КВАЛІФІКАЦІЙНА РОБОТА

Група ІПЗс-20

Огородник С. І.

2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук
Кафедра інформаційних технологій

на правах рукопису

Огородник Сергій Ігорович

УДК 004.42

Розробка веб-платформи для обслуговування в сфері відеомонтажу

Спеціальність 121 – «Інженерія програмного забезпечення»
Кваліфікаційна робота на здобуття кваліфікації бакалавра

Нормоконтроль	Студент
_____Стисло О.В.	_____Огородник С.І.
(підпис, дата, розшифрування підпису)	(підпис, дата, розшифрування підпису)
Допускається до захисту Завідувач кафедри	Керівник роботи
_____к.т.н., доц. Ващишак С.П.	_____к.ф-м.н., доц. Бойчук А.М.
(підпис, дата, розшифрування підпису)	(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«____» _____ 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Огороднику Сергію Ігоровичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:

Розробка веб-платформи для обслуговування в сфері відеомонтажу

керівник роботи:

Бойчук Андрій Михайлович, кандидат фізико-математичних наук

затверджена наказом вищого навчального закладу від « 12 » березня 2024 року № 19/1

2. Термін подання студентом роботи 05.06.2024

3. Вихідні дані роботи: мова програмування Ruby, HTML, CSS, JS

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

   1. Опис та порівняння існуючих аналогів
   2. Проектування архітектури та план розробки сайту
   3. Програмна реалізація веб-платформи

5. Дата видачі завдання 14.03.2024

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Дослідження існуючих аналогів	25.03.2024	Виконано
2.	Проектування архітектури та створення дизайну веб-сайту	18.04.2024	Виконано
3.	Програмна реалізація веб-сайту	07.05.2024	Виконано
4.	Формування висновків	13.05.2024	Виконано
5.	Оформлення графічного матеріалу та підготовка до захисту роботи	22.05.2024	Виконано

Студент _____ Огородник С.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Бойчук А.М.
(підпис) (прізвище та ініціали)

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

25	Tailwind CSS	40	Сторінка замовлення “S.VE”
26	Будова User Story	41	Вкладка “Ваші тікети” “S.VE”
42	Тікет “S.VE”		

АНОТАЦІЯ

Кваліфікаційна робота присвячена створенню веб-сайту по обслуговуванню відеомонтажу, із можливістю замовляти тікети, шляхом використання Ruby on Rails.

В першому розділі проведено загальний опис вимог, і аналіз існуючих конкурентів. Виділивши їхні переваги та недоліки, порівнюючи їх наповнення та інструментів взаємодії з користувачем.

В другому розділі проведено аналіз обраної платформи, методів, та мови програмування, з написанням User Stories можна краще зрозуміти всю ідею та структуру сайту. Розроблено макет архітектури бази даних, та проектування інтерфейсу веб-сайту.

В третьому розділі описано процес реалізації веб-сайту, розглядаючи розробку основним частин Ruby on Rails, та технологій які він використовує.

КЛЮЧОВІ СЛОВА: ПОСЛУГИ ВІДЕОМОНТАЖУ, RUBY, RUBY ON RAILS, POSTGRESQL.

SUMMARY

The qualification work is devoted to a video editing service website, with the possibility of ordering tickets, by using Ruby on Rails.

In the first section, a general description of the requirements and an analysis of existing competitors are provided. Highlighting their advantages and disadvantages, comparing their content and user interaction tools.

In the second section, an analysis of the chosen platform, methods, and programming language is carried out, with the writing of User Stories you can better understand the whole idea and structure of the site. The layout of the database architecture and the design of the website interface were developed.

The third chapter describes the website implementation process, considering the development of the main parts of Ruby on Rails and the technologies it uses.

KEYWORDS: VIDEO EDITING SERVICES, RUBY, RUBY ON RAILS, POSTGRESQL.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Стрімінгові сервіси – це онлайн-платформи, які надають користувачам можливість перегляду аудіо та відео контенту без необхідності завантаження файлів на свій пристрій.

БД – бази даних

Продакшн – Це виробничий процес, який включає в себе створення товарів або послуг. Зазвичай відноситься до всього процесу створення відеоконтенту, включаючи планування, зйомки, постпродакшн і випуск.

RoR – Ruby on Rails

Грід (сітка) – UI/UX елемент графічного інтерфейсу користувача, що забезпечує табличний вигляд даних.

UI – user interface

Хедер (header) – це елемент веб-сторінки, що знаходиться вище області контенту. Якщо описати веб-сторінку «архітектурними» термінами, то футер – це підвал сайту, область контенту – стіни і вікна, а хедер – дах.

MVC – Model–view–controller

Якір, або «Якірні посилання» – це спосіб навігації по сайту, який дозволяє швидко переміститися з одного місця сторінки на інший.

СКБД – Система управління базами даних

Discord – це платформа обміну миттєвими повідомленнями та цифрового розповсюдження інформації з функціями VoIP. Користувачі спілкуються за допомогою голосових дзвінків, відеочатів, текстових повідомлень, медіа та файлів у приватних чатах або в рамках спільнот, які називаються «серверами».

ВСТУП

Актуальність теми. У сучасному цифровому світі відео перетворилося на один із найпопулярніших та найвпливовіших засобів комунікації. Це явище можна спостерігати на різних платформах, від соціальних мереж, таких як Facebook, Instagram, та TikTok, до стрімінгових сервісів YouTube, Netflix і Amazon Prime. Зростання популярності відеоконтенту створює величезний попит на різноманітні види відео, включаючи розважальні, освітні, маркетингові та інформаційні матеріали.

В умовах такого підвищеного інтересу до відео, якість самого контенту набуває надзвичайної важливості. Високоякісне відео привертає увагу, утримує глядачів та сприяє досягненню цілей, будь то залучення аудиторії, підвищення продажів чи поширення інформації. Якість відеоматеріалу багато в чому визначається професійним підходом до його створення та монтажу. Професійний монтаж відео включає в себе не тільки технічні аспекти, такі як правильний вибір кадрів, звуковий супровід та ефекти, але й художню складову – здатність передати настрій, емоції та сенс відео.

Успішний відеомонтаж передбачає не тільки технічні знання і вміння, а й високий рівень креативності. Це включає в себе здатність вибирати музику, що підкреслює емоційну складову відео, професійно обробляти звук, коригувати і збалансовувати колірну гаму для досягнення естетичної гармонії, а також майстерно розповідати історію через образи та кадри.

У роботі відеомонтажу існують різні галузі, де використовується цей процес для досягнення конкретних цілей.

Деякі з них включають:

1. Кіноіндустрія: відеомонтаж є ключовим етапом у виробництві фільмів, документальних стрічок, серіалів та короткометражок. Монтажний редактор відповідає за створення повністю розробленого сюжету, обробку сцен та вибір музики для створення потрібного настрою.

2. Реклама та маркетинг: у світі реклами відеомонтаж використовується для створення рекламних роликів, промо-відео та коротких рекламних кліпів. Важливою задачею є здатність привернути увагу аудиторії та ефективно комунікувати повідомлення бренду.

3. Медіа та телебачення: в телевізійній індустрії відеомонтаж використовується для збірки новинних сюжетів, розважальних програм, телевізійних шоу та спортивних подій.

4. Веб-контент та соціальні мережі: відеомонтаж стає невід'ємною частиною вмісту для YouTube, Instagram, TikTok та інших платформ. Від естетично виконаних відео для особистих блогів до професійних відеороликів для бізнесу – монтаж допомагає залучити аудиторію та збільшити вплив.

5. Освіта: відеомонтаж використовується в освітніх цілях для створення навчальних відеоматеріалів, онлайн-курсів, відеоуроків тощо.

6. Події та виставки: відеомонтаж може бути використаний для створення відеорепортажів та виставок, фестивалів і інших подій.

У більшості по статистиці для таких завдань замовляють талановитих спеціалістів – фрілансери відеомонтажу. Вони пропонують свої послуги як професіонали, готові втілити в життя ідеї клієнтів, так і талановиті аматори, які шукають можливості для самовираження та розвитку.

Бути фрілансером відеомонтажу - це отримати унікальну можливість працювати над різноманітними проектами від різних клієнтів, не прив'язуючись до конкретного робочого місця чи жорсткого офісного графіку. Такий формат роботи надає відчуття свободи та гнучкості, дозволяючи самостійно обирати проекти, які вас цікавлять, планувати свій робочий час та місце роботи за власним розсудом.

Фрілансер відеомонтажу може облаштувати свій робочий простір вдома, у затишній кав'ярні, чи навіть працювати під час подорожей, за умови наявності доступу до необхідного обладнання, програмного забезпечення та Інтернету. Ця мобільність дає змогу поєднувати роботу із особистими вподобаннями та

стилем життя, що може значно підвищити продуктивність, ефективність та задоволення від роботи.

Крім того, фрілансер має можливість працювати з клієнтами з усього світу, розширюючи свою професійну мережу та отримуючи досвід у різних культурних та професійних контекстах. Це дозволяє постійно вдосконалювати свої навички, бути в курсі нових тенденцій та технологій у сфері відеомонтажу, а також відкриває нові перспективи для розвитку кар'єри.

Фріланс дає можливість встановлювати власні тарифи, вибирати проекти, що відповідають вашим інтересам і професійним цілям, а також самостійно керувати своїм робочим навантаженням та особистим графіком. Такий підхід дозволяє досягти оптимального балансу між роботою та особистим життям, роблячи роботу не лише засобом заробітку, а й джерелом творчого задоволення і професійного росту.

Мета і завдання дослідження. Розробка веб-застосунку для обслуговування в сфері відеомонтажу.

Об’єкт роботи – процес покращення послуг обслуговування для клієнтів.

Предмет роботи – Платформа для збереження портфоліо та список замовлень клієнтів на відеомонтаж.

Практичне значення отриманих результатів. Результатом виконання кваліфікаційної роботи є створений на Ruby on Rails веб-сайт, призначенням якого є демонстрація готових робіт відеомонтажера, та можливість обслуговування клієнтів у єдиному зручному середовищі.

Апробація результатів дослідження. Матеріали кваліфікаційної роботи були представлені на XI Міжнародній науковій конференції «Студентські наукові дискусії поза форматом», яка відбулася 11 квітня 2024 року в Університеті Короля Данила.

Структура роботи. Розділи – 3. Обсяг основної частини – 48 сторінок. Список використаних джерел містить – 20 позицій.

РОЗДІЛ 1. ОПИС ТА ПОРІВНЯННЯ ІСНУЮЧИХ АНАЛОГІВ

1.1 Загальний опис вимог до веб-сайту обслуговування

У сучасному світі Інтернет став невід'ємною частиною нашого повсякденного життя. Суспільство активно користується ним для отримання інформації, розваг, а також для доступу до різноманітних послуг, серед яких особливо виділяється обробка медіа та відеомонтаж. Відеомонтаж є ключовим етапом у процесі створення відеоконтенту, включаючи такі важливі аспекти, як обрізка, з'єднання, редагування та додавання ефектів для створення якісного кінцевого продукту.

Відеомонтаж дозволяє перетворити сирий матеріал у завершений, професійний продукт, який може захопити глядача, передати задум режисера або автора, і створити емоційний резонанс. Завдяки відеомонтажу, автори можуть підкреслити важливі моменти, створити динамічний ритм, покращити візуальну та звукову якість матеріалу.

Технологічний розвиток значно вплинув на сферу відеомонтажу, зробивши її більш доступною та ефективною. Сучасні технології також спростили процес співпраці над відеопроектами. Завдяки хмарним сервісам, таким як Adobe Creative Cloud, фрілансери та команди можуть працювати над проектами спільно, незалежно від географічного розташування. Це значно покращує ефективність роботи, скорочує час на обмін файлами та дозволяє оперативно вносити правки.

Весь процес відеомонтажу поділяється на кілька етапів:

– завантаження відеофайлів з камер, смартфонів чи інших пристроїв на комп’ютер;

– обрізка і з'єднання відеокліпів, додавання переходів, текстів, графіки та ефектів;

– поліпшення і корекція кольору для досягнення бажаного візуального стилю;

– редагування та синхронізація звукової доріжки, додавання музики та звукових ефектів;

– створення кінцевого відеофайлу у необхідному розмірі та форматі для розповсюдження.

Все це – основні вимоги, щоб створити відео та аудіо контент. Цей процес потребує часу як на вивчення напряму по відеомонтажу, так і на самий проект. Тому у сучасному світі існують різноманітні сервіси, які готові за плату відтворити в реальність любий медіаконтент, який забажає сам замовник.

Мета кваліфікаційної роботи – створити середовище(веб-сайт), у якому можна переглянути портфоліо фахівця, зареєструвати свій аккаунт та мати можливість замовити послуги. Загальним призначенням веб-сайту є забезпечення зручного та ефективного обслуговування для клієнтів.

Веб-сайт обслуговування відеомонтажу повинен мати ряд ключових характеристик, щоб забезпечити зручність користування та ефективне представлення послуг.

Загальний опис вимог до такого веб-сайту:

– привабливий дизайн: важливо, щоб дизайн сайту був привабливим і професійним. Це повинно відображати не лише відповідають найвищим стандартам якості, але й втілюють справжню естетичну красу.

– простота навігації: користувачі мають легко знаходити інформацію про послуги, портфоліо, ціни, контактні дані та інші важливі розділи. Чітка та логічна навігація допомагає в цьому. Інтерфейс не повинен бути перевантажений зайвими модулями та текстом. Він має бути простим для користувачів, приємним для очей та інтуїтивно зрозумілим у використанні.

– портфоліо робіт: веб-сайт повинен містити розділ з портфоліо, де клієнти можуть переглянути приклади відеомонтажу, щоб оцінити рівень майстерності та стиль роботи компанії/фахівця.

– форма зворотного зв'язку: наявність форми зворотного зв'язку для замовлення консультації або запиту додаткової інформації. Стосовно цього на сайті у футері будуть посилання на мої соцмережі.

– мультимедійні можливості: можливість відтворення відео на веб-сайті, які демонструють роботу компанії, з використанням відеоплеєра.

– SEO-оптимізація: важливо оптимізувати веб-сайт для пошукових систем, щоб забезпечити його високий рейтинг у результатах пошуку.

Виконання всіх наведених вимог є ключовим чинником для створення успішного середовища для надання послуг відеоконтенту. Вебсайт повинен викликати довіру та забезпечувати зручний доступ до замовлень і чатів з клієнтами. Кожен користувач може створити власний обліковий запис, налаштувати його згідно власних уподобань і створити список своїх замовлень з вказівками для фахівця щодо обробки матеріалу.

Крім того, кожен виконаний проект може бути оцінений замовником, а також може бути надане посилання на платформу, де буде опублікований оброблений відеоролик. Дизайн веб-сайту повинен бути простим, легким у експлуатації та приємним для очей. Для досягнення цієї мети будуть використані темні тони, які нададуть сайту сучасний вигляд і зроблять його приємним для відвідувачів (рис. 1.1).

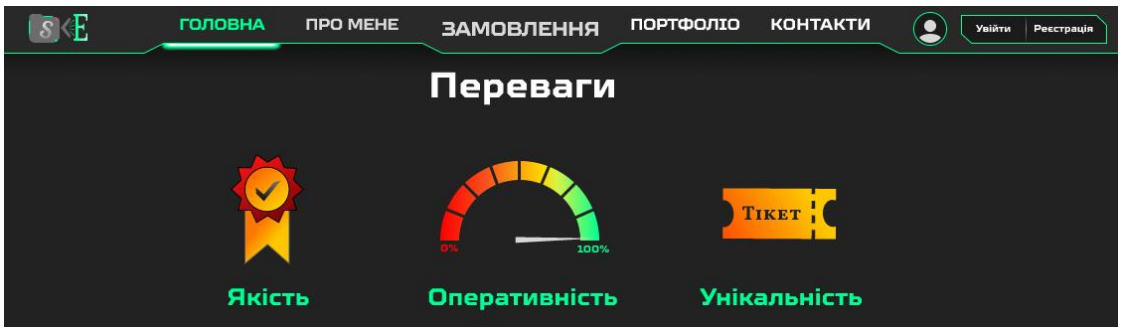


Рисунок 1.1 – Інтерфейс власного веб-сайту “S.VE”

1.2 Огляд існуючих сервісів відеомонтажу

Станом на зараз є безліч варіантів замовити послугу по відео монтажу:

– соціальні мережі;
– електронна пошта;
– веб-сайти для фрілансерів;
– приватні веб-сайти фахівця/команди.

Для початку розглянемо перший аналог моєї роботи, професійний веб-сайт команди монтажерів “lumiereproduction.tv” (рис.1.2).

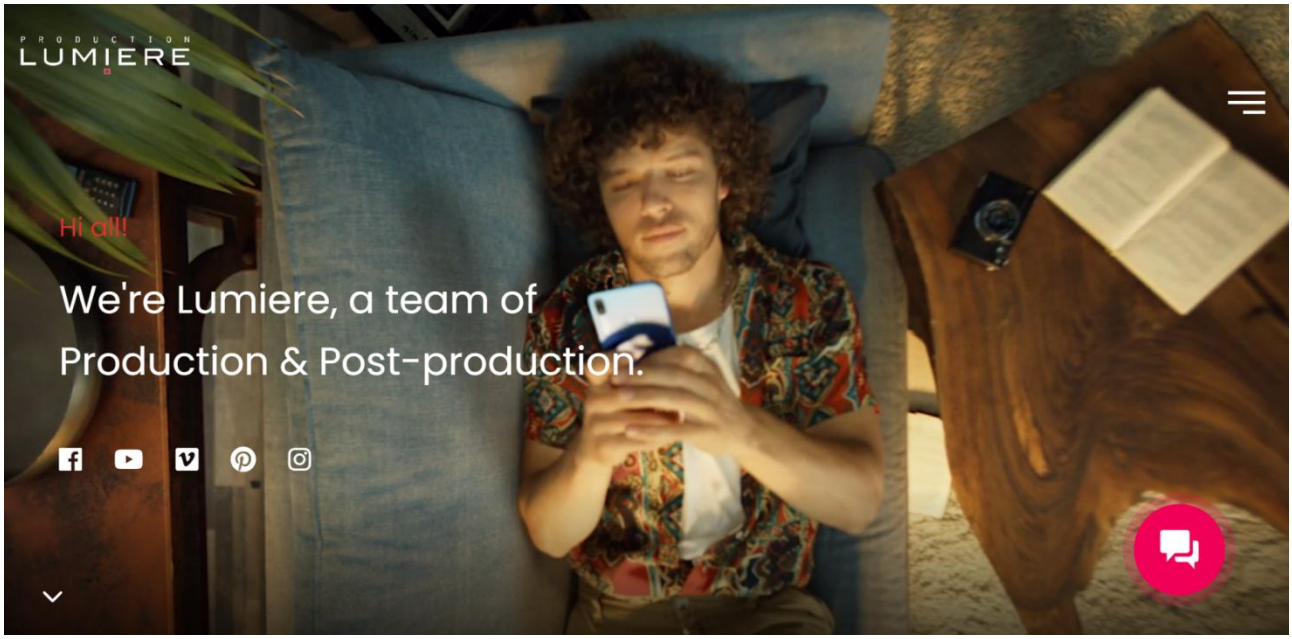


Рисунок 1.2 – Веб-сайт “Lumiere Production”

Lumiere Production [1] – це кіностудія або продюсерська компанія, яка займається створенням, фінансуванням та продюсуванням фільмів, телесеріалів або інших медійних проектів. Lumiere Production займаються такими основними видами діяльності:

1. Розробка ідеї: робота над створенням концепту фільму або серіалу, написання сценарію та підготовка проекту до зйомок.

2. Фінансування: Залучення фінансових ресурсів для виробництва проекту.

3. Виробництво: організація та управління зйомками, включаючи підбір акторів, знімальної групи та всього необхідного обладнання.

4. Постпродакшн: редагування, монтаж, додавання спеціальних ефектів, звуковий супровід та інші процеси, які завершують створення фільму або серіалу.

5. Маркетинг та розповсюдження: просування готового проекту і забезпечення його показу в кінотеатрах, на телебаченні або в онлайн-стрімінгових сервісах.

Одним з найпоширеніших недоліків стосовно сайтів відео обслуговування – це не інтуїтивно-зрозуміле орієнтування на платформі.

Багато хто старається виділитись від всіх конкурентів, і деколи реалізовує настільки складну в розумінні структуру сайту, що для клієнта потрібен час для розуміння де він знаходиться.

Не завжди з першого разу клієнт зрозуміє де замовити сервіс, де знаходяться відгуки до портфоліо і скільки часу зайняло на проект. Серед недоліків є відсутність вимкнення аудіо у відео-заставці на фоні, що змушує користувача шукати спосіб вимкнення звуку самої вкладки, або найгірше – остаточний вихід з сайту.

Ще один недолік – велике перенасичення ефектів та переходів, що буквально змушує сайт довго загружати сторінку, і це може стати проблемою якщо клієнт має слабкий ПК. Серед цікавих особливостей хочеться виділити сторінку з портфоліом під головною сторінкою, яка представляє собою довжелезну стрічку з готовими проектами та відеороликами (рис 1.3).

Рисунок 1.3 – Портфоліо “Lumiere Production”

Реалізація перегляду портфоліо цікава та лаконічна, але в ній, на мою думку, містить за мало інформації про продукт.

Хочеться бачити справжній відгук про роботу від замовника, та скільки часу було витрачено на весь проект.

Ще однією перевагою цього сайту це використання відео-хостингу “Vimeo” як сховище портфоліо, так і демонстрація самої роботи за допомогою плеєра на сайті.

Vimeo (рис.1.4) — це платформа для спільного перегляду, розміщення та відкриття відео. Вона дає змогу будь-якому професіоналу, команді й організації розкрити силу відео для створення, співпраці та спілкування [2].

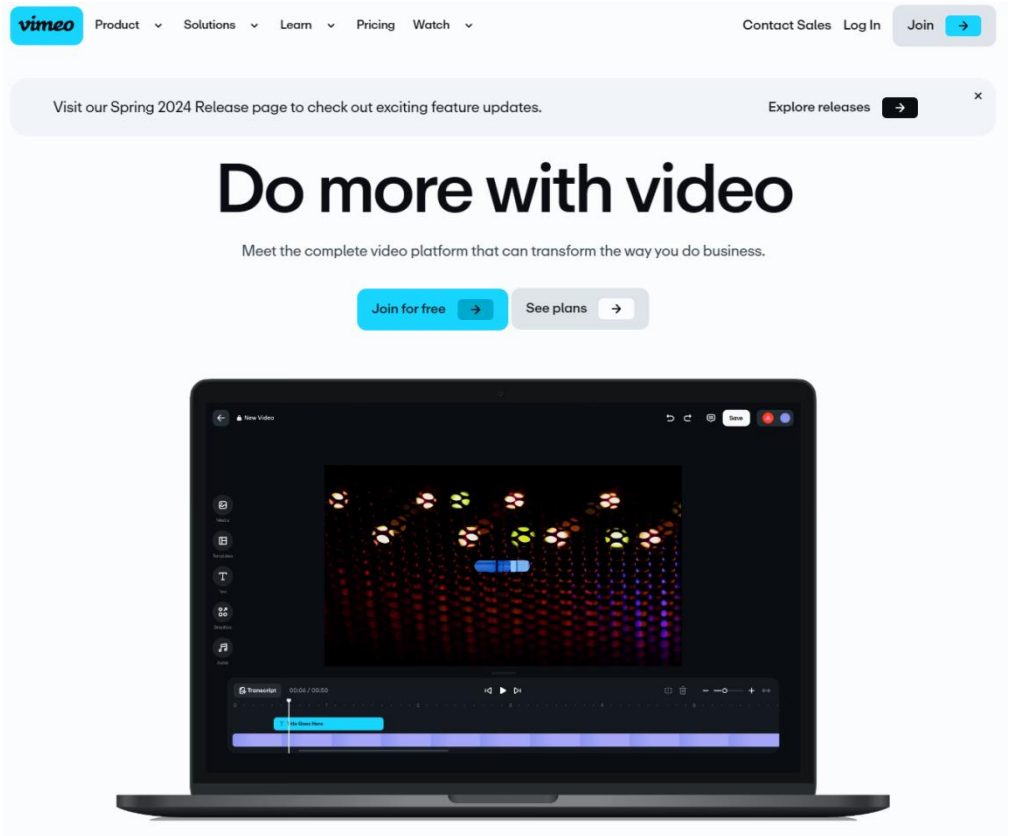


Рисунок 1.4 – Сайт “Vimeo”

Перейдемо до найважливішого недоліку (рис.1.5) – відсутня реєстрація аккаунту. Контакт замовника з фахівцями реалізована звичайною припискою

ел.пошти та ім’ям, тобто надалі увесь діалог буде відбуватись банально на електронній скриньці що насправді зовсім не професійно.

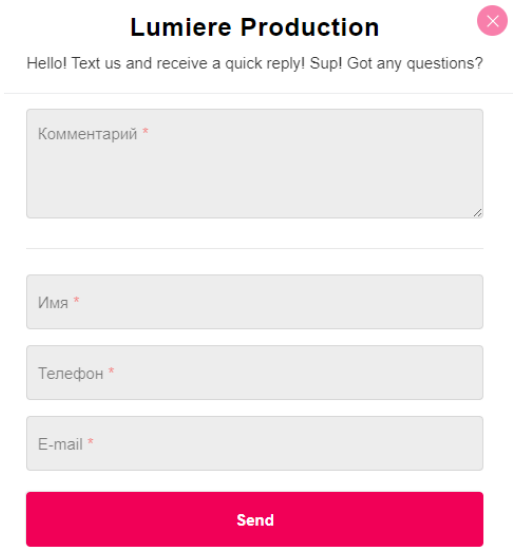


Рисунок 1.5 – Зв'язок з замовником

Цей недолік критичний тим, що потрібно занадто багато маніпуляцій, щоб сконтактуватись з замовником та демонструвати йому прототип продукту. Аккаунт необхідно мати на сайті, задля зберігання усіх ваших замовлень та матеріалів до них.

Наступний конкурент – молода компанія “Mammoth Digital Media” що обслуговує клієнтів медіа-контентом (рис.1.6).

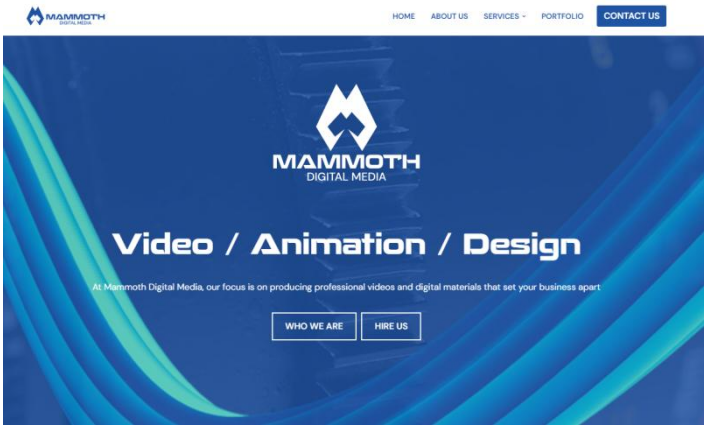


Рисунок 1.6 – Веб-сайт “Mammoth DM”

Mammoth DM (Digital Media) [3] – це компанія, що спеціалізується на створенні цифрового контенту та медіа-проектів. Ось основні напрями діяльності, які можуть охоплювати компанії типу Mammoth DM:

1. Виробництво відеоконтенту: створення відеороликів, корпоративних відеопрезентацій, фільмів, серіалів та інших форм візуального контенту.

2. Постпродакшн: монтаж, редагування, додавання спеціальних ефектів, кольорокорекція, звукорежисура та інші аспекти, пов’язані з завершенням створення відеопродукції.

3. Анімація та візуальні ефекти (VFX): розробка анімаційних елементів і спеціальних ефектів для кіно, рекламних кампаній та ігрової індустрії.

4. Розробка мультимедійних проектів: створення інтерактивних проектів, таких як віртуальна реальність (VR), доповнена реальність (AR), інтерактивні відео та інші інноваційні медійні формати.

5. Дизайн і графіка: розробка графічного дизайну для брендів, включаючи логотипи, банери, постери та інші візуальні матеріали.

6. Маркетинг і просування: створення стратегій для просування контенту через різні цифрові платформи, включаючи соціальні мережі, веб-сайти та інші онлайн-ресурси.

Гарною перевагою цього сайту є його простий та приємний для очей дизайн сайту, на якому лаконічно виставлені модулі тексту та медіа контент, також все вирівнюється за допомогою правильного гріду [4]. Немає нічого лишнього, все коротко та чітко розписано та легко орієнтуватись по веб-сайту. Сторінка з портфоліо демонструє усі роботи команди Mammoth, але всі вони так само без відгуків і більшої конкретики.

Аналізуючи даний сайт я натрапив на деякі неприємні недоліки, а саме:

– хедер [5] сайту не прикріплений до верхньої частини екрану. Тобто якщо гортати вниз – він гортається так само, і доводиться гортати до самого верху, щоб знову користуватись хедером;

– відсутня кнопка “Якір” [6] яка дозволяє вмить прогорнути сторінку до самого верху. Щоб повернутись догори, доводиться вручну гортати комп’ютерним маніпулятором;

– контакт з замовником через телефон/пошту/діскорд [7] – гарний варіант зв’язку з клієнтом лише для консультації, але не для правок у роботі.

Саме з власних досліджень, переписуватись з замовником у соц.мережах під час редагування замовлення по темі різних правок і змін, рано чи пізно приведе до повного безладу. Тим більше потрібно якось демонструвати зміни в проекті без його завантаження на власний комп’ютер.

Тому я пропоную необхідність мати інтегрований окремий чат з видом на прототип до кожного замовлення відразу на сайті.

1.3 Постановка задачі

Згідно вищенаведеного, для якісного та ефективного обслуговування клієнтів, необхідно створити середовище, де замовник зможе з легкістю та комфортом замовляти послуги у нескінченних кількостях, та зручно переглядати статус кожної роботи у окремому таблі. Для цього необхідно вирішити такі задачі:

– провести аналіз переваг та недоліків існуючих сайтів-аналогів;

– вибрати технології з мовою програмування;

– розробити сучасний, ефективний, та зручний дизайн застосунку;

– розробити фронтенд і бекенд веб-сайту з використанням технології Ruby on Rails.

Висновки до розділу 1

В розділі були проведені дослідження та аналіз вже існуючих сайтів-аналогів, за допомогою яких буде створено зручний сайт. Описані загальні можливості та характеристики сайтів інших конкурентів. Описані їхні як

недоліки, так і переваги. В результаті проведених досліджень в подальшій роботі буде використовуватись опис вищеперерахованого з даного розділу, для покращення переваг, та запобігання недоліків під час проектування системи.

РОЗДІЛ 2. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА ПЛАН РОЗРОБКИ САЙТУ

2.1 Вибір мови та технології архітектури

У сфері відео-послуг веб-сайти демонструють різні роботи у своєму портфоліо, містять в собі власну інформацію, та пропонують замовити проект.

Аналізуючи існуючі аналоги веб-сайтів які надають сервіс по монтажу, було помічено, що вони мають певні недоліки, котрі у свою чергу можуть призвести до деяких ускладнень для перегляду сайту зі сторін клієнтів.

Для реалізації сайту “S.VE” сервіс по відеомонтажу була обрана мова Ruby з фреймворком до неї Ruby on Rails (рис.2.1) [8], та з допомогою HTML 5, CSS 3, JS. Основна і найбільша частина була написана саме на Ruby on Rails.

По-перше, Ruby on Rails сприяє продуктивності та швидкості розробки завдяки вбудованим рішенням для стандартних завдань, таких як маршрутизація, робота з базами даних та автентифікація користувачів. Це дозволяє прискорити процес розробки, зменшити обсяг коду і забезпечити високу ефективність роботи команди розробників.

Також Ruby on Rails відрізняється зрозумілим і простим синтаксисом, що робить його легким для читання. Це сприяє швидкому освоєнню фреймворку новими розробниками та полегшує підтримку і розширення сайту [9].

Варто звернути увагу на велике співтовариство розробників Ruby on Rails, яке значно впливає на вибір цієї мови.

Існує безліч доступних бібліотек, модулів та розширень, що сприяють створенню функціональності сайту відеомонтажу. Крім того, активне співтовариство полегшує отримання допомоги, порад та вирішення проблем, які можуть виникнути під час розробки.

Рисунок 2.1 – Логотип фреймворку Ruby on Rails

Також Ruby on Rails має свою гордість – надійна безпека, оскільки він забезпечує вбудовані механізми захисту веб-додатків від SQL-ін'єкцій і міжсайтового скриптингу (XSS).

SQL ін’єкція [19] – це тип атаки на веб-додатки, під час якої зловмисники використовують слабкі місця в обробці користувацького вводу для впровадження та виконання шкідливого SQL коду на сервері бази даних. Вона виникає через недостатню перевірку та фільтрацію введення, і це дає змогу зловмисникам змінювати SQL запити через інтерфейс веб-сторінки та отримувати несанкціонований доступ до даних.

XSS (міжсайтовий скриптинг) [20] – один із різновидів атак на веб-системи, що передбачає впровадження шкідливого коду на певну сторінку сайту та взаємодію цього коду з віддаленим сервером зловмисників при відкритті сторінки користувачем.

Активна підтримка та регулярні оновлення з боку спільноти розробників гарантують безпеку та стабільність для сайту обслуговування.

Крім того, Ruby on Rails має вбудовані засоби для підтримки масштабованості веб-додатків. Він надає зручні інструменти для розширення

функціональності, оптимізації продуктивності та розподілення навантаження, що є особливо важливим для сайту обслуговування з можливою великою кількістю відвідувачів та значним обсягом даних.

Однією з ключових переваг використання Ruby on Rails є його архітектура, зокрема патерн проектування Model-View-Controller (MVC) [10]. Цей підхід розділяє додаток на три основні компоненти, як показано на рисунку 2.2, що покращує організацію коду і спрощує його розуміння та підтримку в майбутньому.

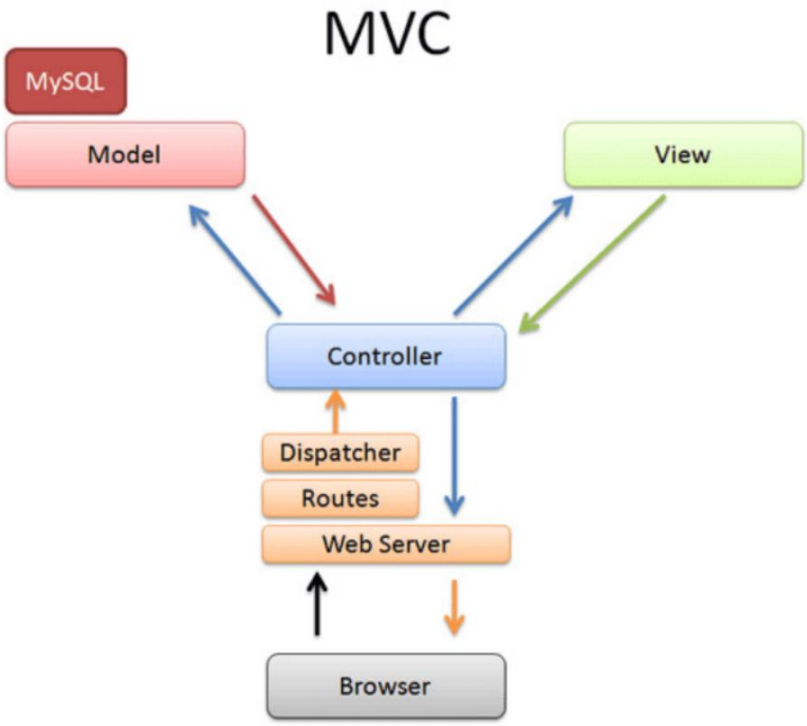


Рисунок 2.2 – Архітектура проекту з використанням MVC

Модель (Model) відповідає за управління даними в додатку. Вона включає логіку доступу до бази даних, виконання операцій зі збереження, оновлення, видалення даних та виконання запитів. Модель відображає структуру і взаємозв'язки даних у додатку, таких як клієнти, тощо. Вона забезпечує безпеку та цілісність даних і може включати бізнес-логіку для підтримки певних правил та процесів.

Вид (View) відповідає за відображення даних користувачеві. Його основне завдання – представити інформацію з моделі у зручному та зрозумілому форматі. Представлення можуть бути реалізовані у вигляді HTML-шаблонів, форм введення або інших компонентів інтерфейсу користувача. Вони відокремлюють відображення даних від бізнес-логіки, що дозволяє змінювати зовнішній вигляд додатку без впливу на роботу моделі та контролера.

Контролер (Controller) відповідає за обробку запитів користувачів і управління потоком даних у системі. Він приймає запити від користувача, виконує необхідну логіку та взаємодіє з моделлю і представленням. Контролер визначає, які дані слід відобразити користувачеві, які дії потрібно виконати у відповідь на запит і які дані слід передати в модель для збереження або оновлення. Він реагує на події, ініційовані користувачем, і координує взаємодію між моделлю і представленням.

Ця архітектура, розділена на компоненти, дозволяє чітко розподілити відповідальність між різними частинами системи. Вона сприяє модульності, розширюваності та повторному використанню коду. Зміни в одному компоненті не впливають на інші частини системи, що полегшує розробку, тестування і підтримку проекту. Крім того, цей підхід дозволяє розробникам працювати паралельно над різними аспектами проекту, що підвищує ефективність команди.

MVC є популярним підходом для веб-розробки на Ruby on Rails завдяки своїй простоті та ефективності. Він забезпечує зручну структуру для розробки та підтримки веб-сайту обслуговування, дозволяючи розбити його на логічні компоненти. Це забезпечує гнучкість і розширюваність системи, що є важливим для динамічних і масштабованих веб-додатків.

Завдяки розподілу відповідальностей між моделлю, видом та контролером, MVC допомагає розробникам створювати чітко структуровані та легкі для підтримки додатки. Така організація коду полегшує внесення змін і розширень у проект, зберігаючи при цьому високу якість і надійність системи.

Для взаємодії з сайтом використовується RESTful API (Representational State Transfer), що дозволяє здійснювати запити через HTTP-протокол, такі як GET, POST, PUT та DELETE, що забезпечує гнучкість та ефективність взаємодії з системою.

У розробці веб-сайту обслуговування на Ruby on Rails для стилізації та дизайну інтерфейсу, крім CSS, буде використаний один з популярних фреймворків – Tailwind CSS. Цей фреймворк надає готові компоненти та класи для швидкої розробки і привабливого вигляду веб-сайту [11]. Завдяки своїй філософії "utility-first", Tailwind CSS дозволяє розробникам легко стилізувати елементи інтерфейсу, використовуючи класи замість написання власного CSS.

Саме це спрощує роботу зі стилями, прискорює процес розробки та забезпечує єдність стилів на всьому сайті (рис.2.3).

Отже, використання Tailwind CSS у розробці веб-сайту на Ruby on Rails дозволяє швидко і зручно стилізувати інтерфейс, забезпечувати єдність стилів та пристосовувати дизайн до потреб проекту. Це сприяє швидкому розвитку та покращує візуальний вигляд веб-сайту.

Також варто зазначити, що Ruby on Rails має добре розвинену систему тестування. Це дозволяє проводити автоматизоване тестування коду, забезпечуючи високу якість та надійність роботи веб-сайту по обслуговуванню. Тести допомагають виявляти та усувати помилки на ранніх етапах розробки, що сприяє підвищенню стабільності та ефективності веб-сайту.

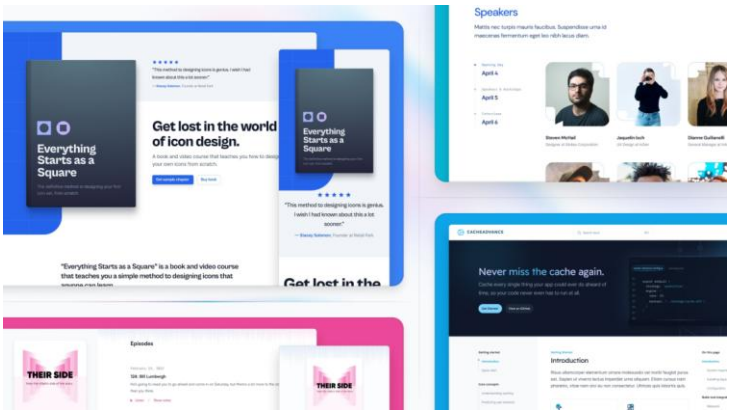


Рисунок 2.3 – Tailwind CSS

2.2 Написання User Stories

У процесі розробки веб-сайту обслуговування на Ruby on Rails я активно застосовував підхід, заснований на написанні User Stories [12]. Цей метод, що ґрунтується на Agile-методології розробки програмного забезпечення, дозволив мені, як розробнику, краще зрозуміти потреби користувачів та створювати продукт, який відповідає їхнім очікуванням.

1. Основні принципи User Stories.

Підхід із написання User Stories базується на коротких, простих описах функціональності системи з точки зору кінцевого користувача. Головна мета цього підходу – створення чіткого розуміння того, як система повинна працювати та які результати мають бути досягнуті для задоволення потреб користувачів. Кожна User Story містить три ключові елементи:

– що? – Короткий опис бажаної функціональності;
– хто? – Актор, якому ця функціональність потрібна;
– чому? – Корисність, яку ця функціональність надає (рис.2.4).

Рисунок 2.4 – Будова User Story

2. Переваги використання User Stories.

Використання User Stories у процесі розробки веб-сайту дозволило мені зосередитися на впровадженні функціоналу, який найкраще задовольняє потреби користувачів. Кожна User Story стала основою для розробки відповідних моделей, контролерів та представлень, забезпечуючи необхідний функціонал для веб-сайту обслуговування відеомонтажу.

3. Реалізація на Ruby on Rails.

Підхід із написання User Stories був ключовим етапом у реалізації веб-сайту обслуговування на Ruby on Rails. Він дозволив мені, як розробнику, краще зрозуміти потреби користувачів та створити продукт, який відповідає їхнім очікуванням. Кожна User Story чітко визначала функціональні вимоги, що спрощувало процес розробки та гарантувало, що всі аспекти веб-сайту узгоджуються з потребами кінцевих користувачів.

Застосування Agile-методології та підходу із написання User Stories допомогло забезпечити успішну розробку веб-сайту на Ruby on Rails. Це сприяло створенню продукту, що відповідає високим вимогам користувачів, забезпечуючи їх задоволення та позитивний досвід використання.

Ось кілька User Stories, які були написані під час розробки сайту, для всіх користувачів:

1. Як новий замовник, я хочу бачити усі роботи відеомонтажера, щоб ознайомитись з багатьма його навичками.

2. Як новий замовник, я хочу бачити детальну інформацію щодо кожного проекту в портфоліо, щоб побачити всі відгуки від замовників та витрачений час на реалізацію проекту.

3. Як новий замовник, я хочу мати свій аккаунт, щоб зберігати мої усі мої замовлення на самому сайті.

4. Як клієнт, я хочу мати контакти фахівця, щоб отримати відповідь на всі мої запитання по темі сервісу.

5. Як клієнт з великим обсягом замовлень, хочу мати середовище для зберігання усіх проектів, щоб легко було до кожного скинути окремий матеріал та опис до нього.

6. Як клієнт, я хочу мати доступ до налаштувань свого акаунта, щоб в любий час змінити пошту, пароль, псевдонім та аватарку.

7. Як клієнт, я хочу щоб сайт повідомляв мене через електронну пошту про зміни в тікеті, щоб постійно не заходити на сайт і перевіряти статус роботи.

8. Як клієнт, я хочу мати в тікеті відео-прототип, щоб краще уявити майбутній продукт, та передати правки відеомонтажеру.

9. Як клієнт, я хочу мати можливість закидувати в чат різноманітні медіа, щоб мати змогу краще демонструвати мої потреби в реалізації продукту.

10. Як адмін, я хочу мати можливість добавляти кілька версій прототипу, щоб краще виконати потреби клієнта.

11. Як адмін, я хочу мати сторінку із всіма тікетами від усіх клієнтів.

2.3 Архітектура бази даних

Архітектура бази даних та вибір оптимальної бази даних є критично важливими аспектами для будь-якого веб-додатку, особливо якщо мова йде про сайт, що надає послуги відеомонтажу. Це зумовлено тим, що сайт “S.VE” зберігає велику кількість інформації про користувачів, всі створені тікети, портфоліо, та інші дані. Тому обрана база даних повинна забезпечувати надійне зберігання, ефективне управління та захист цих даних.

При створенні нового проекту в Ruby on Rails (RoR), за замовчуванням використовується база даних SQLite. Однак, для реалізації мого сайту на RoR було обрано реляційну базу даних PostgreSQL (рис. 2.5). PostgreSQL відома своєю надійністю, можливістю масштабування та підтримкою складних запитів та зв'язків між даними, важливими для індустрії [13].

Попри те, що PostgreSQL є потужним рішенням, одним з її найпопулярніших аналогів є MySQL. MySQL також є реляційною базою даних з великою підтримкою розробників та широким використанням у веб-розробці, включаючи проекти на Ruby on Rails. Вона забезпечує високу продуктивність і масштабованість, хоча в деяких аспектах може поступатися PostgreSQL в потужності та оптимізації.

Ще однією популярною альтернативою є MongoDB, яка належить до категорії NoSQL баз даних. MongoDB зберігає дані у форматі документів, схожих на JSON, що дозволяє гнучко зберігати та обробляти інформацію без

необхідності використовувати жорстко визначені схеми. Це робить її привабливим вибором для проектів, де необхідно ефективно зберігати та обробляти великі обсяги неструктурованих даних.

Іншим варіантом є SQLite, легкий та вбудовуваний реляційний двигун бази даних, який не потребує окремого сервера для роботи. Вся база даних зберігається у одному файлі, що спрощує розгортання та управління. SQLite підтримує мову SQL та має достатні функціональні можливості для багатьох веб-проектів. Однак, у порівнянні з PostgreSQL, SQLite має свої обмеження. Вона не підтримує деякі розширені можливості, такі як реплікація даних, масштабування та паралельне виконання запитів. Крім того, SQLite використовує менш потужні алгоритми оптимізації та має обмежену підтримку конкурентного доступу до бази даних.

Зважаючи на все вищесказане, вибір бази даних для веб-додатку значною мірою залежить від специфічних вимог проекту. PostgreSQL була обрана для сайту “S.VE” завдяки її надійності, масштабованості та потужній підтримці складних запитів, що є ключовими для збереження і обробки великої кількості даних про користувачів та їх діяльність. Однак, інші бази даних, такі як MySQL, MongoDB та SQLite, також мають свої переваги та можуть бути використані в залежності від конкретних потреб проекту.

Рисунок 2.5 – Логотип СКБД PostgreSQL

У Ruby on Rails для оновлення структури бази даних використовуються міграції. Це файлові скрипти, які описують зміни у структурі бази даних, такі

як створення таблиць, додавання або видалення стовпців та інше. Міграції дозволяють керувати версіями структури бази даних і полегшують оновлення на робочому сайті без необхідності ручного втручання. Вони автоматично відстежуються та зберігаються у вигляді файлів у папці db/migrate у самому проекті. Кожна міграція має унікальне ім'я та містить методи, які визначають необхідні зміни структури бази даних.

Наприклад, для створення нової таблиці використовується метод create_table, для додавання стовпця – метод add_column, а для видалення таблиці або стовпця – метод drop_table або remove_column.

Якщо потрібно скасувати зміни, внесені міграцією, Ruby on Rails дозволяє виконати відкат (rollback). Це забезпечує безпеку та цілісність бази даних, запобігаючи її пошкодженню. Виконання rollback повертає базу даних до попереднього стану, видаляючи або змінюючи структуру відповідно до скасованої міграції. Наприклад, якщо ви створили таблицю за допомогою міграції, rollback видалить цю таблицю. Це особливо важливо, коли необхідно видалити міграційний файл; без відкату база даних може залишитися в неконсистентному стані і зламатися, особисто бували випадки коли забував про відкат змінюючи один символ в міграційному файлі, що потім призводило до поломок бази даних.

Що не менш важливо це зв’язки між таблицями, що є ключовим аспектом розробки веб-додатків на базі Ruby on Rails, який забезпечує ефективну взаємодію з базою даних через систему ORM (Object-Relational Mapping) ActiveRecord. Rails надає декілька типів зв'язків між таблицями: один до одного, один до багатьох і багато до багатьох. Кожен з цих типів зв'язків дозволяє розробникам визначати та моделювати взаємозалежності між різними об'єктами додатка, які прописуються в моделі.

Зв'язок "один до одного" використовується, коли один запис у таблиці відповідає одному запису в іншій таблиці. Наприклад, кожен користувач може

мати один профіль. Це реалізується через метод has_one у моделі User і belongs_to у моделі Profile. У свою чергу, зв'язок "один до багатьох" застосовується, коли один запис у таблиці може мати багато записів в іншій таблиці, наприклад, один автор може мати багато статей. Для цього використовується метод has_many у моделі Author і belongs_to у моделі Article.

Зв'язок "багато до багатьох" використовується, коли кожен запис у одній таблиці може бути пов'язаний з багатьма записами в іншій таблиці і навпаки. Наприклад, студенти можуть відвідувати багато курсів, а кожен курс може мати багато студентів. Це реалізується через проміжну таблицю (join table), яка забезпечує зв'язок між студентами та курсами. Найкраще використовувати модель для проміжної таблиці, оскільки це забезпечує більшу гнучкість та можливість додавання додаткових атрибутів.

Таким чином, зв'язки між таблицями в Rails дозволяють легко моделювати складні взаємозалежності між даними і забезпечують простий та інтуїтивно зрозумілий спосіб роботи з базою даних. Правильне розуміння і використання цих зв'язків є основою ефективної розробки веб-додатків на базі Ruby on Rails.

В самому проект реалізовано кілька таблиць, таблиця users зберігає інформацію про всіх користувачів сайту, включаючи адміністраторів. Кожен запис у цій таблиці містить такі поля:

– id: унікальний ідентифікатор користувача;
– name: ім'я користувача;
– email: електронна адреса користувача;
– password_digest: хеш пароля для безпечного зберігання паролів;
– admin: булеве поле, яке вказує, чи є користувач адміністратором.

Таблиця tickets зберігає інформацію про тікети, створені користувачами для відеомонтажу. Кожен тікет може мати пов'язані повідомлення в чаті та прототип відеомонтажу. Поля включають:

– id: унікальний ідентифікатор тікета;

– user_id: ідентифікатор користувача, який створив тікет (зв'язок із таблицею users);

– title: Назва тікета;

– description: опис тікета;

– status: статус тікета (наприклад, "в очікуванні", "в процесі", "завершено");

– created_at, updated_at: дати створення та оновлення тікета.

Таблиця chats зберігає повідомлення, що обмінюються між користувачами та адміністраторами у межах конкретного тікета. Поля включають:

– id: унікальний ідентифікатор повідомлення;

– ticket_id: ідентифікатор тікета, до якого належить чат (зв'язок із таблицею tickets);

– user_id: ідентифікатор користувача або адміністратора, який надіслав повідомлення (зв'язок із таблицею users);

– message: текст повідомлення;

– created_at, updated_at: дати створення та оновлення повідомлення.

Таблиця video_prototypes зберігає прототипи готових відеомонтажів, пов'язаних з тікетами. Кожен прототип прикріплений до конкретного тікета. Замість зберігання URL-адреси відеопрототипу, використовується Active Storage для зберігання відеофайлів в обласному сховищі. Поля включають:

– id: унікальний ідентифікатор відеопрототипу;

– ticket_id: ідентифікатор тікета, до якого належить відеопрототип (зв'язок із таблицею tickets);

– video_attachment: поле для збереження прив'язаного відеофайлу за допомогою Active Storage;

– created_at, updated_at: дати створення та оновлення відеопрототипу;

Для забезпечення цілісності даних та зв'язків між таблицями використовуються ключі, асоціації та звязки. Основні зв'язки включають:

– один користувач може мати багато тікетів (user has_many :tickets);

– один тікет належить одному користувачу (ticket belongs_to :user);

– один тікет може мати багато повідомлень у чаті (ticket has_many :chats);

– одне повідомлення належить до одного тікета (chat belongs_to :ticket);

– один тікет може мати багато відеопрототип (ticket has_many:video_prototype);

– один відеопрототип належить до одного тікета (video_prototype belongs_to :ticket).

Ця структура забезпечує ефективне керування даними та логічні зв'язки між різними компонентами сайту портфоліо відеомонтажу, забезпечуючи зручність та функціональність системи з використанням можливостей Active Storage для зберігання відеофайлів.

Таким чином, правильно спроектована архітектура бази даних є фундаментом для успішного веб-додатка, а правильні зв'язки між таблицями в цій архітектурі грають ключову роль у забезпеченні її ефективності, масштабованості та надійності.

2.4 Проектування інтерфейсу

Проектування інтерфейсу є критично важливою частиною розробки програмного продукту або системи. Врахування потреб та очікувань користувачів, структурування інформації, вибір елементів керування та використання зручних графічних компонентів допомагають створити зручний, ефективний та приємний інтерфейс для користувачів. Регулярне тестування та вдосконалення інтерфейсу дозволяє покращувати його якість та відповідність потребам користувачів.

Загальна структура сайту з відображенням зв’язків та переходів між сторінками була змодельована на сайті Draw.io [16] та показана на (рис. 2.7).

Використовуючи хедер на головній сторінці, де буде розміщено більшість посилань сайту, користувачі можуть швидко переходити на потрібну сторінку.

Використовуючи аналіз із першого розділу, можна ефективно розробити інтерфейс веб-сайту. Важливими етапами в процесі проектування UI/UX інтерфейсу є створення скетчів та прототипів, оскільки вони дозволяють візуалізувати ідеї та концепції інтерфейсу і перевірити їх перед реалізацією.

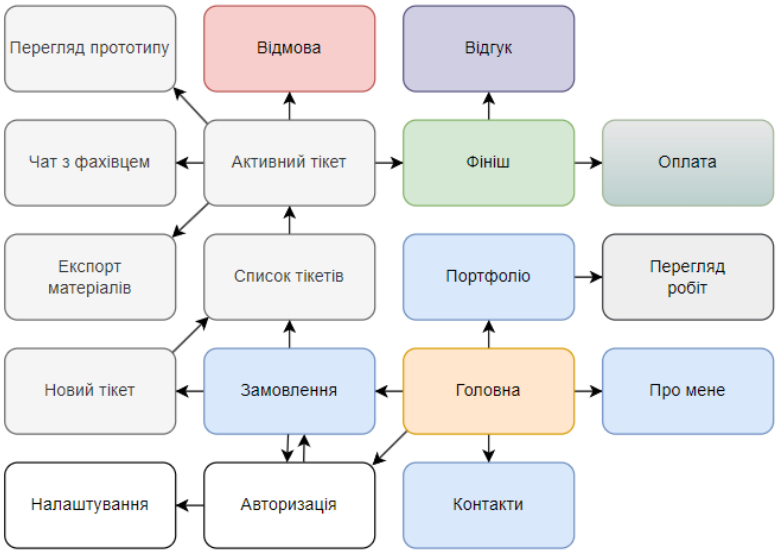


Рисунок 2.7 – Структура сайту

1. Створення скетчів.

Скетчі використовуються для швидкого начерку концепцій та ідей інтерфейсу. Вони можуть бути створені вручну за допомогою паперу та олівця або за допомогою спеціальних програм для створення скетчів, яких в інтернеті є багато. Основна мета скетчів – зосередитися на структурі, розміщенні елементів та логіці взаємодії без деталізації. Це дозволяє швидко перевірити різні варіанти розташування та взаємодії елементів інтерфейсу.

2. Створення прототипів.

Прототипи є більш деталізованими варіантами інтерфейсу, які можуть бути інтерактивними. Вони створюються за допомогою спеціального програмного забезпечення для прототипування, яке дозволяє моделювати

взаємодію з різними елементами інтерфейсу. Прототипи дозволяють випробувати функціональність інтерфейсу, перевірити його робочий процес та збирати відгуки від користувачів. Це допомагає зрозуміти, як буде виглядати та працювати інтерфейс перед розробкою його на реальному рівні, виявити проблеми та внести необхідні зміни в дизайн та функціональність, що економить час та ресурси у майбутньому.

3. Використання варфреймів.

У процесі проектування інтерфейсу для сайту обслуговування важливим інструментом є варфрейми.

Вайрфрейми – в деталях показують, яка інформація, контент і елементи управління повинні виводитися на кожній сторінці інтерфейсу будь якої системи. Також у вайрфреймах вже розставлені акценти щодо різних елементів інтерфейсу: кнопок, зображень, заголовків, тексті, тощо [18].

Вони використовуються для візуалізації структури та розташування елементів у інтерфейсі, не звертаючи уваги на деталі дизайну, кольори та графічні ефекти. Варфрейми допомагають визначити структуру сайту, включаючи головне меню, розділи та розташування різних функцій і інформації. Вони також сприяють виявленню потреб користувачів, візуалізуючи розташування кнопок для створення замовлення та інших важливих функцій.

4. Переваги варфреймів.

Крім того, варфрейми можуть бути використані для узгодження з командою проекту та збору відгуків. Вони дозволяють швидко показати загальну структуру та функціональність інтерфейсу, отримати відгуки та пропозиції щодо його вдосконалення. Варфрейми також можуть бути застосовані для проведення тестування з користувачами, щоб отримати відгуки щодо зручності навігації, розташування елементів та загального враження від інтерфейсу. Це допомагає забезпечити логічну та зручну структуру, відповідну потребам користувачів та бізнесу.

5. Створення варфреймів.

Варфрейми можуть бути створені вручну за допомогою паперу та олівця або за допомогою спеціалізованих програм для дизайну інтерфейсу. Вони можуть бути як статичними, так і інтерактивними, дозволяючи навігацію між сторінками та елементами. Це робить їх надзвичайно корисним інструментом для створення ефективного та привабливого інтерфейсу веб-сайту.

Використання скетчів, прототипів та варфреймів у проектуванні користувацького інтерфейсу для сайту обслуговування є ключовими етапами для успішної реалізації проекту. Ці інструменти допомагають візуалізувати та перевірити ідеї, виявити потенційні проблеми на ранніх стадіях, зібрати відгуки від користувачів та команди проекту. Це сприяє створенню інтуїтивного та ефективного інтерфейсу, який відповідає потребам як користувачів, так і бізнесу. На рисунку 2.8 я розробив прототип головної сторінки, яка демонструє основний стиль веб-сайту за допомогою Figma [17].

Моя мета – створити мінімалістичний та привабливий для очей техно-функціональний дизайн, який забезпечує зручну навігацію та залучає користувачів до взаємодії з сайтом. У моєму прототипі головної сторінки я використав сучасний мінімалістичний стиль, фокусуючись на краплю футуристики, темних відтінків та ніжно-м'ятного ключового кольору.

На початку головної сторінки буде програватись відео, яке продемонструє усю динаміку роботи та важливі процеси, за допомогою яких буде видно результат проекту.

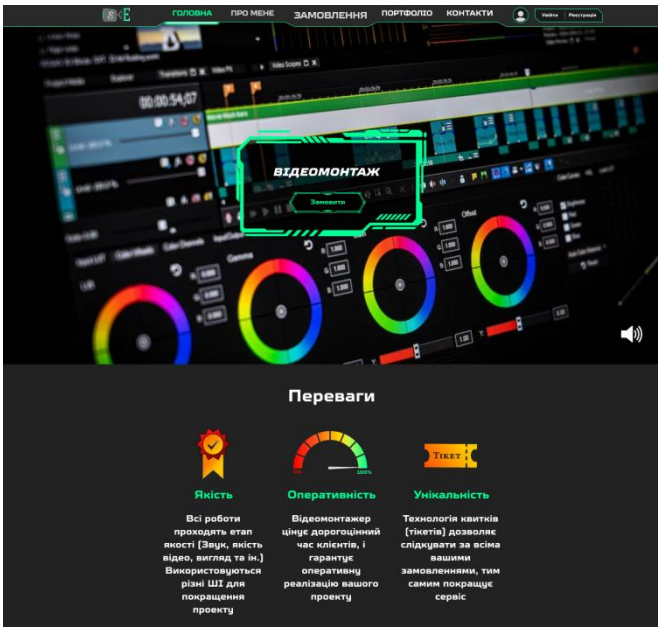


Рисунок 2.8 – Прототип головної сторінки “S.VE”

Окрім цього, я врахував важливі елементи, які сприяють користувацькому досвіду, такі як проста навігація, використання іконок для ілюстрації функціональності та вирівнювання елементів по гріду, задля зручного перегляду веб-сайту обслуговування.

В результаті роботи над прототипом головної сторінки, я створив просте інформаційне веб-середовище яке демонструє переваги та останні роботи фахівця, за допомогою яких можуть залучити користувачів та розпочати взаємодію з новими тікетами. Цей прототип на даний момент є фінальним варіантом та буде використаний на кінцевому продукті.

На рисунку 2.9 наведено ілюстрацію інформаційної сторінки, що відображає фахівця у сфері відеомонтажу, а також його детальний опис, який відкриває перед замовником повний образ автора. Це допомагає визначити, з ким саме замовник буде взаємодіяти.

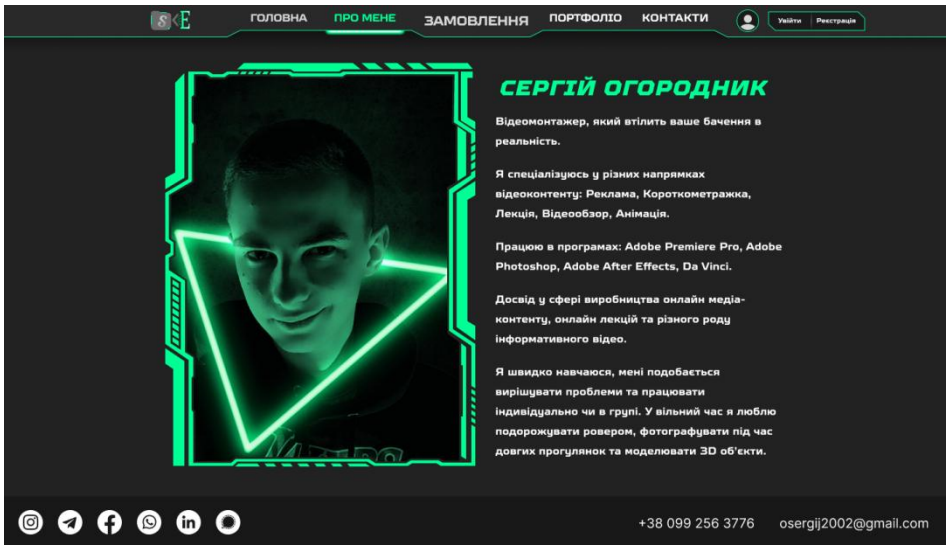


Рисунок 2.9 – Інформаційна сторінка “S.VE”

У моєму прототипі я також врахував значення доступності та використання універсальних дизайнерських рішень. Я здійснив контрастність кольорів, щоб полегшити читання, і використав тематичні шрифти з гарною читабельністю.

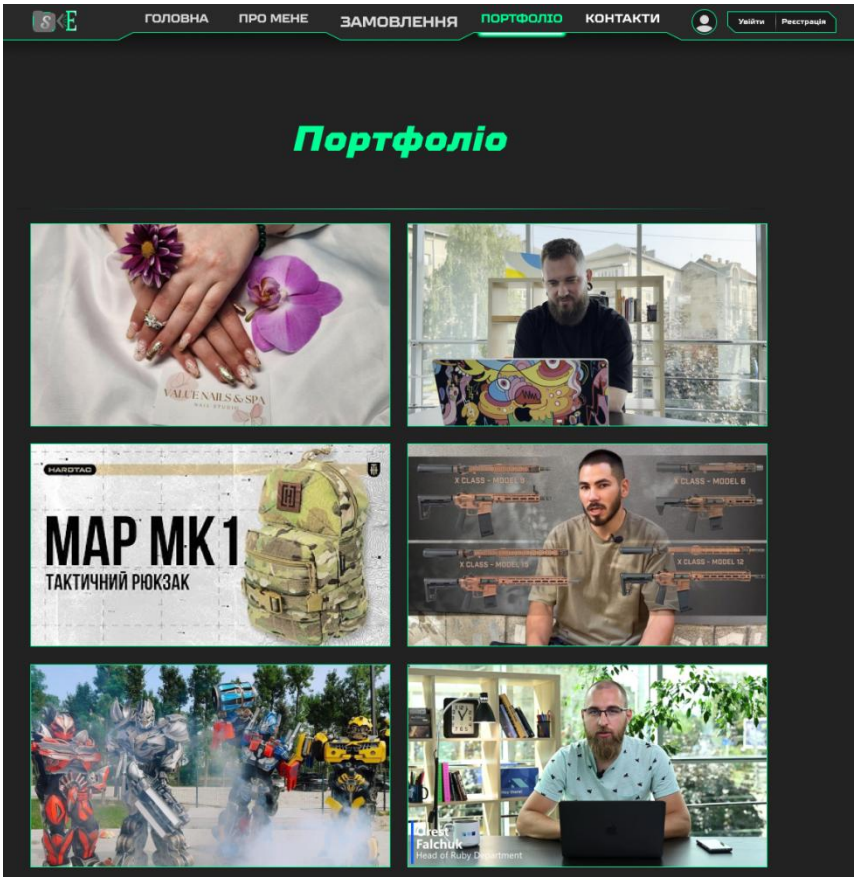


Рисунок 2.10 – Портфоліо “S.VE”

Наступна сторінка на рисунку 2.10 вміщує в собі зручну таблицю з усіма проектами фахівця, у кожної якої з них є інформація про замовника, гіперпосилання, відгук а також шкала часу, яка інформує наскільки багато пішло часу на проект.

Усі проекти відкриваються зручним методом Popup (не велике спливаюче вікно), що у свою чергу не переносить користувача на нову сторінку, тим самим забезпечує швидкий перегляд, та зручність в користуванні.

При огляді будь-якого портфоліо відкривається спливаюче вікно, на якому видно усі деталі проекту рисунку 2.11:

– відеоматеріал, який демонструє навички фахівця;
– замовник;
– посилання на відеоролик в мережі (при наявності);
– відгук від замовника;
– шкала часу, яка інформує скільки було часу потрачено на проект.

На мою думку, наявність усіх цих аспектів в портфоліо є обов’язковим пунктом, тому що всі задіяні деталі додадуть життя в саме портфоліо, та забезпечить більшу довіру замовників до фахівця. Саме цих пунктів буде достатньо, щоб оцінити любу роботу портфоліо.

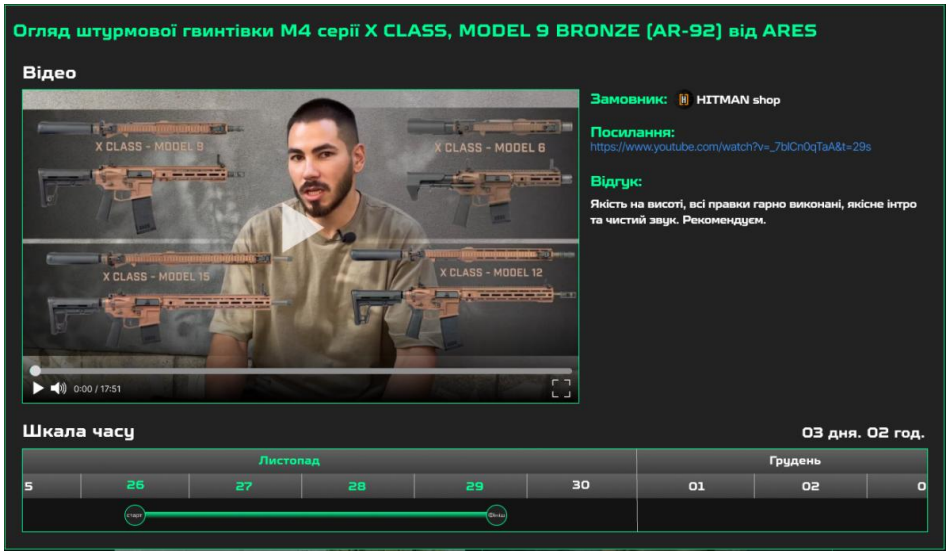


Рисунок 2.11 – Проект “S.VE”

На рисунку 2.12 можна побачити сторінку контактних даних фахівця веб-сайту обслуговування відеомонтажу. Кожна особа зможе звернутись до відеомонтажера з необхідності відповісти на питання по темі сервісу.

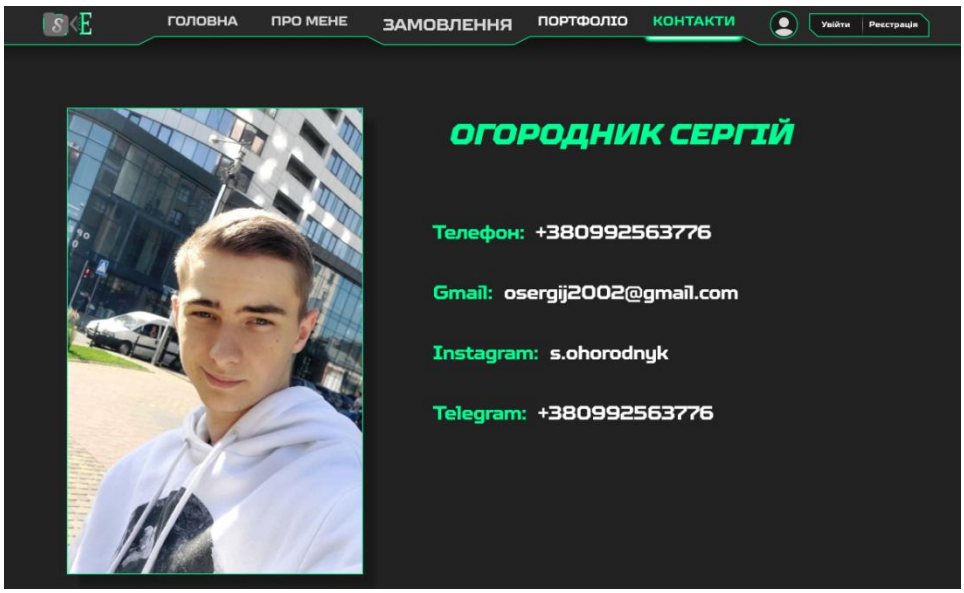


Рисунок 2.12 – Контакти фахівця “S.VE”

Увесь основний функціонал веб-сайту по відеомонтажу знаходиться саме на сторінці “замовлення” (рис. 2.14). Саме тут і відбувається увесь сервіс для замовників. Але перед тим, як замовити послугу у фахівця – необхідно авторизувати свій акаунт на веб-сайті (рис. 2.13), для того, щоб зберігати усі власні тікети на своєму акаунті.

Ця дія дозволяє створювати список замовлень, та зберігати всі свої матеріали на сайті.

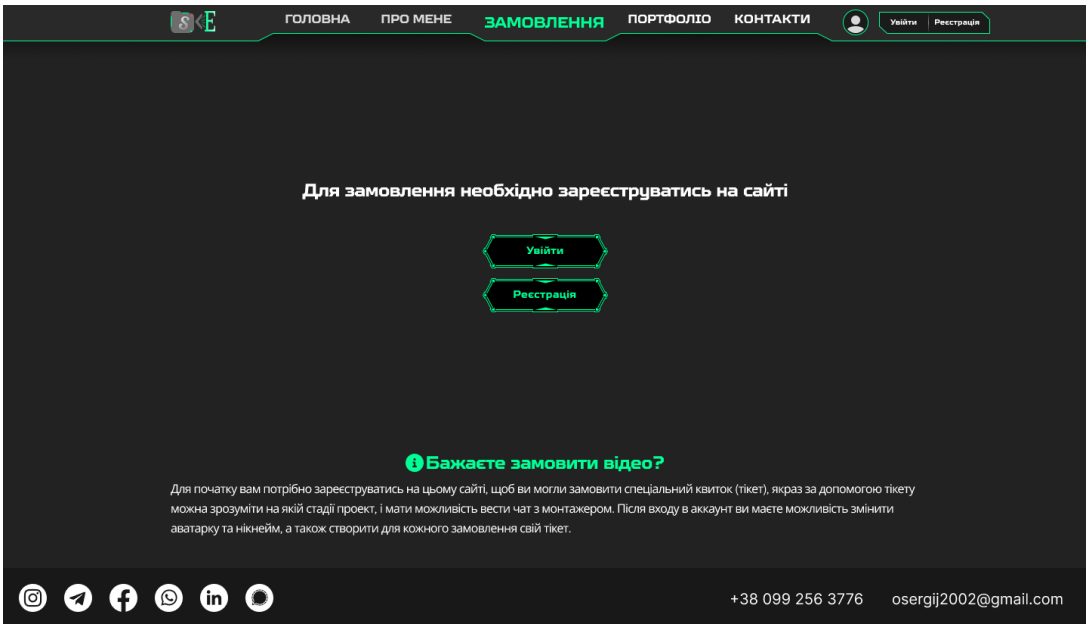


Рисунок 2.13 – Сторінка авторизації “S.VE”

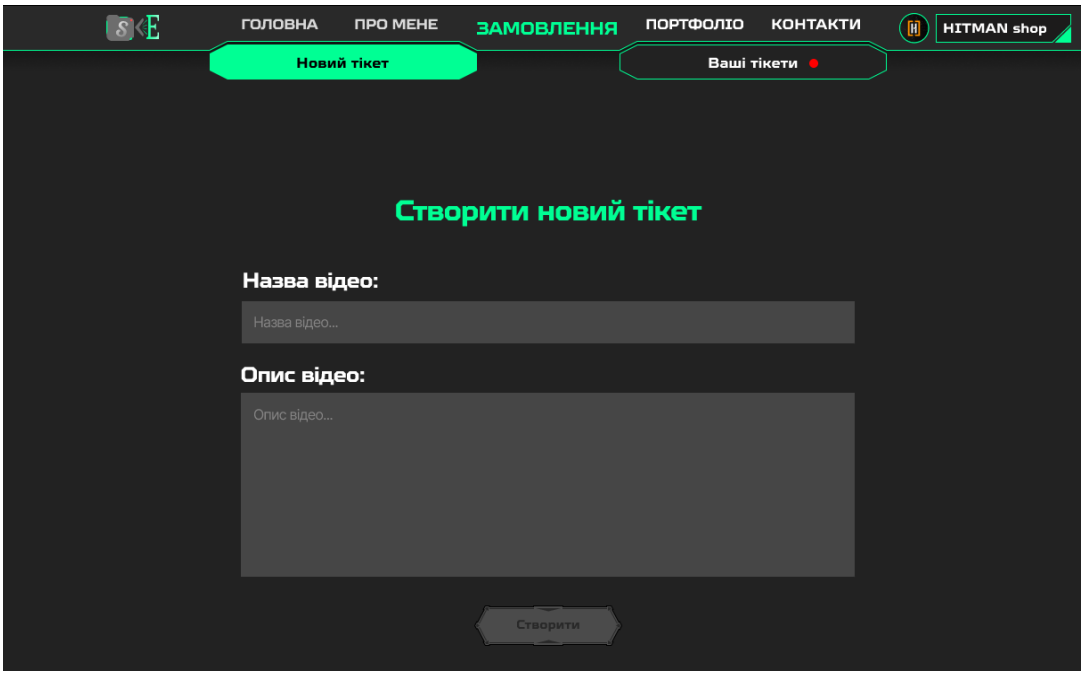


Рисунок 2.14 – Сторінка замовлення “S.VE”

На сторінці “замовлення” з'являються нові вкладки: “новий тікет” та “ваші тікети”. Вкладка “Новий тікет” у свою чергу дає можливість користувачу створити нове замовлення, та вказати як назву проекту, так і його опис. У фінальній версії веб-сайту буде додана функція експорту відеоматеріалу, щоб при створенні замовлення з матеріалами вже можна було взаємодіяти фахівцем.

Як тільки користувач створить свій тікет, його відразу перекине на наступну вкладку під назвою “ваші тікети” (рис. 2.15), це місце, де надалі будуть зберігатись усі його замовлення. Саме це рішення має перевагу в тому, що замовник, маючи при собі всі замовлення в одному місці, зможе переглядати статус кожного тікету, та легко переключитися з одного замовлення на інше.

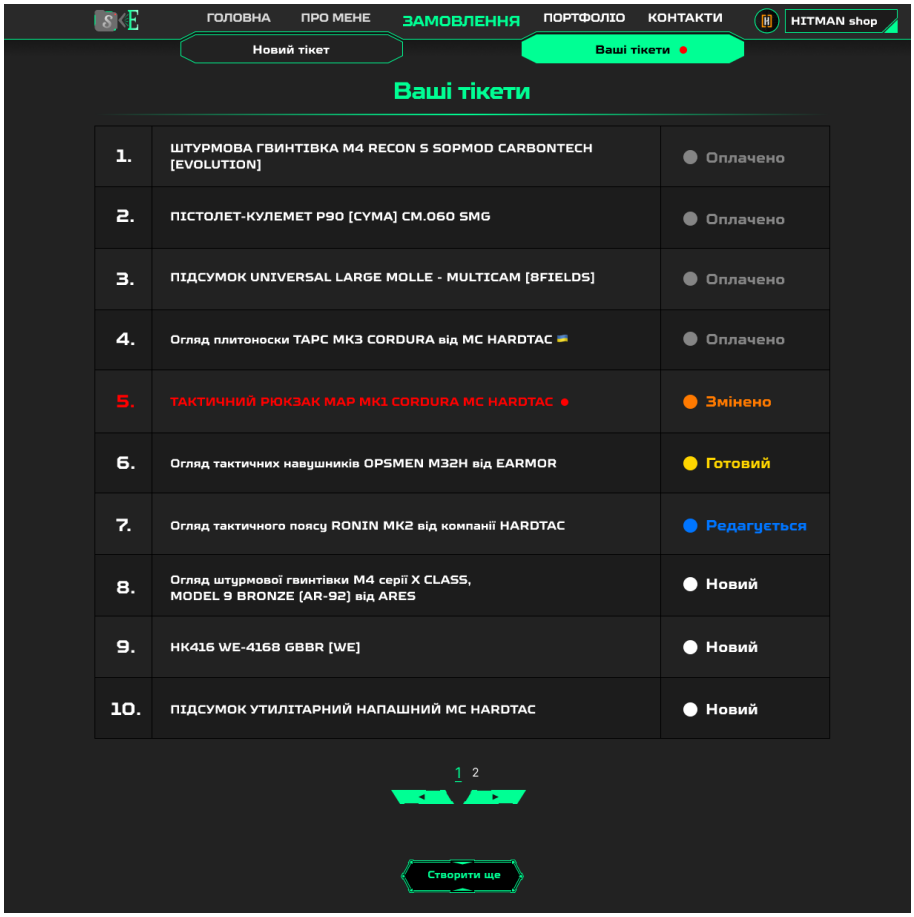


Рисунок 2.15 – Вкладка “Ваші тікети” S.VE

При виборі одного з тікетів, відкривається нове спливаюче вікно, яке вміщує в собі наступні елементи у рисунку 2.16:

– чат замовника з відеомонтажером, де необхідно інформувати фахівця про бачення за проект, та вказувати усі правки, якщо першопочаткова версія продукту була не вдалою;

– прототип продукту, який дозволяє переглянути результат та з часом буде змінюватись в залежності від правок;

– статус тікету, який буде вказувати фахівець;

– вирішальні кнопки “Фініш” та “Відмова”.

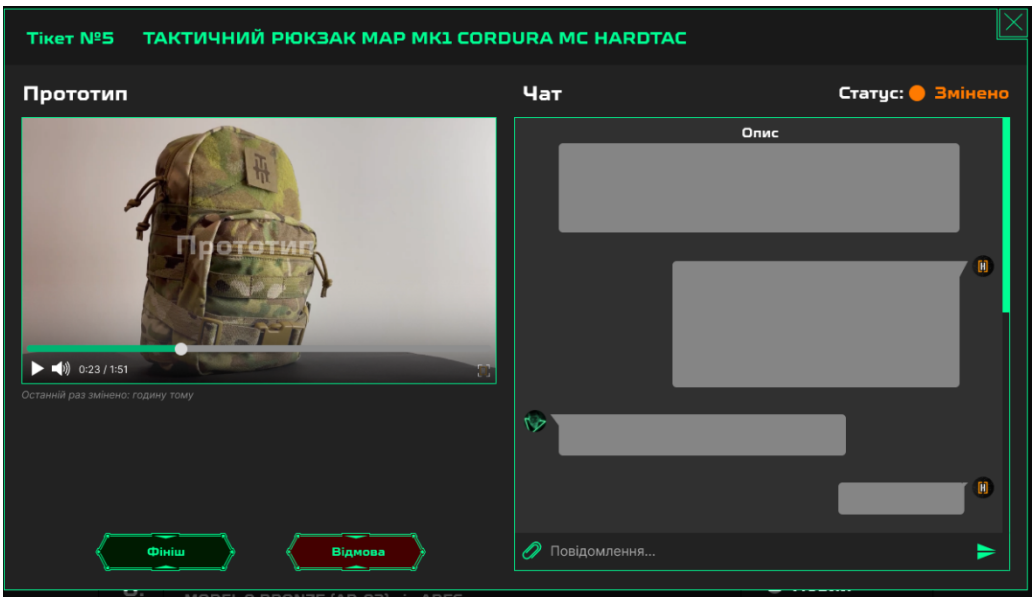


Рисунок 2.16 – Тікет “S.VE”

Висновки до розділу 2

Результатом цього розділу є розробка загальної архітектури веб-сайту обслуговування, а також визначення додаткових інструментів, які полегшують процес розробки. Щоб краще зрозуміти, що і як потрібно розробити, в розділі 2.2 було наведено кілька історій користувачів (User Stories). Також була створена та описана структура бази даних, а також вказано, яка база буде використовуватись для зберігання даних. Зокрема, було змодельовано дизайн та структуру інтерфейсу, щоб краще уявити, яким буде виглядати та яким буде функціонал веб-сайту.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-САЙТУ

3.1 Створення базової аплікації з налаштуваннями

На початковому етапі розробки сайту для послуг відеомонтажу на Ruby on Rails, правильний вибір технології значно спростив завдання. Мій перший крок був спрямований на освоєння основних концепцій та структури RoR, що дозволило мені глибше зрозуміти фреймворк. Початок роботи над веб-сайтом передбачав детальне планування архітектури, створення моделей та міграцій бази даних.

Крім того, я вивчив використання HTML, CSS та JavaScript для створення зручного та естетичного інтерфейсу користувача. Для розробки проекту я використовував Visual Studio Code, що надавав зручний інтерфейс та широкі можливості редагування коду. Початок роботи з RoR зазвичай розпочинається зі створення папки проекту, в якій розміщується структура файлів проекту, яка буде відображати всі його складові.

Початкова реалізація проекту була здійснення з допомогою команди: `rails new media --database=postgresql -T`

Де дані параметри виконують такі функції:

1. `rails new` – ця команда створює новий проект, із базовою структурою.
2. `media` – ця задає назву проекту, не тільки назву директиви з проектором, а назві заголовків, і подібного.
3. `--database=postgresql` – цей параметр визначає, що для проекту буде використовуватися PostgreSQL як основна база даних. Rails автоматично підключає необхідні бібліотеки для роботи з цією базою даних і створює конфігураційний файл з налаштуваннями підключення.
4. `--T` – цей параметр означає відмову від стандартного тестового фреймворку Minitest, який включається за замовчуванням. Замість нього для

написання тестів буде використовуватися RSpec, один з найпопулярніших і найпотужніших інструментів для тестування в Ruby.

Вказування бази даних відбувається ще з тих причин що за замовчуванням в Rails проектах використовується SQLite, тому це просто зміна архітектури на етапі, коли це ще дуже легко зробити, тим більше ще одним явним недоліком SQLite є те що, багато хостингових ресурсів не працюються з цією базою, тому часто її просто міняють на production.

Так як ми використовуємо ще tailwind css, то одразу після ініціалізації проекту, і додавання в gemfile команди gem "`tailwindcss-rails`", яка добавляє залежність нашого проекту до цієї технології, можем інсталювати і його сами написавши – rails tailwindcss:install.

А таким чином прописуємо tailwind.config.js:

```
const defaultTheme = require('tailwindcss/defaultTheme')
    module.exports = {
      content: [
        './public/*.html',
        './app/helpers/**/*.rb',
        './app/javascript/**/*.js',
        './app/views/**/*.{erb,haml,html,slim}'
      ],
      theme: {
        extend: {
          fontFamily: {
            sans: ['Inter var', ...defaultTheme.fontFamily.sans],
          },
        },
      },
      plugins: [
        require('@tailwindcss/forms'),
        require('@tailwindcss/typography'),
        require('@tailwindcss/container-queries'),
      ]
    }
```

Цей конфігураційний файл Tailwind CSS забезпечує налаштування та розширення стандартної теми, вказує, де шукати класи CSS, та підключає додаткові плагіни для розширення функціональності CSS у вашому проекті.

3.2 Створення автентифікації

Автентифікація є критично важливою складовою будь-якого веб-додатка. Впровадження автентифікації на початкових етапах розробки забезпечує безпеку, контроль доступу, покращений користувацький досвід, легкість інтеграції та підтримки додатка. Використання потужних інструментів, таких як Devise [15], дозволяє швидко і ефективно налаштувати автентифікацію, забезпечуючи надійну основу для подальшого розвитку вашого проекту.

Для встановлення його спершу потрібно добавити його в Gemfile, після чого проінсталювати всі залежності з допомогою команди bundle install.

Командою rails generate devise User була створена модель User з необхідними для автентифікації полями, такими як email і password, але перед запуском міграції потрібно добавити і свої зміни, добавивши кілька нових полів:

```
# frozen_string_literal: true
     class DeviseCreateUsers < ActiveRecord::Migration[7.1]
       def change
         create_table :users do |t|
           ## Database authenticatable
           t.string :name,              null: false, default: ""
           t.string :email,              null: false, default: ""
           t.string :encrypted_password, null: false, default: ""
           ## Recoverable
           t.string   :reset_password_token
           t.datetime :reset_password_sent_at
           ## Rememberable
           t.datetime :remember_created_at
           t.boolean :admin, default: false, null: false
```

```
      t.timestamps null: false
    end
    add_index :users, :email,                unique: true
    add_index :users, :reset_password_token, unique: true
  end
end
```

Щоб обмежити доступ до певних дій або контролерів, можна використовувати хелпер before_action з Devise. Наприклад, щоб вимагати автентифікацію для всіх дій у контролері ArticlesController, додайте наступний рядок:

```
class ArticlesController < ApplicationController
  before_action :authenticate_user!

end
```

Цей хелпер гарантує, що користувачі повинні бути автентифікованими, щоб отримати доступ до дій у контролері ArticlesController.

Devise надає готові представлення для різних дій автентифікації. Щоб налаштувати ці представлення під свої потреби, згенеруйте їх локальні копії:

```
rails generate devise:views
```

Ця команда створить папку views/devise з усіма необхідними представленнями, які ви можете редагувати відповідно до дизайну вашого додатка.

3.3 Створення контроллера для тікетів

Контролери в Ruby on Rails є невід'ємною частиною архітектури MVC (Model-View-Controller), яка структурно розділяє різні аспекти веб-додатка. Контролери обробляють вхідні HTTP-запити, взаємодіють з моделями для отримання або оновлення даних, а також відповідають за передачу даних до

представлень, які потім відображаються користувачеві. Правильне налаштування і використання контролерів є важливим для забезпечення ефективної та зручної розробки.

Контролери є логічними обробниками запитів, які користувачі надсилають до веб-додатка. Вони визначають, яку дію потрібно виконати залежно від типу запиту (GET, POST, PUT, DELETE) і URL. Кожна дія контролера, яка відповідає певному маршруту, може взаємодіяти з моделями для виконання CRUD (Create, Read, Update, Delete) операцій та передавати дані до відповідних представлень для відображення.

Одним з основних переваг Rails є можливість автоматичної генерації коду, включаючи контролери. Використання генераторів значно скорочує час розробки, створюючи базову структуру контролера та необхідні файли. Процес генерації контролера включає створення відповідних файлів для тестування, представлень і маршрутизації. Створити контролер можна з допомогою команди rails generate controller Tickets, де Tickets назва контролера відповідно до створеної бази даних з моделлю, додатково ще можна вказати ім'я функцій які ми хочем використи в, в дану випадку, для тікетів це був index, show, і create, де index рендерить сторінку зі всіма тікетами, що буде корисно як для адміна так і для замовників, show рендеритиме сам тікет після того як вибрали із списку, і create логіка яка створює новий тікет, тому в нашому випадку команда матиме вигляд rails generate controller Tickets index create, результатом буде створений контролер і файли до нього, а код в цих функціях ми напишем такий:

```
def index
    @rooms = current_user.rooms
  end
def show
  @current_user = current_user
  @single_ticket = Ticket.find(params[:id])
  @users = User.all_except(@current_user)
```

```
  @Ticket = Ticket.new

  render "index"
end

def create
  @Ticket = Ticket.create(name: params["ticket"]["name"])
end
```

Можна було би ще створити такі сторінки як new, яку часто створюють для проектів, але нам не має необхідності виводити віддільну сторінку, якщо в нас для цього буде використовуватись stimulus, з допомогою якого створить модальне вікно. Stimulus є відмінним вибором для розробників, які прагнуть додати динамічність до своїх веб-додатків, зберігаючи при цьому простоту і читабельність коду. Його легкість, простота інтеграції та організація коду роблять його ідеальним інструментом для сучасних веб-додатків, що потребують інтерактивних елементів без зайвого ускладнення. Використання Stimulus дозволяє ефективно керувати поведінкою користувачів на сторінці, залишаючи HTML-розмітку чистою і зрозумілою.

Маршрути в Rails визначають, який контролер і яку дію слід викликати для конкретного URL. Це дозволяє зручно керувати навігацією веб-сайту і забезпечувати доступ до різних функцій. Налаштування маршрутів виконується в спеціальному файлі конфігурації, де можна легко додавати, змінювати або видаляти маршрути для самого контролера. Маршрути на сайті зберігаються у файлі з назвою routes.rb, в нашому випадку, він матиме такий вигляд:

```
Rails.application.routes.draw do
  devise_for :users
  root "pages#home"
  get 'about', to: 'pages#about'
  get 'portfolio', to: 'pages#portfolio'
  get 'contact', to: 'pages#contact'
  resources :tickets do
```

```
    resources :messages
  end
end
```

– рядок devise_for :user автоматично створює всі необхідні маршрути для автентифікації користувачів за допомогою гем Devise. Включає маршрути для реєстрації, входу, виходу, відновлення паролю тощо;

– root “pages#home” встановлює кореневий маршрут вашого додатка. Коли користувач відвідує головну сторінку (кореневий URL), викликається метод home у контролері PagesController, що по суті говорить нашому додатку, що home, є головною сторінкою;

– для команд де на початку get, ми просто вказуємо маршрут до відповідно сторінки, на них не буде ніякої логіки, чи контакту з базою даних, але там знаходиметься інформація.

Контролери можуть мати безліч дій, кожна з яких відповідає за обробку певного запиту. Наприклад, дія для відображення списку всіх елементів, дія для відображення конкретного елемента, дія для створення нового елемента, а також дії для оновлення або видалення існуючих елементів. Важливо ретельно організувати ці дії, щоб код залишався читабельним і підтримуваним.

Контролери взаємодіють з моделями для отримання, зберігання або видалення даних. Моделі в Rails представляють структуру бази даних і містять логіку бізнес-правил. Контролери використовують методи моделей для виконання необхідних операцій з даними, таких як пошук записів, їх створення або оновлення. Важливо правильно налаштувати взаємодію між контролерами і моделями, щоб забезпечити ефективне та безпечне управління даними.

Не мало важливим моментом є асоціації, і валідація, що присутні в базі даних, модаль тікетів:

```
class Ticket < ApplicationRecord
  # Associations
  belongs_to :user
```

```
  has_one :chats, dependent: :destroy
  has_many :video_prototype, dependent: :destroy

  # Validations
  validates :title, presence: true
  validates :description, presence: true
  validates :status, presence: true
end
```

В даній моделі асоціації виконують функції зв'язків між таблицями, як приклад те що тікети належать користувачам, і можуть мати один чат, де в свою чергу чат належить тікету, і те що тікети можуть мати багато прототипів, для того щоб адмін міг редагувати прототип в залежності від бажать замовника.

Після обробки запиту і взаємодії з моделями, контролери передають отримані дані у представлення. Представлення відповідають за відображення цих даних користувачам у зручному форматі. Це можуть бути HTML-сторінки, JSON-відповіді або інші формати. Контролери часто використовують змінні інстанцій для передачі даних у представлення, що дозволяє легко доступатися до цих даних у шаблонах представлень.

Безпека даних є критично важливою, і Rails використовує концепцію сильних параметрів для захисту від масового присвоєння атрибутів. Це означає, що ви повинні явно визначати, які параметри дозволені для обробки у ваших контролерах. Це допомагає запобігти зловмисним атакам і забезпечує, що лише дозволені параметри будуть використовуватися при створенні або оновленні записів у базі даних.

Тестування контролерів є важливою частиною процесу розробки, оскільки воно допомагає виявити помилки на ранніх етапах і забезпечити належну роботу додатка. Rails надає вбудовані інструменти для тестування, які дозволяють створювати тести для перевірки роботи різних дій контролера. Це включає тестування відповідей, перевірку правильності маршрутизації, а також перевірку коректності взаємодії з моделями і представленнями.

Контролери в Ruby on Rails є центральною частиною архітектури додатка, що відповідає за обробку запитів і керування потоком даних. Використання контролерів дозволяє розробникам створювати добре організовані, масштабовані та безпечні веб-додатки.

Не мало важливим моментом є реалізація чату в ruby on rails, для інтеграції чату в реальному часі, Action Cable використовується для створення двонаправленого зв'язку між сервером і клієнтами. Це дозволяє миттєво передавати повідомлення між користувачами та адміністраторами.

У Rails створюється канал чату, який обробляє підписки та трансляції повідомлень:

```
class ChatChannel < ApplicationCable::Channel
  def subscribed
    stream_from "chat_#{params[:ticket_id]}"
  end

  def unsubscribed
    # Будь-які дії при відписці від каналу
  end

  def send_message(data)
    Chat.create!(
      ticket_id: data['ticket_id'],
      user_id: current_user.id,
      message: data['message']
    )
  end
end
```

Після чого модель Chat зберігає повідомлення в базі даних і транслює їх у відповідний канал:

```
class Chat < ApplicationRecord
  belongs_to :ticket
  belongs_to :user

  after_create_commit do
```

```
    broadcast_append_to "chat_#{ticket_id}"
  end
end
```

Тут можна побачити асоціацію про яку говорилось раніше, що чат належить тікетам. А вже на стороні клієнта використовується JavaScript для підписки на канал і відправки повідомлень.

Висновки до розділу 3

Результатом цього розділу є детальний опис процесів на початку розробки веб-сайту. Зокрема, було створено базовий контролер з приватними методами для спрощення коду та забезпечення захисту.

Приватні методи використовуються для обробки запитів, що допомагає зберігати логіку обробки в одному місці та полегшує підтримку коду. Розроблено модель користувача з чітко визначеним списком допустимих значень для атрибутів.

Реалізовано валідацію даних перед збереженням, що забезпечує цілісність та коректність даних в базі. Також створено таблицю в базі даних для зберігання інформації про користувачів.

Таблиця містить всі необхідні поля для різних категорій користувачів, що дозволяє зберігати різноманітну інформацію про кожного користувача. В результаті виконаної роботи було розроблено повноцінну частину веб-сайту, яка включає управління користувачами в базі даних.

Це дозволило забезпечити:

– структуроване та безпечне зберігання даних: використання валідації та чітко визначених атрибутів гарантує, що всі дані користувачів зберігаються коректно.

– зручне управління користувачами: розроблений контролер та модель забезпечують ефективне управління даними користувачів, включаючи створення, редагування, видалення та перегляд інформації.

– Масштабованість та гнучкість: створена структура бази даних дозволяє легко додавати нові категорії користувачів та атрибути, що забезпечує можливість розширення функціональності сайту в майбутньому.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи створено веб-сайт для обслуговування у сфері відеомонтажу засобами Ruby on Rails. Основні функціональні можливості цього сайту включають:

– детальне портфоліо фахівця: кожен проект містить інформацію про замовника, його відгук, посилання на відео(за необхідністю) та шкала часу, яка демонструє скільки треба було часу для реалізації продукту.

– система тікетів: реалізація організації кожного замовлення клієнта, надання кожному “кімнатку” з окремим чатом до відеомонтажера та оглядовим прототипом відеопроекту.

Було проведено дослідження та аналіз існуючих сайтів-аналоги, які надають сервіс монтажу відеоконтенту. Це дозволило визначити найкращі практики та уникнути недоліків, властивих іншим системам.

При виборі інструментів для розробки сайту враховувались їх ефективність, а також їхні переваги та недоліки. Було обрано Ruby on Rails як основний фреймворк для розробки через його потужність і гнучкість.

Розроблена архітектура веб-сайту включає:

– моделі: відповідають за управління даними в додатку, забезпечуючи цілісність та безпеку даних;

– контролери: обробляють запити користувачів, виконують необхідну логіку та взаємодіють з моделями і представленнями;

– представлення: відображають дані користувачеві у зручному форматі, забезпечуючи відокремлення від бізнес-логіки.

Для розробки були використані додаткові технології, такі як PostgreSQL для створення структури бази даних. PostgreSQL була обрана через її надійність, масштабованість та підтримку складних запитів і зв'язків між даними.

Написання User Stories на основі Agile-методології дозволило краще розуміти потреби користувачів і створити продукт, що відповідає їх очікуванням. Кожна User Story включала короткий опис бажаної функціональності, актора, якому ця функціональність потрібна, та корисність, яку вона надає.

Створення веб-сайту на Ruby on Rails для зручного обслуговування відеомонтажу з великою кількістю замовлень стало можливим завдяки ретельному дослідженню існуючих рішень, обранню відповідних інструментів та технологій, а також використанню User Stories для чіткого визначення потреб користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Веб-сайт Lumiere Production. URL: https://www.lumiereproduction.tv (дата звернення 27.03.2024)

2. Веб-сайт ITukraine.org.ua. URL: https://itukraine.org.ua/members/vimeo/ (дата звернення 29.03.2024)

3. Веб-сайт Mammoth Digital Media. URL: https://mammothdigitalmedia.com (дата звернення 02.04.2024)

4. Веб-сайт Wikipedia.com Grid. URL: https://uk.wikipedia.org/wiki/Таблиця_(елемент_GUI) (дата звернення 06.04.2024)

5. Веб-сайт Wikipedia.com Header. URL: https://redstone.media/yak-stvoryty-garnyy-heder (дата звернення 11.04.2024)

6. Веб-сайт Wikipedia.com Якір. URL: https://cikt.kubg.edu.ua/як-зробити-навігацію-по-сайту-зручною/ (дата звернення 13.04.2024)

7. Веб-сайт Wikipedia.com Discord. URL: https://uk.wikipedia.org/wiki/Discord (дата звернення 14.04.2024)

8. Веб-сайт Wikipedia.com Ruby on Rails. URL: https://en.wikipedia.org/wiki/Ruby_on_Rails (дата звернення: 17.04.2024)

9. Документація Ruby on Rails URL: https://guides.rubyonrails.org/ (дата звернення: 19.04.2024)

10. Веб-сайт Wikipedia.com MVC. URL: https://en.wikipedia.org/wiki/Model-View-Controller (дата звернення: 21.04.2024)

11. Документація Tailwind CSS. URL: https://tailwindcss.com/docs/installation (дата звернення: 25.04.2024)

12. Веб-сайт Wikipedia.com User Case. URL: https://en.wikipedia.org/wiki/User_story (дата звернення: 28.04.2024)

13. Документація PostgreSQL URL: https://www.postgresql.org/docs/ (дата звернення: 5.05.2024)

14. Веб-сайт Wikipedia.com ActiveRecord. URL: https://en.wikipedia.org/wiki/Active_record_pattern (дата звернення: 09.05.2024)

15. Документація Devise URL: https://github.com/heartcombo/devise (дата звернення: 13.05.2024)

16. Веб-сайт draw.io URL: https://app.diagrams.net/ (дата звернення: 14.05.2024)

17. Веб-сайт Figma URL: https://www.figma.com/ (дата звернення: 17.05.2024)

18. Веб-сайт goldwebsolutions.com. URL: https://goldwebsolutions.com/uk/blog/shho-take-vajrfrejm-wireframe-ta-dlya-chogo-vin-potriben-u-protsesi-rozrobki-sajtu-chi-dodatku/ (дата звернення: 19.05.2024)

19. Документація SQL-ін’єкція URL: https://foxminded.ua/sql-iniektsii/ (дата звернення: 21.05.2024)

20. Документація XSS. URL: https://ukeywaf.com/baza/shho-take-xss-ataka-yak-vona-praczyuye1/ (дата звернення: 23.05.2024)

метадані

Заголовок

Сайт обслуговування по спеціалізації відеомонтажу засобами Ruby on Rails

Автор

Огородник С. І. Науковий керівник / Експерт

підрозділ

King Danylo University

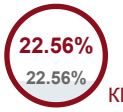
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

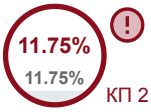
Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	141

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КП 2

25
Довжина фрази для коефіцієнта подібності 2

8366
Кількість слів

63938
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	http://repository.ukd.edu.ua/bitstream/handle/123456789/391/%D0%9F%D0%B0%D1%85%D0%BE%D0%BB%D1%8C%D1%87%D1%83%D0%BA%20%D0%9E.%D0%A0.%20%D0%B4%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0.pdf?sequence=1	154	1.84 %
2	http://repository.ukd.edu.ua/bitstream/handle/123456789/391/%D0%9F%D0%B0%D1%85%D0%BE%D0%BB%D1%8C%D1%87%D1%83%D0%BA%20%D0%9E.%D0%A0.%20%D0%B4%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0.pdf?sequence=1	88	1.05 %
3	http://repository.ukd.edu.ua/bitstream/handle/123456789/391/%D0%9F%D0%B0%D1%85%D0%BE%D0%BB%D1%8C%D1%87%D1%83%D0%BA%20%D0%9E.%D0%A0.%20%D0%B4%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0.pdf?sequence=1	58	0.69 %