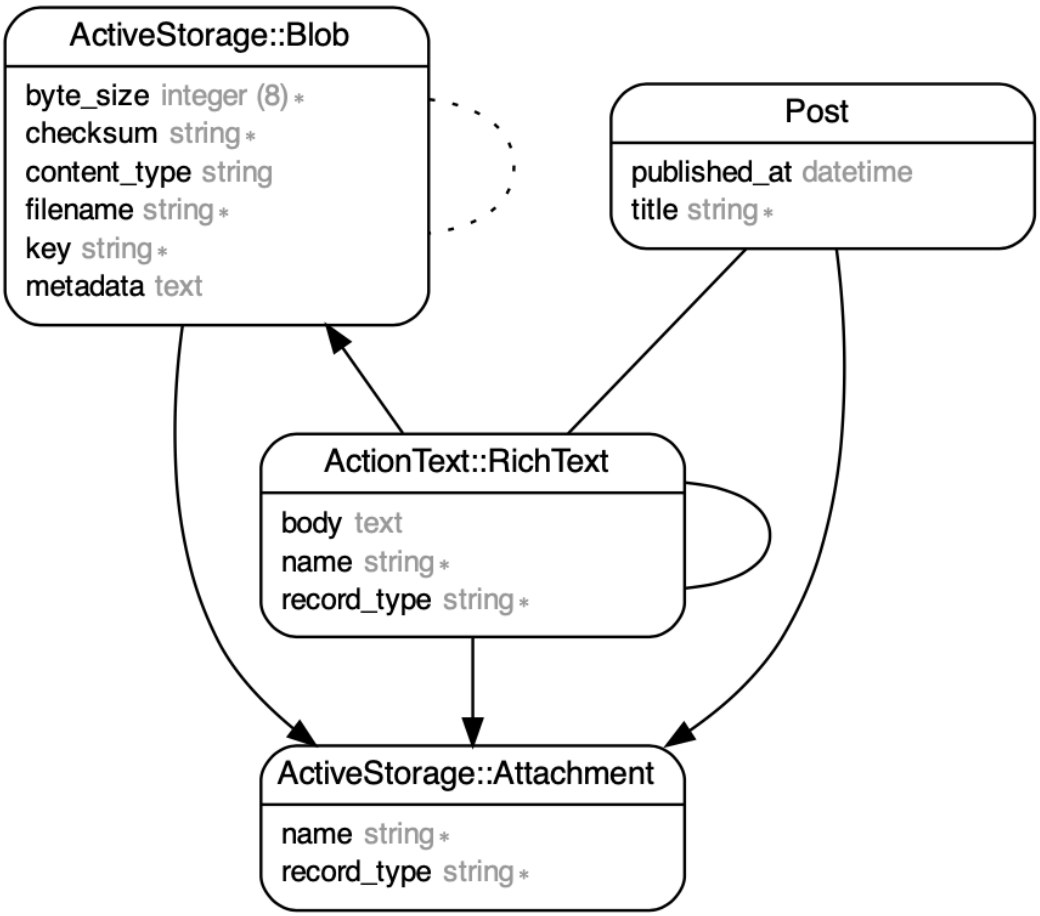

Рисунок 2.8 – Схема бази-даних, що обслуговує блог

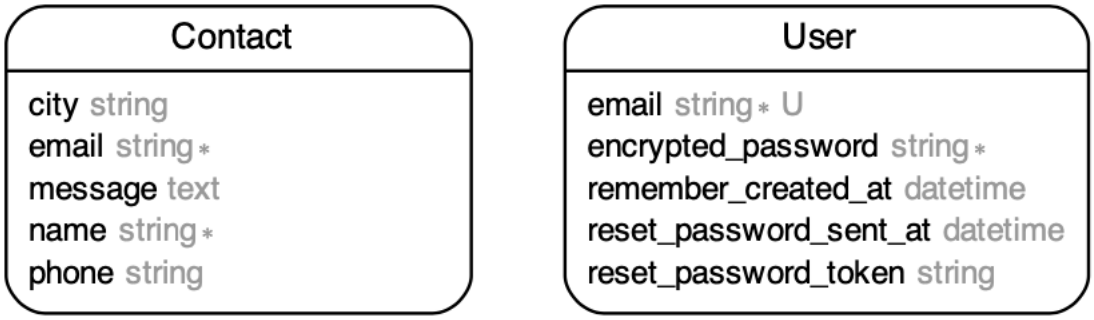

Рисунок 2.9 – Таблиці бази-даних, що зберігають інформацію про адміністратора та контакти клієнтів

Оскільки PostgreSQL - це об'єктно-реляційна база даних, масиви значень можуть зберігатися для більшості існуючих типів даних. Підтримка JSON в PostgreSQL дозволяє перейти до зберігання schema-less даних в SQL базі даних.

Вся структура бази даних знаходиться в **schema.rb** файлі (Додаток А). Цей файл документує останній поточний стан схеми бази даних.

3.5 Bootstrap, як фронтенд інструмент

Найпростіший спосіб стилізації веб-сторінки - це використання фреймворку. Існує безліч фреймворків, що дозволяють стилізувати веб-сторінку. Обравши Bootstrap, ми відразу вирішуємо велику кількість проблем. Це не просто набір готових інструментів (HTML фрагментів, класів, компонентів і плагінів), а й добре спроектований фронтенд фреймворк, який досить просто можна налаштовуватись за допомогою редагування Scss змінних.

Наприклад інтегруючи Bootstrap та використавши стандартні класи надані фреймворком, доволі легко добитися плавної анімації, яка працює гарно, яка гарно працює на повнорозмірному екрані (рис 2.10), і та ж сама частина інтерфейсу, на мобільному додатку (рис 2.11), змінює свою розмірну сітку та користуючись властивостями flexbox, залишається такою ж зручною для користувача.

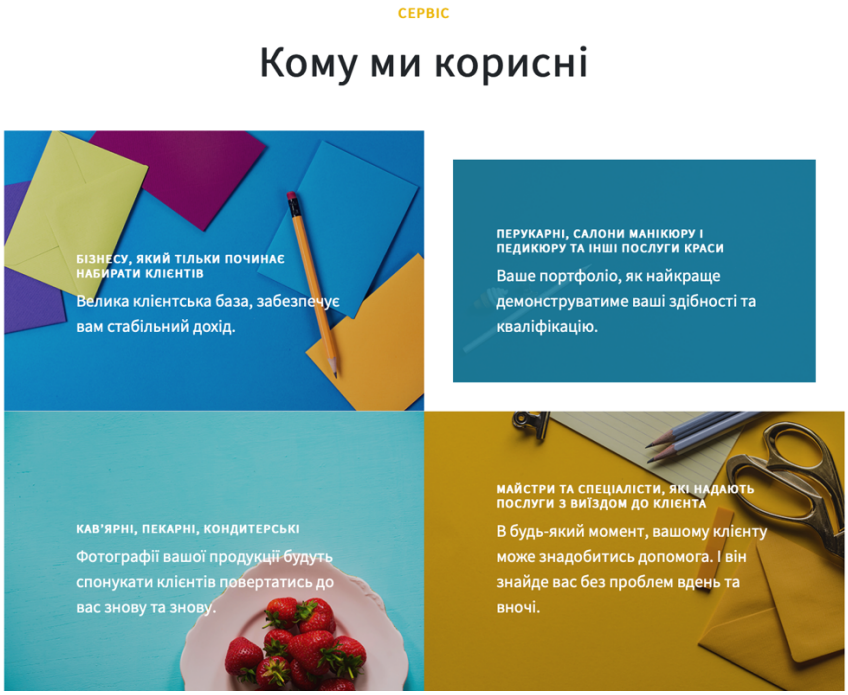

Рисунок 2.10 – Частина веб-додатку на повнорозмірному екрані

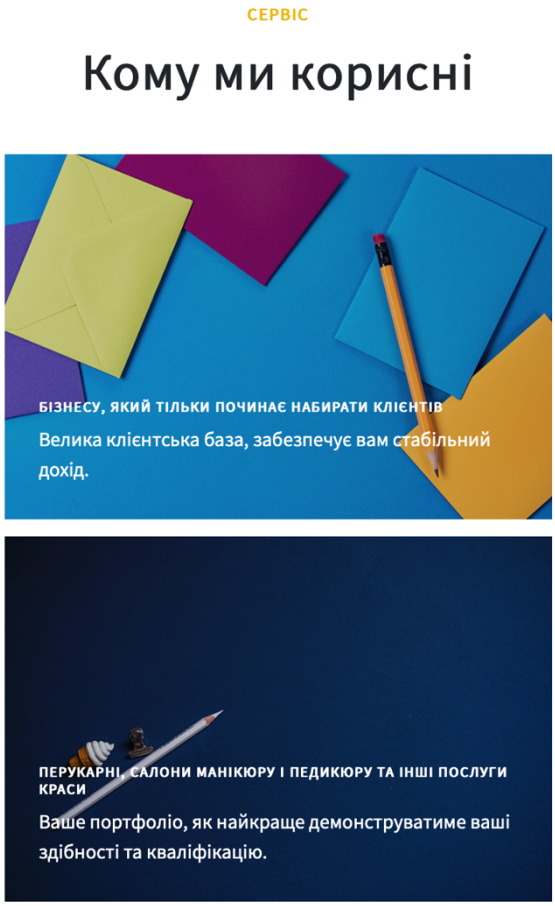

Рисунок 2.11 – Частина веб-додатку на мобільному екрані

Отже, вибір стеку технологій можливо є одним з найключовіших моментів під час розробки будь-якого цифрового продукту. Тому затрачений час на підбір та аналіз, можна назвати інвестицією, так як в подальшому бізнес, як замовник

продукту, отримує гарний результат у вигляді веб-сайту. Найбільше на вибір стеку впливають такі фактори, як розмір проекту, швидкість виходу на ринок, горизонтальна чи вертикальна масштабованість, можливість подальшого обслуговування веб-додатка, експертиза майбутніх розробників та звісно рівень безпеки, який вимагається від програмного продукту. Всі ці фактори в сумі з вимогами від замовника та коректно зумовлюють вибір програмних рішень.

4. ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 MVC архітектурний шаблон додатку

Ruby on Rails надає архітектурний зразок Model – View – Controller для веб-додатків, а також забезпечує їх інтеграцію з веб-сервером і сервером бази даних. Ruby on Rails є відкритим програмним забезпеченням і поширюється під ліцензією MIT.

На прикладі класу **Contact**, розглянемо весь ланцюжок роботи MVC шаблону додатку. Контакт-форма, збирає інформацію у відвідувачів сайту та зберігає інформаці як в базу даних, так і одночасно відправляє два електронних листа – один адміністратору сайту, другий користувачу про успішне заповнення форми.

В моделі ActiveRecord (**app/models/contact.rb**) - він підтримує взаємозв'язок між об'єктами та базою даних та обробляє перевірку, асоціацію. Ця підсистема реалізована в бібліотеці ActiveRecord, яка забезпечує інтерфейс та прив'язку між таблицями в реляційній базі даних та програмним кодом Ruby, який маніпулює записами баз даних. Назви методів Ruby автоматично генеруються з імен полів таблиць баз даних. в якому зберігаються об’єкти, що обробляються.

```
class Contact < ApplicationRecord
  validates :name, presence: true
  validates :email, presence: true, format: { with:
Devise::email_regexp }
  validates :phone, presence: false
  validates :city, presence: false
  validates :message, presence: false
end
```

Таблиця 4.1 – параметри ActiveRecord моделі, що відповідає за контакт-форму

Назва параметра, що містить в собі об'єкт	**Інформація, що зберігається**
name	Ім'я клієнта, обов'язково для заповнення
email	Електронна адреса клієнта, обов'язкова для заповнення, узгоджується зі сталим форматом для емейла
phone	Телефон клієнта, не є обов'язковим для введення
city	Місто в якому проживає клієнт, не є обов'язковим для введення
message	Додаткове повідомлення, не є обов'язковим для введення

Сама база даних, зберігає значення про контакти в файлі **db/schema.rb**, та має деяку додаткову інформацію, про створення або оновлення даних про клієнта і про ті дані, які зберігаються в таблиці бази даних.

```
create_table "contacts", force: :cascade do |t|
  t.string "name"
  t.string "email"
  t.string "phone"
  t.string "city"
  t.text "message"
  t.datetime "created_at", precision: 6, null: false
  t.datetime "updated_at", precision: 6, null: false
end
```

Перегляд (ActionView) контакт - форми, що відображається на головній сторінці веб-додатку розміщенно в **app/views/contacts/_form.slim** (Додаток В).

Для зручності розробки, використовується Ruby Slim - це легкий шаблонізатор для Ruby, мета якого скоротити синтаксис html до основних частин.

Наприклад замість коду в форматі html з великою кількість тегів та вкладених CSS класів, отримуємо

```
.col-lg-8.col-md-10.mx-auto.form-group
  .form-group
    = f.label :name
    = f.text_field :name, placeholder: true
```

де для правильного функціонування головне порядок відступів.

Подання даних у певному форматі, що зазначені в таблиці 4.1 ініційоване рішенням контролера представити дані. Вони являють собою шаблонні системи на основі сценаріїв.

Контролер для класу **Contact** розташований в файлі **app/controllers/contacts_controller.rb** (ActionController) – засіб у додатку, який спрямовує трафік, з одного боку, запитуючи моделі для конкретних даних, а з іншого боку, організовуючи ці дані (пошук, сортування, обмін повідомленнями через емейл) у форму, яка відповідає потребам даного представлення даних.

```
class ContactsController < ApplicationController
  def create
    @contact = Contact.new(contact_params)
    if @contact.save
      # Send email
      AdminMailer.contact_information(@contact).deliver
      redirect_to root_path(anchor: 'contact-form'), notice:
t(:message_sent)
    else
      render template: 'home/index', notice: t(:message_not_sent)
    end
  end

  private
    def contact_params
```

```
            params.fetch(:contact, {}).permit(:name, :email, :phone, :city,
:message)
          end
      end
```

Ця підсистема реалізована у ActionController, який є посередником даних, що знаходиться між ActiveRecord (інтерфейс бази даних) та ActionView (механізм презентацій).

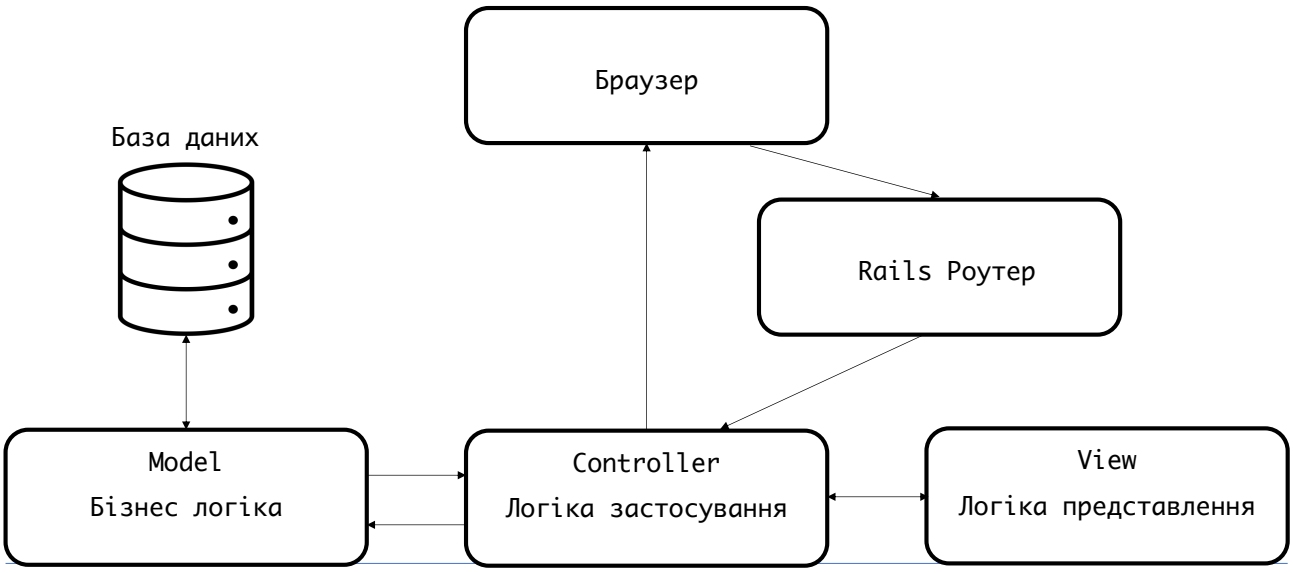

Рисунок 4.1 – Діаграма взаємодії між компонентами MVC

Аналізуючи схему роботи MVC класу **Contact** (рис 2.1) її можна коротко описати наступним чином:

– моделі для обробки даних та бізнес-логіки;

– контролери для обробки користувальницького інтерфейсу та програми;

– представлення для обробки графічних об'єктів інтерфейсу користувача та презентацій;

4.2 Gem файли

Важливими частинами реалізації веб-додатка Ruby on Rails є використання gem файлів, які забезпечують роботу бібліотек (Додаток Г)

При розробці веб-додатка було використано наступні gem файли – таблиця 4.2

Таблиця 4.2 – Головні Gem файли

Назва Gem файла	**Функціонал, який забезпечується даним файлом**
Device gem	Робить авторизацію та аутентифікацію в веб додатку.
Pundit gem	Інструмент, який дозволяє обмежити певні частини Rails додатка авторизованим користувачам, тобто розмежовує веб-додаток.
Slim gem	Дозволяє мінімізувати та оптимізувати HTML об’єкти
Figaro gem	Цей плагін зберігає дані конфігурації та SCM окремо один від одного. Він аналізує файл YAML і завантажує його значення в ENV.
Image_processing gem	бібліотека Ruby, що допомагає читати та писати зображення PNG.
Puma gem	забезпечує роботу веб-сервера.
Pg gem	інструмент для установки пакета PostgreSQL.
Bootsnap gem	підключається до ряду методів Ruby та ActiveSupport та YAML для оптимізації та кешування великих обчислень.
Webpacker	gem, що надає гладку і стандартну інтеграцію Rails зі збирачем webpack і пакетним менеджером yarn.
Sass-rails gem	Плагін для метамови на основі CSS.

4.3 Робота блогу

Робота сайту умовно розділена на дві частини. Перша частина розроблена для клієнта. Він може перемикатись від статті до статті, які присутні в блозі, переходити на сторінки соціальних мереж тощо. Комунікація з зовнішніми сайтами можлива двома шляхами:

- через блок кнопок, що розташовані у футері головної сторінки.
- через навігаційне меню, що розташоване в лівому верхньому куті.

Але для того щоб відвідувач сайту, читав статті, їх потрібно спочатку написати.

Після реєстрації (Рисунок 4.2), авторизований користувач перенаправляється до сторінки з можливістю писати статті.

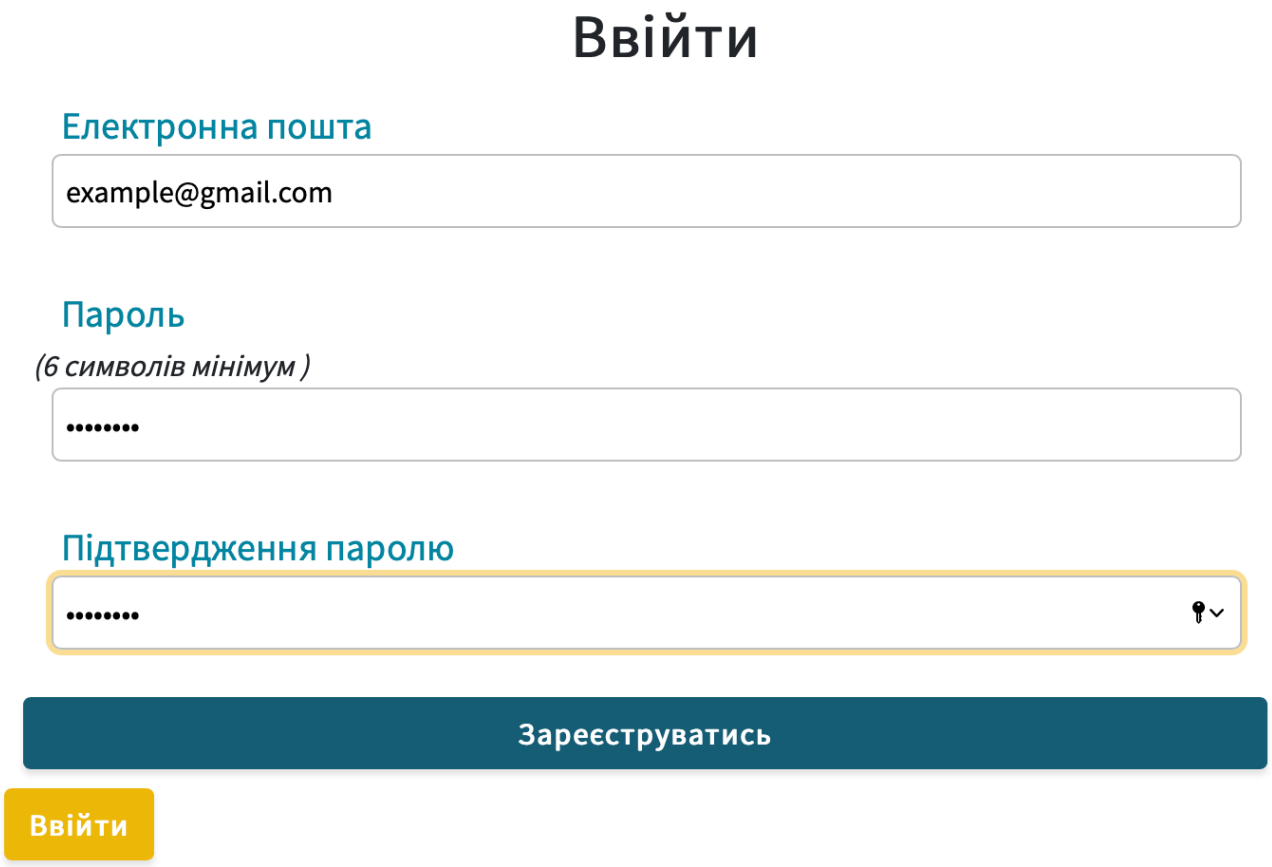

Рисунок 4.2 – Форма реєстрації

За цю частину відповідає клас Posts у файлі **app/controllers/posts_controller.rb** (Додаток Д)

Методи цього класу, дозволяють авторизованому користувачу статті: опубліковувати, редагувати, видаляти, та залишати їх неопублікованими

```
class PostsController < ApplicationController
  before_action :authenticate_user!, except: [:index, :show]
  before_action :set_post, only: [:show, :edit, :update, :destroy,
:publish, :unpublish]
```

Кожна стаття, також має свій функціонал. На рисунку 4.3 ми бачимо інтерфейс написання посту. Статтю можемо умовно розділити на 3 частини – заголовок, попередній перегляд, та сама стаття безпосередньо.

```
.field.form-group
  = f.label :title
  = f.text_field :title, class: "title"
.field.form-group
  = f.label :preview
  = f.rich_text_area :preview
.field.form-group
  = f.label :content
  = f.rich_text_area :content
```

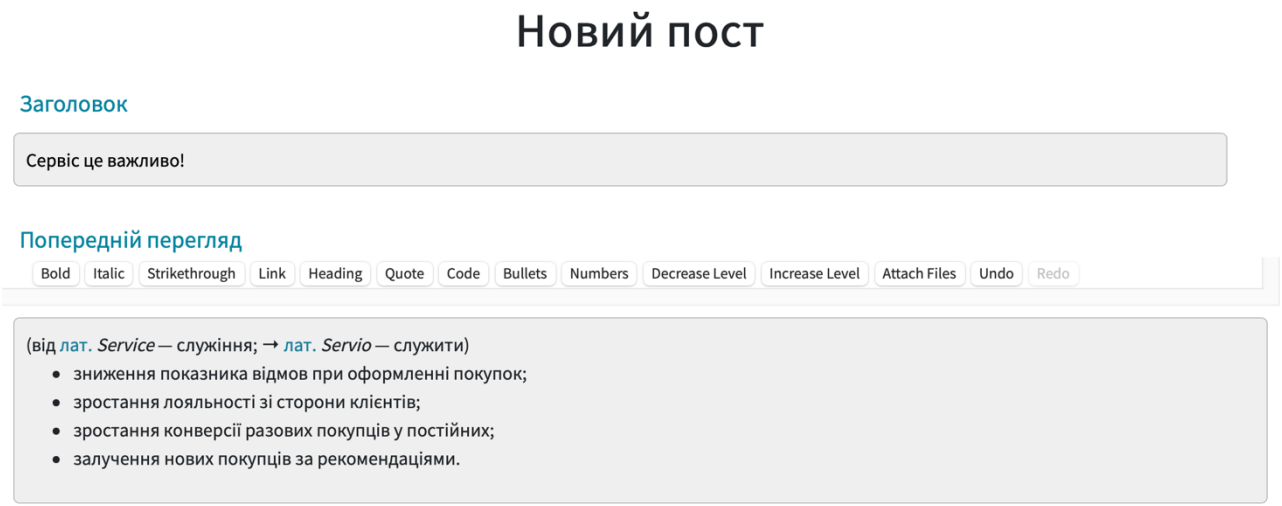

Рисунок 4.3 – Форма для написання блогу

Сам текст можливо модифікувати різними стилями, кольорами, додавати до постів картинки.

Перед опублікованням поста, автор має можливіть перевірити його, внести зміни.

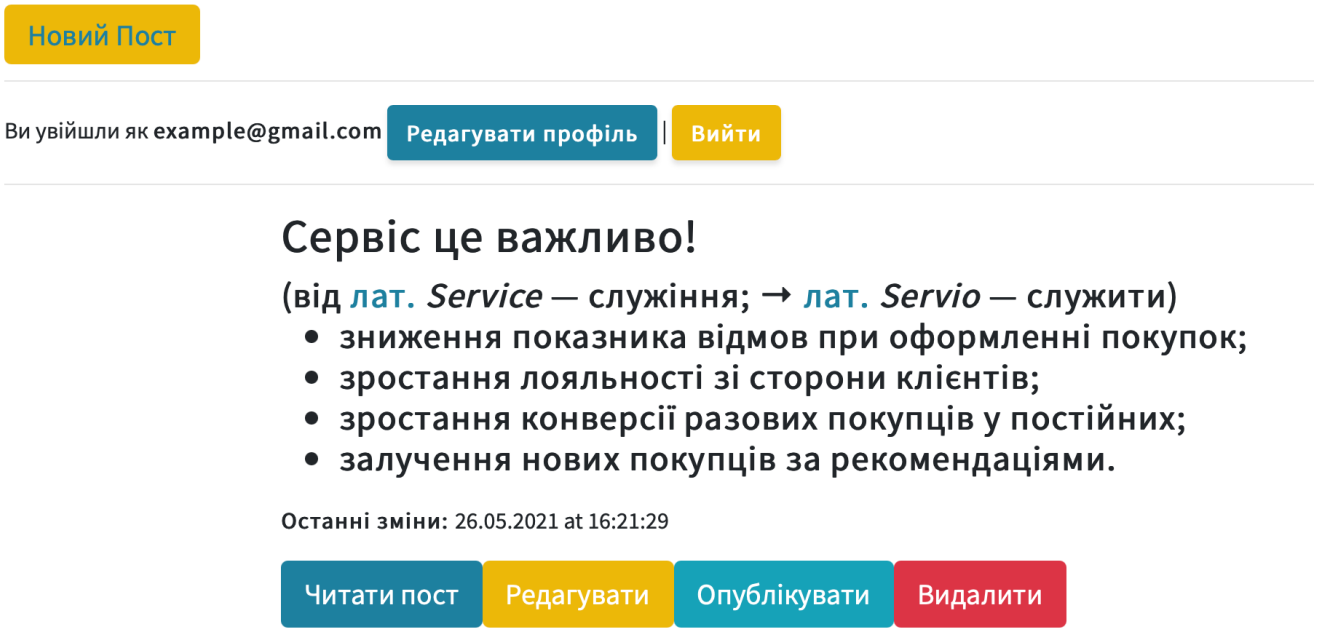

Рисунок 4.4 – Неопублікований пост

Опублікований пост стає доступним для читання всім користувачам. Також користувачі мають можливіть додавати коментарі (рис. 4.5), рекомендувати статті та реагувати на них деяким набором смайлів. Даний функціонал надається інтернет-службою Discus.

```
.container
  #disqus_thread
  javascript:
    var disqus_config = function() {
      this.page.url = "#{post_url(@post)}";
      this.page.identifier = "post-#{@post.id}";
    };

    (function() {
      var d = document, s = d.createElement('script');
      s.src = 'https://siteukraine-com.disqus.com/embed.js';
      s.setAttribute('data-timestamp', +new Date());
    (d.head || d.body).appendChild(s);
    })();
```

```
noscript
  | Please enable JavaScript to view the 
  a href="https://disqus.com/?ref_noscript" comments powered by
Disqus.
```

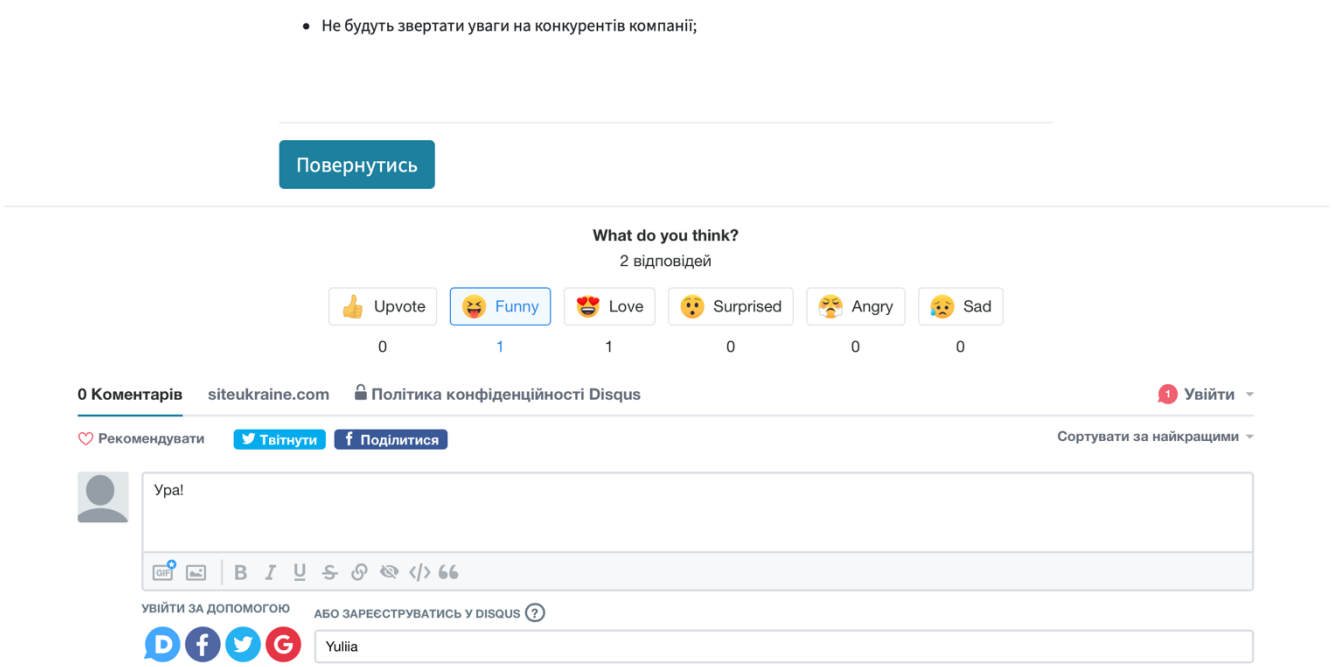

Рисунок 4.5 – Форма для введення коментарів

4.4 Двомовність сайту

При перемиканні кнопок на головній сторінці, можливо перемикати весь функціонал з української мови на російську та навпаки.

```
section.button-lang
  btn.btn-primary.btn-lg = link_to 'UA', { :locale=>'uk' }
  btn.btn-secondary.btn-lg = link_to 'RU', { :locale=>'ru' }
```

До кожного слова, або символа, що потребує переклада, додається **I18n**, разом з ключовим словом. А самі переклади зберігаються у файлах **config/locales/.ru.yml** та **config/locales/uк.yml.**

Управління локалізації відбувається, в файлі **app/controllers/application_controller.rb**, (Додаток Г). При інтернаціоналізації веб-додатка, відвідувач сайту отримує всі повідомленні про реєстрацію, відправку

листів, коментування блога також вибраною ним мовою. А до правильно спроектованого веб-додатку, в майбутньому дуже легко додати інші мови. Так як **I18n**, це комплекс механізмів який дозволяє розробляти багатомовні додатки. Він дозволяє локалізувати текст інтерфейсу, статичні зображення, специфічну верстку і записи таблиць бази даних. Іншими словами, можна локалізувати статичні і динамічні дані. В процесі розробки додатку, вже було додано англійську мову. Тому при необхідності розширювати бізнес за кордон, або плануванні дотримуватись іншої бізнес-стратегії, що включає в себе комунікацію з іноземними користувачами, достатньо на головній сторінці додати відповідну кнопку, що буде перемикати локаль.

ВИСНОВКИ

В ході виконання роботи було проаналізовано та виділено найголовніші задачі, що постають перед веб-сайтом, який обслуговує бізнес. Розглянувши основні концепції побудови веб-додатку, та зваживши всі плюси та мінуси було прийнято рішення, проектувати програмний продукт використовуючи мову програмування Ruby та фреймворк Ruby on Rails.

Дослідивши основну проблематику бізнес-сайтів та структурно описавши їх, до кожного пункта було підібрано відповідне рішення, що обумовлюється певною або функціональною частиною, або концептуальним підходом. Наприклад, двомовність сайту додає більшу кількість лояльних клієнтів та задовільняє вимоги законодавства. А можливість клієнта залишати коментарі, або користуватись контакт-формою для зворотнього зв'язку - дозволяє формувати клієнтську базу.

Гнучкість, економічна ефективність, швидкість, масштабованість, високий рівень захисту даних, надійність, ремонтопридатнісь, доступ до найширшої бази знань та інші фактори - все це переваги Ruby on Rails, які дають конкурентну перевагу стартапам та бізнесу.

Отже, метою роботи було створення веб-додатку для бізнеса, який повністю задовольняв би потреби ринку в сфері просування послуг та товарів в мережі інтренет. Під час розробки веб-додатку були використані наступні технології та поняття:

- мови програмування: Ruby, JavaScript;
- мова розмітки: HTML, SLIM, CSS, SASS;
- фреймворки та бібліотеки: Ruby on Rails, Bootstrap;
- бази даних: PostgreSQL;
- концепції: шаблон дизайну MVC, об'єктно-орієнтовне програмування, функціональне програмування, DRY

Розроблений веб-додаток, в повній мірі відповідає поставленим вимогам: підсилює маркетингові канали бізнесу, допомагає залучати додатковий трафік на сайт, генерувати нових потенційних клієнтів, підтримувати довгостокові результати, та сприяє формуванню бренду.

СПИСОК ЛІТЕРАТУРИ

1. Using Rails for API-only Applications [Електронний ресурс]. – Режим доступу: https://guides.rubyonrails.org/api_app.html– 12.05.2021р.

2. What is a Web Framework [Електронний ресурс]. – Режим доступу: https://www.goodfirms.co/glossary/web-framework/ – 15.10.2020р.

3. Why Blog? The Benefits of Blogging for Business and Marketing [Електронний ресурс]. – Режим доступу: https://blog.hubspot.com/marketing/the-benefits-of-business-blogging-ht/– 17.10.2020р.

4. Incredibly useful advantage of ruby on rails you must know Framework [Електронний ресурс]. – Режим доступу: https://mindstack.in/blog/advantages-ruby-rails/ – 15.10.2020р.

5. Pros and Cons of Ruby on Rails [Електронний ресурс]. – Режим доступу: https://sloboda-studio.com/blog/pros-and-cons-of-ruby-on-rails/– 20.10.2020р.

6. Django vs Laravel vs Rails [Електронний ресурс]. – Режим доступу: https://www.flowkl.com/article/web-development/django-vs-laravel-vs-rails/ – 20.10.2020р.

7. Understanding the Model-View-Controller (MVC) Architecture in Rails [Електронний ресурс]. – Режим доступу: https://www.sitepoint.com/model-view-controller-mvc-architecture-rails/ – 01.11.2020р.

8. Meet Laravel - Why [Електронний ресурс]. – Режим доступу: https://laravel.com/docs/8.x#why-laravel/ – 05.02.2021р.

9. About the Django Software Foundation - Why [Електронний ресурс]. – Режим доступу: https://www.djangoproject.com/foundation/ – 05.02.2021р.

10. TCP/IP Internet protocols [Електронний ресурс]. – Режим доступу: https://www.britannica.com/technology/TCP-IP – 10.05.2021р.

11. What Is Localization, And When Do You Need It? [Електронний ресурс]. – Режим доступу: https://blog.languageline.com/what-is-localization – 29.04.2021р.

12. Ruby on Rails Technology Stack [Електронний ресурс]. – Режим доступу: https://www.railscarma.com/technology-stack/ – 05.05.2021р.

13. What if I Tell You That Ruby on Rails Is Scalable [Електронний ресурс]. – Режим доступу: https://rubygarage.org/blog/ruby-on-rails-is-scalable – 29.04.2021р.

14. State Of Ruby On Rails Web Development At The Beginning Of 2021 [Електронний ресурс]. – Режим доступу: https://www.ideamotive.co/blog/state-of-ruby-on-rails-web-development – 11.05.2021р.

ДОДАТОК А

Лістинг модуля db/schema.rb

```
ActiveRecord::Schema.define(version: 2020_09_23_103845) do

  # These are extensions that must be enabled in order to support
this database
  enable_extension "plpgsql"

  create_table "action_text_rich_texts", force: :cascade do |t|
    t.string "name", null: false
    t.text "body"
    t.string "record_type", null: false
    t.bigint "record_id", null: false
    t.datetime "created_at", precision: 6, null: false
    t.datetime "updated_at", precision: 6, null: false
    t.index ["record_type", "record_id", "name"], name:
"index_action_text_rich_texts_uniqueness", unique: true
  end

  create_table "active_storage_attachments", force: :cascade do |t|
    t.string "name", null: false
    t.string "record_type", null: false
    t.bigint "record_id", null: false
    t.bigint "blob_id", null: false
    t.datetime "created_at", null: false
    t.index ["blob_id"], name:
"index_active_storage_attachments_on_blob_id"
    t.index ["record_type", "record_id", "name", "blob_id"], name:
"index_active_storage_attachments_uniqueness", unique: true
  end

  create_table "active_storage_blobs", force: :cascade do |t|
    t.string "key", null: false
    t.string "filename", null: false
```

```
    t.string "content_type"
    t.text "metadata"
    t.bigint "byte_size", null: false
    t.string "checksum", null: false
    t.datetime "created_at", null: false
    t.index ["key"], name: "index_active_storage_blobs_on_key",
unique: true
  end

  create_table "contacts", force: :cascade do |t|
    t.string "name"
    t.string "email"
    t.string "phone"
    t.string "city"
    t.text "message"
    t.datetime "created_at", precision: 6, null: false
    t.datetime "updated_at", precision: 6, null: false
  end

  create_table "posts", force: :cascade do |t|
    t.datetime "created_at", precision: 6, null: false
    t.datetime "updated_at", precision: 6, null: false
    t.string "title", null: false
    t.datetime "published_at"
  end

  create_table "users", force: :cascade do |t|
    t.string "email", default: "", null: false
    t.string "encrypted_password", default: "", null: false
    t.string "reset_password_token"
    t.datetime "reset_password_sent_at"
    t.datetime "remember_created_at"
    t.datetime "created_at", precision: 6, null: false
    t.datetime "updated_at", precision: 6, null: false
```

```
          t.index ["email"], name: "index_users_on_email", unique: true
          t.index ["reset_password_token"], name:
"index_users_on_reset_password_token", unique: true
        end

        add_foreign_key "active_storage_attachments",
"active_storage_blobs", column: "blob_id"
      end
```

ДОДАТОК Б

Лістинг модуля app/views/home/index.slim

```
.top#page-top
section.button-lang
  btn.btn-primary.btn-lg = link_to 'UA', { :locale=>'uk' }
  btn.btn-secondary.btn-lg = link_to 'RU', { :locale=>'ru' }

.contact-number
  h6.mb-1 = I18n.t 'home.our_phone_number'
  h5.mb-1
    a href="tel:+380953076554" +38 (095) 30-76-554

  = link_to "Англійська", { :locale=>'en' }

a.menu-toggle.rounded href="#"
  i.fa.fa-bars
nav#sidebar-wrapper
  ul.sidebar-nav
    li.sidebar-brand
      a.js-scroll-trigger href="#page-top"
    li.sidebar-nav-item
      a.js-scroll-trigger href="#page-top" = I18n.t ('home.main')
    li.sidebar-nav-item
      a.js-scroll-trigger href="#about" = I18n.t ('home.about')
    li.sidebar-nav-item
      a.js-scroll-trigger href="#services" = I18n.t ('home.servis')
    li.sidebar-nav-item
      a.js-scroll-trigger href="#portfolio" = I18n.t
('home.attendance')
      li.sidebar-nav-item
        a.js-scroll-trigger href="#contact" = I18n.t ('home.contacts')
      li.sidebar-nav-item
        = link_to t(:blog), posts_path
        = link_to t(:avto), new_user_session_path
```

```
    header.masthead.d-flex
      .container.text-center.my-auto
        h1.mb-1 = I18n.t 'home.your_website'
        h3.mb-5
          em = I18n.t 'home.face_of_your_business'
        a.btn.btn-primary.btn-xl.js-scroll-trigger href="#about" = I18n.t
'home.learn_more'
      .overlay

    section.content-section.bg-light#about
      .container.text-center
        .row
          .col-lg-10.mx-auto
            h2 = I18n.t 'home.clear'
            p.lead.mb-3
              br = I18n.t 'home.be_happy'
              = I18n.t 'home.simply'
              |  
              a href="mailto:createsiteukraine@gmail.com?subject=Лист із
сайта www.siteukraine.com" = I18n.t 'home.write'
              |  
              = I18n.t 'home.or'
              |  
              a href='tel:+380953076554' = I18n.t ('home.call_us')
            a.btn.btn-dark.btn-xl.js-scroll-trigger href="#services" =
I18n.t 'home.what_we_offer'

    section.callout
      .container.text-center
        h3.mx-auto.mb-4 = I18n.t 'home.go_to_our_blog'
        btn.btn-primary.btn-xl.btn-blog
          = link_to t(:blog), posts_path
```

```
section.content-section.bg-primary.text-white.text-center#services
  .container
    .content-section-heading
      .text-secondary.mb-0 = I18n.t 'home.servis'
      h2.mb-5 = I18n.t 'home.what_we_offer'
    .row
      .col-lg-3.col-md-6.mb-5.mb-lg-0
        span.service-icon.rounded-circle.mx-auto.mb-3
          i.icon-pencil
        h4 = I18n.t 'home.accessibility'
        p.text-faded.mb-0 = I18n.t 'home.no_burdensome'
      .col-lg-3.col-md-6.mb-5.mb-md-0
        span.service-icon.rounded-circle.mx-auto.mb-3
          i.icon-magnifier
        h4 = I18n.t 'home.loyalty'
        p.text-faded.mb-0 = I18n.t 'home.personal'
      .col-lg-3.col-md-6.mb-5.mb-lg-0
        span.service-icon.rounded-circle.mx-auto.mb-3
          i.icon-screen-smartphone
        h4 = I18n.t 'home.adaptability'
        p.text-faded.mb-0 = I18n.t 'home.mobile_devices'
      .col-lg-3.col-md-6.mb-5.mb-lg-0
        span.service-icon.rounded-circle.mx-auto.mb-3
          i.icon-notebook
        h4 = I18n.t 'home.communication'
        p.text-faded.mb-0 = I18n.t 'home.member_of_your_contacts'

section.content-section#portfolio
  .container
    .content-section-heading.text-center
      h3.text-secondary.mb-0 = I18n.t 'home.servis'
      h2.mb-5 = I18n.t 'home.who_are_we_useful_to'
    .row.no-gutters
      .col-lg-6
```

```
            a.portfolio-item href="#"
              span.caption
                span.caption-content
                  h2 = I18n.t 'home.business'
                  p.mb-0 = I18n.t 'home.customer'
              img.img-fluid
src="#{asset_pack_path('media/images/portfolio-1.jpg')}" alt=""
          .col-lg-6
            a.portfolio-item href="#"
              span.caption
                span.caption-content
                  h2 = I18n.t 'home.hairdressers'
                  p.mb-0 = I18n.t 'home.your_portfolio'
              img.img-fluid
src="#{asset_pack_path('media/images/portfolio-2.jpg')}" alt=""
          .col-lg-6
            a.portfolio-item href="#"
              span.caption
                span.caption-content
                  h2 = I18n.t 'home.coffee_houses'
                  p.mb-0 = I18n.t 'home.photos_of_your_products'
              img.img-fluid
src="#{asset_pack_path('media/images/portfolio-3.jpg')}" alt=""
          .col-lg-6
            a.portfolio-item href="#"
              span.caption
                span.caption-content
                  h2 = I18n.t 'home.masters'
                  p.mb-0 = I18n.t 'home.at_any_time'
              img.img-fluid
src="#{asset_pack_path('media/images/portfolio-4.jpg')}" alt=""

    section#contact.map
```

```
      iframe width="100%" height="100%" frameborder="0" scrolling="no"
marginheight="0" marginwidth="0"
src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d1257.89315641105
42!2d34.795737071874235!3d50.90916991856797!2m3!1f0!2f0!3f0!3m2!1i1024!2i76
8!4f13.1!3m3!1m2!1s0x4129018995d952b9%3A0x0!2zNTDCsDU0JzMzLjAiTiAzNMKwNDcnN
DguMCJF!5e0!3m2!1suk!2sua!4v1581949578697!5m2!1suk!2sua"></iframe>


    footer.footer
      .container
        .content-section-heading.text-center
          h3.text-secondary.mb-0 = I18n.t 'home.contacts'
          h2.mb-5 = I18n.t 'home.how_to_find_us'
        .row
          .col-sm-12
            .form-group.text-center
              p
                span.icon-map
                  |  
                  = I18n.t ('home.address')
              p
                span.icon-call-in
                |  
                a href="tel:+380953076554"
                  | +38 (095) 30-76-554
              p
                span.icon-envelope
                |  
                a href="mailto:createsiteukraine@gmail.com?subject=Лист
із сайта www.siteukraine.com" createsiteukraine@gmail.com
          .col-sm-12
            #contact-form.h2.mb-5.text-center = I18n.t
'home.your_details'
            = render 'contacts/form'
```

```
        container.text-center
          ul.list-inline.mb-5
            li.list-inline-item
              a.social-link.rounded-circle.text-white.mr-3
href="https://www.facebook.com/siteukraine/" target="_blank"
                i.icon-social-facebook
            li.list-inline-item
              a.social-link.rounded-circle.text-white.mr-3
href="https://www.instagram.com/siteukraine/" target="_blank"
                i.icon-social-instagram
            li.list-inline-item
              a.social-link.rounded-circle.text-white.mr-3 href="/posts"
target="_blank"
                i.icon-note
```

ДОДАТОК В

Лістинг модуля app/views/contact/_form.slim

```
.container
  .row
    .col-lg-8.col-md-10.mx-auto.form-group
      - if notice
        p.alert.alert-danger = notice

      = form_for @contact, method: :post, html: { class: 'form-
horizontal' } do |f|
        - if @contact.errors.any?
          ul.error_explanation
            - @contact.errors.full_messages.each do |message|
              li = message
          br

        p *
        = I18n.t ('contacts.required_fields')

        .form-group
          = f.label :name
          = f.text_field :name, placeholder: true

        .form-group
          = f.label :email
          = f.text_field :email, placeholder: true

        .form-group
          = f.label :phone
          = f.text_field :phone, placeholder: true

        .form-group
```

```
  = f.label :city
  = f.text_field :city, placeholder: true

.form-group
  = f.label :message
  = f.text_area :message, placeholder: true

.actions
  = f.submit class: "btn btn-primary"
```

ДОДАТОК Г

Лістинг модуля Gemfile

```
source 'https://rubygems.org'
git_source(:github) { |repo| "https://github.com/#{repo}.git" }

ruby '2.6.3'

# Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
gem 'rails', '~> 6.0.3', '>= 6.0.3.2'
# Use postgresql as the database for Active Record
gem 'pg', '>= 0.18', '< 2.0'
# Use Puma as the app server
gem 'puma', '~> 4.1'

# For running Rails server and webpacker at the same time
gem 'foreman'

# Use SCSS for stylesheets
gem 'sass-rails', '>= 6'
# Transpile app-like JavaScript. Read more:
https://github.com/rails/webpacker
gem 'webpacker', '~> 4.0'

# HTML templating
gem 'slim-rails'

# App secrets
gem 'figaro'

# Authentication
gem 'devise'

# Authorization
gem 'pundit'
```

```
    # Use Active Storage variant
    gem 'image_processing', '~> 1.2'

    # Reduces boot times through caching; required in config/boot.rb
    gem 'bootsnap', '>= 1.4.2', require: false

    group :development do
      # Access an interactive console on exception pages or by calling
'console' anywhere in the code.
      gem 'web-console', '>= 3.3.0'
      gem 'listen', '~> 3.2'
      # Spring speeds up development by keeping your application running
in the background. Read more: https://github.com/rails/spring
      gem 'spring'
      gem 'spring-watcher-listen', '~> 2.0.0'

      # Open letters in browser on development
      gem 'letter_opener'

      # Generate PDF with database schema when running rake db:migrate
      gem 'rails-erd'
    end
```

ДОДАТОК Д

Лістинг файлу app/controllers/ post_controller.rb

```
class PostsController < ApplicationController
  before_action :authenticate_user!, except: [:index, :show]
  before_action :set_post, only: [:show, :edit, :update, :destroy,
:publish, :unpublish]

  # GET /posts
  def index
    authorize Post
    @posts = policy_scope(Post)
  end

  # GET /posts/1
  def show
    @post = Post.find(params[:id])
    authorize @post
  end

  # GET /posts/new
  def new
    @post = Post.new
    authorize @post
  end

  # GET /posts/1/edit
  def edit
    authorize @post
  end

  # POST /posts
  def create
    @post = Post.new(post_params)
```

```
      authorize @post

      if @post.save
        redirect_to @post, notice: 'Post was successfully created.'
      else
        render :new
      end
    end

    # PATCH/PUT /posts/1
    def update
      authorize @post
      if @post.update(post_params)
        redirect_to @post, notice: 'Post was successfully updated.'
      else
        render :edit
      end
    end

    # DELETE /posts/1
    def destroy
      authorize @post
      @post.destroy
      redirect_to posts_url, notice: 'Post was successfully destroyed.'
    end

    # PUT /posts/1/publish
    def publish
      authorize @post
      if @post.update(published_at: DateTime.current)
        redirect_to posts_path, notice: 'Post was successfully
published.'
      else
```

```
            redirect_to posts_path, alert: 'Post was not published due to
validation error.'
          end
        end

        # PUT /posts/1/unpublish
        def unpublish
          authorize @post
          if @post.update(published_at: nil)
            redirect_to posts_path, notice: 'Post was successfully removed
from published.'
          else
            redirect_to posts_path, alert: 'Post was not removed from
published due to validation error.'
          end
        end

        private
          # Use callbacks to share common setup or constraints between
actions.
          def set_post
            @post = Post.find(params[:id])
          end

          # Only allow a list of trusted parameters through.
          def post_params
            params.fetch(:post, {}).permit(:content, :preview, :title)
          end
      end
```

Лістинг файлу app/controllers/application_controller.rb

```
class ApplicationController < ActionController::Base
  include Pundit

  rescue_from Pundit::NotAuthorizedError, with: :user_not_authorized

  around_action :switch_locale

  def switch_locale(&action)
    locale = params[:locale] || I18n.default_locale
    I18n.with_locale(locale, &action)
  end

  def default_url_options
    { locale: I18n.locale }
  end

  private
    def user_not_authorized
      flash[:alert] = 'You are not authorized to perform this
action.'
      redirect_to(request.referrer || root_path)
    end
end
```