ПВНЗ Університет Короля Данила

Кафедра Інформаційних технологій та програмної інженерії

УДК 004

ДИПЛОМНА РОБОТА

Тема ***Інформаційно-довідковий web-ресурс з проведення фестивалю Franek Shop***

Напрям підготовки

(код і назва спеціальності

Студент

Соболь Ю.М.

(підпис) (дата) (розшифрування підпису)

Керівник проекту

к.т.н. *Мельничук С.І.*

(посада) (підпис) (дата) (розшифрування підпису)

к.т.н. *Мануляк І.З.*

Нормоконтроль

(посада) (підпис) (дата) (розшифрування підпису)

Допускається до захисту

Завідувач кафедри

д.т.н.,доц. *Мельничук С.І.*

(посада) (підпис) (дата) (розшифрування підпису)

					ДР.ПІс– 20.00.000 ПЗ		
Зм.	*Лист*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		*Соболь Ю. М.*			*Інформаційно-довідковий web-ресурс з проведення фестивалю Franek Shop*	Літ.	Арк. 6 / Аркушів 95
Перевір.		*Мельничук С. І.*					
Реценз.						*УКД, ПІс – 2015*	
Н. Контр.		*Андрейко В. М.*					
Затверд.		*Мельничук С. І.*					

ЗМІСТ

ВСТУП

Для людей, які потребують комунікації на відстані, інтернет є ідеальним засобом, так як дає змогу обмінюватись інформацією в режимі реального часу, а також завдяки своїй доступності обсягом цієї інформації можуть бути практично нескінченними. Мережа дає можливість з мінімальними витратами проінформувати цільову аудиторію про певну подію. Крім того, перевагами інтернету є те, що він дозволяє передавати текстову, графічну, аудіо та відео інформацію, а також оцінювати ефективність заходів за рахунок зворотного зв'язку із цільовою аудиторією.

Зовнішній вигляд сайту є надзвичайно важливим, оскільки, насамперед, сторінки сприймаються візуально. Великі фрагменти тексту виглядають нудно, їх важко читати, вони вимагають від користувача більшого розумового напруження, ніж альтернативні методи подання інформації. Також важливо вміти правильно структурувати дані на сторінці, щоб важлива інформація була першочерговою.

Структура сайту є найважливішим технічним інструментом з точки зору SEO. Неправильне побудова структури сайту значно ускладнює просування користувачів по сайту. Тому при розробці архітектури ресурсу, необхідно аналізувати розміщення кожного розділу і підрозділу, щоб все зробити грамотно, задовольнивши потреби користувача і відповівши на вимоги пошукових роботів.

Сайт, який здійснює інформування через Інтернет, зможе швидко та якісно ознайомити користувачів з подією.

1 ОГЛЯД ЗАСОБІВ РЕАЛІЗАЦІЇ ВЕБ-ЗАСТОСУНКУ

1.1 Аналіз інформаційних розважальних порталів

Люди та компанії створюють сайти для різних цілей: щоб продавати товари та послуги, розміщувати й знаходити інформацію, здобувати знання, спілкуватися з іншими користувачами, розважатися тощо.

Компанії сайт згодиться, щоб створити репутацію, знайти нових клієнтів, збільшити продажі. На корпоративному сайті можна розповісти про діяльність компанії та її співробітників, представити асортимент товарів або послуг, провести рекламну кампанію чи відповісти на запитання користувачів.

Приватній особі сайт згодиться, щоб знайти роботу або привабити нових клієнтів. На особистому сайті можна поділитися професійним досвідом, прорекламувати себе чи свій бізнес, продавати товари чи послуги онлайн [11].

Основними завданнями веб-сайту є:

– реклама продукції, послуг, ідей. Правильно зроблений веб-сайт із легкістю приведе клієнта до висновку про необхідність покупки товару, або послуг, або ідей, що пропагуються на ньому;

– продаж товарів, послуг, інформації, ідей. У сучасної людини немає багато часу для ходіння по магазинах. Тому можливість замовлення товарів і послуг, не відходячи від комп'ютера, значно розширює можливості і клієнта, і продавця;

– безкоштовне надання інформації або послуг. Насправді надання інформації або послуг — це засіб залучення відвідувачів до даного ресурсу для здобуття, наприклад, статистичної інформації або ж для показу реклами, якщо це рекламний майданчик;

– підтримка клієнтів.

Відповідно до завдань існують такі типи веб-сайтів:

– рекламні веб-сайти;

– веб-сайти-продавці;

– веб-сайти-альтруїсти;

– веб-сайти для підтримки.

Складові веб-сайту:

– дизайн сайту;

– контент сайту;

– код сайту.

На сьогоднішній день існують кілька етапів розробки веб-сайту, виділимо з них основні:

– проектування сайту;

– створення макетів сторінок;

– верстка сторінок і дизайнів;

– програмування;

– тестування та внесення коригувань;

– відкриття проекту на хостингу;

– обслуговування сайту.

Визначення структури сайту містить у собі планування розділів, системи навігації по сайті. Як буде виглядати меню сайту, де і як будуть розташовані ті або інші модулі. Відповіді на ці питання дозволяють уже на етапі проектування представити, як буде виглядати майбутній сайт [15].

На проектуванні сайту визначаються всі завдання, які ставляться перед командою, що розробляє проектування або, як ще кажуть, прототипування сайту. В результаті командної роботи, замовник отримує готовий прототип (макет) сайту, із зручною для користувача навігацією, продуманою структурою і концепцією дизайну, також повне ТЗ (технічне завдання) [13].

Макет показує, як будуть виглядати різні сторінки, де буде розташовуватися інформація, де буде перебувати навігаційні й керуючі елементи і т. і. Нарешті, на основі макета проробляється зовнішній вигляд

кожної категорії сторінок. Наприклад, розробляється дизайн сторінок з новинами, дизайн сторінок з інформацією про продукт і т.і. Фактично дизайнер вивчає технічне завдання і малює шаблони для всього, що там описано.

Інтерфейс сайту, який розроблений дизайнером, – це ще тільки макет остаточного інтерфейсу сайту. Фактично, він складається просто з набору малюнків. Для того, щоб його можна було використати в програмному продукті, потрібно провести верстку – розрізати макет інтерфейсу на складові його графічні компоненти і описати правила розташування всіх цих елементів на сторінці [16].

Структура сайту є найважливішим технічним інструментом з точки зору SEO. Неправильне побудова структури сайту значно ускладнює просування користувачів по сайту. Тому при розробці архітектури ресурсу, необхідно аналізувати розміщення кожного розділу і підрозділу, щоб все зробити грамотно, задовольнивши потреби користувача і відповівши на вимоги пошукових роботів [25].

На жаль, чіткого визначення якою має бути правильної структура не існує. Вона залежить від виду сайту, семантичного ядра і цільової аудиторії, тому завжди індивідуальна. Однак існують рекомендаційні типи структур, а також основні правила по її розробці.

Структура сайту – це логічна побудова всіх сторінок ресурсу. Схема, за якою розподіляється шлях до папок, категорій, підкатегорій. З технічної точки зору, навігація ресурсу являє собою набір URL-ів, логічно вибудуваних в певній послідовності (рисунок 1.1). Структура взаємопов'язана з семантичним ядром. Саме воно говорить про те, які папки і документи повинні бути присутніми на сайті. Тому, зібравши семантику, вже можна зробити начерки схеми побудови кожного майбутнього URL-а [1].

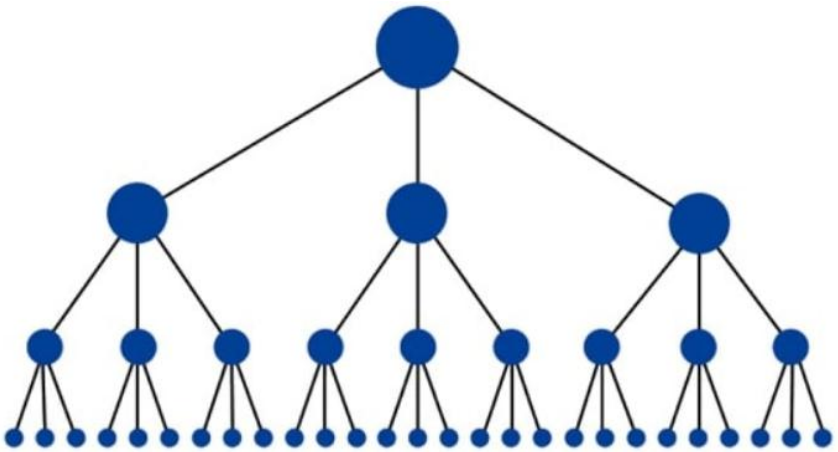


Рисунок 1.1 – Загальний вигляд структур веб-сайтів

Пошукові роботи проходять по посиланнях сайту. І чим правильніше, простіше буде складена структура сайту, тим менше пошуковий робот витрачає свої ресурси і швидше проведе обхід ресурсу. Це призведе до найбільш швидкої індексації сайту. Більш того, правильно і логічно розміщені документи на сайті можуть потрапити в індекс пошукової системи вже на наступний день. Отже, структура безпосередньо впливає на ранжування. Чим вона краще, тим швидше відбудеться індексація. Це є невід'ємним фактором для SEO-просування [17].

Пошукова система проводить аналогію між сайтом і ставленням користувачів до нього. Чим привабливіший ресурс для споживачів, тим він привабливіший і для пошукових систем. Якщо алгоритм Google бачить, що сайт має низький час перебування відвідувача на ньому, поганий показник коефіцієнта кліків, то він сприйме ресурс, як неякісний, що дуже погано позначиться на просуванні і пошуковій видачі веб-сайту.

Правильна структура фокусує користувача на головних діях на сайті. Вона виключає складність пошуку людиною потрібної йому сторінки. У зворотному випадку, це безповоротно знизить конверсію, так як користувач не знайде те, що шукає, і, відповідно, не стане клієнтом.

Також бувають ситуації, коли розробник не зазнає ніяких проблем з кодом, сайт нормально функціонує, але не приносить бажаних результатів у

вигляді грошової прибутку, відвідувань і т. д. В таких випадках, як правило, починають шукати недоліки в контенті, кількості зворотних посилань і сильних конкурентів.

Необхідність структурування сайту (як зовнішнього, так і внутрішнього) відчули багато веб-розробники і SEO-оптимізатори. Часто до такого висновку приходять через сумний досвід, і щоб отримувати якомога менше, слід позначити основні причини дотримання грамотної організації сайту.

Продумуючи внутрішню і зовнішню структуру сайту, необхідно враховувати вимоги і переваги, як людей, так і пошукових роботів. Слідуючи простим рекомендаціям, можна серйозно підняти популярність свого ресурсу.

Якщо є семантичне ядро, і вся інформація, яка буде задіяна на ресурсі, то повинна вибудовуватися структура сайту у вигляді схеми. Іншими словами, ієрархія, логічний і послідовний ланцюжок побудови і подачі інформації.

Грамотна структура сайту допомагає пошуковим системам індексувати всю інформацію, а відвідувачам – відчувати себе комфортно, користуючись ресурсом.

Перевірити структуру свого або чужого сайту можна самостійно, «пройшовшись» по всім сторінкам або вивчивши меню. Однак набагато зручніше скористатися спеціально призначеними для такої перевірки сервісами. Вони забезпечать наочну подачу інформації, а також забезпечать додаткові відомості про досліджуваний ресурсі.

Якщо структура веб-сайту добре організована, а навігація інтуїтивна, відвідувачі з легкістю знайдуть шлях до дій, які ви від них очікуєте

З точки зору ієрархії структура ділиться на рівні, починаючи з першого. Першим рівнем буде головна сторінка і основні категорії, другим – підкатегорії і так далі. Ієрархічна побудова структури дозволить користувачеві зручніше і швидше знайти шуканий їм товар. Наприклад, користувач шукає електричний чайник у великому онлайн-магазині електроніки [18].

Шлях пошуку чайника зображено на рисунку 1.2.

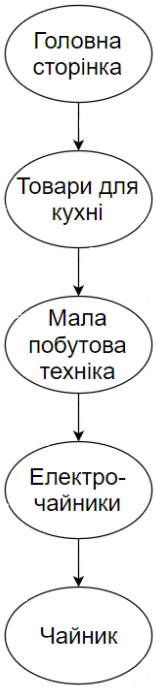


Рисунок 1.2 – Структура пошуку товару

Шлях від головної сторінки до товару займає всього 3 кліки, що максимально спрощує споживачеві пошук продукту. Загальний вигляд чотирьохрівневої структури сайту зображено на рисунку 1.3.

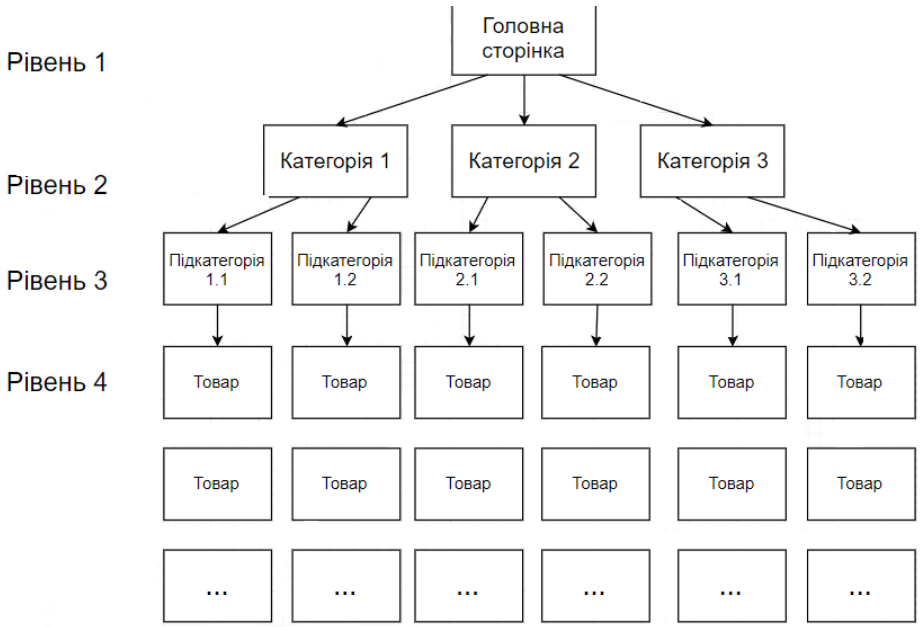


Рисунок 1.3 – Чотирьохрівнева структура сайту

Однак подібний тип структури сайту не завжди доречний, так як він безпосередньо залежить від цілей сайту. Виходячи з них, можна вибрати відповідний організацію структури з декількох існуючих.

1.2 Огляд сайтів-аналогів інформаційного супроводу розважальних заходів

Для того, щоб чітко розуміти, що потрібно реалізувати, потрібно оглянути існуючі рішення. На даний час існує безліч веб-ресурсів, які висвітлюють інформацію про конференції, фестивалі і т. д. . Сайти містять, як і просто необхідну інформацію про поточні події, а їй про попередні та заплановані. Також більшість ресурсів містять фотогалереї та відеоматеріали з подій.

Для огляду було обрано сайти Івано-Франківських конференцій Design Village та IT Rally, тому що аудиторія сайту Franek Shop може складатися з аудиторії названих подій.

Сайт Design Village є односторінковим, тобто увесь необхідний код - HTML, JavaScript, та CSS - завантажується разом зі сторінкою, або динамічно довантажується за потребою, зазвичай у відповідь на дії користувача. Тому сайт буде розглянути по блоках (екрани).

Екран заголовку (рисунок 1.4) складається з наступних блоків :

- меню (заховане під написом «Меню»);
- посилання на соціальні мережі;
- перемикач мов;
- зображення з інформацією про подію;
- посилання на купівлю квитків.

Рисунок 1.4 – Екран заголовку сторінки

Меню при розгортанні має вигляд, який зображено на рисунку 1.5 складається з наступних пунктів:

– квитки;
– програма;
– спікери;
– воркшопи;
– виставки;
– portfolio review;
– локація;
– сувенірна лавка;
– партнери.

Меню є зручним та зрозумілим у використанні. Відвідувач сайту легкістю знайде всю необхідну інформацію про конференцію та придбати квитки.

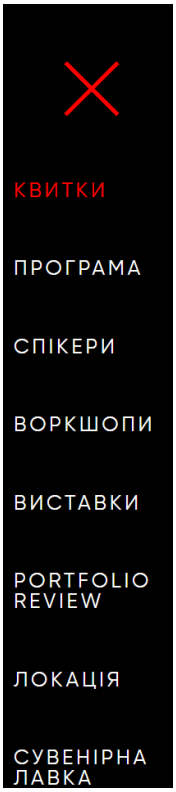


Рисунок 1.5 – Меню сайту

Інформаційний екран (рисунок 1.6) складається з блоку інформаційного тексту про подію та посилання на купівлю квитків. Також елементи меню, перемикача мов та посилання на соцмережі, відображаються на усіх екранах.

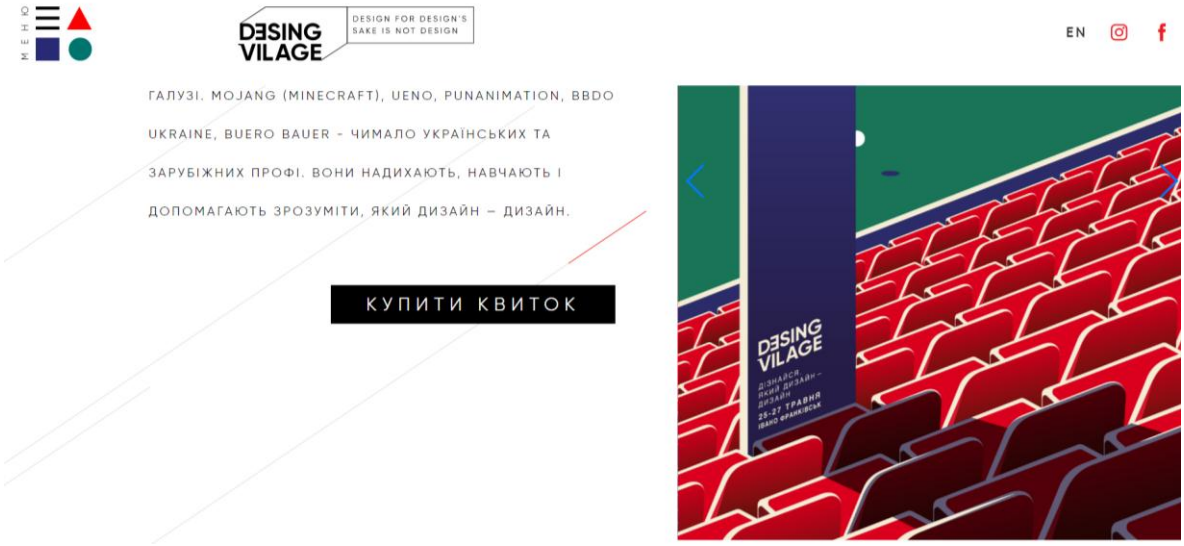


Рисунок 1.6 – Інформаційний екран

Наступні кілька екранів відображають програму події (рисунок 1.7). Зображається дата, час, тема доповіді та спікер.

Рисунок 1.7 – Екран програми події

Екран інформації про спікерів (рисунок 1.8) відображає список спікерів та конкретну інформацію про кожного. Ім’я спікера, фото, опис діяльності та посилання на соцмережі.

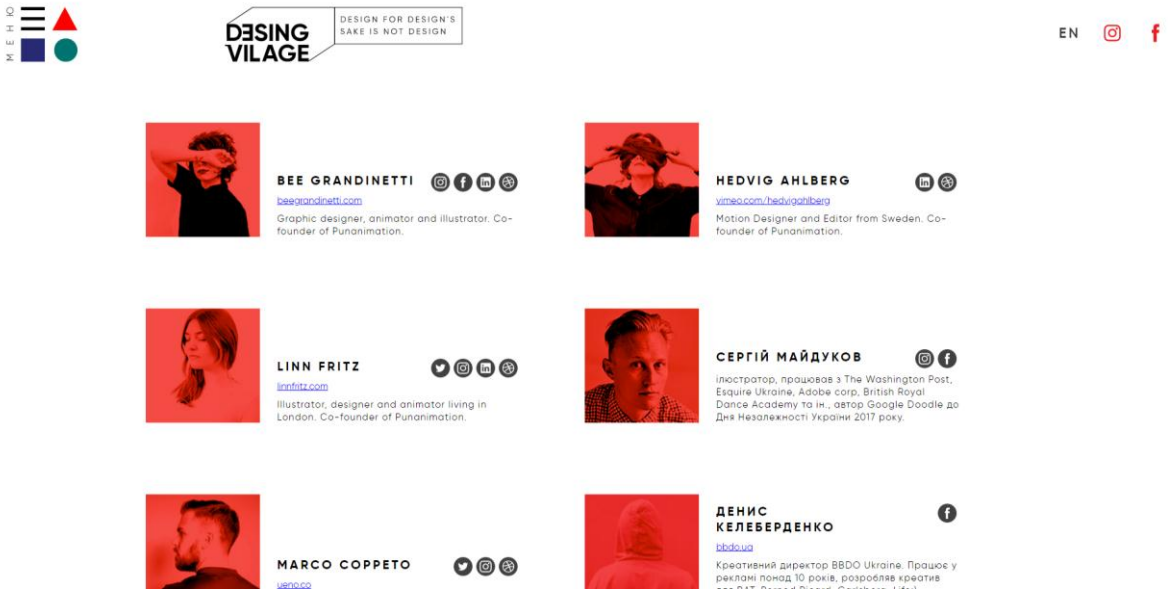


Рисунок 1.8– Екран інформації про спікерів

Екран воркшопів містить список воркшопів (рисунок 1.9) та відображає наступну інформацію про воркшопи:

– дата;
– час;
– тривалість;
– кількість місць;
– ціна;
– посилання на подію facebook;
– назва;
– хто проводить воркшоп;
– вартість.

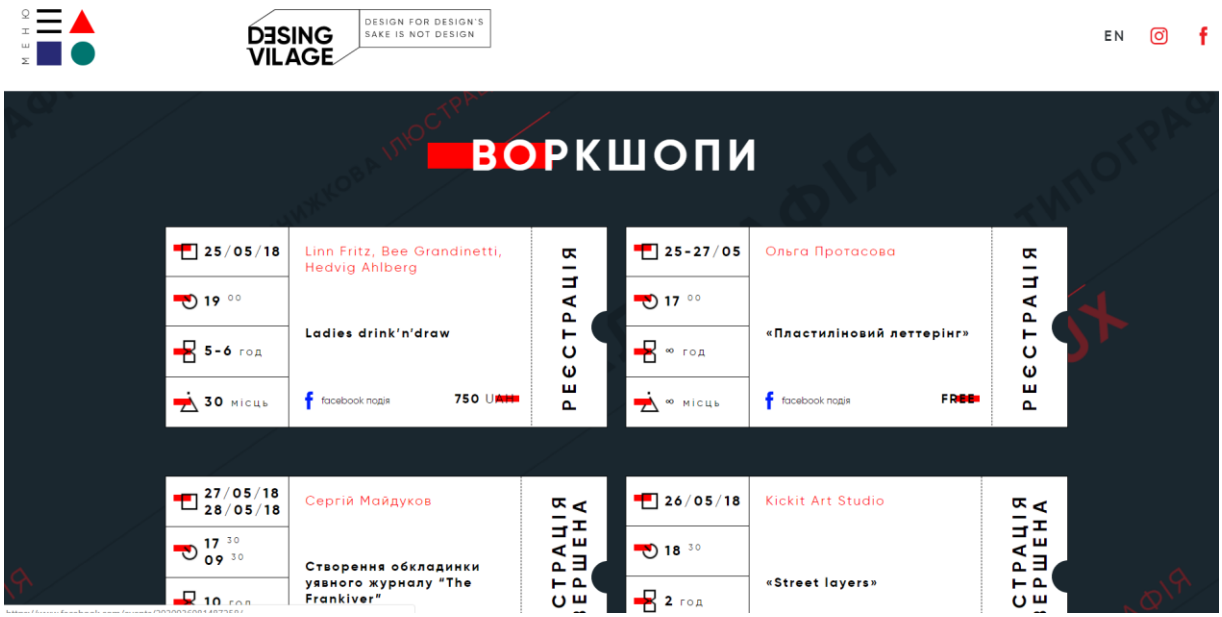


Рисунок 1.9 – Екран воркшопів

Екран виставок зображено на рисунку 1.10. Відображається список виставок, про кожну виставку подана наступна інформація:

– фото;
– посилання на подію у facebook;
– назва;

– автор;

– локація.

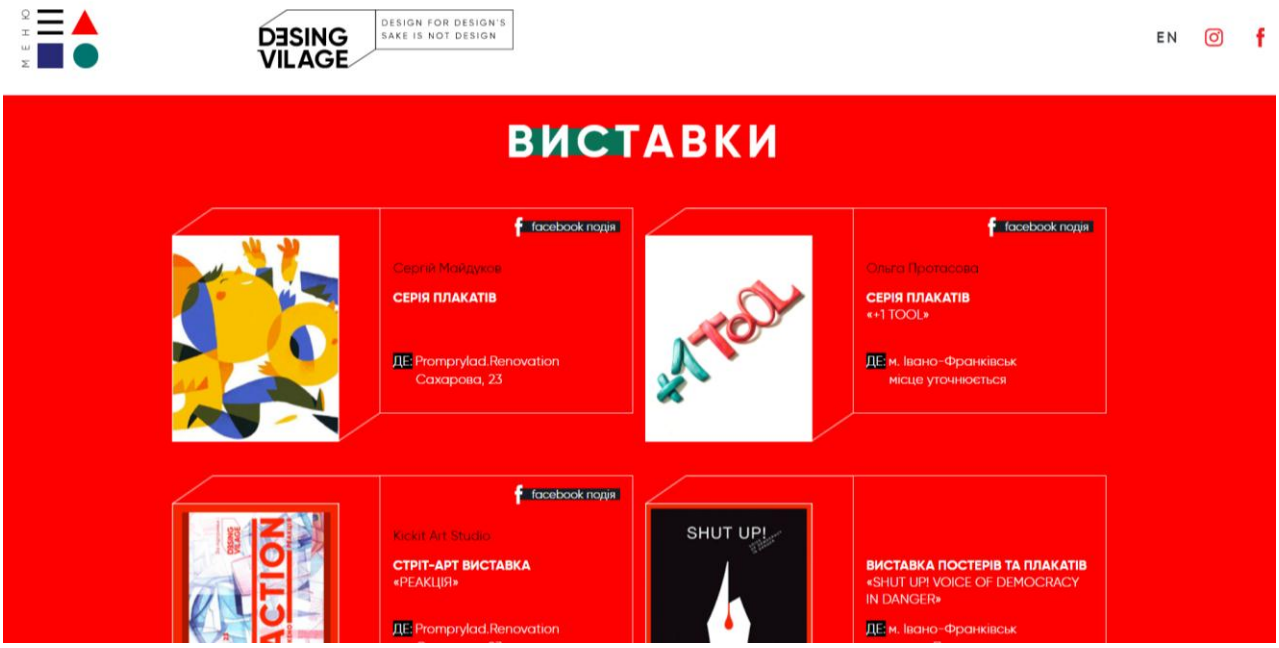


Рисунок 1.10 – Екран виставок

Також важливим є екран зі списком партнерів (рисунок 1.11). На ньому зображені логотипи партнерів.

Рисунок 1.11 – Екран партнерів

Також є футер з стандартною інформацією про сайт. Рік створення, контактна інформація та хто створив.

Сайт IT Rally є багатосторінковим, тому кожна сторінка буде розглядатись окремо. Головна сторінка сайту складається з наступних блоків (меню на головній сторінці складається з таких самих блоків):

– організовуємо;
– організатори;
– попередні;
– партнери.

Блок «Організовуємо» (рисунок 1.12) складається з двох останніх доданих подій, це можуть бути події, що відбудуться або вже відбулись.

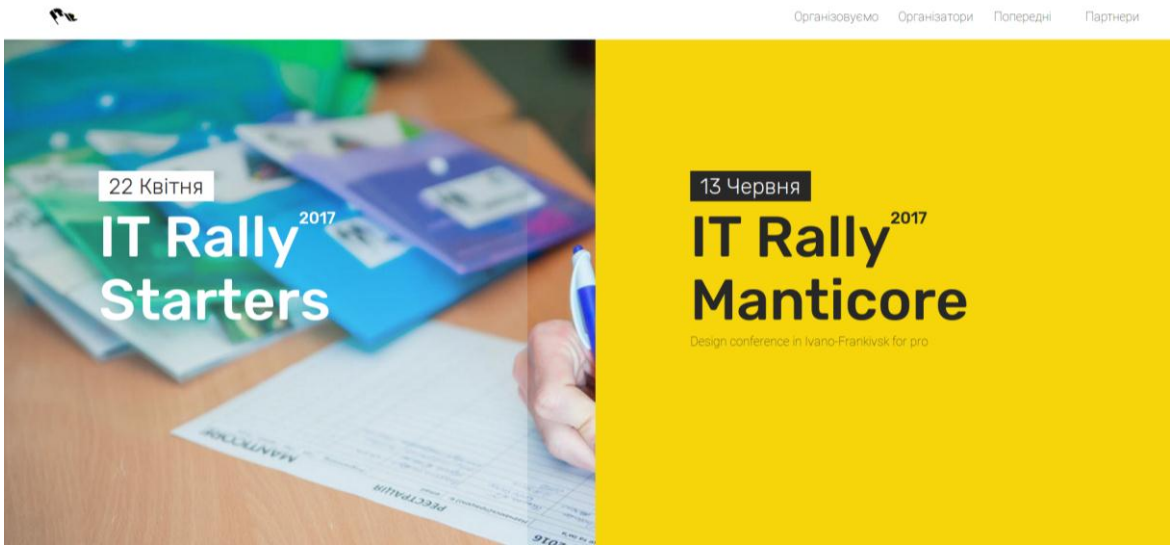


Рисунок 1.12 – Блок «Організовуємо»

При настисканні на одну з подій відбувається перехід на неї. Блок, що містить інформацію про організаторів та співорганізаторів, скаладється з списку фотографій та назв організаторів. Така інформація є дуже корисною для людей, які хочуть взяти участь в події. Вони зможуть дізнатись хто є організаторами та вияснити всі деталі та умови конференції у них. (рисунок 1.13).

Рисунок 1.13 – Блок «Організатори»

Блок «Попередні конференції» (рисунок 1.14) складається з статистичної інформації про конференції нанесуну на хронологічну шкалу. Про подію відображається наступна інформація:

– назва;
– фото;
– кількість спікерів;
– кількість відвідувачів;
– рік.

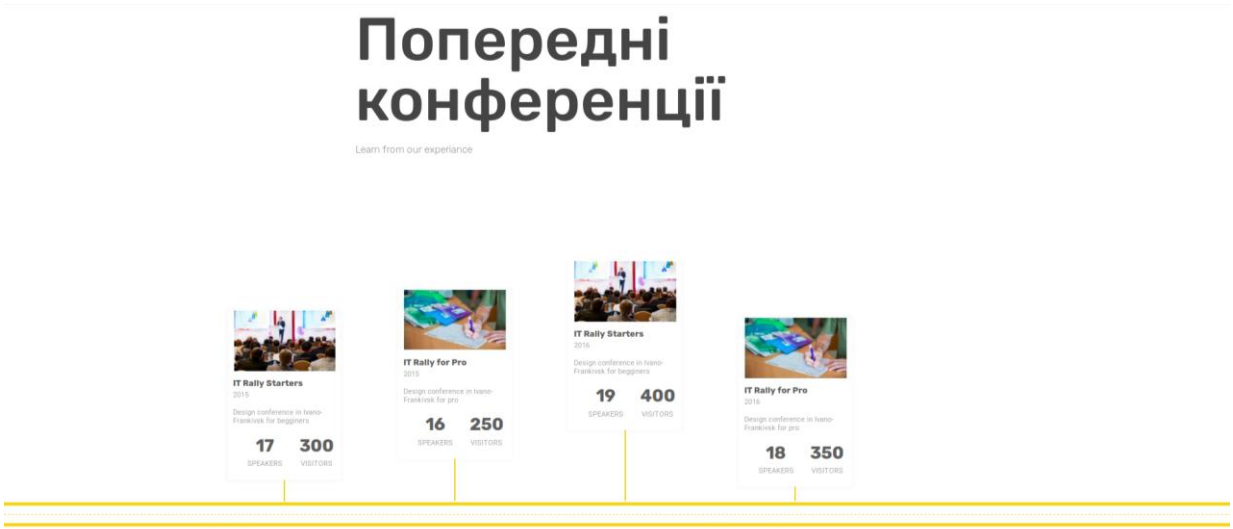


Рисунок 1.14 – Блок «Попередні конференції»

Блок зі списком партнерів (рисунок 1.15). На ньому зображені логотипи партнерів.

Рисунок 1.15 – Блок «Партнери»

Також є футер з контактною інформація.

Сторінка конкретної конференції складається з наступних блоків:

– заголовок з інформацією про конференцію;
– спікери;
– розклад;
– команда.

Варто звернути увагу на блок, який містить інформацію про команду, так як попередні блоки схожі з інформацією, яка відображається на сайті Design Village.

Блок «Команда» (рисунок 1.16) складається зі списку організаторів та волонтерів, які допомагають у проведенні конференції.

Про них відображається наступна інформація:

– фото;
– прізвище та ім’я;
– посилання на сторінки в соцмережах.

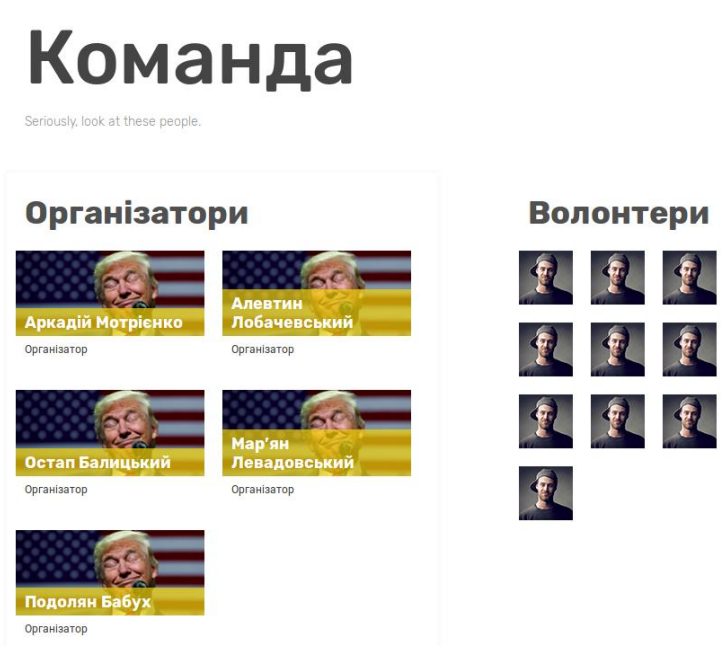


Рисунок 1.16 – Блок «Команда»

Користувач відвідавши сайт має швидко отримати інформацію про подію яка його цікавить. Також сайт повинен володіти хорошою структурою, яка значно спрощуватиме просування користувачів по сайту. Тому при розробці, необхідно аналізувати розміщення кожного розділу і підрозділу, щоб все зробити грамотно, задовольнивши потреби користувача і відповівши на вимоги пошукових роботів.

В оглянутих сайтах відсутні форми для зв'язку з користувачами, що є недоліком, так як сайт не можу забезпечити комунікацію з кінцевим користувачем. На сайті Design Village на відміну від IT Rally відсутня інформація про попередні конференції, це не дає користувачу більш повної інформації про події.

Також відчутним недоліком є відсутність фото галереї та відео з подій. Не реалізований функціонал реєстрації на подію, ні для волонтерів ні для учасників. Але більшість схожих блоків, таких як: «розклад», «спікери» та «партнери» є корисними та функціональними та потребують інтеграції у сайті Franek Shop.

1.3 Вибір технологій розробки

Створити сайт можна як за допомогою CMS (система керування вмістом) так і за допомогою мови розмітки HTML, мови для опису зовнішнього вигляду сторінок CSS та інших веб-орієнтованих мов програмування. Для опису використовується спеціалізована мова HTML мова розмітки гіпертекстових документів. Готові зверстані HTML шаблони далі використовуються в наступних етапах реалізації проекту [19].

На цій стадії графічна картинка нарізається на окремі елементи і з використанням технологій HTML і CSS трансформується в код, який можна переглядати за допомогою браузера.

HTML - це не мова програмування. Це форма збереження даних. Мова програмування відрізняється від мови розмітки тим, що мова програмування програмує дещо на виконання деяких дій, а мова розмітки готує певним чином деякий документ для того, щоб абияка програма могла його використовувати.

Таким чином, створивши HTML-документ, він передається програмі (здебільше, HTML-документи використовуються для створення веб-сайтів, отже, зазвичай, програма для обробки HTML-документа є браузер), що обробляє його згідно правил, що в неї запрограмовані. Отже, коли браузер читає HTML-документ, то він малює на екрані його вміст, керуючись правилами, що в нього закладені для показу файлів, що розмічені мовою HTML [12].

Елементи являють собою базові компоненти розмітки HTML. Кожен елемент має дві основні властивості: атрибути та зміст (контент). Існують певні настанови щодо кожного атрибута та контент елемента, які треба виконувати задля того, щоб HTML-документ був визнаний валідним.

У елемента є початковий тег, який має вигляд <element-name>, та кінцевий тег, який має вигляд </element-name>. Атрибути елемента

записуються в початковому тегу одразу після назви елемента, контент елемента записується між його двома тегами [21].

Як і HTML, CSS не є справжньою мовою програмування. Це лише мова таблиць стилів, яка дозволяє задавати стилі обраним елементам у HTML документах.

Каскадна або блочна верстка прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки — розділення змісту сторінки (даних) та їхньої візуальної презентації [20].

CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг — можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS) [6].

Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш структурованим та простим, прибрати повтори тощо. CSS також дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою (КПК), у роздрукованому вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових браузерах та ін.).

Selector (Селектор) — назва елементу HTML на початку правила. Селектор вибирає елемент чи елементи, які будуть стилізовані (у нашому випадку, елементи p).

Declaration (Визначення) — одне правило на зразок color: red; вказує, яку з властивостей елементу ви бажаєте стилізувати.

Properties (Властивості) — шляхи, якими ви можете стилізувати даний HTML елемент. (У цьому випадку, color — це властивість елементів p).

JavaScript — кроссплатформна об'єктно-зорієнтована мова сценаріїв (скриптів). Рушій JavaScript підключається до об'єктів свого середовища виконання (зазвичай, веб-переглядача) та надає можливість керування ними [5].

Мова JavaScript має стандартну бібліотеку об'єктів (таких як Array, Date та Math) і основний набір елементів мови програмування, таких як оператори, керівні структури та вирази.

Ядро JavaScript може бути розширене для різних потреб шляхом доповнення його додатковими об'єктами. Наприклад:

На стороні клієнта JavaScript розширює ядро мови, додаючи об'єкти керування переглядачем і його об'єктною моделлю документа — Document Object Model (DOM). Наприклад, клієнтські розширення дозволяють застосункам розміщувати елементи на HTML-формі та реагувати на дії користувача, такі як клацання миші, введення даних у форму і пересування сторінками [6].

На стороні сервера JavaScript розширює ядро мови шляхом додавання об'єктів, що стосуються роботи JavaScript на сервері. Наприклад, серверні розширення дозволяють застосункам взаємодіяти з базою даних, забезпечувати безперервність потоку інформації від одного запущеного застосунку до іншого, або виконувати маніпуляції з файлами на сервері.

JavaScript стандартизований Європейською асоціаіцєю з стандартизації інформаційних та комунікаційних систем — Ecma International (спершу складноскорот ECMA позначав European Computer Manufacturers Association). Ця стандартизована версія JavaScript, що має назву ECMAScript, однаково працює у всіх застосунках, що підтримують стандарт. Різні компанії можуть використовувати цей відкритий стандарт для розробки своїх реалізацій JavaScript [14].

Одним із архітектурних шаблонів програмного забезпечення **є** архітектура клієнт-сервер. Клієнт-сервер є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними.

Основні компоненти:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Найчастіше веб-сервер і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі.

Схожою з клієнт-серверною трьохрівневою архітектурою є шаблон MVC (рисунок 1.17).

MVC — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення. Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача [22].

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності [8].

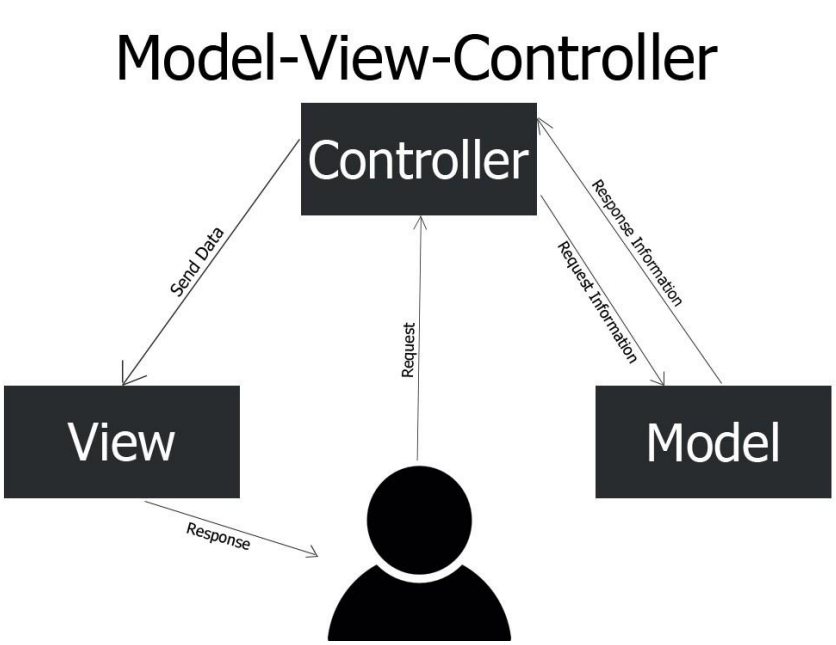


Рисунок 1.17 – MVC

Так як потрібно просто реалізувати сайт з великою швидкодією, для цього пригодиться мова програмування Ruby. Ruby інтерпретована, повністю об'єктно-орієнтована мова програмування з чіткою динамічною типізацією.

Мова вирізняється високою ефективністю розробки програм і увібрала в себе найкращі риси Perl, Java, Python, Smalltalk, Eiffel, Ada і Lisp. Ruby поєднує в собі Perl-подібний синтаксис із об'єктно-орієнтованим підходом мови програмування Smalltalk. Також деякі риси запозичено із мов програмування Python, Lisp, Dylan та CLU. Багатоплатформова реалізація інтерпретатора мови Ruby поширюється як Вільне програмне забезпечення. Мова Ruby дозволяє використовувати нотацію, що легко читається, для визначення семантики веб-застосунків [23].

Для розробки додатку буде використано об'єктно-орієнтований програмний каркас (фреймворк) для створення веб-додатків, написаний на мові програмування Ruby. Цей каркас має назву Ruby on Rails.

Ruby on Rails повністю задовольняє усі вимоги, як поставлені у задачі, так і які диктуються світовими стандартами розробки програмного забезпечення.

Дані, які будуть опрацьовуватись сайтом необхідну організовувати. Сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами називається базою даних. В загальному випадку базою даних можна вважати будь-який впорядкований набір даних [24].

Бази даних класифікують за різними критеріями. За моделлю організації даних розрізняють такі бази даних:

– Ієрархічна. Ієрархічна база даних може бути представлена як дерево, що складається з об'єктів різних рівнів. Між об'єктами існують зв'язки типу «предок-нащадок». При цьому можлива ситуація, коли об'єкт не має нащадків або має їх декілька, тоді як у об'єкта-нащадка обов'язково тільки один предок.

– Мережна. Така база даних подібна до ієрархічної, за винятком того, що кожен об'єкт може мати більше одного предка.

– Реляційна. Реляційна база даних зберігає дані у вигляді таблиць. Найвживаніші СКБД використовують реляційну модель даних.

– Об'єктно-орієнтована. У базі даних цього виду дані оформляють у вигляді моделей об'єктів.

Для розробки сайту підійде база даних об'єктно-орієнтована типу. Тому що об'єктно-орієнтована база даних зазвичай рекомендовані для тих випадків, коли потрібна високопродуктивна обробка даних, що мають складну структуру. Об'єктно-орієнтованою базою даних, яка працює з Ruby on Rails є MongoDB [7].

Отже для розробки обраний наступний стек технологій:

– HTML;

– CSS;

– JavaScript;

– Ruby;

– Ruby on Rails;

– MongoDB.

1.4 Постановка задачі

Метою розробки є створення сайту, який буде швидко і детально інформувати користувача сайту про фестивалі Franek Shop. Користувачі зможуть дізнатись коротку історію створення фестивалю, дати та місця проведення та придбати квитки на подію. А також реалізувати функціонал для розміщення фото, відео та форм реєстрації.

Сайт повинен бути розробленим за допомогою стандартних методів та технологій веб-розробки. Основними етапами розробки повинні бути:

– створення макетів сторінок;

– верстка сторінок і дизайнів;

– програмування, розробка функціональних інструментів (для веб на стороні клієнта і сервера).

Забезпечити швидке завантаження сторінок сайту при великій кількості запитів, для максимальної взаємодії з користувачем.

Інтерфейс повинен бути максимально простим, та не має містити багато елементів. Якщо структура веб-сайту добре організована, а навігація інтуїтивна, відвідувачі з легкістю знайдуть шлях до дій, які ви від них очікуєте. В даному випадку, це є покупка квитка та реєстрація себе як учасника або волонтера. На сайті має бути максимально ефективне управління: просте, швидке і зручне.

Для роботи ресурсу не потрібні додаткові встановлення програмного забезпечення, або конфігурації браузера. Сайт повинен бути максимально простий у користуванні. Відвідувач сайту має швидко та чітко зрозуміти всю подану інформацію. Для сайту фестивалю це є дуже важливим критерієм, так як від цього залежить кількість відвідувачів. Неправильно структуризовані сайті призводять до того, що користувач просто не може знайти потрібні дані.

Сайт повинен містити наступні блоки та сторінки:

– заголовок;
– інформаційний блок;
– окремі сторінки для кожного фестивалю;
– список попередніх фестивалів;
– партнери;
– команда;
– організатори;
– розклад;
– список учасників;
– фотогалереї для конкретного фестивалю;
– відео для конкретного фестивалю.

Також для спілкування з користувачами повинні бути реалізовані наступні форми:

– форма реєстрації волонтера;
– форма реєстрації учасника;
– форма реєстрації відвідувача;
– форма зворотнього зв’язку.

Для роботи з контентом сайту потрібно реалізувати адміністравну панель з наступними функціями:

– CRUD операції;
– автоматична валідація форм;
– пошук та фільрування;
– експорт даних;
– аутентифікація;
– авторизація;
– історія дій користувача.

2 РОЗРОБКА СТРУКТУРИ САЙТУ ТА ФУНКЦІЇ РЕЄСТРАЦІЇ КОРИСТУВАЧІВ

2.1 Розробка структури сайту

Сайт складається з інформативних сторінок та сторінок на яких користувач зможе здійснювати певні дії, наприклад зареєструватись як волонтер (рисунок 2.1). До інформативних сторінок належать наступні сторінки:

– головна сторінка;

– сторінка з інформацією про події.

До сторінок дій належать:

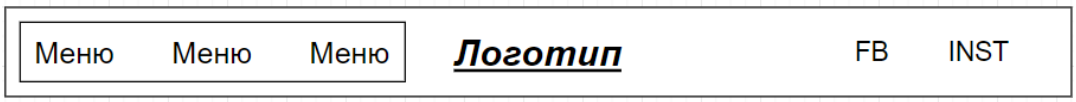
– сторінки адмінпанелі сайту;

– сторінка реєстрації волонтера (рисунок 2.11);

– сторінка реєстрації учасника (рисунок 2.12).

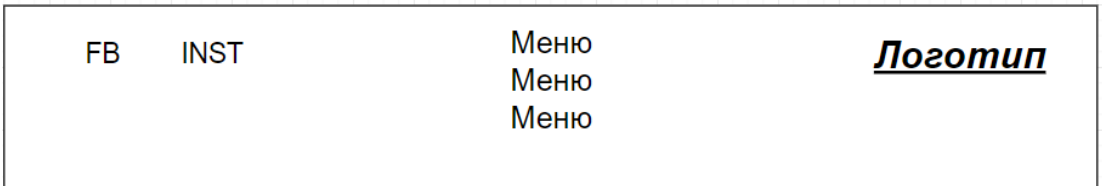
Рисунок 2.1 – Загальна структура сторінок сайту

Також на усіх сторінках крім адмінпанелі буде відображено навігаційне меню сайту разом з посиланнями на соцмережі та логотипом (рисунок 2.2).



Риунок 2.2 – Меню сайту

Ще однми елементом, який буде відображено на усіх сторінках крім адмінпанелі є футер. У футері буде розміщено посилання на соцмережі, меню та логотип (рисунок 2.3).



Риунок 2.3 – Футер сайту

На головній сторінці буде розміщено кілька інформаційних блоків:

– заголовок та анонс події (рисунок 2.4);
– попередні події (рисунок 2.5);
– партнери (рисунок 2.6).

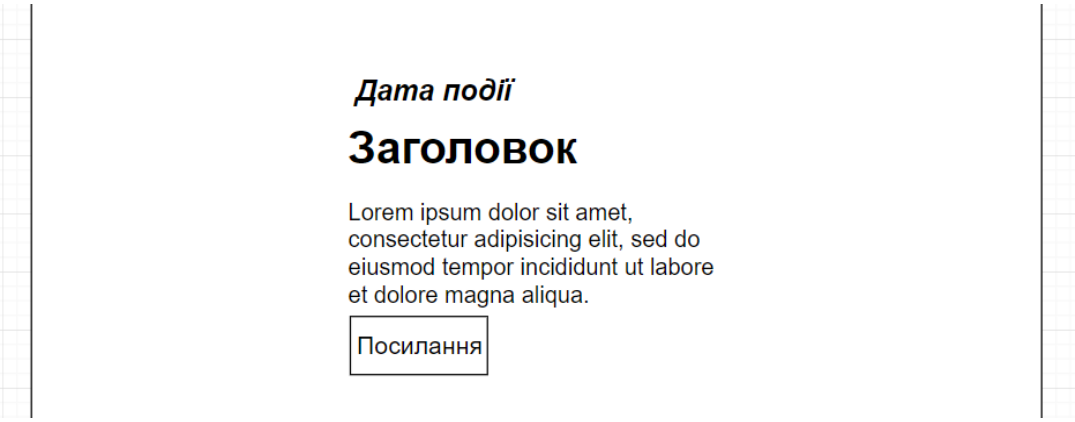


Рисунок 2.4 – Блок заголовку та анонсу

Блок заголовку та анонсу міститиме:

– короткий опис події;

– заголовок;

– дату;

– посилання на подію, якщо вона відбулась.

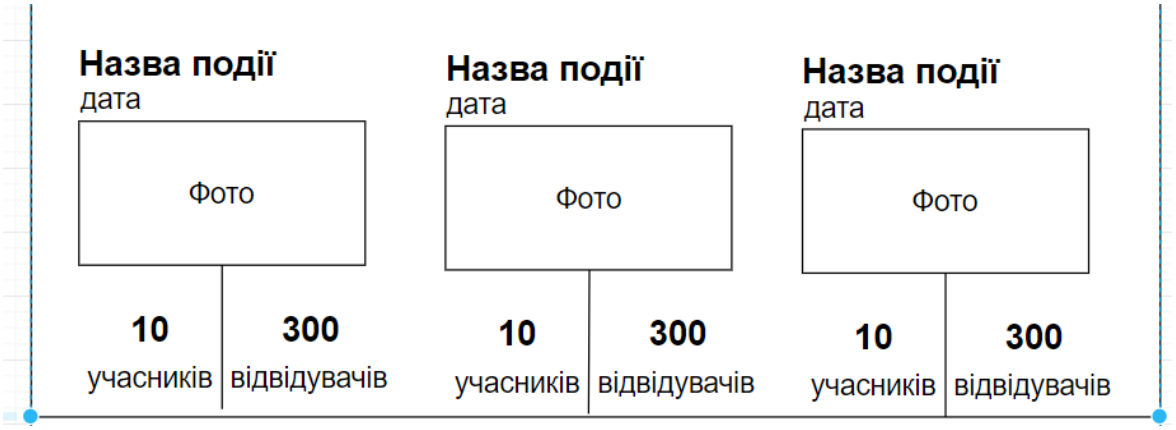


Рисунок 2.5 – Блок попереднійх подій

У блоці попередніх подій відображатиметься список останніх трьох подій, де буде розміщений короткий опис подій, статистика та місця проведення. Також, відвідувачі зможуть побачити галерею з фотографіями попередніх фестивалів. Ця інформація є важливою для потенційних учасників та партнерів фестивалю. Тут вони зможуть дізнатись скільки людей відвідало захід, скільки брендів було представлено. Це допоможе їм визначити, чи потрібно брати участь у події та зрозуміти, наскільки захід є популярним та яка його цільова аудиторія.

Про події відображатиметься наступна інформація:

– назва події;

– фото з події;

– дата події;

– кількість учасників;

– кількість відвідувачів.

Рисунок 2.5 – Блок партнери

У блоці партнери буде відображатись заголовок блоку, а також список партнерів (поточних або постійних).

Список партнерів відображається у вигляді логотипів кожного партнера з посиланням на сторінку з інформацією про партнера, поза сайтом. Так, відвідувачі сайту зможуть відвідати сайт та переглянути інформацію про партнера. Даний блок є дуже важливим для підтримки хороших зв’язків з партнерами.

Сторінка інформаці про окрему конференції складається з наступних блоків:

– заголовок (рисунок 2.6);
– учасники (рисунок 2.7);
– розклад (рисунок 2.8);
– команда (рисунок 2.9);
– медіа матеріали (слайдер з фотографіями та відеоролик) (рисунок 2.10).

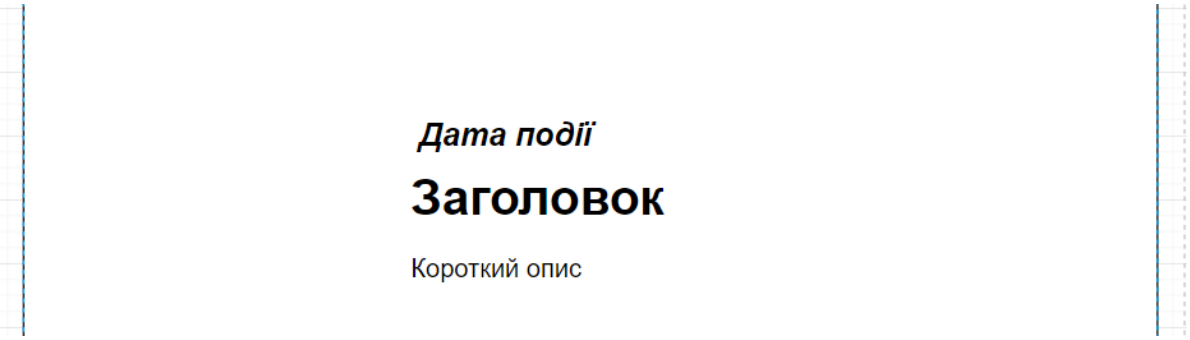


Рисунок 2.6 – Блок заголовку події

У блоці заголовку міститься наступна інформація:

– дата події;

– заголовок (назва);

– короткий опис (інформація про місце проведення заходу).

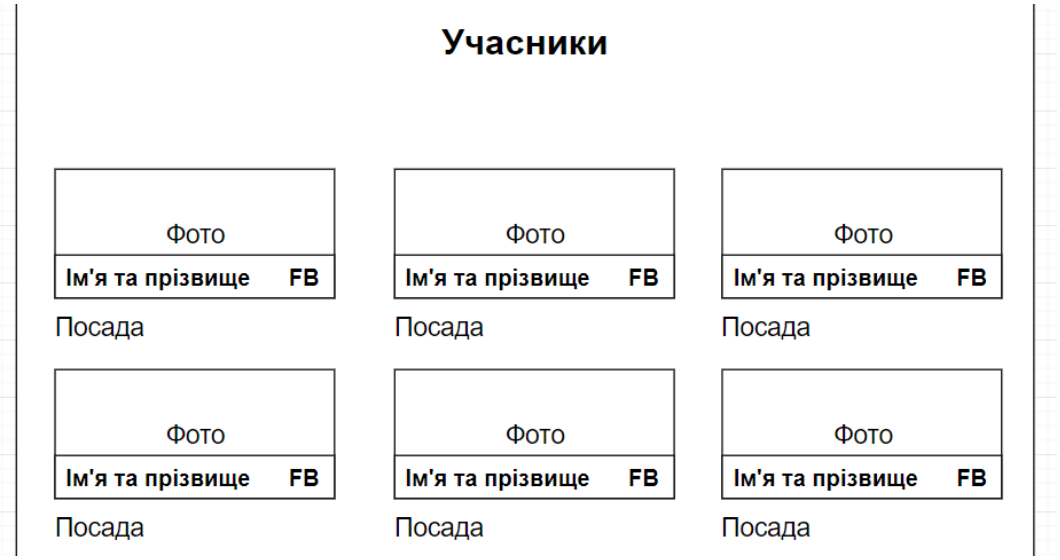


Рисунок 2.7 – Блок учасників події

У блоці учасників відображається заголовок блоку та список учасників. Також, буде розміщенна коротка інформація про учасника та історія створення бренду. Нижче будуть прикріплені декілька фото з роботами учасників. Така функція допоможе відвідувачам фестивалю детальніше ознайомитися з брендами та товарами, які будуть представленні на події.

Про кожного учасника відображається наступна інформація:

– фотографія;

– ім’я та прізвище;

– посилання на сторінку в соцмережах;

– посада або місце роботи.

Рисунок 2.8 – Блок розкладу події

Розклад відображаєть у вигляді таблиці. Він має бути максимально простим для користувача, щоб він з легкістю міг зрозуміти всю інформацію місце проведення лекції, інформацію про лектора та про що він буде розповідати.

В таблиці наведено наступну інформацію:

– час;

– інша інформація елементу розкладу, яка включає дані про назву лекції та її короткий опис, лектора та місце проведення.

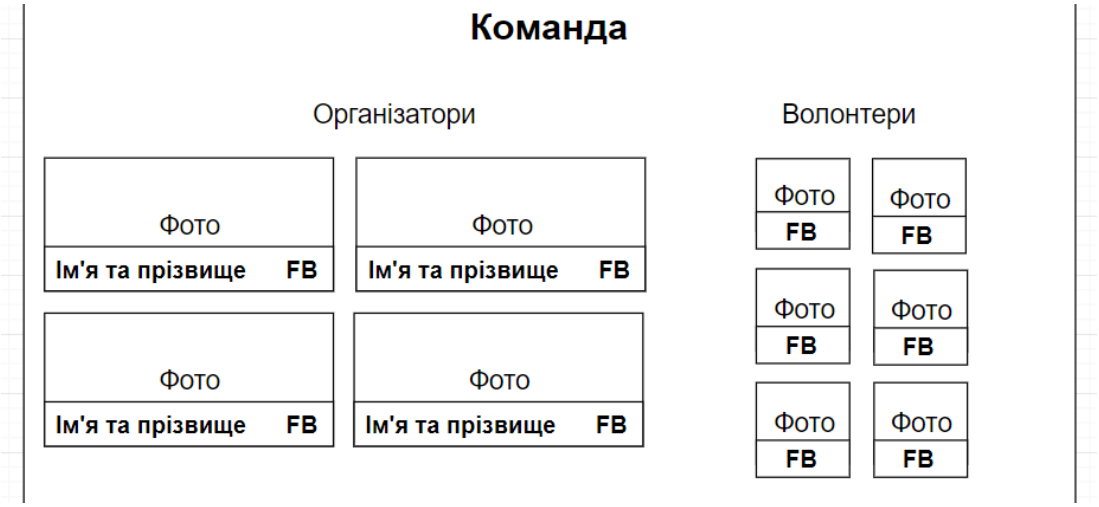


Рисунок 2.9 – Блок команди події

Блок команди складається з двох підрозділів: організатори та волонтери. Про організаторів відображається наступна інформація:

– фотографія;

– ім’я та прізвище;

– посилання на сторінку в соцмережах.

Про волонтерів відображається аналогічна інформація за виключенням імені та прізвище.

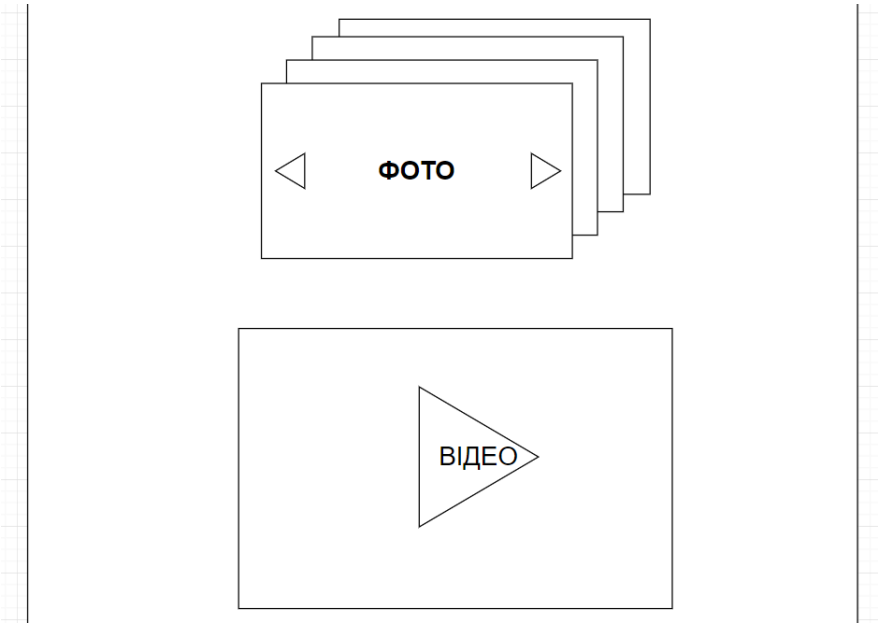


Рисунок 2.10 – Мультимедійни блок

Мультимедійний блок складається з : відео, яке можна вставити з youtube та галереї фотографій, яка працює у вигляді popup слайдера. Слайдер змінюватиме зображення за допомогою кнопок для прокрутки. Наступне зображення відображатиметься меншим.

Таким чином, користувачам та відвідувачам сайту буде зручно переглянути фото та відео з попередніх фестивалів.

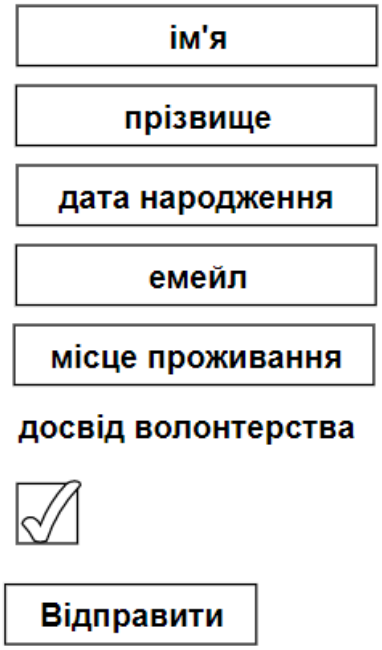


Рисунок 2.11 – Форма реєстрації волонтера

На сторінці реєстрації волонтера крім меню та футера знаходиться форма реєстрації. Перелік полів:

- ім’я (текстове);
- прізвище (текстове);
- дата народження (дата);
- електронна пошта (текстове);
- місце проживання (текстове);
- чекбокс чи наявний досвід волонтерства (булеве).

Усі поля форми реєстрації є обов’язковими.

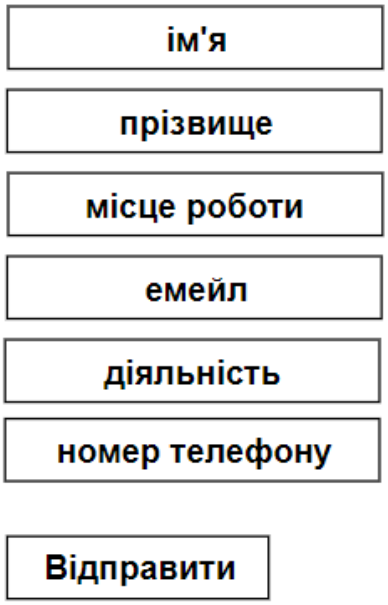


Рисунок 2.12 – Форма реєстрації учасника

Сторінка реєстрації учасника крім меню та футера містить форму реєстрації. Реєстрація дає змогу організаторам дізнатись про потенційних учасників коротку інформацію, оцінити вироби людини та вирішити, чи підходить даний учасник для фестивалю.

Усі поля форми є обов’язковими. Перелік полів:

– ім’я (текстове);
– прізвище (текстове);
– місце роботи (текстове);
– електронна пошта (текстове);
– діяльність (текстове);
– номер телефону (текстове).

Сторінки адмінпанелі автоматично згенеровані гемом rails_admin, приклад головної сторінки адмінпанелі зображено на рисунку 2.13.

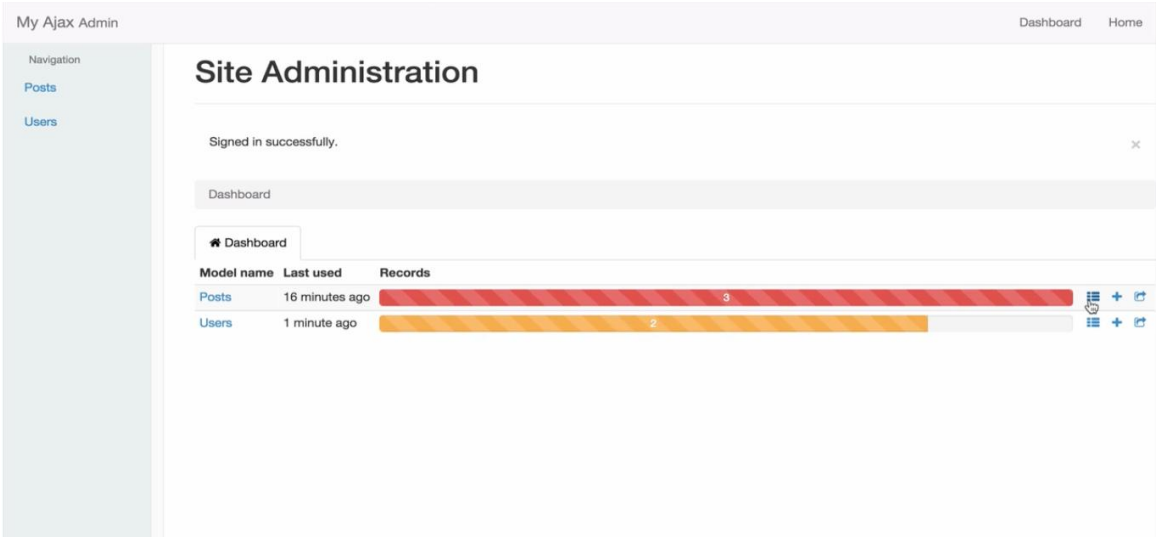


Рисунок 2.13 – Приклад головної сторінки адмінпанелі

На сторінці міститься декілька блоків:

– головне меню (верх сторінки);

– бічне меню (ліва сторона);

– статистичний блок (середина).

Головне меню складається з пунктів:

– заголовок адмінпанелі;

– посилання на головну сторінку адміпанелі (dashboard);

– посилання на головну сторінку сайту (home).

Бічне меню складається з:

– заголовок;

– підзаголовки;

– список налаштованих сутностей та конфігурацій доступних у адмінпанелі.

Статистичний блок містить:

– перелік сутностей та посилання на їх налаштування;

– назви сутностей та конфігурацій;

– кількість створених сутностей у числовому та графічному вигляді;

– поле з датою останніх змін відповідно до кожної сутності;

– кнопки для додавання нових сутностей;
– кнопки для експорту сутностей.

2.2 Розробка Use Cases діаграм роботи

У розробці програмного забезпечення та системному проектуванні для опису поведінки системи, як вона відповідає на зовнішні запити, використовують Use Case (варіант використання).

У системному проектуванні різновиди використання застосовуються на більш високому рівні ніж при розробці програмного забезпечення, часто представляючи цілі зацікавлених осіб або місії. На стадії аналізу вимог різновиди використання можуть бути перетворені на ряд детальних вимог і задокументовані за допомогою діаграм вимог SysML або інших подібних механізмів.

Кожен різновид використання зосереджується на описі того, як досягти мети або завдання. Для більшості програмних проектів це означає, що потрібно безліч різновидів використання щоб визначити необхідний набір властивостей нової системи. Ступінь формальності програмного проекту і його стадії буде впливати на необхідний рівень деталізації, для кожного різновиду використання.

Різновиди використання не можна плутати з поняттям властивостей системи (Feature). Різновид використання може бути пов'язаний з однією або більше властивістю системи, і властивість може бути пов'язана з одним або більше різновидом використання.

Деяким процесам розробки програмного забезпечення достатньо простого різновиду використання для визначення вимог до системи. Іншим необхідно багато деталізованих різновидів використання. У загальному випадку чим більше і складніше проект, тим більше ймовірно, що буде необхідно використовувати багато деталізованих різновидів

Різновид використання визначає взаємодії між зовнішніми агентами і системою, спрямовані на досягнення мети. Актор (Actor) являє собою роль, яку грає людина або річ, взаємодіючи з системою. У випадку даного веб-сайту буде розгялнути 4 ролі:

– адміністратор;
– учасник;
– волонтер;
– відвідувач.

Різновиди використання розглядають систему як «чорний ящик», і взаємодії з системою, включаючи системні відповіді, описуються з точки зору зовнішнього спостерігача. Це — навмисна політика, тому що це змушує автора зосередитися на тому, що система повинна зробити, а не, як це повинно бути зроблено, і дозволяє уникати створення припущень про те, як функціональні можливості будуть реалізовані.

Різновиди використання можуть бути описані на абстрактному рівні (діловий різновид використання, іноді званий ключовим різновидом використання), або на системному рівні (системний різновид використання).

Діловий різновид використання не зачіпає технологій, розглядає систему як «чорний ящик» і описує бізнес-процес, який використовується діловими акторами (людьми, або системами, зовнішніми до бізнесу) для досягнення своїх цілей (наприклад, обробка оплати, схвалення авансового звіту, управління корпоративними нерухомим майном). Діловий різновид використання описує процес, цінний для ділового агента, описує що саме робить процес. На діаграмах різновидів використання в UML різновид відображається у вигляді еліпса. Всередині еліпса або під ним вказується ім'я елемента [29].

Системний різновид використання зазвичай описується на рівні функцій системи (наприклад, створіть ваучер), і визначає функцію або сервіс, що надаються системою для користувача. Системний різновид використання описує що актор може зробити взаємодіючи з системою. З цієї причини

рекомендується, щоб системні випадки використання починалися з дієслова (наприклад, створіть ваучер, виберіть платежі, скасуйте ваучер) [10].

Для опису Use Case-ів системи буде використано системний різновид, з огляду на наявний функціонал для більшості ролей.

Use Case повинен:

– описувати що саме система має зробити, щоб актор досяг своєї мети;

– не торкатися деталей реалізації;

– мати достатній рівень деталізації;

– не описувати інтерфейси та екрани. Це робиться під час дизайну користувальницького інтерфейсу.

В уніфікованій мові моделювання відносини між усіма або частиною різновидів використання й акторами представлені у вигляді діаграми різновиду використання або діаграмах.

На діаграмах різновидів використання в UML різновид відображається у вигляді еліпса. Всередині еліпса або під ним вказується ім'я елемента [9].

До різновиду використання в UML застосовують наступні види відносин:

– Асоціація (Association) — Може вказувати на те, що актор ініціює відповідний варіант використання.

– Розширення (Extend) — Різновид відносини залежності між базовим варіантом використання та його спеціальним випадком.

– Включення (Include) — Визначає взаємозв'язок базового варіанту використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

– Узагальнення (Generalization) — моделює відповідну спільність ролей.

На рисунку 2.14 зображено відповідності між акторами та їхніми цілями.

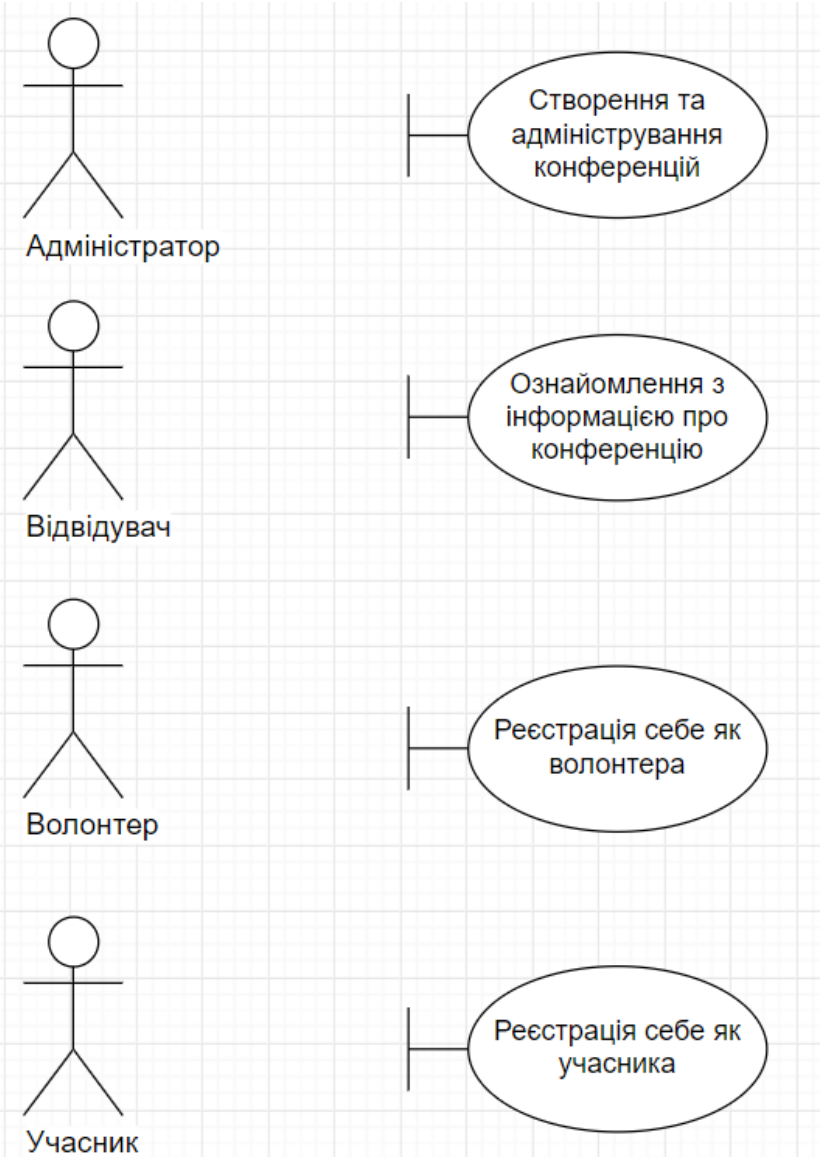


Рисунок 2.14 – Актори та їх цілі

Беручи за основу для побудови Use Case рисунок з описом цілей акторів, побудовано UML-діаграми (рисунок 2.15).

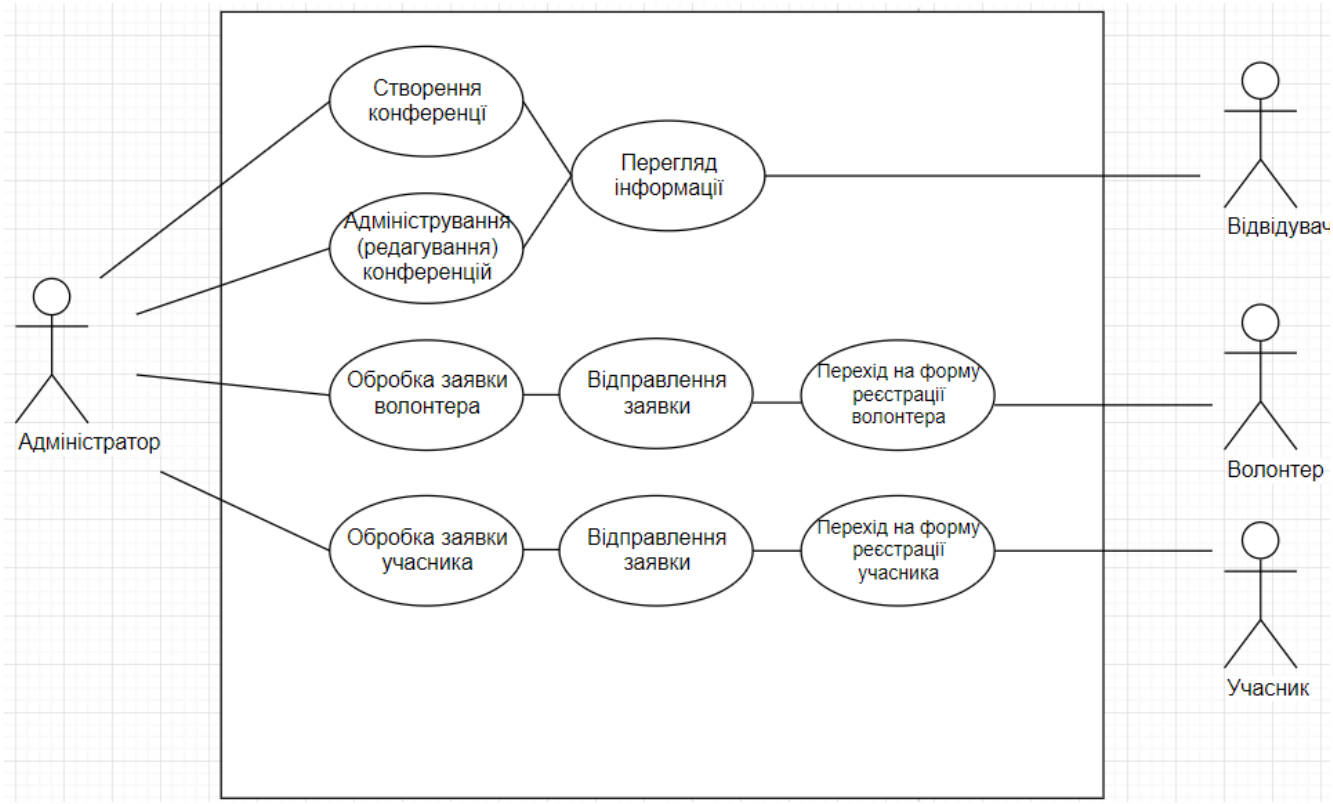


Рисунок 2.15 – UML-діаграми

На рисунку зображено діаграму Use Case-ів потенційних користувачів сайту, відповідно до їхнії вимог.

Історія користувача (user story) - це одне чи більше речень, звичайною мовою предметної області, які описують чого користувач хоче досягти. Історії користувача використовуються в гнучких методологіях для з'ясування базових функцій що будуть реалізовуватись. Кожна історія користувача достатньо коротка і записується на карточках приблизно 7 на 12 сантиметрів, що гарантує те, що вона не стане занадто великою. Історії користувача пишуться споживачами програмного забезпечення і є основним інструментом їх впливу на розробку програми. Вони описують вимоги у простий та точний спосіб [30].

Історії користувача − швидкий спосіб оперування вимогами користувача, без необхідності застосування занадто формалізованих документів, та виконання адміністративних задач пов'язаних з опрацюванням цих документів. Наміром з яким використовують історії користувача є швидше та менш накладне реагування на швидко змінювані вимоги реального світу.

Сценарії використання, на відміну від історій користувача, описують процес та його кроки докладно, і можуть бути сформульовані з точки зору формальної моделі. Історії користувача − це невелике і зручне в роботі уявлення інформації. Вони сформульовані на повсякденній мові користувача і містять невеликі деталі, таким чином залишаючись відкритими для інтерпретації. Вони допомагають читачеві розуміти що повинна робити система. До кожної історії користувача в якийсь момент має бути прикріплено одне або більше приймальне тестування. Це дозволяє розробнику дізнатися, коли для користувача історія готова і як клієнту перевірити це. Приклад user story зображений на рисунку 2.16.

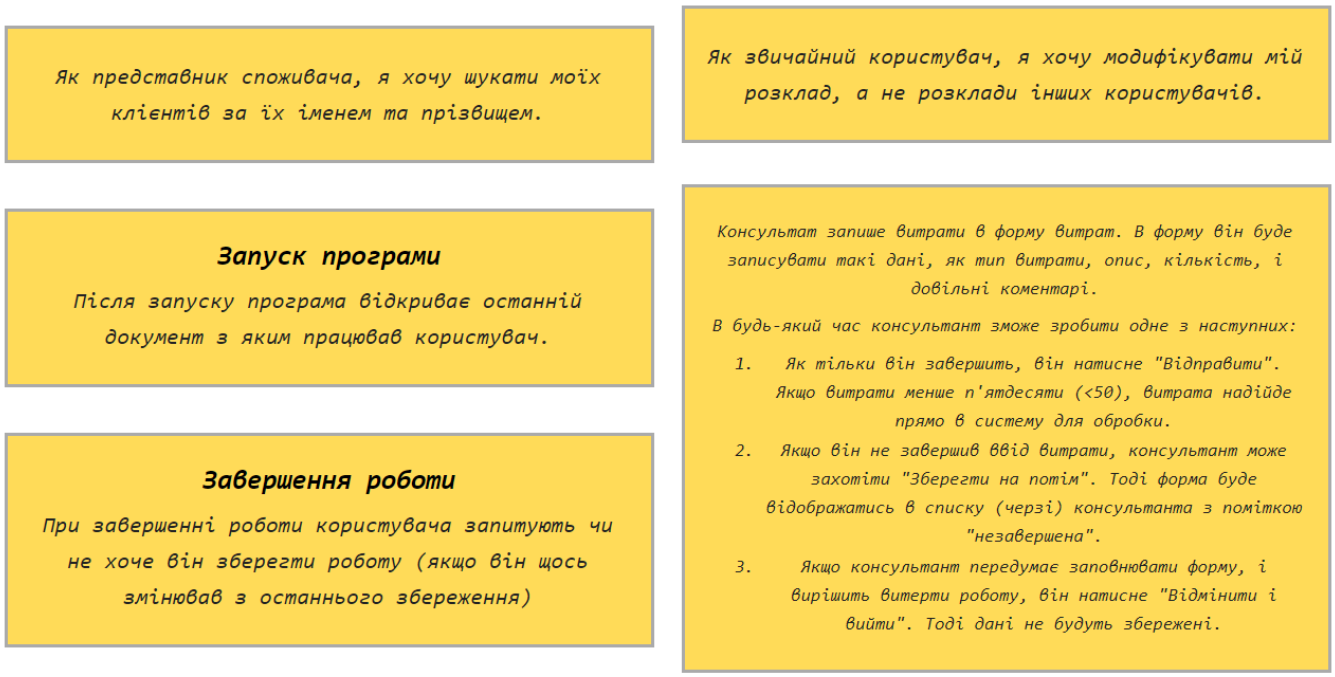


Рисунок 2.16 – Приклад user story

Усі user story розроблені за поданим прикладом та описують детальні кроки роботи користувачів на сайті. User story звичайного відвідувача сторінки сайту Franek Shop зображена на рисунку 2.17.

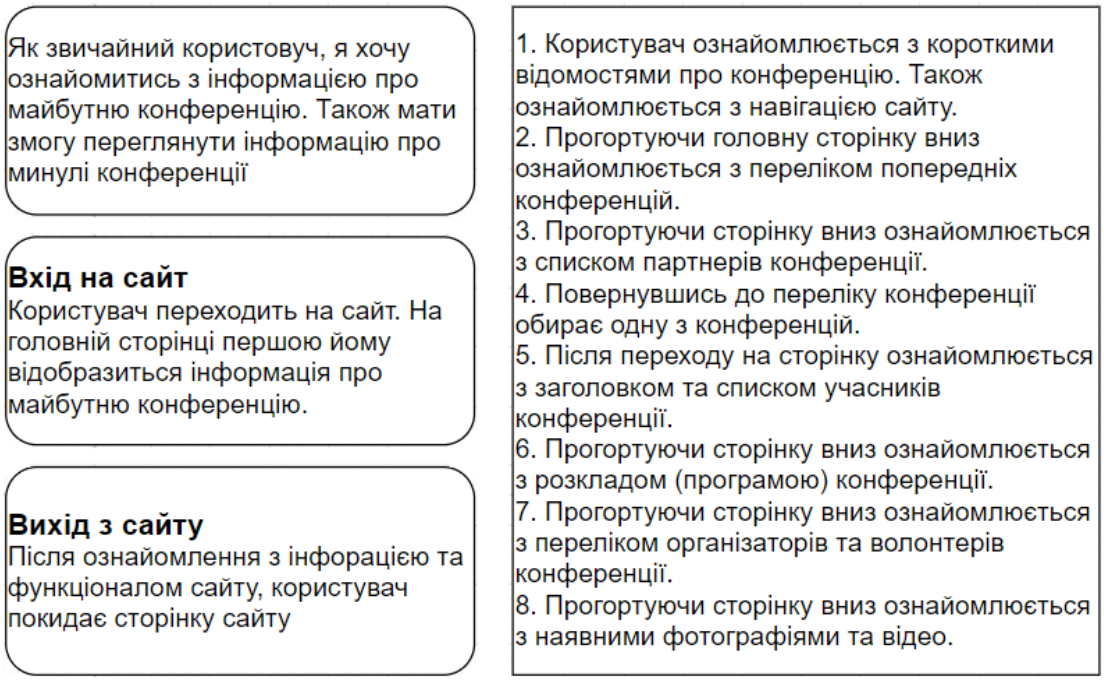


Рисунок 2.17 – User story відвідувача сайту

Дана user story описує діяльність звичайного відвідувача, який хоче ознайомитись з інформацією про попередні та майутні конференції. User story волонтера на сторінці сайту Franek Shop зображена на рисунку 2.18.

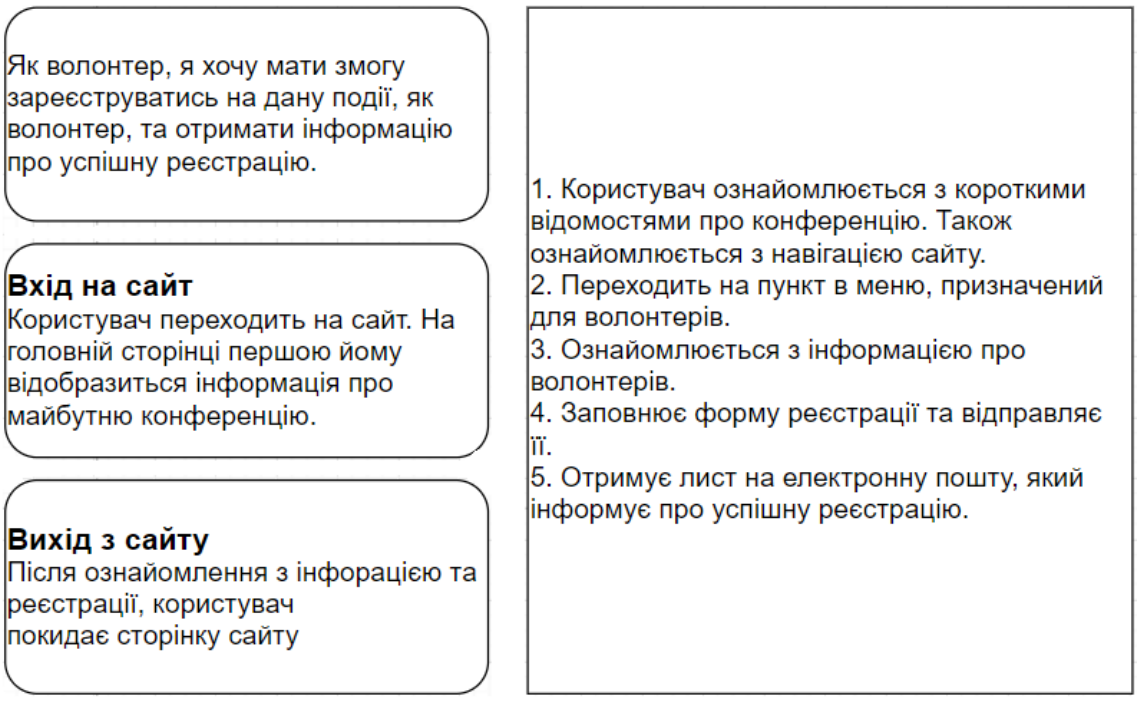


Рисунок 2.18 – User story волонтера

Дана user story описує роботу волонтера, який хоче зареєструватись. User story учасника на сторінці сайту Franek Shop зображена на рисунку 2.19.

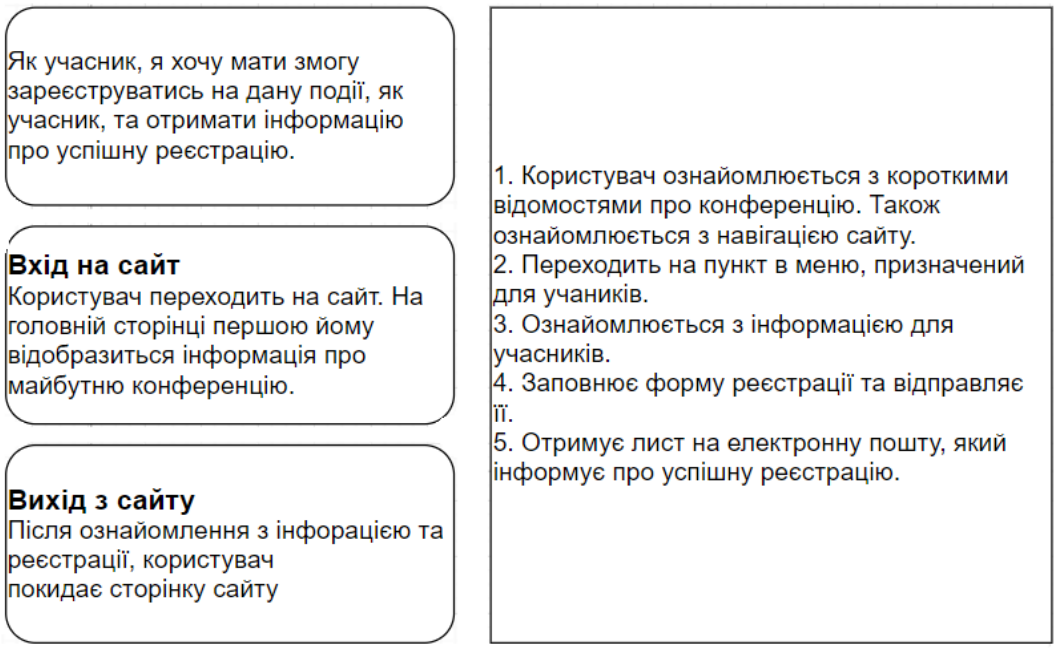


Рисунок 2.19 – User story учасника

Дана user story описує роботу на сайті учасника, який хоче зареєструватись. Описані user story кінцевих користувачів сайту, діяльність адміністратора не описується, так як цільовою аудиторією ресурсу є волонтери, потенційні учасники та звичайні відвідувачі.

2.3 Розробка системи реєстрації користувача

Один з кращих гемів для аутентифікації в rails-додатках є Devise. Devise – це ruby–гем, що надає можливості для аутентифікації в rails-додатках. Devise працює в зв'язці з гемом Warden, який в свою чергу надає сам механізм для аутентифікації в rack-базованих ruby-додатках. Основними особливості Devise є:

– розроблений на Rack;

– є довершеним MVC-рішенням, розробленим на Rails;

– дозволяє вхід в систему по декількох моделях одночасно;

– модульний, тобто використовує тільки те, що вам дійсно необхідно.

Devise потрібен для реєстрації користувачів та, авторизації адміністратора в адмінпанель. Інтеграцію гема розподілено на два етапи: встановлення та налаштування.

Для встановлення потрібно вказати гем в Gemfile, можна також вказати точну версію для використання [2]. Наприклад:

```
gem 'devise', '3.2.2'
```

Для встановлення гему потрібно виконати команду:

```
bundle install
```

Як залежностей будуть встановлені додатково наступні геми:

– warden – сполучне ПЗ, яке надає можливість аутентифікації для Rack-додатків;

– orm_adapter – надає єдину точку входу для використання основних функцій Ruby ORMs;

– bcrypt-ruby – надає просту обгортку для роботи з паролями. В основі лежить криптографічна хеш-функція bcrypt ();

– thread_safe – надає потоко-безпечні колекції і утиліти для Ruby;

– railties – внутрішні компоненти Rails, такі як завантажувачі додатків, плагіни, генератори і rake-завдання.

Devise має в своєму арсеналі зручні генератори. Виконати інсталяцію Devise можна шляхом запуску наступного генератора:

```
rails generate devise:install
```

Цей генератор встановить ініціалізатор, в якому описані всі конфігураційні налаштування Devise, необхідні для роботи, а також файл з базовою локаллю (англійська мова). Також інсталятор пропонує виконати базове налаштування.

Так як будуть використовуватись сповіщення, які надходитимуть на електронну пошту, після реєстрації нового користувача необхідно задати налаштування мейлера (відправника пошти) для кожного середовища виконання. Для середовища розробки необхідно додати наступний рядок в файл config/environments/development.rb:

```
config.action_mailer.default_url_options = { :host => 'localhost:3000' }
```

Після входу користувача в систему, реєстрації, підтвердження аккаунта або поновлення пароля, Devise буде шукати шлях для подальшого перенаправлення. За замовчуванням виконає перенаправлення до user_root_path, якщо той існує. В іншому випадку Devise виконає перенаправлення до root_path. Тому цей шлях повинен бути обов'язково визначений у програмі. config/routes.rb повинен містити рядки такого вигляду:

```
root 'home#index'
```

Можна налаштувати файли вигляду під свої потреби. Для цього необхідно їх скопіювати з гема до свого додатку шляхом запуску наступної команди:

```
rails generate devise:views
```

В директорії app/views/devise появляться всі використовувані гемом файли вигляду. Можна налаштувати їх за своїм бажанням, під загальний стиль додатку.

Для користувача, який буде виконувати аутентифікацію, тобто адміністратора (модель Admin), потрібно згенерувати модель та зв'язати її з гемом. Також потрібно створити моделі для користувачів, які будуть здійснювати реєстрацію, це моделі Volunteer та Member. Генерація моделей для гему відбувається командами:

```
rails generate devise Admin
rails generate devise Volunteer
rails generate devise Member
```

Даний генератор створить нову модель, якщо її не існувало раніше і зконфігурує її з урахуванням використовуваних за замовчуванням модулів. Результатом виконання команд є створення файлів, які описують класи в app/models/. Приклад згенерованого коду моделі Admin:

```
class Admin < ActiveRecord::Base
  include Mongoid::Document
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable and :omniauthable
  devise :database_authenticatable,
         :registerable,
         :recoverable,
         :rememberable,
         :trackable,
         :validatable

end
```

Devise має в своєму арсеналі 10 модулів. За замовчуванням підключено 6 модулів. Можна відредагувати цей список. Опис всіх доступних модулів нижче:

1. Database Authenticatable: надає можливість входу в систему на основі зашифрованого і збереженого в базі даних пароля. Вхід може бути

виконаний за допомогою відправки POST-запиту або за допомогою HTTP Basic Authentication.

2. Omniauthable: додає підтримку Omniauth.

3. Confirmable: дозволяє відправляти лист з інструкціями для підтвердження аккаунта, створеного під час реєстрації.

4. Recoverable: дозволяє відновлювати забутий пароль. Відправляє інструкції по відновленню на пошту.

5. Registerable: керує реєстрацією користувачів, дозволяє редагувати і видаляти акаунти.

6. Rememberable: дозволяє запам'ятовувати користувачів на основі cookies. Керує створенням і видаленням токенів.

7. Trackable: веде статистику кількості входів, враховує час і IT-адреси.

8. Timeoutable: відповідає за тривалість сесії активності користувача в системі;

9. Validatable: надає інструменти валідації e-mail і пароля. Модуль може бути легко налаштований, можете визначити власні валідатори.

10. Lockable: блокує акаунт після зазначеної в налаштуваннях кількості невдалих спроб авторизації. Аккаунт може бути розблоковано за допомогою email або через певний період часу.

Також виконана команда автоматично додасть до моделі наступні поля:

– Email (Електронна пошта).
– Encrypted_password (Зашифрований пароль).
– Reset_password_token (Токен для зміни паролю).
– Reset_password_sent_at (Дата відправлення зміни паролю).
– Remember_created_at (Дата збереження cookie).
– Sign_in_count (Дата входу в акаунт).
– Current_sign_in_at (Дата поточної авторизації).
– Last_sign_in_at (Дата останнього входу в акаунт).

– Current_sign_in_ip (Поточна IP-адреса авторизації).

– Last_sign_in_ip (Остання IP-адреса авторизації).

Для користувачів Volunteer та Member потрібно за коментувати усі модулі крім Registerable, для того, щоб вони могли тільки зареєструватись. Також потрібно додати в моделі відповідні поля, які є обов’язковими для реєстрації. Для налаштування форми реєстрації потрібно відкрити згенерований шаблон форми реєстрації та виправити його.

Також для відправки повідомлень на електронну пошту потрібен Action Mailer. За допомогою нього можна розсилати повідомлення з додатку використовуючи класи і відображення розсильника.

У конкретного класу, який буде наслідувати Action Mailer та реалізовувати свої функції для розсилання повідомлень, існує і ряд необхідних методів для роботи.

Для розробки в середовищі development потрібно буде зімітувати роботу SPTM сервера використавши гем mailcatcher, але для production середовища, потрібно буде використати або власний, або вже реалізований SPMT сервер.

SMTP-сервер − це мережевий протокол, який служить для відправки електронної пошти. SMTP розшифровується як Simple Mail Transfer Protocol, що в перекладі означає «простий протокол передачі пошти».

За протоколом SMTP відправник листа зв'язується з одержувачем за допомогою командного рядка і спеціальних каналів, роль яких зазвичай виконує TCP-з'єднання. Будь-яка SMTP-сесія складається з двох провідних компонентів: команд від клієнта і відповідних їм відповідей сервера. При відкритій сесії обидві цих складових обмінюються її параметрами. Подібний обмін може включати як нуль, так і більше SMTP-операцій (транзакцій).

3 РОЗРОБКА ІНТЕРФЕЙСУ КОРИСТУВАЧА ІНФОРМАЦІЙНОГО САЙТУ

3.1 Розробка інтерфейсу для користувачів типу волонтер та учасник

Засобом зручної взаємодії користувача з інформаційною системою, в даному випадку з сайтом, є інтерфейс користувача. Сукупність засобів для обробки та відбиття інформації, якнайбільше пристосованих для зручності користувача; у графічних системах інтерфейс користувача, втілюється багатовіконним режимом, змінами кольору, розміру, видимості (прозорість, напівпрозорість, невидимість) вікон, їхнім розташуванням, сортуванням елементів вікон, гнучкими налаштуваннями як самих вікон, так і окремих їх елементів (файли, теки, ярлики, шрифти тощо), доступністю багатокористувацьких налаштувань.

Виділяють наступні основні види інтерфейсів:

– Інтерфейс прямої обробки, це назва загального класу інтерфейсів користувача, який дозволяє користувачам маніпулювати наданими їм об'єктами, з використанням дій, котрі принаймні, відповідають фізичному світу.

– Графічні інтерфейси користувача (GUI) приймають вхідні дані за допомогою таких пристроїв, як комп'ютерна клавіатура та миша, й забезпечують графічний висновок на моніторі комп'ютера. У GUI-дизайні, широко використовуються як мінімум, два різні принципи: об'єктно-орієнтовані інтерфейси користувача(OOUI) й інтерфейси, орієнтовані на додатки.

– Веб-інтерфейси користувача або веб-інтерфейси користувача (WUI), які приймають вхідні дані та забезпечують виведення, створенням веб-сторінок, які передаються Інтернетом і проглядаються користувачем за допомогою програми веб-браузера. Адміністративні веб-інтерфейси для веб-серверів, серверів і мережевих комп'ютерів, часто називаються панелями керування.

– Сенсорні екрани — дисплеї, які приймають введення, дотиком пальців або стилусом. Використовуються у все більшій кількості мобільних пристроїв і багатьох засобах точок продажу, промислових процесах і машинах, пристроях самообслуговування тощо.

– Апаратні інтерфейси — фізичні, просторові інтерфейси, присутні на виробах у повсякденному житті від тостерів, до приладових панелей автомобілів чи кабін літаків. Вони, як правило, є поєднанням ручок, кнопок, слайдерів, перемикачів і сенсорних екранів.

– Інтерфейси командного рядка, де користувач надає вхідний сигнал, введенням командного рядка за допомогою комп'ютерної клавіатури, а система забезпечує виведення, шляхом відбиття тексту на моніторі комп'ютера.

З розвитком DHTML та JavaScript набув популярності підхід до розробки інтерфейсної частини веб-застосунків, названий AJAX. Серцем технології є здатність веб-сторінки зініціювати запит до веб-сервера і отримати потрібні дані, так щоб інтерфейс не перезавантажував сторінку цілком, а лише довантажують необхідні дані і змінив потрібні частини сторінки, що робить їх більш інтерактивними і продуктивними.

Веб-інтерфейси зручні тим, що дають можливість вести спільну роботу співробітникам, які не перебувають в одному офісі. Веб-інтерфейс дає можливість універсального віддаленого доступу до служб та пристроїв, у цьому технології практично нема альтернатив. Але водночас, оскільки такий інтерфейс доступний усім, постають серйозні питання забезпечення безпеки, зокрема автентифікація та авторизація користувачів, шифрування переданих даних від сторонніх очей, модерація вмісту тощо.

Однією з основних вимог до веб-інтерфейсів є їх однаковий зовнішній вигляд і однакова функціональність при роботі в різних браузерах.

Для взаємодії з інформацією на сайті, потрібно розробити веб-інтерфейс. Класичним і найпопулярнішим методом створення веб-інтерфейсів є використання HTML із застосуванням CSS і JavaScript, як правило за допомогою скриптових мов на стороні сервера. Проте різна реалізація HTML,

CSS, DOM і інших специфікацій в браузерах викликає проблеми при розробці веб-застосунків і їхньої подальшої підтримки.

Стандартнj. Мовj. розмітки для створення веб-сторінок і веб-додатків є HTML. HTML (Hypertext Markup Language — мова гіпертекстової розмітки) служить для опису Web-сторінки, що зберігається у виді звичайного текстового файлу з розширенням *.htm або *.html. Головна мета HTML — описати формат вмісту Web-сторінки, він описується з допомогою дескрипторів (tag) HTML. Дескриптори визначають способи форматування тексту, служать розпізнавальними знаками зображень або таблиць, дозволяють зв'язувати слова або фрази з іншими документами в Internet.

Але так як, більшість програмістів використовує інфраструктуру програмних рішень, що полегшують розробку складних систем. Так звані фреймворки, які можна вважати своєрідною комплексною бібліотекою, але при цьому вона має ряд обмежень, що задають правила створення структури проекту та написання коду. То для розмітки, задання стилів та роботи з додатком будуть використані наступні технології:

– HAML;
– Sass;
– JQuery.

HAML (HTML Abstraction Markup Language) — це мова для написання шаблонів, головною метою якої є написання простого і легкого для сприйняття коду, який інтерпретується в звичайний HTML. Haml дає змогу писати динамічний код для HTML [28].

HAML також дозволяє писати код, що буде виконаний протягом генерації HTML і отримати динамічний код. Розширення файлів з кодом − .haml. Такий підхід до роботи схожий на файли .erb (eRuby), що дозволяють вставляти код, написаний на Ruby для генерації коду веб-застосунків. Під час обробки коду Haml користується тими ж правилами, що й Ruby версії 1.9 і новіше.

HAML може працювати через командний рядок як окремий модуль Ruby або ж використовуватися в Ruby on Rails, що робить цю мову гнучкою та зручною для створення багатьох видів застосунків [26].

Розмітка має відповідати принципу DRY. Вона повинна:

– Уникати непотрібних повторень

– Приділяти увагу чистоті коду

Для початку роботи з HAML потрібно встановити гем haml-rails, додавши наступний код в Gemfile:

```
gem "haml-rails", "~> 2.0"
```

Гем забезпечить:

– Підчас генерації контроллерів, також згенеруються відображення або поштові шаблони, типу HAML замість ERB.

– Під час завантаження Rails додатку HAML завантажеться та ініціалізується автоматично.

– Шаблони HAML відповідають шаблонам кешування digestor.

Для роботи також потрібно переіменувати розширення попередньо згенерованих відображень з .html.erb на .html.haml. Для цього необхідно виконати команди:

```
rails generate haml:application_layout convert
rails haml:erb2haml
```

Відповідно до структури роботи Ruby on Rails, кожна дія контролера може генерувати HTML. Сторінки які повинні бути згенеровані знаходяться у відповідних директоріях з назвами контролерів та дій (рисунок 3.1).

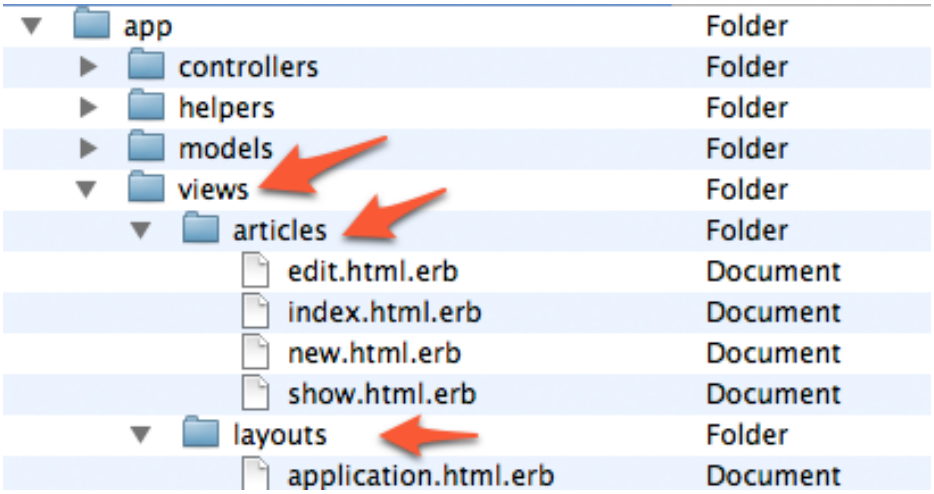


Рисунок 3.1 – Структура директорій відображень

В Rails є узгодження імен відображень. Як правило, ім'я відображення збігається з відповідною дією контролера. Наприклад, індекс контролера в статті articles_controller.rb буде використовувати файл в index.html.erb в директорії app/views/articles. Повний HTML, повернений клієнтом, складається з комбінації цього файлу, шаблону макета, який обгортає його, і всіх партіалів (partials), які можуть бути використані. Шаблони розміщені в спеціально відведених директоріях як partials layouts [27].

Для об’єднання та мінізації або зтискання асетів JavaScript і CSS, Rails надає фреймворк Asset Pipeline. Asset Pipeline увімкнутий в проекті по замовчуванню. Це дозволяє автоматично комбінувати асети програми з асетами інших гемів.

Для роботи з Asset Pipeline потрібно налаштовувати файли манфести, щоб фреймворк розумів, які файли враховувати, але для зручності, файли також розміщаються у директоріях з іменем відповідного типу асету (javascript, css) та відповідного контроллера.

Розглянемо структуру сторінок на основі блоків головної сторінки сайту. На рисунку 3.2 зображено блок з заголовком, головної сторінки сайту.

Рисунок 3.2 – Блок заголовку головної сторінки сайту

Для головної сторінки створено контролер під назвою Main. В якому міститься тільки одна дія (action) index. Але вміст сторінки складається не тільки з коду одного файлу шаблону дії. Також міститься application_layout.html.haml в якому описане меню сайту, тому що воно зустрічається на усіх сторінках. Partial використовується і для заголовку сторінки.

Відповідно до назви контролера, відображення шаблону сторінки зберігається у файлі index.html.haml, в директорії app/view/main. Уривок HAML коду сторінки наведено далі.

```
- if @event.present?
  .home
    .conf
      .show-conf
        .show-one-conf
          %h4.date=              date_range(@event.start_date,
@event.end_date, format: :long, show_year: false)
          %h2.big-heading
            = @event.name
            %sup
              %small= @event.start_date.year
          %span.start-name= @event.second_name
          %h5.description= @event.description
```

Змінна @event містить в собі усю потрібну інформацію про конференцію. Як видно з коду, якщо змінна існує, тобто подія, яка доступна до перегляну головній сторінці є, тоді можна виводити інформацію про подію. Текстові поля, такі як description, name і т.д. не викличуть помилок, якщо будуть порожніми, але якщо поля з датою, будуть порожніми, то виникнуть помилки, тому що вони надалі обробляються. Тому у сутності події, поля дати є обов'язковими. Особливості HAML дозволяють компактно та ясно описати структури сторінок. Код application_layout.html.haml сайту, наведений нижче:

```
!!!
%html
  %head
    %meta{:content => "text/html; charset=UTF-8", "http-
equiv" => "Content-Type"}/
    %meta{name: 'viewport', content: 'width=device-width,
initial-scale=1, minimum-scale=1.0, user-scalable=yes'}
    %title
      =@title
    = csrf_meta_tags
    = stylesheet_link_tag 'application', media: 'all',
'data-turbolinks-track': 'reload'
    = stylesheet_link_tag
'https://fonts.googleapis.com/css?family=Roboto:100,100i,300,300i,
400,400i,500,500i,700,700i,900,900i&amp;subset=cyrillic'
    = stylesheet_link_tag
'https://fonts.googleapis.com/css?family=Rubik:300,300i,400,400i,5
00,500i,700,700i,900,900i'
  %body
    .overlay.overlay-hugeinc
      = render 'partials/menu'
    = yield
    = render 'partials/footer'
= javascript_include_tag 'application'
```

application_layout є головним шаблоном усіх сторінок сайту, який по замовчуванню інтегрується на всіх сторінках. Тому мета теги, стилі, java script, та заголовки описуються тут. Для заголовків передається змінна @title, яка для кожного контролера мое бути іншою. Підключені деякі зовнішні шрифти. Підключений файл application.js, який у свою чергу в собі містить усі інші

скрипти. Також за допомогою методу render вказується, який partial генерувати для футера та меню.

Partial меню міститься в директорії partials під назвою _menu.html.haml.

```
%section.menu-section
  .container
    %ul.main-menu
      %li>
        = link_to member_path
          %h3
            Учасникам
      %li>
        = link_to volunteer_path
          %h3
            Волонтерам
      %li>
        = link_to '#past_events'
          %h3
            Події

    =link_to '', root_path, class: "main-logo"
    %a.overlay-close.close{href: "#"}
    .instagram
       = link_to t('menu.inst_link')
    .facebook
       = link_to t('menu.fb_link')
```

У коді для посилань використовуються 3 різних типа, але всі посилання створюються за допомогою метода link_to, який генерує тег посилання з зазначеними параметрами. Типи використаних посилань у меню:

– шлях який генерує Rails;

– якір на блок на сторінці;

– посилання, яке задається вручну через адмінпанель.

Яскравим прикладом використання partials є сторінка з організаторами конференції (рисунок 3.3).

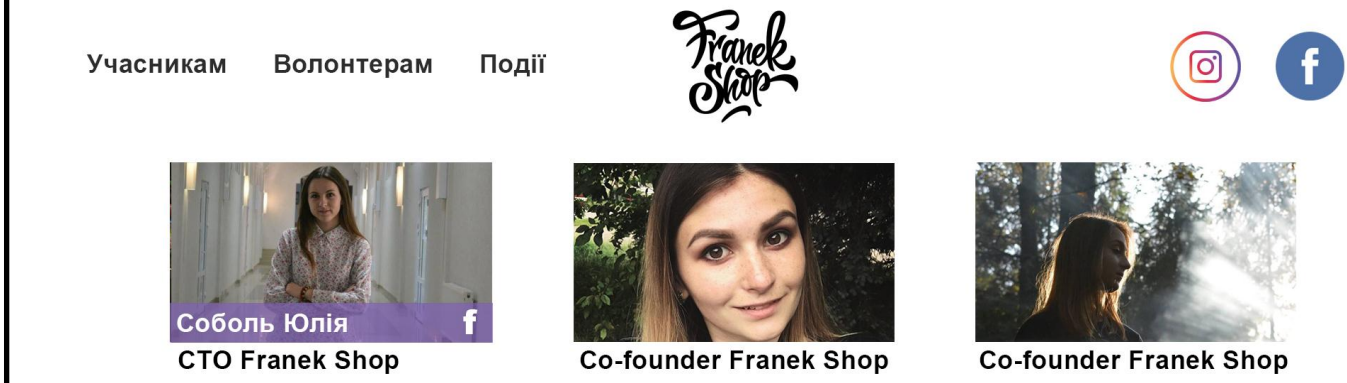


Рисунок 3.3 – Сторінка організаторів конференції

На сторінці завдяки шаблону пере використано меню, а також кожен учасник є всього лиш згенерованим partial шаблоном, просто з різними даними. Реалізація partial-у використаного для сторінки, наведена нижче:

```
        .members
          - @event.members. each do |member|
            .member-item__container
              .member-item-wrap
                . member-item
                  . member-img
                    = image_tag member.image. member.url
                  %h3. member-name= member.full_name
                  %a.facebook{href:      member.fb_link,      target:
'_blank'}
                    = image_tag('facebook.png', alt: "Facebook")
                %h5.member-position= member.position
```

Також, для зручності користувачів, часткова інформація про учасника є захованою, та з’являється тільки при наведенні курсора миші на учасника. Це створено для того, щоб не преобтяжувати користувача інформацією, він зможе прочитати короткий опис та переглянути фото робіт учасника, якщо зацікавиться.

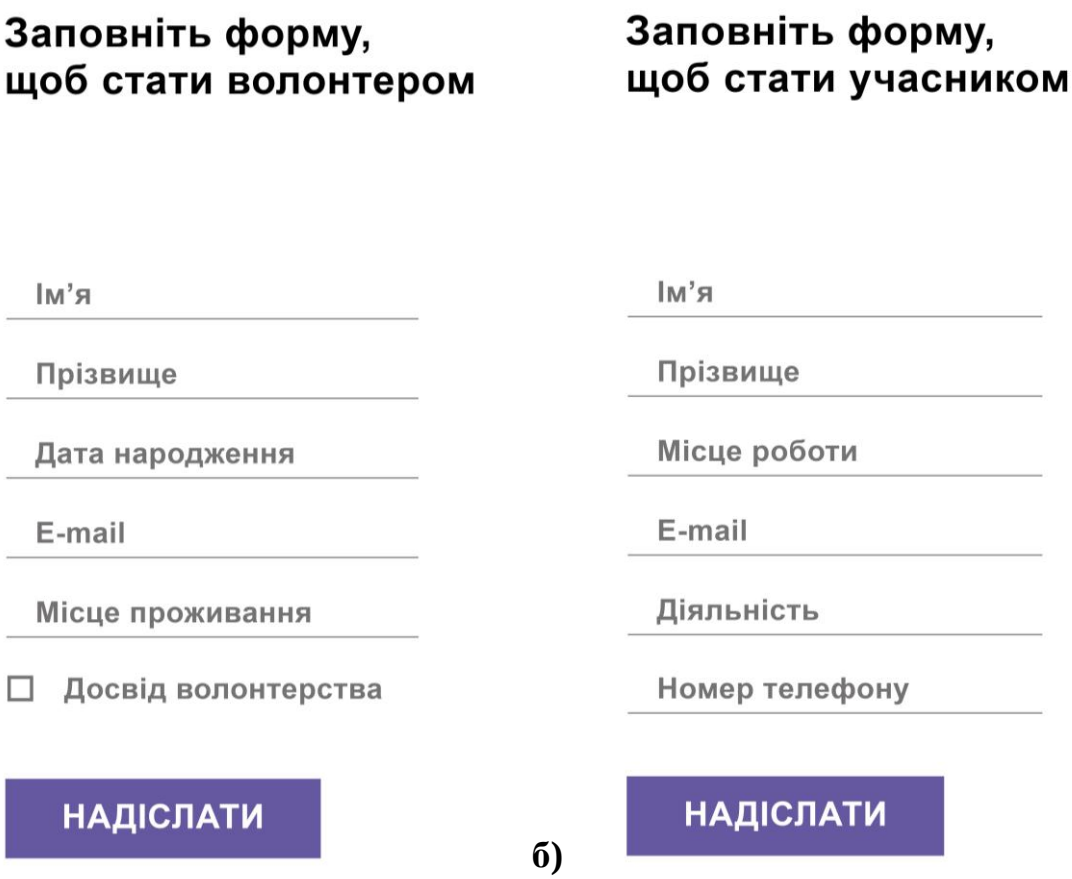
Детальна реалізація усіх стилів сторінок за допомогою Sass, наведена у додатках.

На даній сторінці виводиться масив учасників конференції та інформація про них.

3.2 Реалізація форм реєстрації

Для розробки форм реєстрації необхідні не тільки уміння HAML та CSS, у більшості випадків за функціонування форм відповідає і Java Script. Java Script відповідає за відправлення даних, обробку відповідей від серверу та валідації даних, які вводяться у форму.

Перш за все створено загальну структуру форм, з усіма інпутами відповідно до типу даних, які вони повинні отримувати. Реалізовані форми реєстрації зображено на рисунку 3.4.



а) **б)**

Рисунок 3.4 – Форми реєстрації а) волонтера; б) учасника

Форма волонтера (рисунок 3.4 а) містить такі елементи:

- input type text;
- input type date;
- input type email;

– checkbox;

– button type submit.

Якщо якесь з полів пропущене, або невірно заповнене, то поле після відправлення форми буде підсвічуватись червоним кольором. Усі поля форми є обов’язковими до заповнення.

Форма реєстрації учасника деякими елементами відмінна від форми реєстрації волонтера.

Форма учаниска (рисунок 3.4 б) містить такі елементи:

– input type text;

– input type email;

– input type tel;

– button type submit.

Так само як і в попередній формі, при виникненні помилок, побудуть підсвічуватись червоним кольором.

Валідація є важливим процесом, який відповідає за визначення відповідності розроблюваного програмного забезпечення між очікуваннями і потребами користувача, вимогам до системи.

У випадку роботи форм, валідація перевіряє значення, які вводить користувач і відображає знайдені помилки. Основним завданням валідції є - зробити так, щоб користувач не зробив помилку і валідація не знадобилася.

Для цього потрібно:

– обмежити вибір завідомо неправдивих значень в списку: блокувати їх або не показувати в списку;

– обмежити введення невідповідних символів. Якщо в поле потрібно вводити тільки цифри, і це очевидно користувачеві, ігнорувати введення букв замість того, щоб показати помилку. Використовувати маски в полях, де у значень відомий формат;

– писати підказки для заповнення форми. Наприклад, плейсхолдер в полях введення.

Форми розроблялись з дотриманням наведених правил, тому кожне поле містить плейсхолдер, абу підпис. При введені даних у форму, з’являється маска, там де у даних специфічна форма введення. У полях з номером телефону, неможливо ввести невідповідні символи. А також поле для введення дати не дає змоги ввести значення невалідного типу, тому що, при введені даних з’являється календар з якого необхідно обрати дату.

Існує три види валідації:

- миттєва;
- при втраті фокусу;
- при відправці форми.

Чим раніше інтерфейс повідомляє про помилку, тим користувачеві простіше повернутися і виправити помилку.

Найшвидший спосіб повідомити про помилку це миттєва валідація. Але вона можлива тільки в тих випадках, коли в процесі введення зрозуміло, що значення некоректне. Зазвичай такі помилки пов'язані з неправильною розкладкою клавіатури (кирилиця замість латиниці) або введенням букв в цифрове поле. Для цих випадків використовуються поля з масками: введення невідповідних символів в них заблоковано. Зразок миттєвої помилки наведено в рисунку 3.5. Тому в наших інтерфейсах є усі види валідації.

Рисунок 3.5 – Миттєве повідомлення про помилку

Під час валідації при втраті фокусу не валідуються поля на пустоту, тому що користувач, може повернутись до цього поля і тоді заповнити його, отже перевірку на те чи поле заповнено варто проводити тільки після відправлення форми.

Валідація спрацьовує одразу після втрати фокусу, якщо значення заповнене, якщо знайдено помилку, то поле підсвічується червоним і фокус в поле не повертається (рисунок 3.6).

Рисунок 3.6 – Повідомлення про помилку при втраті фокусу

Поле з помилкою підсвічується, навіть якщо воно отримало фокус, і його значення не виправили, а потім воно втратило фокус. Підсвічування знімається з поля, як тільки користувач почав виправляти помилкове значення.

Валідація форми після відправки форми, здійснюється наприклад у випадку, коли потрібно перевірити чи заповнені обов’язкові поля або коли перевіряється інформація на унікальність, наприклад коли потрібно, щоб електронна пошта не повторювалась.

Перевірка відбувається після того, як користувач натиснув кнопку відправки даних: всі поля з помилками на формі підсвічуються, сторінка прокручується до першого поля з помилкою, фокус переміщається в це поле, курсор встає в кінець рядка, поруч з полем з'являється виринаючі підказки.

За допомогою такого функціоналу до організаторів буде приходити тільки правильна та точна інформація, яка суттєво скоротить відбір учасників на фестивалі. Користувачам необхідно точно та коректно вводити свої дані, якщо вони хочуть представити свій бренд на події.

Схематичне відображення помилки зображене на рисунку 3.7.

Рисунок 3.7 – Схема відображення помилки після відправлення форми

У невеликих формах замість перевірки заповнення обов'язкових полів можна блокувати кнопку відправки форми. Дана поведінка використовується, коли очевидно, чому кнопка відправки форми неактивна. Наприклад, на формі входу в адмінпанель.

Виринаючі підказки, або тултіпи працюють за настпуним принципом.

Тултіп з'являється:

– при наведенні на поле з помилкою;

– коли поле з помилкою отримує фокус.

Тултіп зникає:

– курсор вийшов за область поля з помилкою;

– поле з помилкою втратило фокус.

Тултіп з'являється з правого боку або зверху, щоб не перекривати поле вводу.

Для роботи з валідаціями використано Java Script бібліотеку, а саме JQury Validation Plugin. В стандартних функціях JQury теж містяться валідації, але в даному плагіні розширені можливості валідації, додано наступні функції:

– перевірка Url та E-mail;

– перевірка максимальних та мінімальних значень, діапазонів чисел та дат;

– автоматичне відображення помилок.

Для підключення потрібно завантажити та підключити в application.js файл jquery.validate.min.js. Наступним кроком є ініціалізація валідатора для конкретної форми, це робиться за допомогою id форми.

```
($("#volunteerform").validate())
```

В атрибуті rules вказуються поля по їх імені і задаються правила для конкретного поля, по яких і буде відбуватись валідація. Фрегмент валідації форми волонтерів наведено нижче:

```
$('#volunteerform).validate({
  rules: {
    'first_name': {required: true},
    'second_name': {required: true},
    'birthday': {
          required: true,
          date: true
     },
    'address': {required: true},
    'email': {
          required: true,
          email: true
     }
  }
});
```

З наведеного фрагменту коду видно, що майжу усі поля є обов'язковими та те що для полів e-mail та дата народження стоять додаткові перевірки, які перевіряють чи введені дані відповідають даті та формату запису e-mail-у.

В параметрі messages задається текст повідомлення про помилку, для кожного правила окремо, зразок повідомлень для форми реєстрації волонтера наведено нижче:

```
    messages: {
        'first_name':   {required:   "Це   поле   обов'язкове   для
заповнення"},
        'second_name':   {required:   "Це   поле   обов'язкове   для
заповнення"},
        'birthday': {
```

```
            required: "Це поле обов'язкове для заповнення",
            date: "Дата введена в невірному форматі"
        },
        'address':  {required:  "Це  поле  обов'язкове  для
заповнення"},
        'email': {
            required: "Це поле обов'язкове для заповнення",
            email:  "Електронна  пошта  вказана  в  невірному
        форматі"
        }
    }
```

Для вказання зовнішнього вигляду повідомлень помилки в CSS, потрібно працювати з класом error. Форма реєстрації учасника налаштована за таким принципом, також додатково вказана маска для роботи з полем номеру телефону. Валідація форми учасника наведена в додатках.

3.3 Розробка панелі адміністратора сайту

Адмін панель розроблена на базі RailsAdmin. RailsAdmin є механізмом Rails, який реалізує легкий у використанні інтерфейс для керування вашими даними.

Особливості та функції:

- CRUD операції;
- користувальницькі дії;
- автоматична валідація форм;
- пошук та фільрування;
- експорт даних в CSV/JSON/XML;
- аутентифікація з використанням Devise та інших гемів;
- авторизація за допомгою CanCanCan або Pundit;
- історія дій користувача;
- підтримка ORM (ActiveRecord Mongoid).

Кроки інсталяції RailsAdmin:

1. Додати гем gem 'rails_admin', '~> 2.0.0.beta'.
2. Запустити bundle install.
3. Запустити rails g rails_admin:install.
4. Налаштувати простір імен для маршрутизації.
5. Налаштувати авторизацію.
6. Запустити Rails сервер та перейти за налаштованим шляхом.

Після успішного встановлення гему, варто налаштувати маршрутизацію, це робиться у файлі routes.rb [3].

```
mount RailsAdmin::Engine => '/admin', as: 'rails_admin'
```

У фрагменті коду, наведено налаштування маршруту для RailsAdmin в проекті. Наступний крок це налаштування авторизації за допомогою гему Devise. Налаштування авторизації вказуються в файлі ініціалі заторі rails_admin.rb. Код для впровадження авторизації наведено нижче.

```
config.authenticate_with do
  if !logged_in? || !current_user.is_superadmin?
    raise ActionController::RoutingError.new('Not Found')
  end
end
config.current_user_method(&:current_user)
```

Також додатково налаштовано, що тільки користувачі з правами суперадміна можуть користуватися адмінпанеллю. У випадку, якщо користувач, який не авторизувався, або не має прав, спробує зайти в панель, то його перенаправить на повідомлення, що даної сторінки не існує. Це робиться в цілях безпеки, тільки організатори події мають доступ до адмінпанелі сайту, де вони зможуть виконувати необхідні операції та бачити відомості про кількість об’єктів на сторінці.

Першою сторінкою при переході в адмінпанель є dashboard (рисунок 3.8).

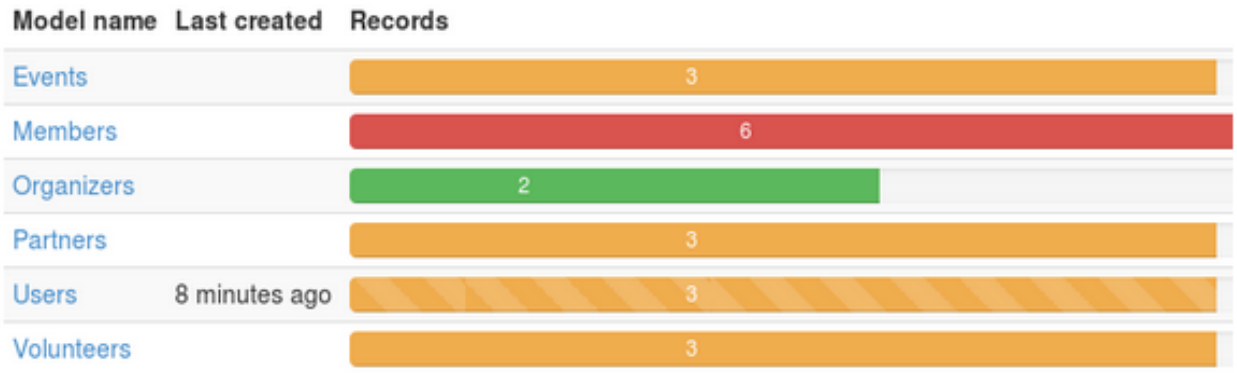


Рисунок 3.8 – Сторінка dashboard

Наведено список усіх моделей адмінпанелі та відомості про кількість об’єктів в базі про кожну модель. Кількість зображена також у вигляді кольорових графіків.

Налаштування об’єктів для адміністрування теж відбувається у файлі rails_admin.rb. Приклад конфігурацій моделі події наведено нижче:

```
config.model 'Event' do
  list do
    exclude_fields :_id
  end
  edit do
    field :title
    field :name
    field :second_name
    field :description, :ck_editor
    field :start_date
    field :end_date
    field :visitors
    field :photogallery
    field :schedule
    field :members
    field :organizers
    field :volunteers
  end
end
```

Вказується назва моделі, потім окремо налаштовуються усі блоки, які відображаються. Блок який відповідає за відображення атрибутів об’єкту в загальному списку об’єктів моделі називається list. В даному випадку вказано,

що будуть відображатися усі атрибути крім id. Результат налаштування зображено на рисунку 3.9.

Title	Name	Second name	Description
Franek Shop	Franek Shop (зимовий)	в Університеті Короля Данила	<p>Офлайн-платформа для моло…
Franek Shop	Franek Shop (літній)	на палаці Потоцьких	<p>Офлайн-платформа для моло…
Franek Shop	Franek Shop	у парку Шевченка	Офлайн-платформа для молодих …

Рисунок 3.9 – Список подій

Блок edit відповідає за відображення атрибутів моделі під час редагування та створення. Вказано усі атрибути крім id, Редагування події зображено на рисунку 3.10.

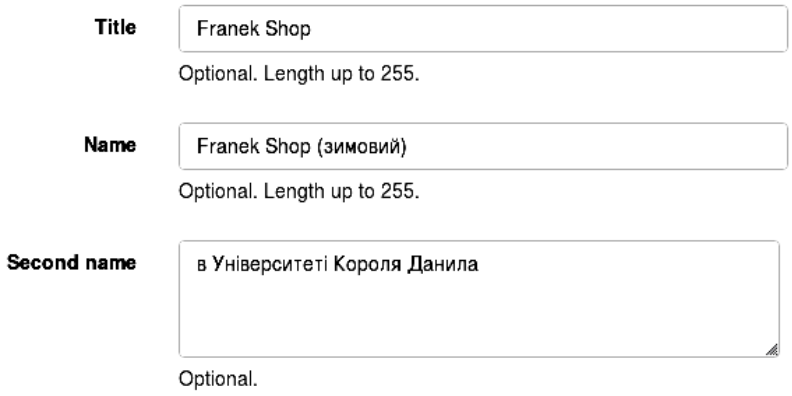


Рисунок 3.10 – Редагування події

Для атрибуту description вказано додатковий параметр ck_editor. CKEeditor це бібліотека, яка реалізує компонентний редактор, в якому можна додавати різноманітний контент [4], правити шрифти, колір тексту, обгортати його в HTML і т.д. Редактор для атрибуту description наведено на риснку 3.11.

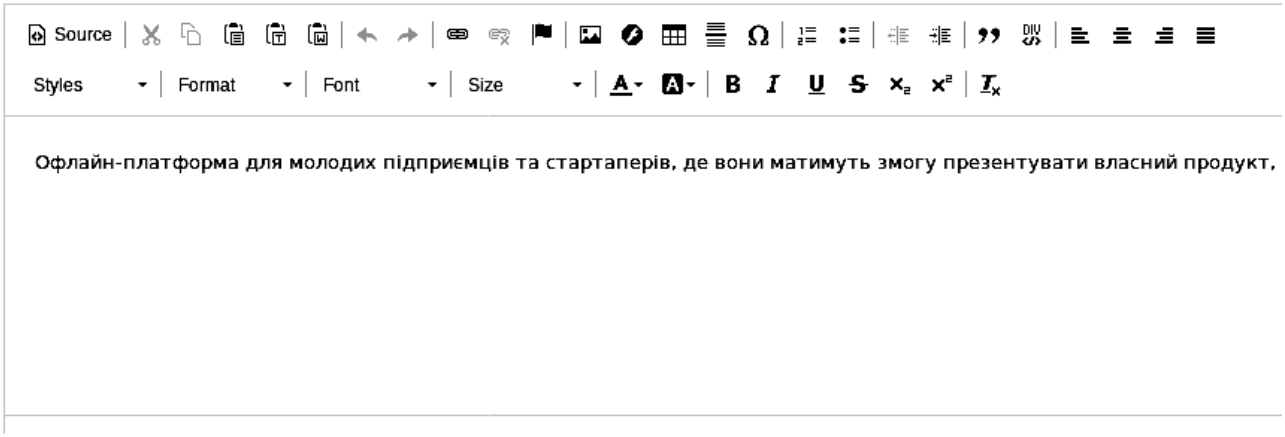


Рисунок 3.11 – Редактор контенту CKEditor для атрибуту description

Також RailsAdmin створює спеціальні поля в залежності від зв’язків. Наприклад, якщо між моделями реалізований, якийсь зв'язок. Реалізація поля для встановлення зв’язку багато до багатьох між моделями Event та Volunteer зображено на риснку 3.12.

Рисунок 3.12 – Поле зв’язку моделей Event та Volunteer

Як видно на рисунку створено модульне поле, в якому можна переміщати конкретні об’єкти з списку в список (належать події або ні), також реалізований пошук по об’єктах та є можливість створити об’єкт волонтера безпосередньо з моделі події.

Для того, щоб об’єкти у таких списках мали вигляд зрозумілий для людини, використовується метод object_label_method. Цей метод використано для моделей Member та Volunteer. Код налаштування моделі Member наведено нижче.

```
config.model 'Member' do
    object_label_method do
      :user_name
    end
    list do
      exclude_fields :_id
      exclude_fields :events
    end
    edit do
      field :first_name
      field :second_name
      field :weight
      field :company
      field :activity do
        read_only true
      end
      field :phone_number do
        read_only true
      end
      field :approved
      field :position
      field :fb_link
      field :photo
    end
  end
```

У методі object_label_method вказується і'мя атрибуту, який описуватиме дану модель. Атрибутам можна налаштовувати безліч функцій, наприклад зробити поле тільки для читання, це робиться за допомогою блоку read_only, який вказується атрибуту (налаштування наведено у фрагменті коду вище). Вигляд поля, яке призначене тільки для читання, зображене на риснку 3.13.

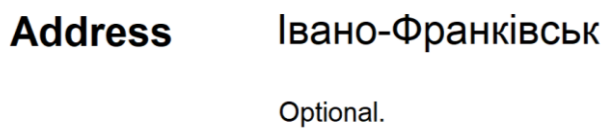


Рисунок 3.13 – Поле призначене тільки для читання

Поле відображається не у формі, а у вигляді звичайного тексту на сторінці. Також адмінпанель підтримує функцію завантаження зображень, вони

відображаються як і при редагуванні об’єкту так і у списках. Завантаження зображення наведено у рисунку 3.14.

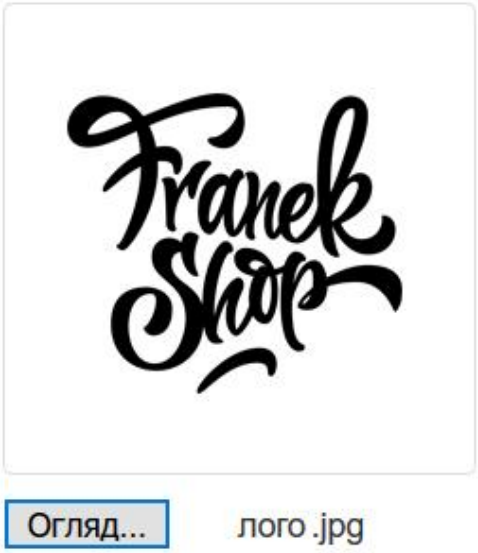


Рисунок 3.14 – Завантаження зображення

Кожен з об’єктів володіє набором стандартних функцій за CRUD. Їх можна створити, видалити, редагувати, відображати їх списки, виділяти і т.д.. Цими функціями можна маніпулювати, їх можна забороняти для окремих моделей. Заборона створення волонтера, наведена в коді нижче.

```
config.actions do
    dashboard
    index
    new
      except [Volunteer']
    end
    export
    bulk_delete
    show
    edit
```

Для налаштування, достатньо вказати назву моделі у списку доступних дій. Також можна реалізовувати власні події. Це робиться за допомогою додавання назви події у список стандартних. Також потрібно створити файли, які реалізовують логіку та відображення, якщо потрібно.

Реалізовано функцію, яка дозволяє відписати людині, що зареєструвалась, як учасник, або волонтер. Код реалізації наведено у додатку. А результат виконання зображено на рисунку 3.15.

Рисунок 3.15 – Custom action відповіді на реєстрацію

Результатом виконання буде надсилання повідомлення на електронну пошту людини, що зареєструвалась, від імені організаторів конференції.

Відправлення повідомлень відбувається за допомогою портативного SMTP серверу, який надається сервісом Gmail. Після підключення сервісу до відповідної електронної пошти, потрібно тільки вказати деякі налаштування для початку роботи з поштовим сервером. Приклад налаштувань наведено в додатках.

Адмінпанель та сайт коректно працюють, усі дані занесені через адмін. Панель, правильно відображаються на сторінках сайту. Реалізований функціонал адмінпанелі сповна задовольняє поставлені вимоги, для адміністрування та модерування сайту.

ВИСНОВКИ

В дипломній роботі розроблено сайт, який швидко і детально інформує користувача про фестивалі Franek Shop. А також реалізовано функціонал для розміщення фото, відео та форм реєстрації.

Дану систему розроблено відповідно до потреб потенційних користувачів. Потреби користувачів будувались на основі порівняльного аналізу інформаційних ресурсів. Побудовано use case для усіх типів користувачів.

Сайт розроблений за допомогою стандартних методів та технологій веб-розробки. З огляду вимоги специфікації, клієнтська сторона створена за допомогою фреймворку для розмітки HAML, каскадних стилів CSS та сценаріїв сторінок JavaScript. Також використано бібліотеку для роботи з даними, яка використовується в якості адмінпанелі сайту.

Розробка сайту зумовила розширення знань в області створення веб-додатків з використанням Ruby on Rails.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Структура веб-сайту. URL: https://naurok.com.ua/urok-struktura-veb-saytiv-riznovidi-veb-saytiv-riznovidi-veb-storinok-33309.html (дата звернення: 20.03.2019)
2. Github devise. URL: https://github.com/plataformatec/devise (дата звернення: 25.03.2109)
3. Rails admin . URL: https://github.com/sferik/rails_admin (дата звернення: 02.04.2019)
4. Ruby on Rails: руководство по стилю оформления. URL: https://github.com/arbox/ruby-style-guide/blob/master/README-ruRU.md (дата звернення: 07.04.2019)
5. INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF0). Draft Federal Information Processing Standards Publication 183 ,1993 December 21.
6. Вступ в Java Script. URL: https://developer.mozilla.org/JavaScript/Guide/Introduction (дата звернення: 21.04.2019)
7. Основи CSS. URL: https://developer.mozilla.org/uk/docs/Learn/CSS_basics (дата звернення: 25.04.2019).
8. Mongo DB. Матеріал з Вікіпедії - вільної енциклопедії. URL https://ru.wikipedia.org/wiki/MongoDB (дата звернення: 05.05.2019)
9. MVC. Матеріал з Вікіпедії – вільної енциклопедії. URL: https://uk.wikipedia.org/wiki/модель-вид-контроллер (дата звернення: 10.05.2019).
10. UML. Матеріал з Вікіпедії – вільної енциклопедії. URL: https://uk.wikipedia.org/wiki/Unified_Modeling_Language (дата звернення: 12.05.2019).

11. Сценарій використання. Матеріал з Вікіпедії – вільної енциклопедії. URL: https://uk.wikipedia.org/wiki (дата звернення: 20.05.2019)
12. Ашманов И.С., Продвижение сайта в поисковых системах. М.: Вильямс, 2007. 304 с.
13. Гото К. Веб-редизайн. М.: Символ-Плюс, 2006. 416 с.
14. Гото Келлі Веб-редизайн, 2-е видання. СПб.: Символ Плюс, 206. 416 с.
15. Кантор Марри. Управление программными проектами. Издательство: Вильямс, 2002 г. 176 с.
16. Петлюшкин А.В., HTML в Web-дизайне. СПб.: БВХ-Петербург, 2004. – 400 с.
17. Постановка задачи при проектировании сайта. URL: http://site-manager.ru/webmasters/lib/110/ (дата звернення: 28.05.2019)
18. Програмування по-українськи - А що таке HTML? URL: http://programming.in.ua/web-design/html/73-html-introdution.html. (дата звернення: 28.05.2019)
19. Прототип сайту – як він допомагає оцінювати і розробляти сайти. URL: http://evergreens.com.ua/ua/articles/prototype-site.html (дата звернення: 25.05.2019)
20. Севостьянов И.О., Поисковая оптимизация. Практическое руководство по продвижению сайта в Интернете. СПб.: Питер, 2010. 240 с. 9
21. Современный учебник Javascript Введение в – AJAX. URL: https://learn.javascript.ru/ajax-intro (дата звернення: 28.05.2019)
22. Типи сайтів. URL: https://internetdevels.ua/blog/most-common-types-of-websites (дата звернення: 29.05.2019)
23. Уэйншенк С. Интуитивный веб-дизайн. М.:, Эксмо, 2011.160 с.
24. Фрейн Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств. СПб.: Питер, 2014. 304 с.
25. Ruby on Rails. Матеріал з Вікіпедії – вільної енкиклопедії. URL: https://ru.wikipedia.org/wiki/Ruby_on_Rails (дата звернення: 29.05.2019)

26. HTML. Матеріал з Вікіпедії – вільної енкиклопедії. URL: https://ru.wikipedia.org/wiki/HTML (дата звернення: 29.05.2019)
27. HAML. URL: http://haml.info/tutorial.html (дата звернення: 30.05.2019)
28. Use case. Матеріал з Вікіпедії – вільної енциклопедії. URL: https://en.wikipedia.org/wiki/Use_case (дата звернення: 30.05.2019).
29. User story. Матеріал з Вікіпедії – вільної енкиклопедії. URL: https://en.wikipedia.org/wiki/User_story (дата звернення: 30.05.2019)