СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра програмування та математики

ПОЯСНЮВАЛЬНА ЗАПИСКА
до кваліфікаційної випускної роботи

освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)
спеціалізація „Інженерія програмного забезпечення”

на тему „Розробка веб-платформи для створення сучасних односторінкових сайтів”

Виконав: студент групи ІПЗ-16д _____ Д.О. Савченко
(підпис) (ініціали і прізвище)

Керівник _____ Д.М. Марченко
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ В.О. Лифар
(підпис) (ініціали і прізвище)

Рецензент_____

Сєвєродонецьк – 2020

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра програмування та математики

Освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)
спеціалізація „Інженерія програмного забезпечення”
(назва спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМ,
д.т.н., доцент
_____ Лифар В.О.
“____” _____2020 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

Савченко Денис Олександрович
(прізвище, ім’я, по батькові)

1.Тема роботи Розробка веб-платформи для створення сучасних односторінкових сайтів.

Керівник роботи професор, д.т.н.Марченко Дмитро Миколайович,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджений наказом університету від “___”_____20__року №___

2. Строк подання студентом роботи 20 травня 2020 р.

3. Вихідні дані до роботи Об'єктом даної роботи є процес розробки сучасного односторінкового сайту.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ. Аналітичний огляд, з висвітленням наступних питань: браузери та протоколи, основи веб-розробки, протоколи, API та формати даних, клієнт і сервер. Основна частина, в якої висвітлити: збір вимог та проектування, етап даталогічного проектування. Висновки. Перелік використаних джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслеників)

6. Консультанти розділів роботи

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

7. Дата видачі завдання__30 березеня 2020 року._____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Одержання завдання на виконання роботи	30.03.20	
2	Укладання і погодження з керівником плану і етапів виконання роботи	06.04.20	
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	13.04.20	
4	Аналіз шляхів виконання завдання. Вибір і погодження керівником оптимального шляху	20.04.20	
5	Укладання та тестування програмного продукту	27.04.20	
6	Укладання, оформлення та погодження пояснювальної записки з керівником	04.05.20	
7	Здача готової пояснювальної записки на кафедру	12.05.20	
8	Укладання доповіді і презентації	30.05.20	

Студент _____ ___Д.О. Савченко_____
(підпис) (ініціали і прізвище)

Керівник роботи _____ ____Д.М. Марченко_____
(підпис) (ініціали і прізвище)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. ІПЗ-16д Савченко Д.О.

Науковий керівник

Доцент, д.т.н. _____ Марченко Д.М.

Оцінка наукового керівника: _____

Рецензент _____

ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

_____

Голова ЕК

_____ Лифар В.О.

підпис

РЕФЕРАТ

Робота містить: 47 сторінок основного тексту, 13 сторінок додатків, 43 рисунка, 1 таблицю, 10 використаних джерел.

Метою випускної кваліфікаційної роботи є вивчення особливостей роботи конструкторів сайтів, методів їх реалізації, оптимальних рішень, в плані програмування і підбору продуктивних інструментів.

Було проаналізовано багато конструкторів сайтів, їх принцип роботи, перелік наданих можливостей і кращі рішення взаємодії з користувачами.

Багато часу було приділено технічній частини, а саме підбору продуктивного PHP фреймворку і СКБД, яка підійде під данній проект.

В результаті виконаної роботи, було реалізовано веб-додаток, що дозволяє створювати сучасні односторінкові сайти, без навичок програмування для користувача.

Веб-додаток може бути легко масштабованим для подальшої розробки даного проекту, завдяки шаблону MVC.

Система реалізована відповідно всім вимогам технічного завдання.

Зроблено детальний опис процесу розробки системи, а також, продемонстрована робота готового веб-додатка.

ЗМІСТ

ВСТУП

Актуальність. У той час як традиційні системи управління вмістом (CMS) були розроблені для адміністрування великих веб-сайтів, конструктори веб-сайтів розроблялися з урахуванням менших веб-сайтів.
Це означає, що користувачеві конструктора сайтів не потрібні спеціальні навички програмування чи інші технічні знання, наприклад такі, як встановити CMS, застосувати шаблон або створити базу даних. Можна створити повністю функціональну присутність в Інтернеті за допомогою конструктора веб-сайтів. Оновлення тексту, на створенному веб-сайті або додавання нових статей, зображень або мультимедійного вмісту, таких як відео чи аудіо, також легко зробити через зручний інтерфейс. [1]

Об’єкт дослідження: процес розробки сучасного односторінкового сайту.

Предмет дослідження: веб-сервіс для розробки односторінкових сайтів.

Мета дослідження: розробка веб-платформи для створення сучасних односторінкових сайтів без навичок програмування.

Задачі дослідження:

1. Ознайомитися з принципом роботи сучасних конструкторів сайтів і сформувати список основних функцій.
2. Сформувати найкращий стек технологій і мов, для розробки власного конструктора односторінкових сайтів.
3. Розробити веб-платформу для створення сучасних односторінкових сайтів.

1 АНАЛІТИЧНИЙ ОГЛЯД

Веб-розробка поставляється з величезним набором правил і методів, про які повинен знати кожен розробник веб-сайту. Розробка веб-сайту зазвичай зводиться до знання 3 основних мов: JavaScript, CSS і HTML, а також.

Веб-технології - це загальний термін, що до багатьох мов і мультимедійних пакетів, які використовуються в поєднанні один з одним для створення динамічних веб-сайтів. Кожна окрема технологія досить обмежена сама по собі і вимагає подвійного використання принаймні ще однієї такої технології. Можна зробити висновок, що всі компоненти, складові сайт, взаємозалежні один від одного.

1.1. Браузери

Браузери подають запит на інформацію, а потім вони показують інформацію в доступному виді.

Найпопулярніші з них:

- Google Chrome - в даний час найпопулярніший браузер
- Safari - веб-браузер Apple
- Firefox - браузер з відкритим кодом, підтримуваний Mozilla Foundation
- Internet Explorer - браузер Microsoft

1.2. HTML & CSS

HTML - це мова для опису структури веб-сторінок. HTML дає авторам засоби для:

- Публікація онлайн-документів з заголовками, текстом, таблицями, списками, фотографіями тощо.
- Отримати інформацію онлайн через гіпертекстові посилання, одним натисканням кнопки.

- Розробка форм для проведення транзакцій з віддаленими сервісами, для використання при пошуку інформації, бронюванні, замовленні товарів тощо
- Додати електронні таблиці, відео, звуки та інші додатки безпосередньо в свої документи.

CSS - це мова для опису представлення веб-сторінок, включаючи кольори, макет та шрифти. Це дозволяє адаптувати вид до різних типів пристроїв, таким як великі екрани, маленькі екрани або принтери. CSS не залежить від HTML і може використовуватися з будь-якою мовою розмітки на основі XML. Відділення HTML від CSS полегшує обслуговування сайтів, спільне використання таблиць стилів на різних сторінках і адаптацію сторінок до різних середовищ. Це називається відділенням структури (або вмісту) від уявлення. [2]

1.3. Основи веб-розробки

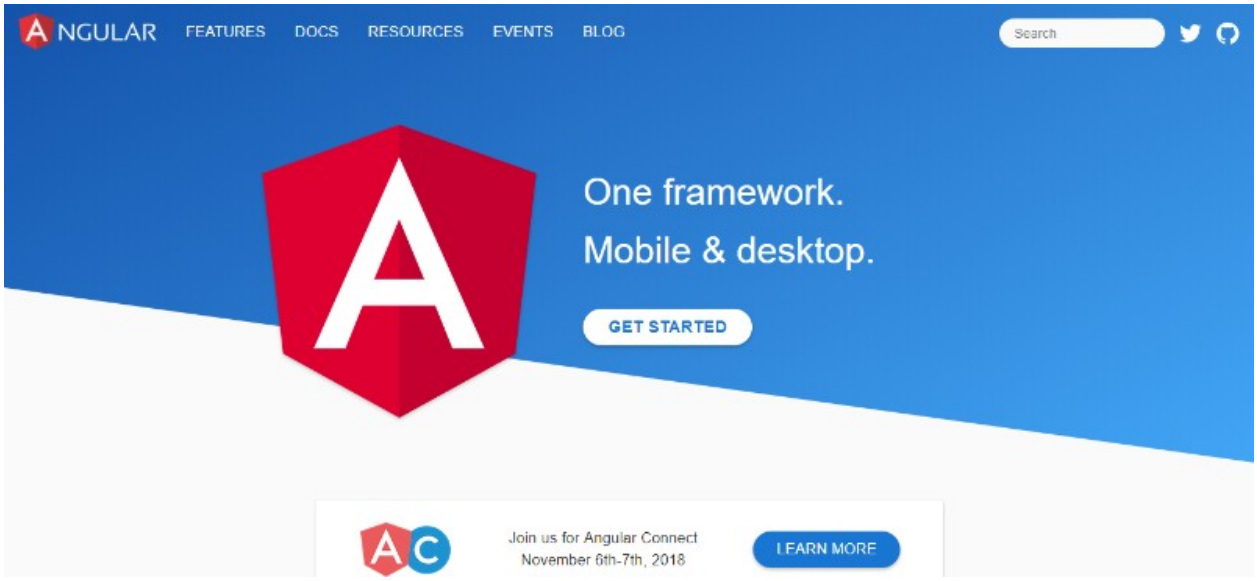

Рисунок 1.1. Фреймворк Angular

Angular - це одна з новітніх веб-технологій, розроблена спеціально для розробки динамічних веб-додатків. За допомогою цього фреймворка можна легко створювати інтерфейсні додатки без необхідності використання інших фреймворків або плагінів. В число функцій входять добре зроблені шаблони, архітектура MVC, генерація коду, розбиття коду тощо. Всі вирази схожі на

фрагменти коду, укладені у фігурні дужки, і не використовують ніяких циклів або умовних операторів.

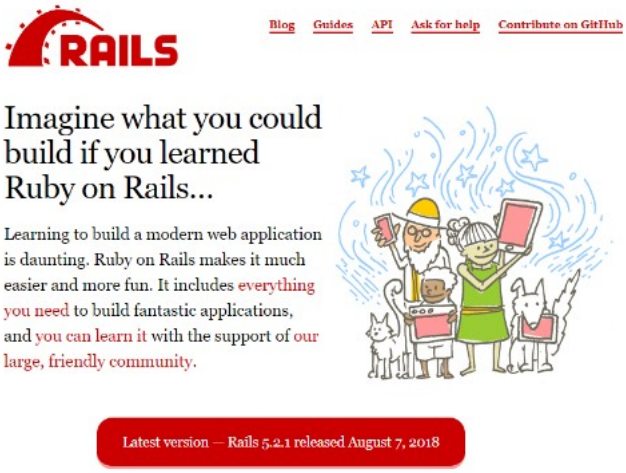

Рисунок 1.2. Фреймворк Ruby on Rails

Ruby on Rails - це серверна веб-технологія, яка робить розробку додатків набагато простіше і швидше. Те, що дійсно відрізняє цю платформу, - це можливість багаторазового використання коду, а також деякі інші цікаві функції, які допоможуть вам виконати роботу в найкоротші терміни.

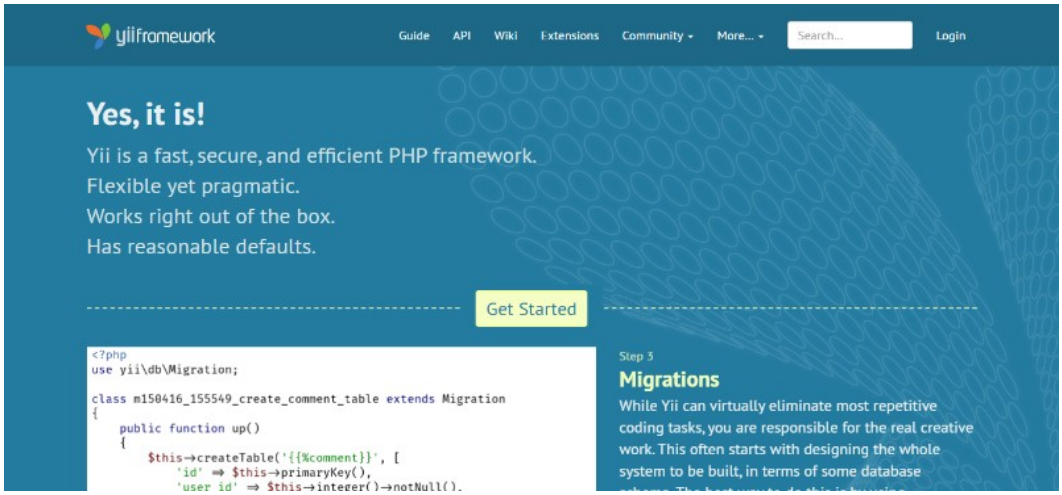

Рисунок 1.3. Фреймворк Yii

Yii - це середовище розробки веб-додатків з відкритим вихідним кодом, побудована на PHP5. Він оптимізований по продуктивності і поставляється з низкою відмінних інструментів для налагодження і тестування програм. Ще один плюс в тому, що він досить простий і зручний у використанні.

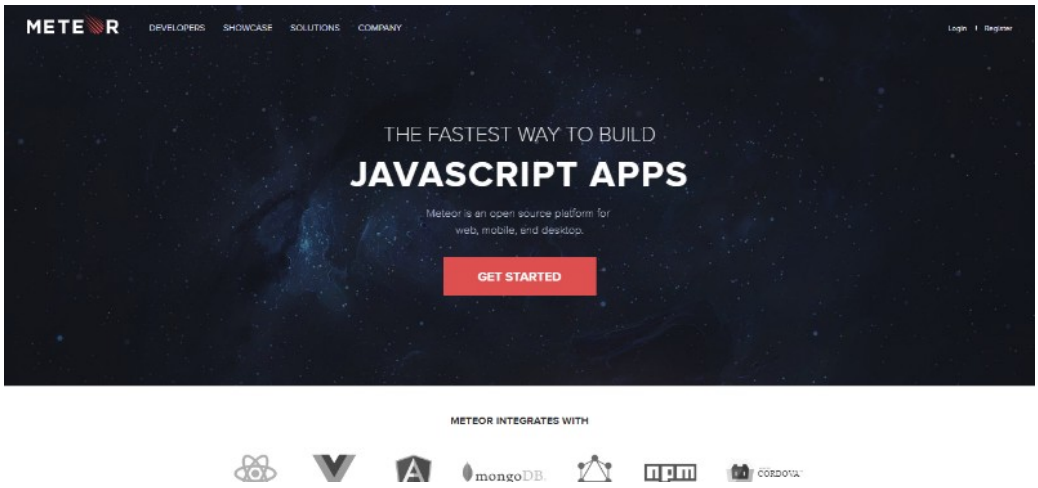

Рисунок 1.4. Фреймворк Meteor JS

Meteor JS написаний на Node.js і дозволяє створювати веб-додатки в реальному часі для різних платформ. Середовище для створення простих веб-сайтів для особистого використання дійсно виділяється з Meteor JS. Це ізоморфна веб-інфраструктура JavaScript з відкритим вихідним кодом, яка також означає, що час завантаження веб-сторінки значно коротше. Стек JavaScript також дозволяє отримати ті ж результати з меншою кількістю рядків коду, ніж зазвичай.

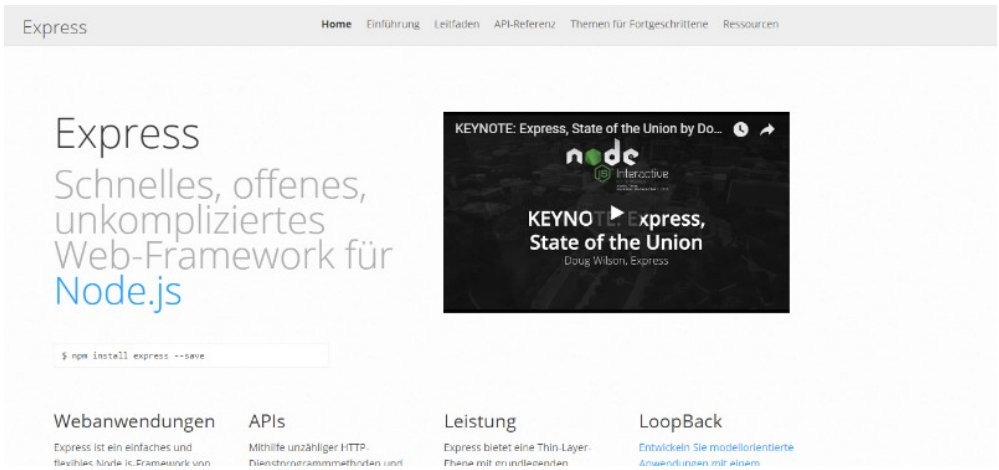

Рисунок 1.5. Фреймворк Express.js

Express.js, створений на Node.js, являє собою мережу для розробки веб-додатків, яка відмінно підходить для тих, кому необхідно розробляти програми і API-інтерфейси якомога швидше. Безліч чудових функцій надається за допомогою плагінів.

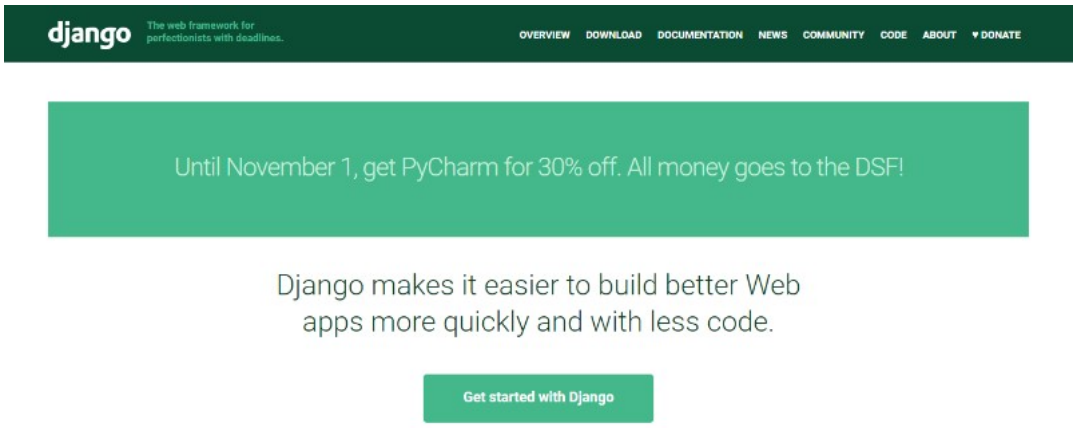

Рисунок 1.6. Фреймворк Django

Django є одним з найпопулярніших фреймворків, написаних на Python, і архітектурі MVC. Це робить процес розробки додатків набагато простіше завдяки своїй простоті. Django значно спрощує використання Python і надає безліч інструментів, що полегшують життя розробникам веб-додатків - наприклад, ORM, Моделі, адміністратор Django, шаблони тощо.

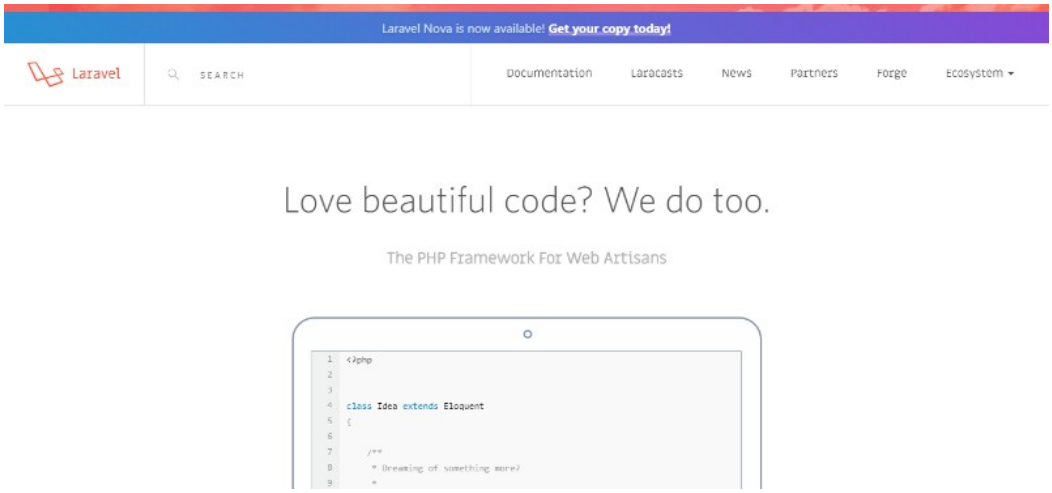

Рисунок 1.7. Фреймворк Laravel

Laravel - це середовище розробки PHP, ідеальний для середніх сайтів. Він поставляється з рядом корисних функцій, включаючи підтримку MVC, об'єктно-орієнтовані бібліотеки, Artisan, техніку авторизації, міграцію бази даних тощо. В даний час це одна з найбільш підтримуваних спільнотою і розроблених співтовариством середовищ, і з огляду на, що PHP має Laravel, одне з найбільших угруповань, є відмінним інструментом, який підтримує як невеликі веб-сайти, так і великі веб-додатки B2B, щодня керуючи мільйонами транзакцій.

1.4. Протоколи

Детальна інформація про передачу інформації між комп'ютерами і пристроями зазвичай називаються протоколами.

HTTP - завдяки цьому протоколом кожен сайт може потрапити в браузер. Протокол запитує сайт з сервера Google, а потім отримує відповідь з HTML, CSS і JavaScript сайту.

DDP - Використовує веб-сокети для створення узгодженої зв'язку між клієнтом і сервером. В результаті буде оновлення сайту в режимі реального часу без необхідності оновлювати браузер.

Використовуваний в основному для API, цей протокол має стандартні методи, такі як GET, POST і PUT, які дозволяють обмінюватися інформацією між додатками.

1.5. API

API (інтерфейс прикладного програмування) дозволяє іншим розробникам використовувати деякі функції програми без спільного використання коду. Кінцеві точки надаються розробниками, в той час як API може контролювати доступ за допомогою ключа API. Прикладами добре зроблених API є ті, які створені Facebook, Twitter і Google для своїх веб-сервісів.

1.6. Формати даних

Дані зберігаються в структурі, званій форматом даних.

JSON - JavaScript Object Notation - це синтаксис для зберігання та обміну даними (як XML). В даний час це стає найпопулярнішим форматом даних там.

XML - переважно використовується системами Microsoft, він був найпопулярнішим форматом даних

CSV - дані, відформатовані запитом; наприклад дані Excel

1.7. Клієнт

Кожен користувач програми називається клієнтом. Клієнтами можуть бути комп'ютери, мобільні пристрої, планшети тощо. Зазвичай кілька клієнтів взаємодіють з одним додатком, що зберігається на сервері.

1.8. Сервер

Код програми зазвичай зберігається на сервері. Клієнти роблять запити до серверів. Потім сервери відповідають на ці запити після збору запитаної інформації. [3]

2 МОДЕЛЬ ПРОЕКТУ ТА ПОСТАНОВКА ЗАДАЧІ

Реалізація проекту «Конструктор сайтів» буде основана на моделі водоспаду.

Реалізація моделі водоспаду (або, інакше, каскадна модель) вимагає високої кваліфікації всіх розробників проекту повного розуміння кожного етапу, прорахунок ризиків «наперед» і таким чином, щоб ризик за кожним етапом був першою задачею, на локалізацію якої націлені перші кроки розробників.

На рисунку 2.1. наведені впорядковані етами моделі водоспаду:

–збір вимог до системи й програмного забезпечення;

–аналіз;

–проектування;

–кодування й тестування модулів;

–інтеграція системи;

–тестування додатка в цілому;

–передача в експлуатацію.

Для переходу до наступного етапу необхідно повністю закінчити роботу над поточним і ретельно описати його результати.

Модель водоспаду використовується не часто, бо по потребує надмірних витрат часу, відслідковування необхідних змін, надмірний ризик і неможливість переглядати вимоги до проекту. Як правило, така модель використовується спеціалізованими організаціями-розробниками, в яких окремі проекті групи займаються роботами за напрямами:

– інтеграцією всіх підсистем і постійний моніторинг роботи окремих додатків в часі;

– організацією і координацією групи експертів, які визначають необхідність зміни проекту на пізніх стадіях життєвого циклу;

– моніторингом ризиків проекту і розробкою пропозицій з їх мінімізації;

– моніторинг, аналіз і рецензування відповідності вимог замовника до бачення виконання задачі розробником (замовник на початку роботи не завжди чітко усвідомлює, що йому треба).

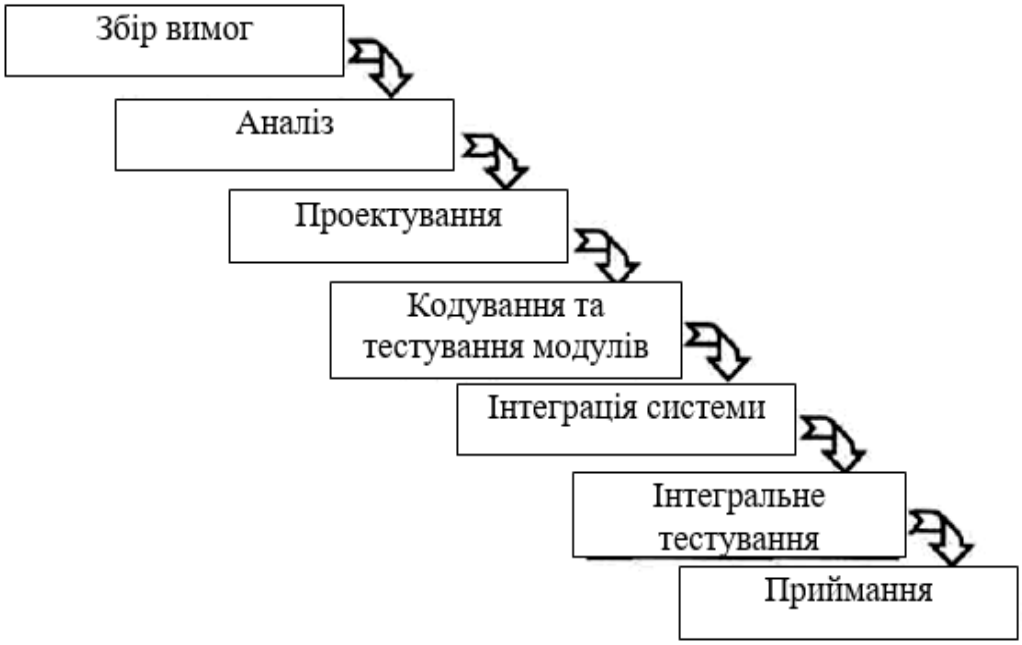

Рисунок 2.1. Модель водоспаду

При реалізації моделі водоспаду часто виникають питання щодо обмеження обміну інформацією між робочими групами на кожній стадії проекту. Перші чотири етапи моделі – великі обсяги паперової роботи, яка і є основним джерелом інформації для інших груп розробників. Часто роботи окремих груп розробників дублюється через необхідність проведення нарад, щоб швидше вводити нових фахівців, що приступають до роботи на новому етапі, в курс справи. Це призводить до того, що виконаний етап роботи не перевіряється, а приймається, як завідомо вірний.

Сутності для БД конструктора:

- сутність «проект»
- сутність «користувач»
- сутність «рахунок»
- сутність «роль»
- сутність «роль користувача»
- сутність «міграція»

2.1 Збір вимог

1. Дизайн. Простий, ненав'язливий, але оригінальний - головні критерії.
2. Грамотний призначений для користувача інтерфейс. Максимально зручний, орієнтований на цільову аудиторію.
3. Логічна навігація. Основні розділи сайту (Головна | Про проект | FAQ | Контакти | Кабінет користувача | Середовище розробки сайтів | Вхід | Реєстрація | Відновлення паролю).
4. Розташування посилань внизу. Необхідно продублювати. Це додатковий спосіб привернути увагу.
5. Якісний та унікальний контент.
6. Контакти. Це обов'язкова умова: зв'язок повинен бути зручним.
7. Зрозуміла документація по використанню конструктора. Схема роботи тарифів.
8. Реєстрація / Підписка. Максимально спрощена реєстрація та підписка.
9. Карта сайту. Допомагає скласти читачеві повне уявлення ресурсу, а також сприяє індексації сторінок.
10. Відокремлення контенту від дизайну. Використання CSS-файлів для того, щоб пошукові системи могли, минаючи надлишкові коди, ідентифікувати контент.
11. Використання HTML / CSS. Крім того, коди повинні бути валідними. Ресурси перевірки коду: W3C Markup Validation Service, W3C CSS Validation Service.
12. Сумісність браузерів. Сайт повинен бути сумісний з усіма браузерами та платформами, з якими працюють користувачі.
13. Оптимізація зображень та коду. Стиснути все медіафайли і призначений для користувача код, для оптимізації швидкості завантаження контенту і зручності користування системою.
14. Статистика. Встановити і налаштувати аналітику поведінки користувачів на сайті. Необхідно для отримання детальної інформації користувацького досвіду. [5]

Якщо сайт не проглядається або відображається некоректно по-четвертим версіях браузерів, то втрачається значна частина відвідувачів. Обсяг HTML-коду повинен прагнути до мінімуму з двох причин. Перша, і найголовніша, викликана швидкістю завантаження вашого HTML-документа користувачем. Користувач не буде довго чекати. Якщо сторінка не з'явилася протягом однієї хвилини, то швидше за все, користувач піде на інший веб-сайт. Друга причина пов'язана з сумісністю з основними версіями браузерів і подальшим супроводом сайту. Практика показує, що чим більший об'єм HTML-коду, тим складніше домогтися, щоб він однаково відображався в різних браузерах. Тут мається на увазі не абсолютний розмір HTML-коду. Справа в тому, що одного і того ж результату можна домогтися різними способами. Наприклад, при форматування вмісту HTML-документа, одного і того ж результату можна домогтися різними вихідним HTML-кодом.

Надалі, після завершення робіт над сайтом, його доведеться підтримувати, тобто доповнювати і змінювати. Для швидкої і успішної модифікації сайту необхідно, щоб:

1. Сайт мав зрозумілу структуру.
2. Вихідний HTML-код повинен бути читабельним.
3. Верхні і нижні колонтитули були винесені в окремий файл.
4. CSS (таблиці каскадних стилів), що використовуються в декількох документах перебували в окремому файлі.
5. Функції JavaScript, що використовуються в декількох документах перебували в окремому файлі.
6. Гіперпосиланнях мали правильні шляхи - абсолютні та відносні.
7. В іменах файлів і адресах гіперпосилань використовувалися тільки цифри і букви англійського алфавіту нижнього регістру, також допускається використання символу підкреслення. Всі інші символи російського алфавіту, англійські символи верхнього регістру значно можуть ускладнити підтримку веб-сайту.

Відповідність існуючим стандартам забезпечує:

- доступність - це означає, що веб-сторінки можна переглядати за допомогою найрізноманітніших браузерів і пристроїв.
- сумісність від низу до верху. Майбутні стандарти будуть створюватися на основі нинішніх, тому у сайтів, які слідують стандартам сьогодні, не буде проблем з відображенням в нових версіях браузерів.

більш швидке завантаження і відображення, так як смислова розмітка дозволяє зменшити загальний обсяг файлу (що впливає на швидкість передачі і завантаження) і скорочує час обробки файлу браузерами, різними пристроями, іншими програмами, в тому числі і пошуковими системами. Крім того, швидке відображення сторінок подобається відвідувачам. [5]

2.2. Проектування

Проектування – це процес створення прообразу системи, пошук такого способу створення системи, що задовольняє вимогам функціональності зважаючи на задані обмеження та наявні технології.

Інтерфейс**:**

1. Розробка інтерфейсу з використанням сучасних інструментів, мов розмітки і програмування.
2. Адаптивна верстка. Коректне відображення інтерфейсу на всіх пристроях.
3. Оптимізований код і медіафайли, для швидкого завантаження і роботи інтерфейсу.

Серверна частина:

1. Розробка серверної частини з використанням сучасних технологій і інструментів, а також стабільних версій мов програмування і модулів веб сервера.
2. Використання популярного і надійного фреймворка Laravel.
3. Використання СКБД MySQL.
4. Використання хостингу з характеристиками, необхідними для стабільного функціонування додатку.

2.2. Етап даталогічного проектування

Виходячи з вимог технічного завдання, в якості СКБД був обраний продукт – MySQL.

MySQL - це система управління реляційними базами даних з відкритим кодом, заснована на структурованій мові запитів (SQL). MySQL доступна у всіх основних операційних системах, включаючи Windows , Linux та Solaris.

Вона безкоштовна для використання для фізичних осіб та виробничих середовищ за загальною ліцензією GNU; однак, якщо використовується комерційно, потрібна комерційна ліцензія.

MySQL, як і інші реляційні бази даних, зберігає дані в таблицях, стовпцях та рядках. Кожен запис визначається унікальним ідентифікатором. Її основою завжди були продуктивність та надійність бази даних. MySQL була розроблена та оптимізована для арени веб-розробок; це, мабуть, найпоширеніша база даних, яка використовується при розгортанні веб-серверів. MySQL дуже добре працює з Apache та PHP і часто переходить до бази даних для розгортання стеків LAMP . Сьогодні MySQL використовує 9 з 10 веб-сайтів в Інтернеті, і ця база даних, обрана Facebook, Twitter та Wikipedia.

MySQL виявилася успішною, оскільки вона проста у використанні, проста в установці та використовує мову запиту, яку легко зрозуміти. Наприклад, команда SHOW DATABASES відобразить список доступних баз даних у MySQL. Вона була розроблена для людей, які не є експертами в управлінні базами даних, надаючи можливості досвідченим користувачам, в той же час пропонуючи достатню складність для обслуговування передових DBA.

Версії MySQL

З 2007 року MySQL почав представляти різні версії свого основного продукту MySQL. Ці видання були зосереджені на спільнотах та корпоративних версіях MySQL. Ця зміна стратегії запропонувала користувачам вибір, оскільки обидва додатки використовують один і той же

вихідний код, але пропонувались з різними рівнями підтримки, доступними для кінцевого користувача.

Видання підтримується глобальною спільнотою з відкритим кодом MySQL через центр документації, онлайн-блоги, канали IRC та форуми. Видання для підприємств підтримуються спеціалізованими командами технічної підтримки, які пропонують виправлення та підтримку пакетів послуг, а останнім часом включають ексклюзивні функції підприємства, такі як Enterprise Backup, Enterprise Monitor, Enterprise Security та Enterprise Audit модулі.

Інструменти управління MySQL

Існує кілька інструментів для управління та підтримки MySQL. У типових установках Linux управління БД здійснюється за допомогою командного рядка за допомогою клієнта MySQL. Клієнт MySQL може використовуватися для створення баз даних, налаштування дозволів та прав доступу. Крім того, існує кілька популярних інструментів графічного інтерфейсу, включаючи SQLyog, phpMyAdmin, браузери MySQL Query, Navicat, MySQL Administrator та MySQL Workbench. [10]

Після опису сутностей і установки зв'язків отримаємо наступну даталогічну модель основної бізнес-логіки:

Рисунок 2.2. Даталогічна модель бази даних проекту

3 ВИБІР МЕТОДІВ ТА РЕАЛІЗАЦІЯ ПОСТАВЛЕНИХ ЗАДАЧ

Laravel - це фреймворк, написаний на PHP.

Як і інші фреймворки, він поширюється безкоштовно і має відкритий вихідний код, який можна знайти на GitHub.

Laravel використовується при створенні додатків з моделлю MVC (Model View Controller - модель-уявлення-контролер); багато хто вважає його одним з кращих MVC фреймворків (в тому числі і тому що у Laravel велика спільнота).

Цей фреймворк стабільно потрапляє в списки найбільш популярних, перспективних і використовуваних PHP фреймворків і отримує такі звання як кращий фреймворк корпоративного рівня і кращий фреймворк для особистих проектів.

3.1. Розробка на Laravel

Додаток на Laravel можна розробляти на будь-якій операційній системі. Можна використовувати IDE (інтегроване середовище розробки) на свій вибір, наприклад PhpStorm (любителі текстових редакторів можуть вибрати Atom або Sublime Text).

Тут ще може стати в нагоді Laravel IDE Helper Generator - пакет, який генерує файл-хелперів, в якому містяться статичні класи фасадів.

Другим корисним інструментом стане Composer, який дозволяє оновлювати фреймворк і завантажувати в проект додаткові пакети. Його можна завантажити за цим посиланням. Більшість PHP-пакетів мають мінімум залежностей і тому можуть бути легко додані в додаток.

При написанні будь-якого проекту його потрібно тестувати - в Laravel представлені функціональні тексти (Feature-тести), перевіряючі функціонал проекту (з точки зору користувача), і модульні тести (Unit-тести), які перевіряють саму логіку проекту.

Ще один корисний інструмент - Laravel Debugbar, пакет, який дозволяє контролювати і налагоджувати код (інтегрується в PHP Debug Bar). Він

відстежує запити, наприклад, дозволяє відстежити SQL-запити для їх оптимізації.

Для складання проекту знадобиться Laravel Mix. Це API використовується для визначення інструкцій збірки Webpack для Laravel додатки.

Laravel - це потужний і універсальний інструмент розробки з можливістю масштабування, хорошим вбудованим механізмом кешування і високою швидкістю розробки.

Laravel йде в ногу з часом, він змінюється і допрацьовується, це сучасний фреймворк, що підходить для широкого кола завдань.

Laravel дозволяє використовувати сервіс-провайдер (service provider), завдяки якому можна централізовано підключати необхідні компоненти програми.

У Laravel можна легко розширювати будь-які компоненти.

Також окремо можна відзначити зручну маршрутизацію і валідацію параметрів.

Laravel дає можливість працювати з різними базами даних, змінювати їх структуру, відкочувати зміни тощо.

Сайти на Laravel відрізняються:

- широким функціоналом (можна зробити проект з практично будь-яким необхідним функціоналом);
- зручною адміністративною панеллю (можна зробити панель конкретно під певний проект і його завдання);
- високим рівнем безпеки баз даних (сайти надійно захищені від SQL-ін'єкцій);
- масштабованість (функціонал проекту можна легко розширити).

[6]

3.2. Особливості Laravel

Laravel пропонує такі основні функції, що робить його ідеальним вибором для розробки веб-додатків:

Модульність

Laravel пропонує 20 вбудованих бібліотек та модулів, що сприяє розширенню програми. Кожен модуль інтегрований із менеджером залежності композитора, який полегшує оновлення.

Заповітність

Laravel включає функції та помічники, що допомагає в тестуванні за допомогою різних тестових випадків. Ця функція допомагає підтримувати код відповідно до вимог.

Маршрутизація

Laravel надає користувачеві гнучкий підхід до визначення маршрутів у веб-програмі. Маршрутизація допомагає краще масштабувати додаток та підвищує його ефективність.

Управління конфігурацією

Веб-додаток, розроблений в Laravel, буде працювати в різних середовищах, а це означає, що буде постійно змінюватися його конфігурація. Laravel забезпечує послідовний підхід до ефективної обробки конфігурації.

Builder запитів та ORM

Laravel включає в себе конструктор запитів, який допомагає в запиті баз даних, використовуючи різні прості методи ланцюга. Він забезпечує ORM (Object Relational Mapper) та реалізацію ActiveRecord під назвою Eloquent.

Схема побудови

Schema Builder підтримує визначення бази даних та схеми в PHP-коді. Він також підтримує відстеження змін щодо міграції бази даних.

Шаблон двигуна

Laravel використовує двигун шаблону Blade, легку мову шаблонів, що використовується для проектування ієрархічних блоків та макетів із заздалегідь визначеними блоками, що містять динамічний вміст.

Електронна пошта

Laravel включає поштовий клас, який допомагає надсилати пошту з багатим вмістом та вкладеннями з веб-програми.

Аутентифікація

Аутентифікація користувача є загальною ознакою у веб-додатках. Laravel полегшує розробку автентифікації, оскільки включає такі функції, як реєстрація, забутий пароль та надіслати нагадування про пароль.

Редіс

Laravel використовує Redis для підключення до існуючого сеансу та кешу загального призначення. Redis безпосередньо взаємодіє з сеансом.

Черги

Laravel включає сервіси черг, такі як надсилання електронної пошти великій кількості користувачів або вказане завдання Cron. Ці черги допомагають у легшому виконанні завдань, не чекаючи, коли попереднє завдання буде виконане.

Подія та командна шина

Laravel включає в себе командну шину, яка допомагає виконувати команди та відправляти події простим способом. Команди в Laravel діють відповідно до життєвого циклу програми. [7]

3.3. Model-View-Control

Модель Model-View-Control (MVC), була сформульована наприкінці 1970-х років, - це модель архітектури програмного забезпечення, побудована на основі збереження подання даних окремо від методів, які взаємодіють з даними. Теоретично, добре розвинена система MVC повинна дозволити front-end розробнику та back-end розробнику, який працює на задньому рівні, працювати в одній і тій же системі, не втручаючись, обмінюючись або редагуючи файли, над якими працює будь-яка зі сторін.

Навіть незважаючи на те, що MVC спочатку був розроблений для персональних обчислень, він був адаптований та широко використовується веб-розробниками через його акцент на розподілену систему. Ця схема заохочує розвиток модульних систем, дозволяючи розробникам швидко оновлювати, додавати або навіть видаляти функціональні можливості.

Розуміння MVC

Назва шаблону - це поєднання трьох його основних частин: Модель, Перегляд та Контролер. Візуальне зображення повного і правильного шаблону MVC виглядає наступним чином:

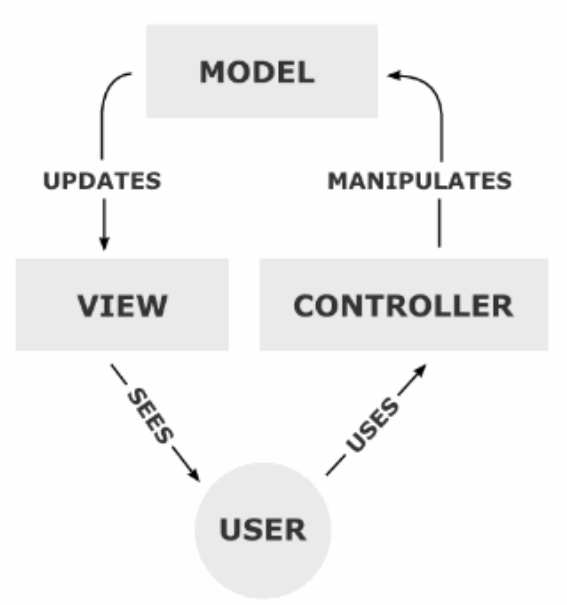

Рисунок 3.1. Шаблон MVC

Зображення показує макет даних одного потоку, як вони передаються між кожним компонентом і, нарешті, як працює взаємозв'язок між кожним компонентом.

Модель

Модель - це ім'я, що надається для постійного зберігання даних, що використовуються в загальній конструкції. Він повинен забезпечувати доступ для перегляду даних, збирання та запису даних і є мостом між компонентом View та компонентом Controller у загальній схемі.

Одним з важливих аспектів Моделі є те, що вона технічно «сліпа» - модель не має ніякого зв’язку або знань про те, що відбувається з даними, коли вони передаються компонентам View або Controller. Він ні дзвонить, ні шукає відповіді з боку інших частин; єдина його мета - обробляти дані в постійне зберігання або шукати та готувати дані для передачі разом з іншими частинами.

Однак модель не може бути просто підведена як база даних або шлюз до іншої системи, яка обробляє процес обробки даних. Модель повинна діяти як воротар самих даних, не задаючи жодних питань, але приймаючи всі запити, які надходять на свій шлях. Часто найскладніша частина системи MVC, компонент Model - це також вершина всієї системи, оскільки без неї не існує зв'язку між контролером та видом.

Вид

Вид - це перегляд даних, запитуваних від Моделі, і визначається її кінцевий вихід. Традиційно у веб-додатках, побудованих за допомогою MVC, Перегляд - це частина системи, де генерується та відображається HTML. Перегляд також запалює реакції користувача, який потім переходить до взаємодії з Контролером. Основний приклад цього - кнопка, створена видом, який користувач натискає та запускає дію в Контролері.

Існують деякі помилки щодо компонентів View, зокрема, веб-розробники, які використовують шаблон MVC для створення свого додатка. Наприклад, багато хто помиляється, що View не має жодного з'єднання з Моделлю, і що всі дані, показані Переглядом, передаються з Контролера. Насправді цей потік повністю ігнорує теорію, що стоїть за схемою MVC. Крім того, опис переглядів як файлу шаблону є неточним. Однак, як вказує Том Батлер, це не вина однієї людини, а безліч помилок багатьох розробників, в результаті яких розробники навчаються MVC неправильно. Потім вони продовжують навчати інших неправильно. Перегляд насправді набагато більше, ніж просто шаблон, однак сучасні натхненні MVC рамки базували погляд майже до того, що ніхто насправді не переймається тим, чи рамка насправді дотримується правильної схеми MVC чи ні.

Також важливо пам’ятати, що частина перегляду ніколи не надає дані контролером. Немає прямого зв'язку між Поглядом і Контролером без Моделі між ними.

Контролер

Заключний компонент тріади - Контролер. Його завдання полягає в обробці даних, які користувач вводить або надсилає, та оновлює модель

відповідно. Життєва кров Контролера - це користувач; без взаємодії з користувачем, Контролер не має мети. Це єдина частина шаблону, з яким повинен взаємодіяти користувач.

Контролер може бути узагальнений просто як збирач інформації, який потім передає його Моделі, яку потрібно організувати для зберігання, і не містить жодної логіки, окрім необхідної для збору вводу. Контролер також підключений лише до одного виду та до єдиної моделі, що робить його односторонньою системою потоку даних, з рукостисканням та сигналізацією в кожній точці обміну даними.

Важливо пам’ятати, що Контролеру даються завдання, які потрібно виконувати лише тоді, коли користувач спочатку взаємодіє з Переглядом, і що кожна функція Контролера є тригером, що відзначається взаємодією користувача з Поглядом. Найпоширеніша помилка, яку роблять розробники, - це заплутати контролер для шлюзу, і, зрештою, призначити йому функції та обов'язки, які повинен мати перегляд (зазвичай це результат того самого розробника, що плутає компонент View просто як шаблон). Крім того, поширеною помилкою є надання функцій контролера, які покладають на нього виключну відповідальність за стискання, передачу та обробку даних від Моделі до перегляду, тоді як у шаблоні MVC це співвідношення має зберігатися між Модель та Вид.

MVC в PHP

Можна написати веб-додаток в PHP, архітектура якого базується на шаблоні MVC.

```php
<?php
class Model
{
    public $string;

    public function __construct(){
        $this->string = "MVC + PHP = Awesome!";
    }
}
```

Рисунок 3.2. Реалізація «моделі» на язику PHP

```php
<?php
class View
{
    private $model;
    private $controller;

    public function __construct($controller,$model) {
        $this->controller = $controller;
        $this->model = $model;
    }

    public function output(){
        return "<p>" . $this->model->string . "</p>";
    }
}
```

Рисунок 3.3. Реалізація «виду» на язику PHP

```php
<?php
class Controller
{
    private $model;

    public function __construct($model) {
        $this->model = $model;
    }
}
```

Рисунок 3.4. Реалізація «контроллера» на язику PHP

Цей проект розпочався з декількох дуже базових класів для кожної частини шаблону. Тепер потрібно встановити відносини між ними:

```php
<?php
$model = new Model();
$controller = new Controller($model);
$view = new View($controller, $model);
echo $view->output();
```

Рисунок 3.5. Реалізація шаблону MVC на язику PHP

Перегляд містить всю функціональність, оскільки приклад призначений виключно для відображення. [8]

Маршрутизація та URL-адреси

Хоча MVC, теоретично, повинен працювати бездоганно у всіх формах комп'ютерного програмування, включення MVC в Інтернет з PHP може бути дещо складним. Перша проблема полягає в маршрутизації URL-адрес. Маршрутизація URL - це аспект, який ніколи не розглядався під час створення MVC, і тому, як Інтернет розвивається та розвивається в міру розширення технологій, так само повинні використовуватися і структури архітектури, які користувач використовує.

Одне рішення - спроба викупити будь-які URL-адреси, потрібні веб-сайту людини, чи веб-додатку, зберігаючи їх у постійному сховищі разом із інформацією про те, яку модель, перегляд та контролер потрібно завантажити для кожної сторінки чи розділу. Потім система отримує запитувану користувачем URL-адресу та завантажує конкретні компоненти, призначені для цієї сторінки, і приступає до роботи. Це ідеально можливе рішення, якщо ви створюєте портфоліо-сайт або статичний веб-сайт, який не покладається на динамічні URL-адреси.

URL-адреси виглядатимуть так:

```
example.com/index.php?page=about
```

Рисунок 3.5. Приклад маршрутизації

На прикладі системи MVC завантажується конкретний набір моделі, перегляду та контролера для потрібної сторінки. Якщо параметр URL "приблизно", тоді відобразиться сторінка About. Якщо параметр "портфоліо", замість цього буде сторінка Портфоліо.

Це основний приклад статичної маршрутизації, яка навіть через її налаштування дуже проста, має деякі недоліки. Одним з найбільш очевидних недоліків є той факт, що масштабованість стає набагато складнішою, оскільки широта веб-сайту обмежується важко закодованим масивом сторінок.

Іншим варіантом є відкриття призначення класів «Модель», «Вид» та «Контролер» і дозволити URL визначати ці параметри. У прикладі статичної маршрутизації було витягнуто ідентифікацію класу з масиву, який, в свою чергу, діяв як дані про маршрутизацію, що надходять з постійного сховища. Заміна масиву елементами перетворює статичну маршрутизацію в динамічну маршрутизацію.

Незважаючи на те, що був витягнутий ключ для кожної асоціації в масиві з змінною URL, зв'язки з відповідними класами були вже визначені; не можна було змішати і співставити значення в кожній клавіші зі статичною маршрутизацією. Ну а для початку не доведеться жорстко кодувати кожен розділ нашої системи. Можна створювати розділи чи сторінки просто шляхом створення взаємозв'язку моделі, перегляду та контролера.

Нова URL-адреса виглядатиме так:

```
example.com/index.php?controller=controllername↦model=modelname&view=viewname&action=actionname
```

Змінна URL-адреса дії вказує системі, яку функцію в Контролері виконувати. Важливо пам’ятати, що коли ця функція передає дані до Моделі, вона передає частину даних Моделі, яка, в свою чергу, вказує, які дії потрібно завантажити. Це може бути змінною URL-адреси дії, але вона також може бути окремою або дані, зібрані Контролером.

Обидва підходи мають свої плюси і мінуси: статична маршрутизація є більш стабільною, швидшою у впровадженні та дозволяє розробникам більш контролювати систему, а динамічна маршрутизація дозволяє створити більш ефективну систему, де існує величезний потенціал масштабованості та портативності.

Динамічна маршрутизація, однак, може покласти більше відповідальності на контролер, ніж статичну маршрутизацію, що може розглядатися як зміна традиційного шаблону MVC. Тим не менш, якщо динамічна маршрутизація реалізована правильно і ефективно, це може зробити Контролер більш бажаним в системі в порівнянні з використанням статичної маршрутизації.

Додавання переднього контролера дозволить системі динамічно завантажувати секції, залежно від того, що хоче завантажити користувач. Алехандро Гервасіо написав фантастичну статтю з двох частин про схему переднього контролера , в якій торкаються ідеї використання переднього контролера з елементами MVC.

Шаблони

Слово "шаблон" може викликати декілька питань для тих, хто раніше бачив веб-рамки MVC, оскільки більшість людей обробляють частину шаблону разом із Переглядом. Компонент View має сенс лише обрати шаблон і передати дані з представлення у цей шаблон. Таким чином він готовий до відображення з використанням блочного макета коду, або з друком або будь-яким іншим вихідним кодом у PHP. Незалежно від того, які з методів використовувати, головне пам’ятати - це, що дані користувача ОБОВ'ЯЗКОВО перебувають у стані готовності.

Крім того, шаблон передається через модель, яка, може забезпечувати динамічне призначення шаблонів залежно від того, що призначено для кожного конкретного представлення. Цей метод шаблонування дозволяє ефективно та впевнено розширювати системи MVC, надаючи нам можливість розділити розробку заднього кінця від розвитку переднього кінця; оригінальна мета шаблону MVC.

Модель MVC була зміною ігор, коли вона була вперше використана для настільного програмування. Це було і досі є дуже дискусійною темою між розробниками щодо інтерпретації певних сценаріїв при розробці з ним. Дебати стають ще більш інтенсивними, коли ви додаєте PHP та Інтернет в суміш, що є фантастичним ознакою того, що все більше розробників зосереджуються на вдосконаленні дисциплін розвитку PHP.

MVC є особистим фаворитом завдяки заохоченню розділення між різними деталями обробки та можливістю поділу між передньою та задньою частинами.

3.4 Опис призначеного для користувача інтерфейсу

Інтерфейс проекту представлений у вигляді веб-інтерфейсу, реалізація якого була виконана на такому стеку технологій, як HTML, CSS, JS та багатьох сторонніх бібліотек.

Для роботи з системою, необхідно створити обліковий запис. Процедура реєстрації максимально проста. Необхідно вказати активний email, своє ім'я, і придумати надійний пароль, який потрібно повторити двічі, щоб уникнути помилок. Далі, відправити форму і система, в разі успішного введення даних створить обліковий запис, менш ніж за одну секунду.

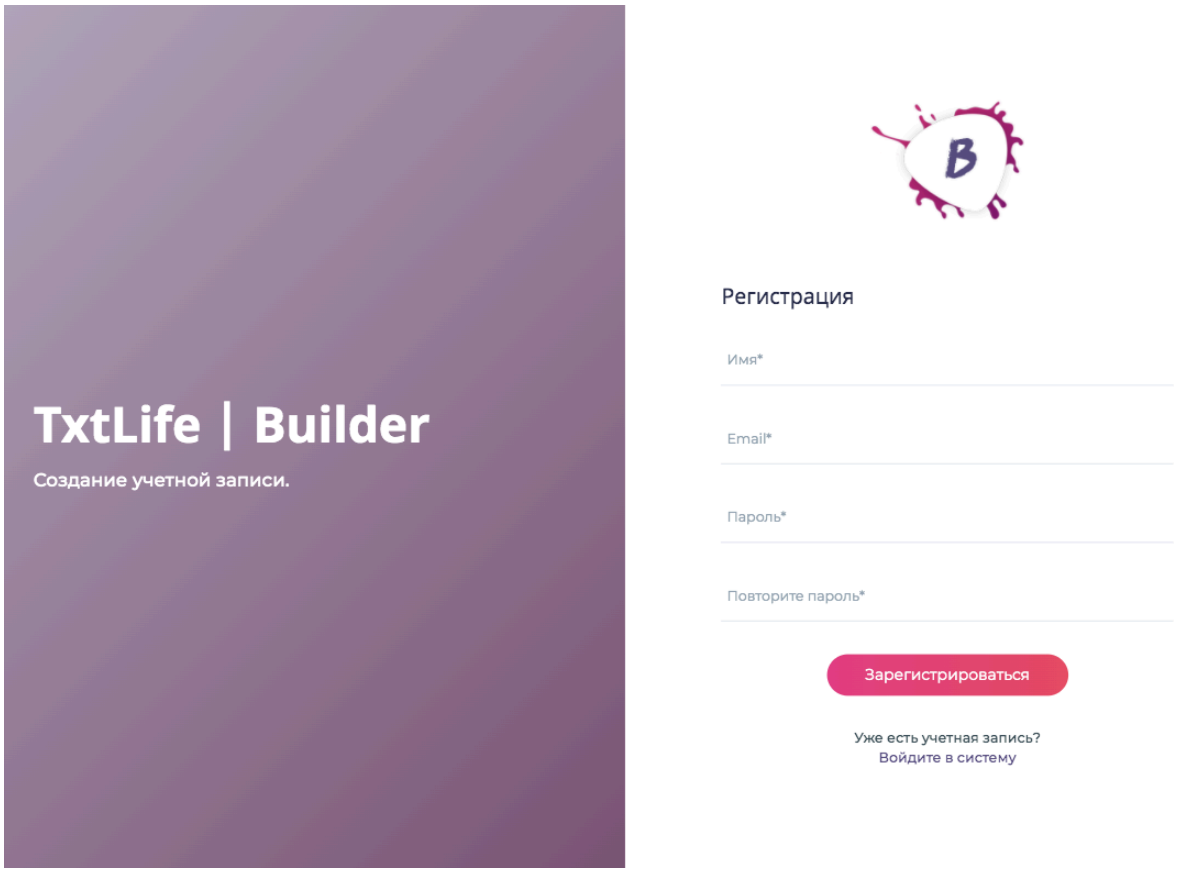

Рисунок 3.6. Сторінка реєстрації користувача

При необхідності, на сторінці є посилання на форму авторизації, якщо обліковий запис вже створений.

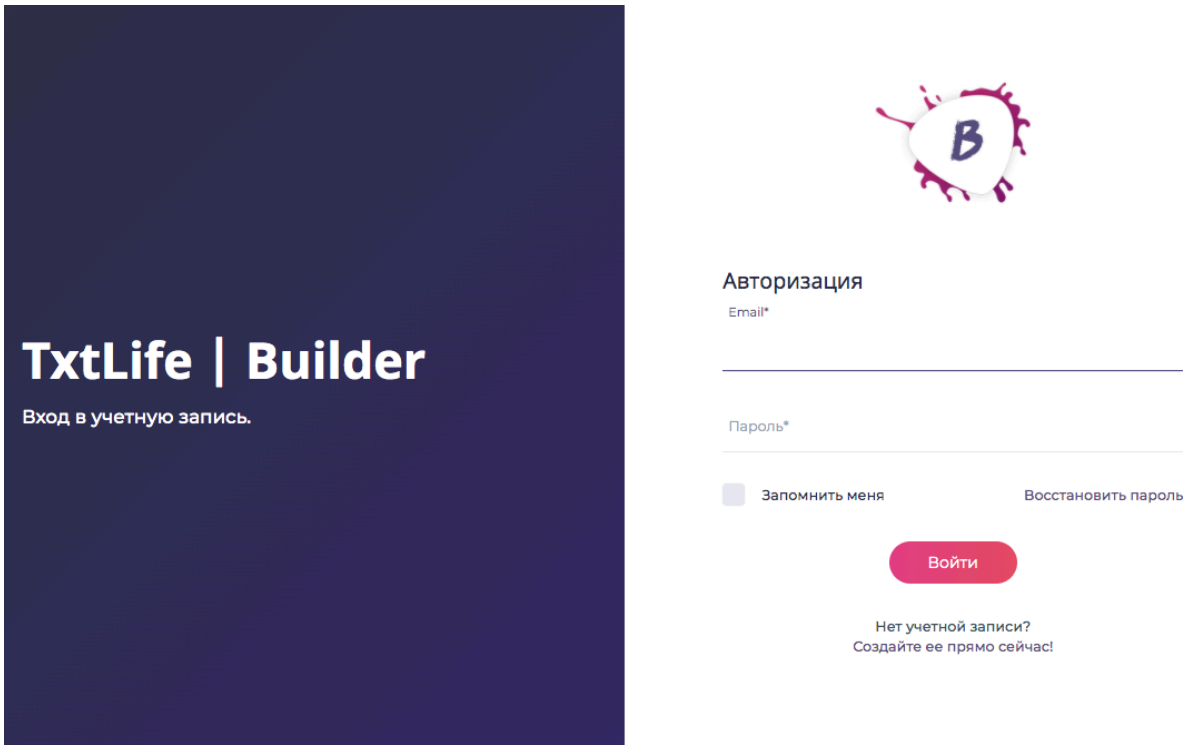

Рисунок 3.7. Сторінка авторизації користувача

На Рисунку 3.7. відображена форма авторизації користувача. Розроблена система має функцію тривалого зберігання сесії. Для активації цієї функції, необхідно поставити галочку напроти пункту "Запам'ятати мене". На сторінці є посилання на форму відновлення пароля і створення облікового запису користувача.

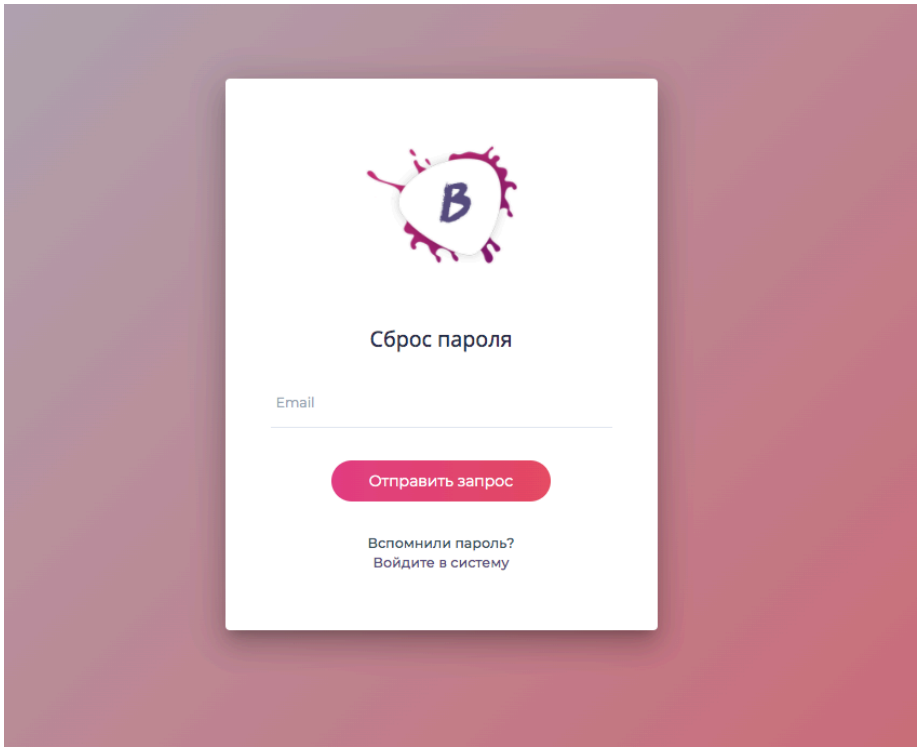

Рисунок 3.8. Сторінка відновлення пароля

На Рисунку 3.7. відображена сторінка відновлення пароля. Для активації функції відновлення пароля, необхідно вказати email користувача, доступ до якого необхідно відновити. Після відправки форми, в разі, якщо користувач із зазначеним email був знайдений, буде автоматично сформований email з одноразовим посиланням на скидання пароля і відправлений на зазначений email.

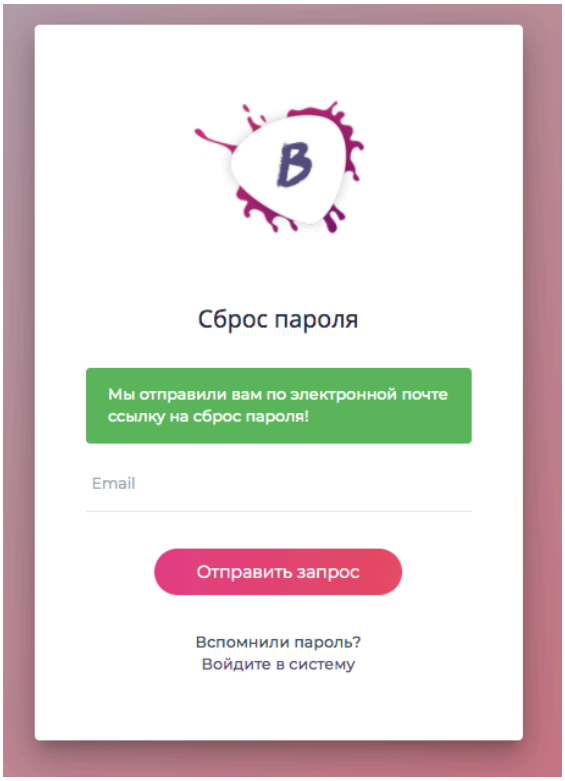

Рисунок 3.9. Успішний статус відправки форми відновлення пароля

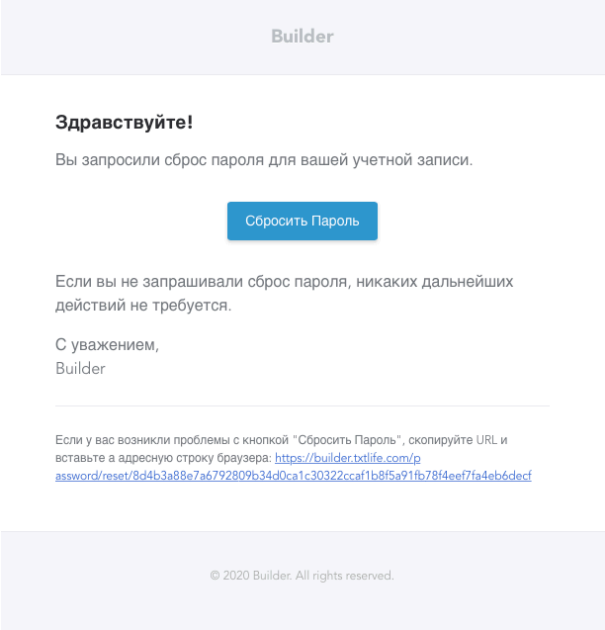

Рисунок 3.10. Email з інструкціями відновлення пароля від облікового запису проекту

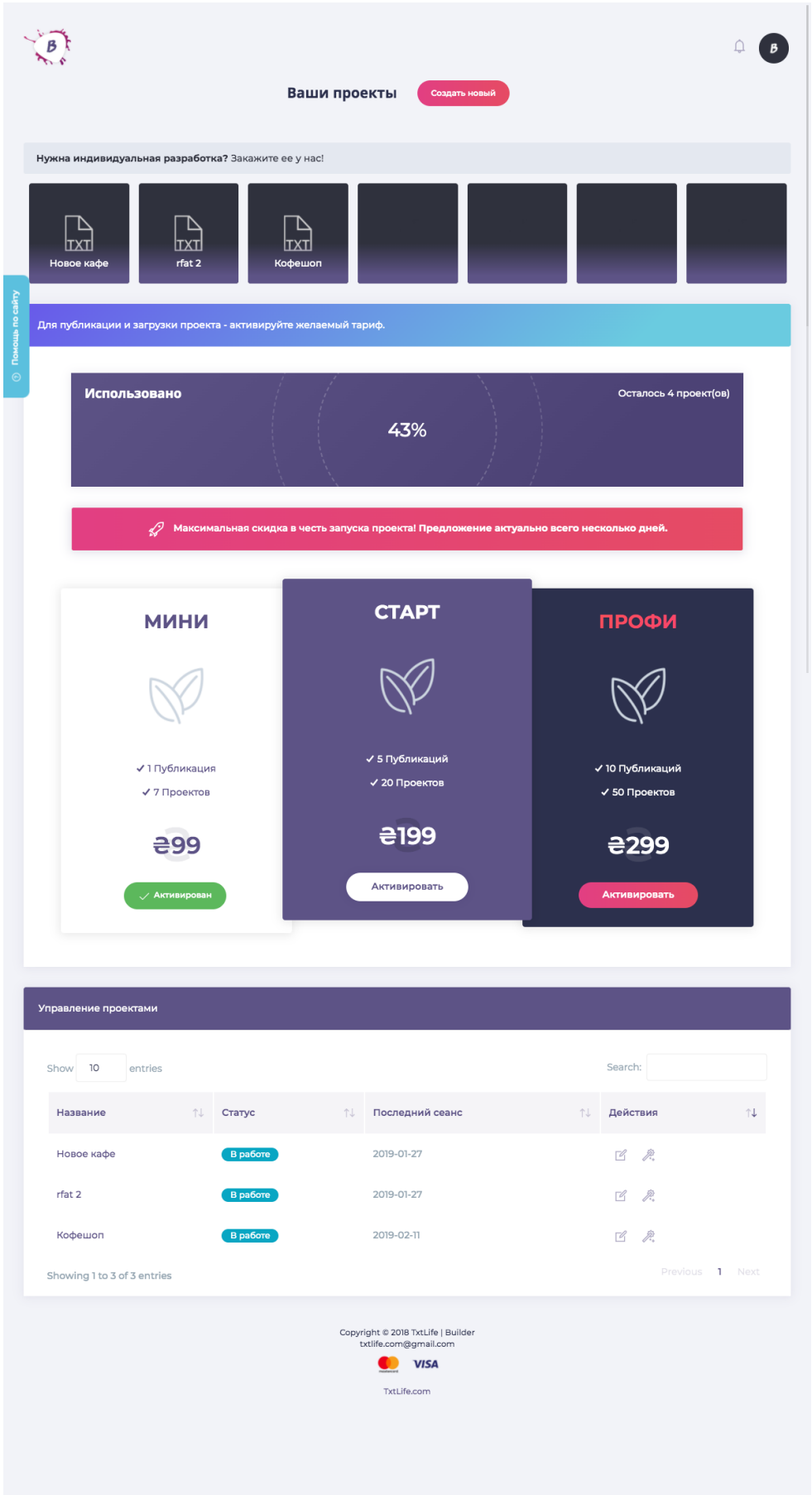

Рисунок 3.10. Домашня сторінка користувача

На Рисунку 3.10. відображена домашня сторінка користувача.

Структура сторінки користувача:

- Приховане меню, яке містить посилання на сторінку "Допомога" і "Реферальная система"
- Посилання на сторінку створення нового проекту
- Панель активних проектів, для швидкого доступу до них. Реалізація виконана у вигляді каруселі, що дозволяє вивести в один ряд безліч проектів і отримувати до них доступ, за допомогою перетягування стрічки
- Віджет, який показує кількість доступних до створення проектів та ту ж статистику в процентному співвідношенні
- Віджет зміни тарифного плану. В момент активації більш високого тарифного плану, користувача буде автоматично перенаправлено на платіжну систему LiqPay, для оплати суми, яка відповідає обраному тарифу. Після проведення оплати, користувача перенаправить назад в кабінет і в разі успішного платежу - активується новий тарифний план з більшою кількістю проектів до створення і публікації в інтернет.
- Віджет управління проектами користувача. У табличному вигляді відображений список поточних проектів користувача, де кожен відображає свою назву, поточний статус, дату останніх робіт над даним проектом, кнопку редагування і публікації проекту в інтернет.
- Посилання на сторінку документації до взаємодії з проектом. Посилання виконане у вигляді фіксованої кнопки і це дозволяє моментально отримати доступ до даної сторінки.

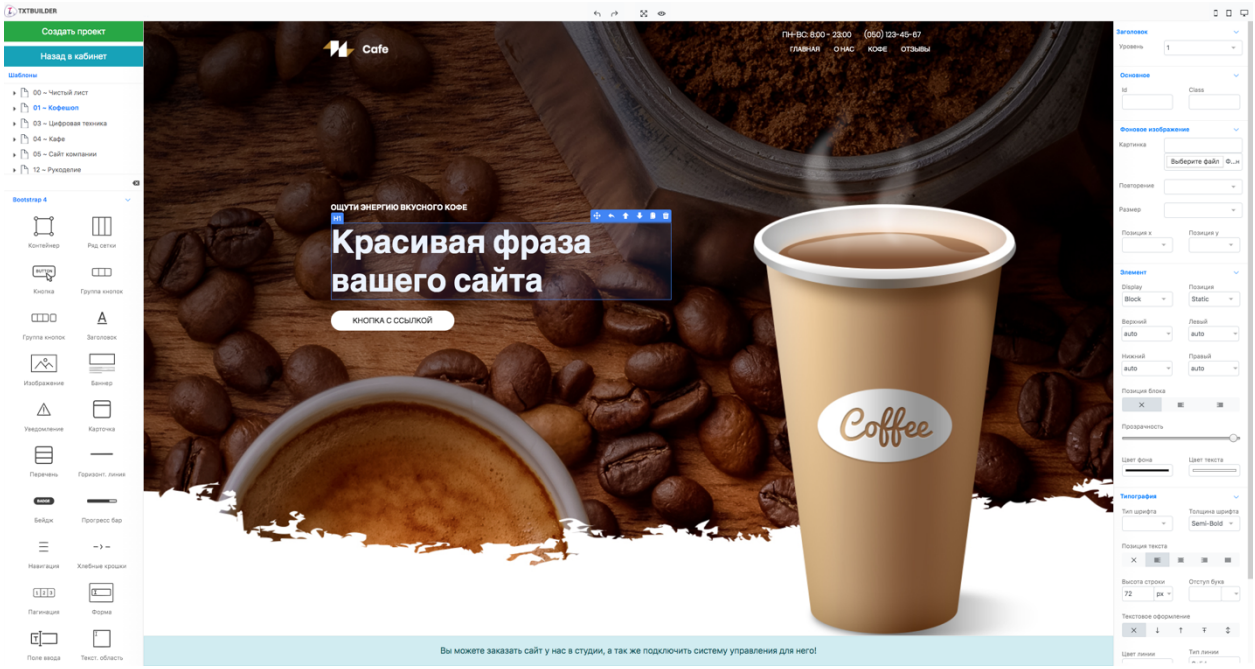

Рисунок 3.11. Сторінка створення сайту

На Рисунку 3.11. відображена сторінка створення сайту, яка складається с безлічі частин.

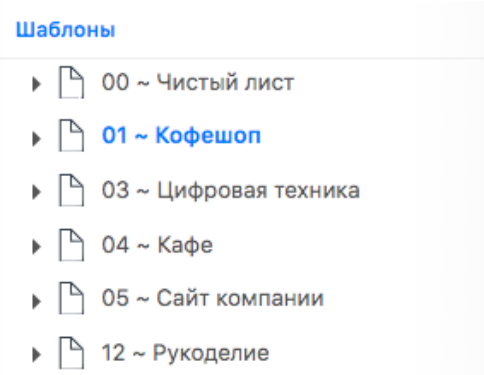

Рисунок 3.12. Віджет "Шаблони"

На Рисунку 3.12. відображений віджет, який представляє собою список завантажених шаблонів, готових до використання, в якості основи майбутнього сайту. Всі шаблони базуються на CSS фреймворку Bootstrap 4. Це дозволяє отримувати максимум можливостей від CSS складової майбутнього сайту. Завдяки підтримці Bootstrap 4, шаблон "з коробки" вже має підтримку адаптивного дизайну.

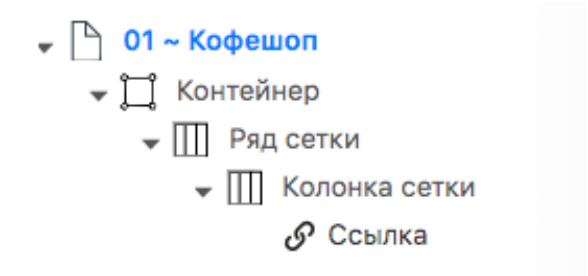

Рисунок 3.13. Демонстрація детального уявлення компонента шаблону

Завдяки єдиній структурі побудови шаблонів, яка базується на Bootstrap 4, система конструктора здатна "розбирати" кожен компонент шаблону на його складові (рисунок 3.13).

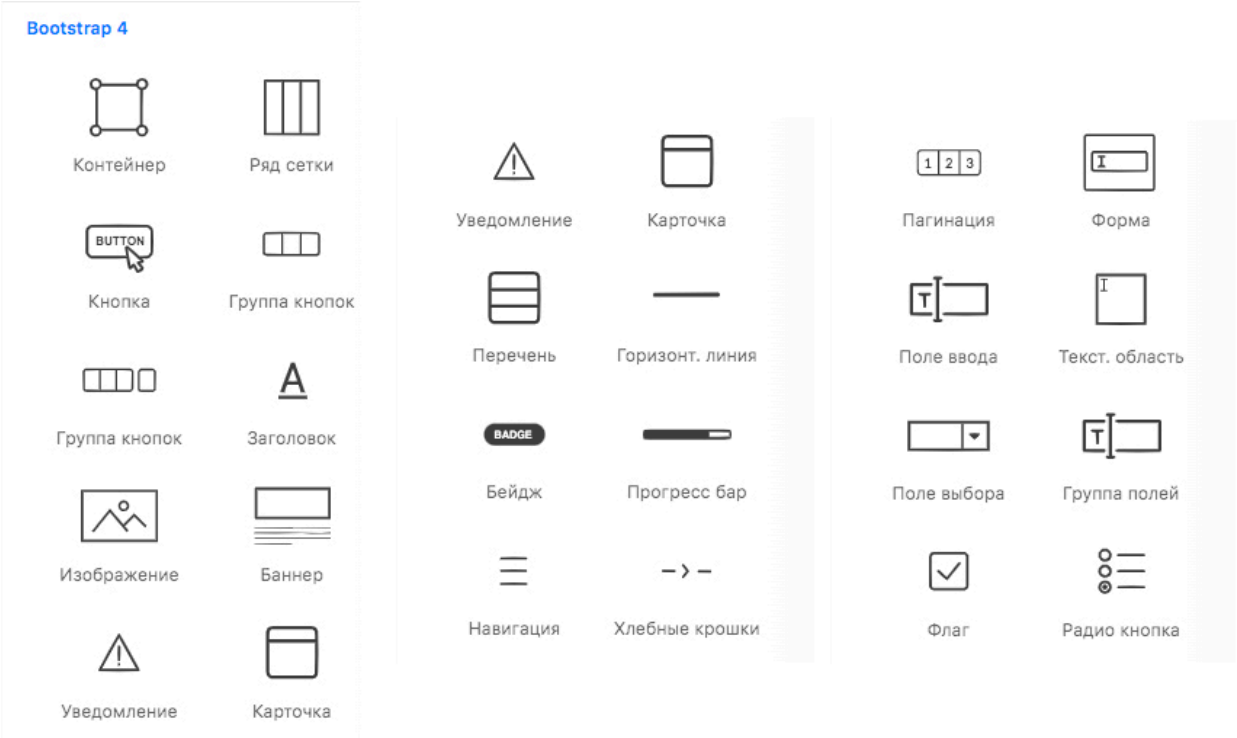

Рисунок 3.14. Віджет «Bootstrap 4 Components»

Оскільки, Bootstrap 4 це CSS фреймворк, а значить, він являє собою великий набір готових класів, для елементів інтерфейсу. Компонуючи класи, можна створити безліч адаптивних елементів, таких як меню навігації, прогрес бари, форми, кнопки тощо.

Практично всі можливі компоненти, які можна створити, використовуючи "чистий" Bootstrap 4 - описує сама його документація. На основі стандарних HTML / CSS Bootstrap 4 компонентів, були розроблені

повноцінні, динамічні JS компоненти, які можна використовувати при інтерактивному режимі розробки сайту (рисунок 3.15).

Рисунок 3.15. Рівні компонента "Хлібні крихти"

Крім компонентів Bootstrap 4, є кілька вбудованих віджетів.

- Карта Google
- Вбудоване відео YouTube, Vimeo, HTML5
- Діаграма на основі Chart.JS
- Facebook Page Plugin
- Пожертвування через Paypal
- Вбудована стрічка Instagram

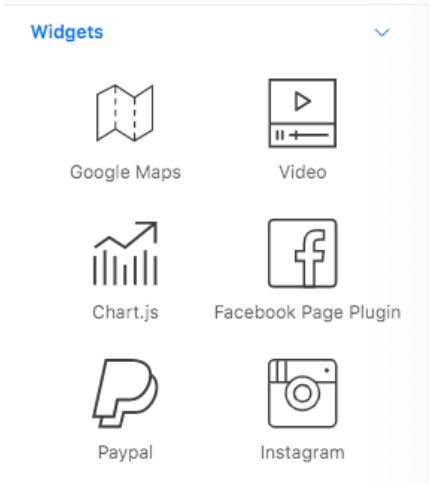

Рисунок 3.16. Вбудовувані віджети

У центральній частині сторінки є кнопки для "Крок вперед", "Крок назад", "Повноекранний режим" і "Демонстрація проекту" (рисунок 3.17).

Рисунок 3.17. Швидкі функції

JS додаток "сканує" весь шаблон перед тим, як користувач почне з ним роботу. Це дозволяє отримати доступ до кожного елементу, для роботи з ним. Під час редагування вибраного елемента, JS в реальному часі зчитує всі внесені зміни і вносить їх в поточний елемент (рисунок 3.18).

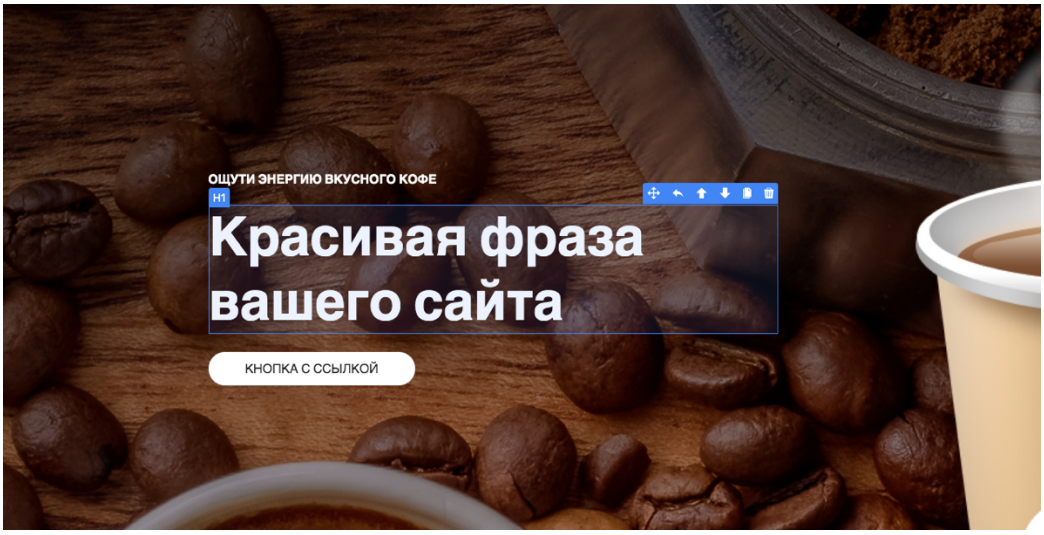

Рисунок 3.18. Редагування обраного елемента

Крім "швидкого" редагування елемента, є величезний набір властивостей кожного для кожного типу елемента. На рисунку 3.19 відображений процес редагування зображення. Крім стандартних властивостей, таких як позиціонувати і ім'я класу є додаткові властивості, які відносяться тільки до типу "Зображення".

Для повноцінної розробки та перевірки шаблону на адаптивність, є 3 формати відображення - комп'ютер, планшет і телефон. Після натискання на будь-яку з цих кнопок, конструктор моментально перебудує вид шаблону в зазначений.

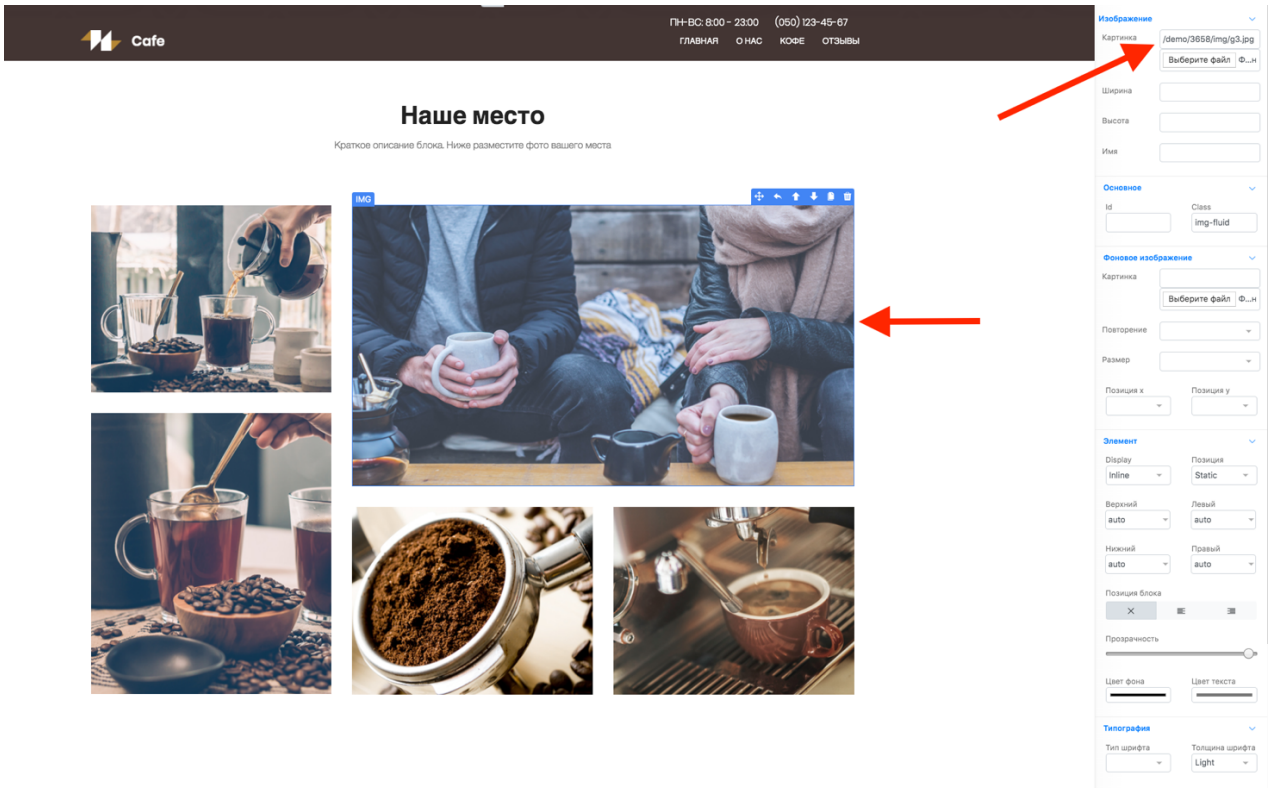

Рисунок 3.19. Зміна властивостей обраного елемента

Якщо була обрана функція створення проекту, перед виходом користувач повинен буде створити проект, давши йому зручне для нього ім'я (рисунок 3.20). Після чого, проект буде збережений з базу даних конструктора і отримає унікальний ідентифікатор. Далі буде запропоновано кілька подальших дій (рисунок 3.21).

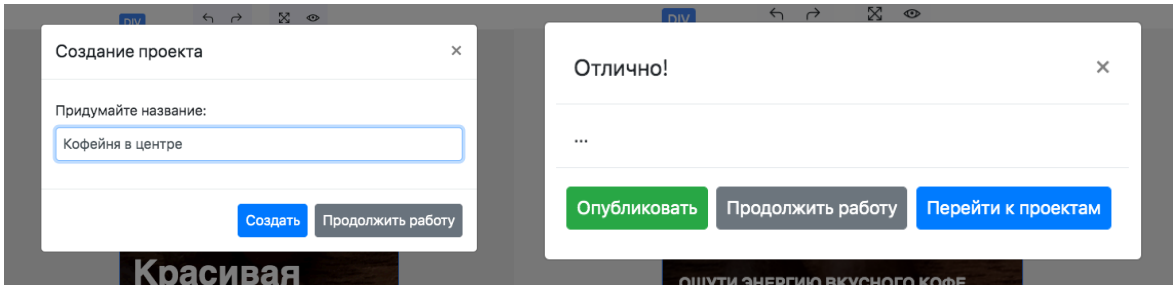

Рисунок 3.20. Створення проекту

Рисунок 3.21. Успішне створення проекту

Логіка конструктора також включає в себе постійну перевірку розміру дисплея користувача. Якщо розмірність дисплея не відповідає заданим властивостям коструктора - користувач побачить попередження про те, що інтерфейс може нестабільно працювати на такому дисплеї (рис 3.22).

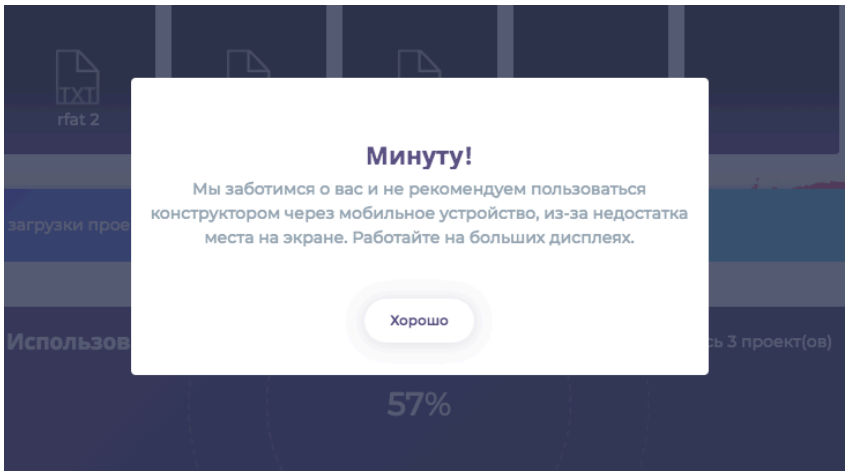

Рисунок 3.22. Попередження про низьку роздільну здатність дисплея

Публікація проекту в інтернет, буде розглядатися на основі створеного шаблону, який описаний вище. Доступ до сторінки публікації можливий лише за умови вільних тарифних одиниць.

Процес публікації розділений на три етапи, для більшої зручності взаємодії. На першому етапі (рисунок 3.23) відображена статистика вільних / зайнятих місць публікації, які передбачені поточним тарифом користувача, а також сам тариф користувача і кнопка "Збільшити", при натисканні на яку, користувача буде перенаправлено на сторінку оплати вищого тарифного плану.

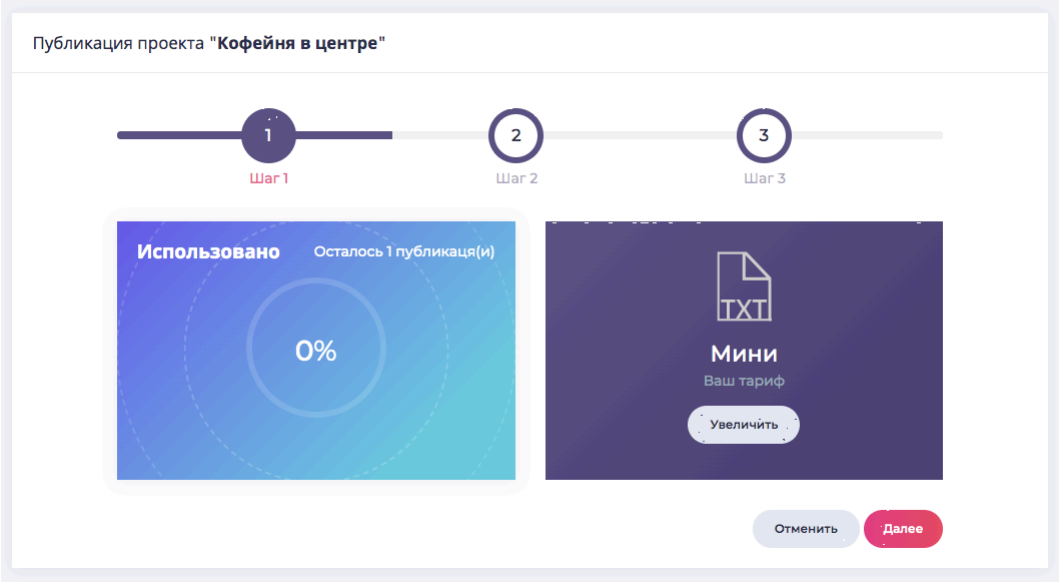

Рисунок 3.23. Перший етап публікації проекту в інтернет

Другий етап публікації полягає у виборі доменного імені майбутнього сайту. Звідси і виходить поділ на два типи публікації - на особистому домені, або на піддомені коструктора (рисунок 3.24).

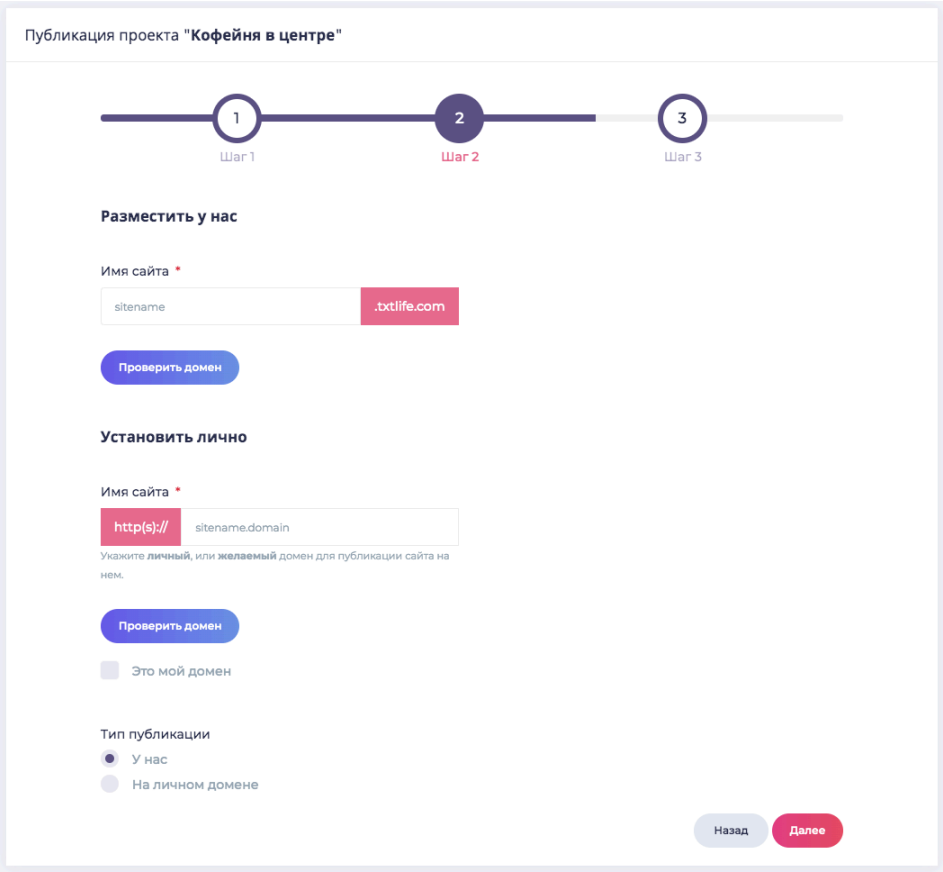

Рисунок 3.24. Другий етап публікації проекту в інтернет

У кожного типу доменного імені є своя перевірка на доступність імені. Що стосується перевірки доступності особистого доменного імені, є можливість активувати папаметр "Це мій домен" (рисунок 3.25) і перевірка через WHOIS сервіси буде проігнорірованна.

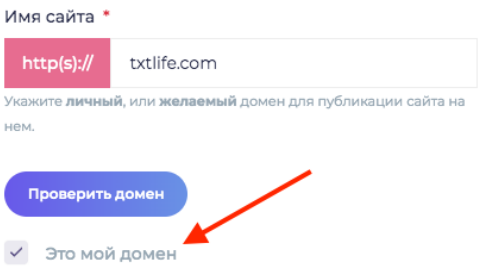

Рисунок 3.25. Демонстрація параметра "Це мій домен"

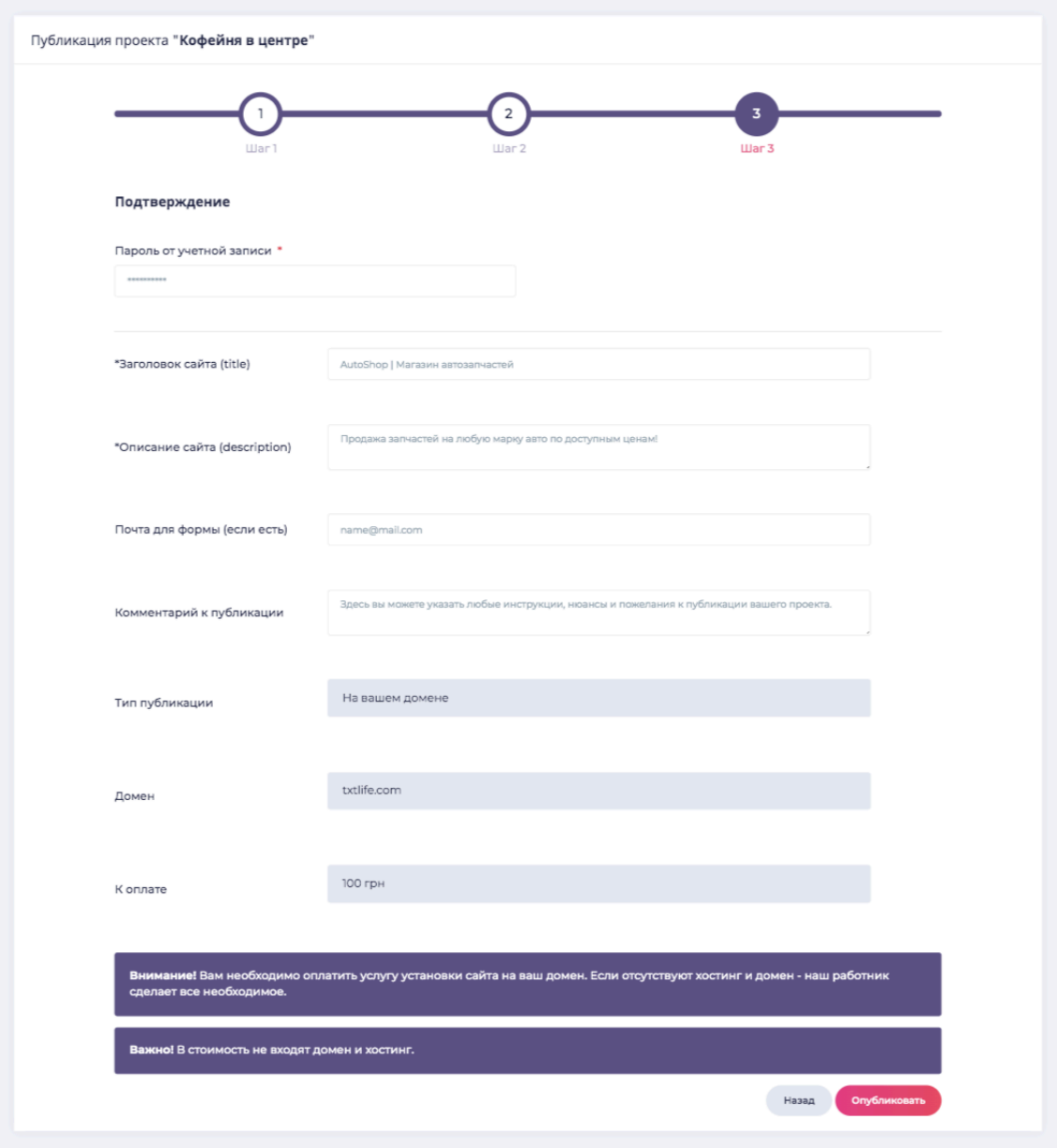

Рисунок 3.26. Третій етап публікації проекту в інтернет

Третій етап є фінальним, тому він містить в собі частину даних, які були вказані на попередньому етапі, а також додаткові поля конфігурацій.

Для підтвердження публікації, необхідно вказати актуальний пароль від поточного облікового запису, цей метод не є достатньо безпечним, оскільки, якщо зловмисник отримає доступ до аккаунту - цей захід безпеки не матиме сенсу.

Даний "недолік" можливо вирішити, впровадженням системи двофакторної аутентифікації, або прив'язкою номера телефону, на який будуть приходити числові ключі, коли це необхідно.

Далі необхідно заповнити два обов'язкових поля - "Тема" і "Опис" сайту. Дані параметри необхідні не тільки для відображення назви в закладці браузера, а й катастрофічно важливі для індексації пошуковими системами.

Якщо в шаблоні, користувач створив форму зворотного зв'язку, він може зповнити поле "Пошта для форми". Даний параметр буде використаний фахівцем, в момент публікації проекту в мережу.

Поле "коментар до публікації" є додатковою інформацією для фахівця, який буде займатися процедурою публікації.

Далі йде фінальна інформація, така, як "Тип публікації", "Домен" і "Сума", яку користувач повинен буде сплатити, якщо він обрав публікацію "На особистому домені".

Доробок в плані автоматизації проекту має бути ще дуже багато. Наприклад, публікацію проекту на наш піддомен можна автоматизувати, шляхом редагування конфігураційних файлів Apache, створення дерикторії і вивантаження проекту в неї.

Публікацію проекту на власному домені, можна реалізувати автоматичним створенням дерикторії проекту на сервері і вивантаженням туди, а прив'язку домена зробити в налаштуваннях користувача, де йому необхідно буде додати відповідний домен і вказати NS сервери констркутора в налаштуваннях доменнного імені на сайті реєстратора.
Автоматизація процесу публікації виключає роботу фахівця.

На сайті присутня реферальная система (рисунок 3.27). Вона сприяє підвищенню числа реєстрацій і покупок тарифів на публікацію. Користувач, за посиланням якого зареєструвався новий, відобразиться на сторінці "Реф. Система". Якщо новий користувач активує будь-який тариф, відсоток від суми платежу надійде на рахунок користувача, за посиланням якого зареєструвалися. Засоби, які були накопичені, можна вивести, використовуючи зв'язок з адміністрацією.

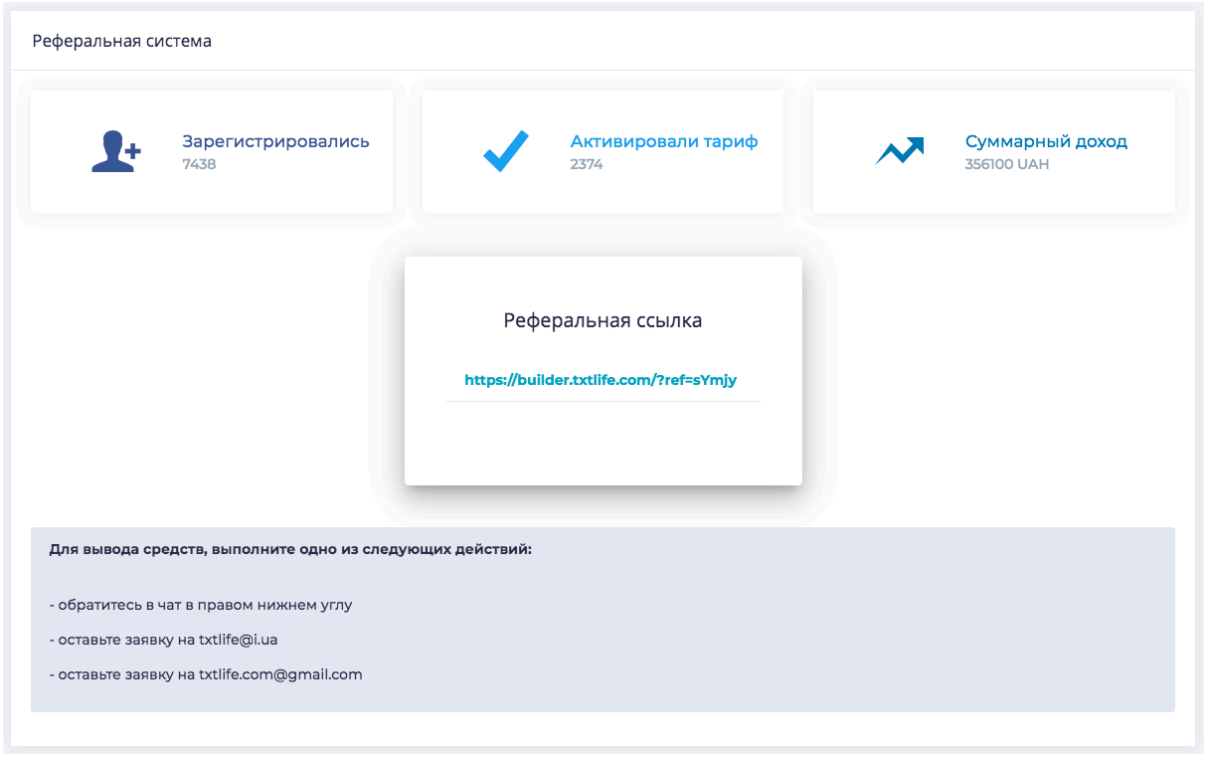

Рисунок 3.27. Реферальна система

Допомога по сайту включає в себе відповіді на найчастіші запитання. Відповідь на кожне питання описанна просто, з відповідними зображеннями (рисунок 3.28).

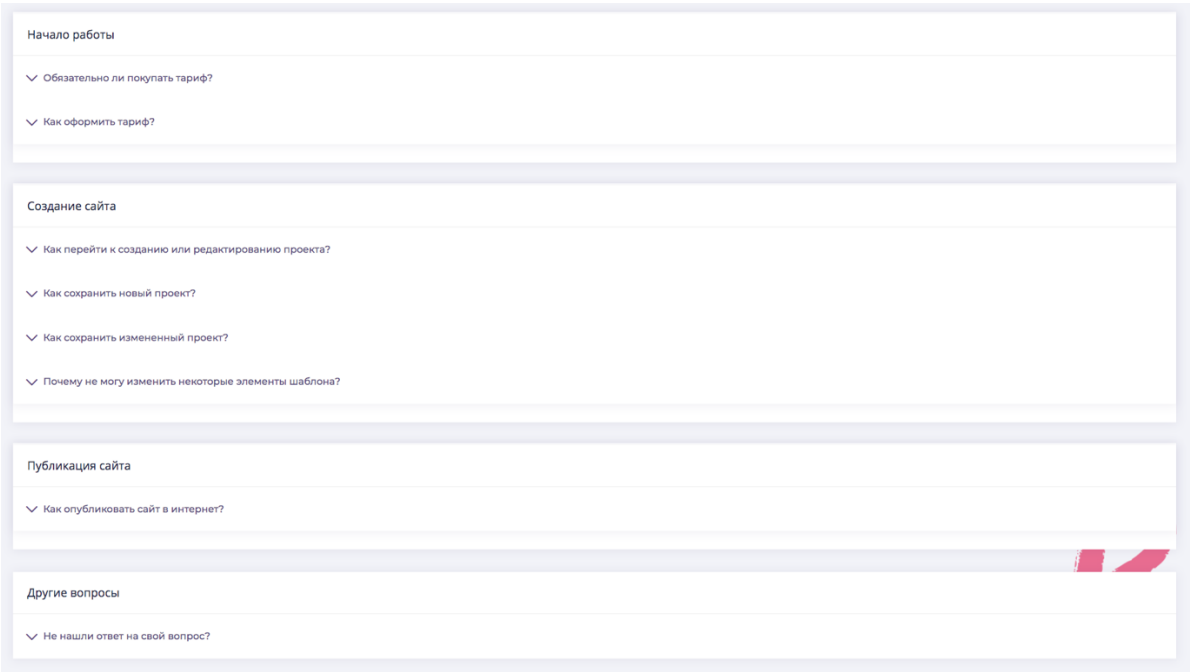

Рисунок 3.28. Допомога по сайту

Панель адміністратора (рисунок 3.29) складається з:

- Лінійної діаграми, яка відображає статистику покупок тарифів
- Короткої інформації про проект (кількість користувачів, куплених тарифів і кількості транзакцій)
- Списку проектів, який розділений по статусах і показаний в табличному вигляді
- -Списку користувачів, який представлений в табличному вигляді

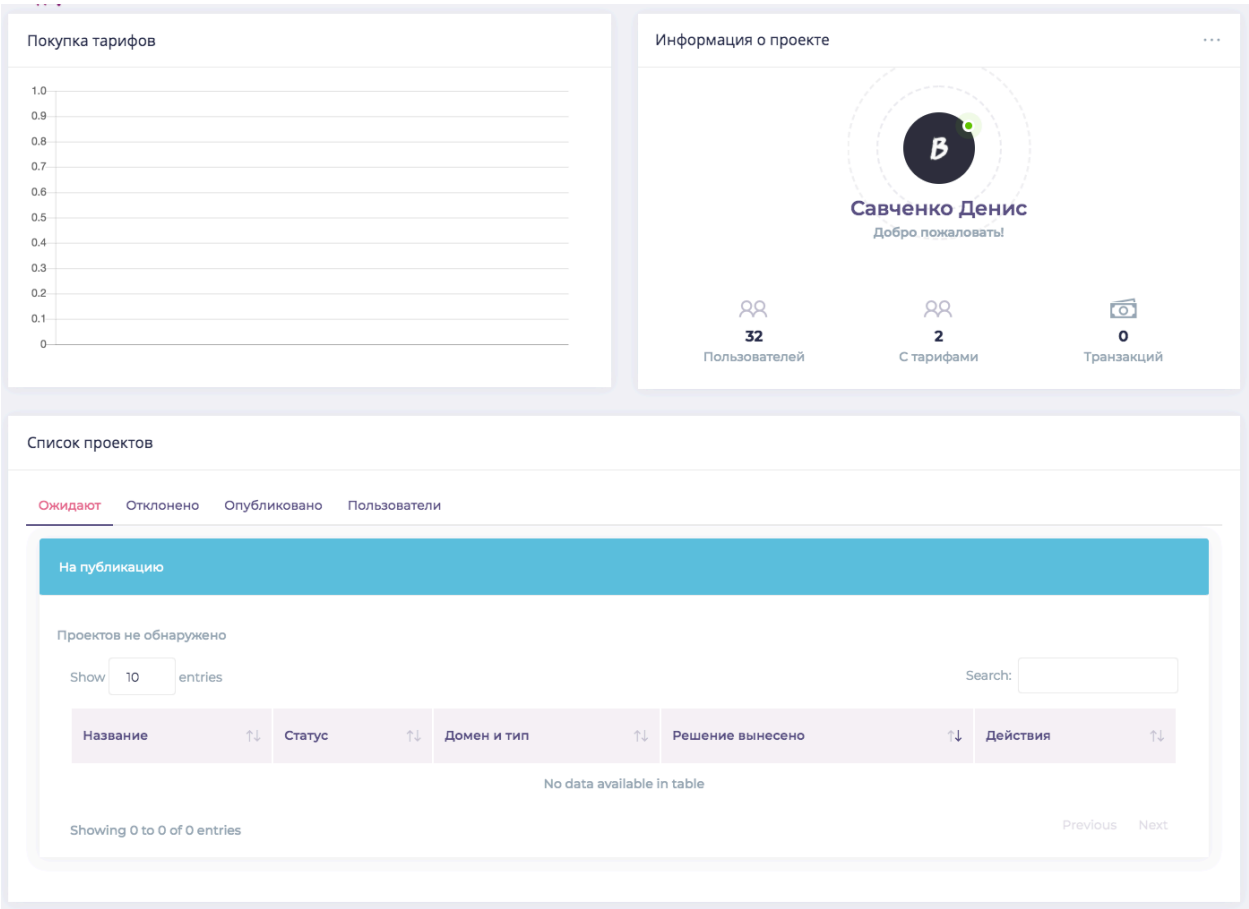

Рисунок 3.29. Панель адміністратора

Після успішного етапу публікації проекту з боку клієнта, в панелі адміністратора з'являється нова заявка на публікацію проекту, яку необхідно обробити. В силу великої довжини форми, далі буде розглянено окремо кожну секцію.

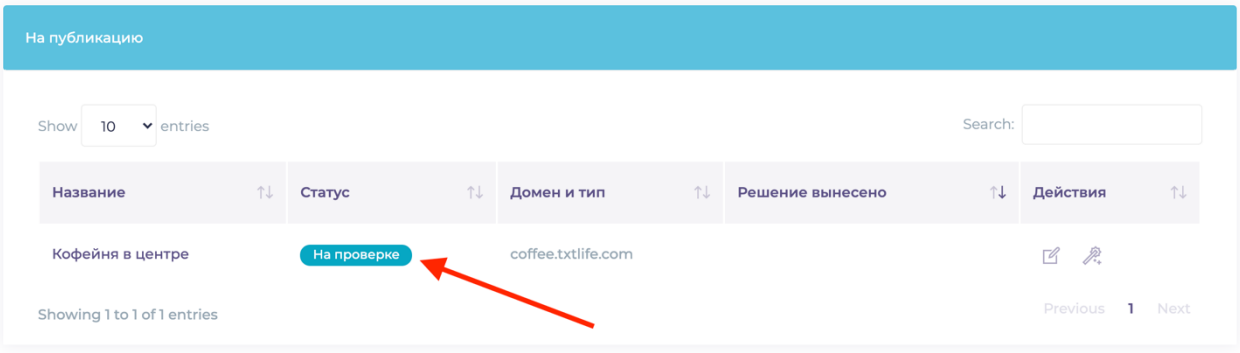

Рисунок 3.30. Нова заявка на публікацію проекту

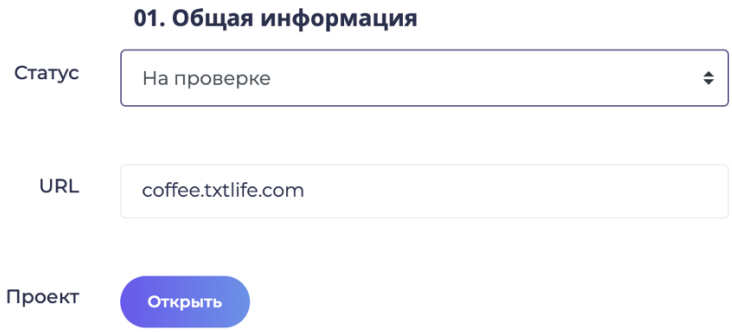

Рисунок 3.31. Форма публікації проекту

На рисунку 3.31 відображена секція загальної інформації про поточний проект до публікації. Адміністратор може редагувати поточний статус проекту, переглянути домен і його тип, а також перейти в редактор даного проекту, для внесення необхідних правок.

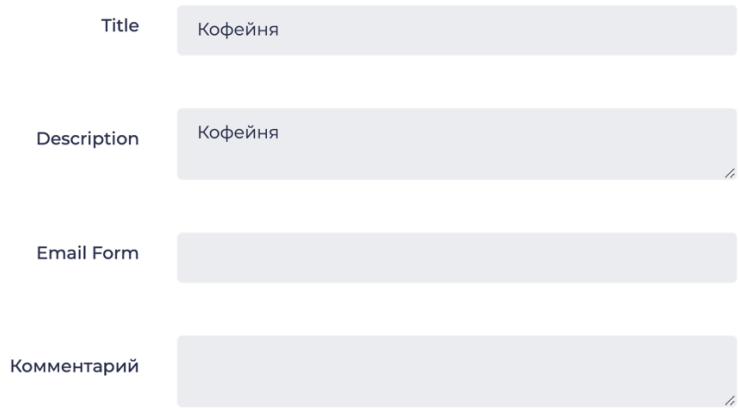

Рисунок 3.32. Форма публікації проекту

Вся інформація, яку вказував користувач, при публікації проекту - відображається на поточній сторінці (рисунок 3.32).

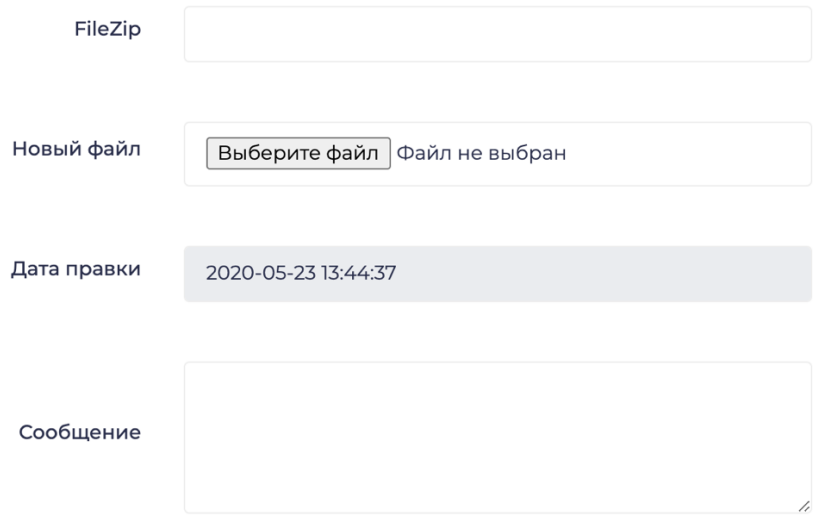

Рисунок 3.33. Форма публікації проекту

Функціонал конструктора дозволяє здійснювати скачування власних проектів до себе на пристрій. Це стає можливим, після того, як адміністратор перевірить шаблон, і вивантажить його на сервер конструктора в форматі ZIP (рисунок 3.33).

Після відправки форми, буде створено унікальне посилання на архів з проектом. Також, є поле "Повідомлення", яке адміністратор заповнює, якщо необхідно відхилити поточний проект до публікації. В даному полі описується причина такого рішення і далі буде показано користувачеві в кабінеті.

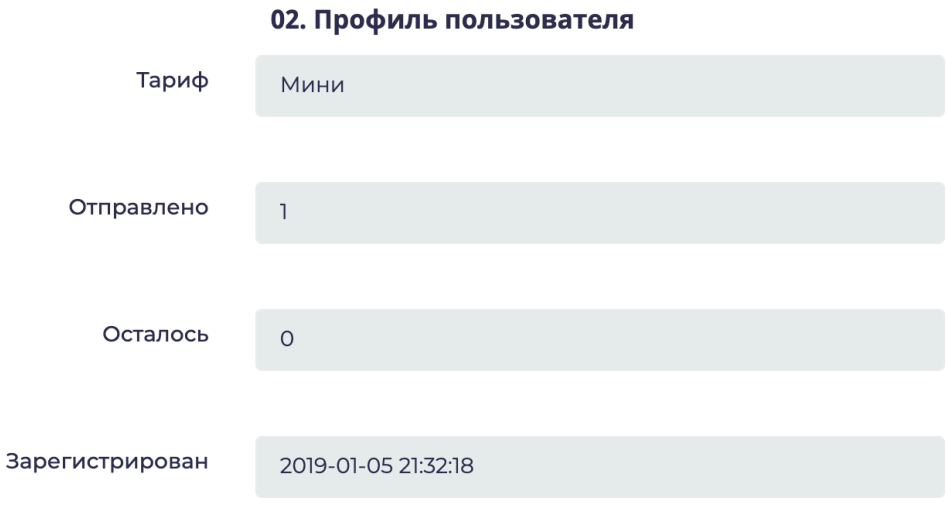

Рисунок 3.34. Форма публікації проекту

Крім інформації про проект, адміністратору відображається основна статистика про користувача, його тариф і кількість проектів, відправлених на публікацію (рисунок 3.34).

ВИСНОВКИ

В процесі виконання дипломної роботи було розглянуто питання архітектури конструкторів сайтів, вибору правильних мов і технологій, які найбільше підходять під завдання подібних проектів.

На основі аналізу, необхідних до зберігання даних, була спроектована даталогічна модель БД. Виходячи з даних даталогічної моделі БД, була спроектована фізична БД.

Виходячи з типу проекту і списку завдань, був обраний і вивчений фреймвок Laravel, його особливості, такі як модульність, заповітність, маршрутизація, управління конфігурацією, builder запитів та ORM тощо.

За результатами вивчених матеріалів і досліджень, була розроблена веб-платформа, що дозволяє створювати односторінкові сайти, без необхідності знань в програмуванні для користувача. Програмна система складається з модулів для обробки і збереження даних, а також сучасного і адаптивного веб-інтерфейсу.

На етапі тестування користувачами, було виявлено ряд технічних недоліків, які не є уразливостями системи. Наприклад, процес публікації сайту можна зробити повністю автоматизованим, шляхом створення додаткових модулів, для роботи з файловою системою дерикторії проектів і модулем Apache, для створення піддоменів та папок проектів. Роблячи висновок з технічних недоліків, можна сказати, що проект не досить автоматизований. У цього є і своє виправдання, в силу тестування актуальності проекту. Немає сенсу витрачати величезні ресурси, роблячи ідеальну у всьому систему, поки актуальність проекту не буде доведена користувачами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке конструтор веб-сайтів? - websitetooltester.com [Електронний ресурс] - Режим доступу: - https://www.websitetooltester.com/en/online-website-builder/
2. HTML & CSS - www.w3.org [Електронний ресурс] - Режим доступу: https://www.w3.org/standards/webdesign/htmlcss.html
3. 9 веб-технологій, які кожен веб-розробник повинен знати в 2020 році - tms-outsource.com [Електронний ресурс] - Режим доступу: https://tms-outsource.com/blog/posts/web-technologies/
4. Переваги побудови динамічних сайтів - hackernoon.com [Електронний ресурс] - Режим доступу: https://hackernoon.com/the-advantages-of-building-dynamic-websites-bc914071a155
5. 15 Основних вимог до веб сайту - studfile.net [Електронний ресурс] - https://timeweb.com https://studfile.net/preview/6438407/page:3/
6. Про фреймворк Laravel - timeweb.com [Електронний ресурс] - Режим доступу: https://timeweb.com/ru/community/articles/o-freymvorke-laravel-i-cms-na-baze-nego-1
7. Laravel – Огляд - tutorialspoint.com [Електронний ресурс] - Режим доступу: https://www.tutorialspoint.com/laravel/laravel_overview.htm
8. Шаблон MVC та PHP, частина 1 - sitepoint.com [Електронний ресурс] - Режим доступу: https://www.sitepoint.com/the-mvc-pattern-and-php-1/
9. Шаблон MVC та PHP, частина 2 - sitepoint.com [Електронний ресурс] - Режим доступу: https://www.sitepoint.com/the-mvc-pattern-and-php-2/
10. Що таке MySQL? - atlantic.net [Електронний ресурс] - Режим доступу: https://www.atlantic.net/what-is-mysql/

ДОДАТКИ

ДОДАТОК А. СТРУКТУРА ТАБЛИЦЬ В БАЗІ ДАНИХ

Таблиця А.1

Схема	Таблиця	Атрибут	Тип
builder	apps	id	INT(10)
builder	apps	id_user	INT(10)
builder	apps	id_app	VARCHAR(255)
builder	apps	title	VARCHAR(255)
builder	apps	status_publish	TINYINT(1)
builder	apps	code	MEDIUMTEXT
builder	apps	created_at	TIMESTAMP
builder	apps	updated_at	TIMESTAMP
builder	apps	my_domain	TINYINT(1)
builder	apps	type_publish	VARCHAR(20)
builder	apps	url_publish	VARCHAR(255)
builder	apps	message_text	TEXT
builder	apps	published_at	DATE
builder	apps	order_id	VARCHAR(255)
builder	invoices	id	INT(11)
builder	invoices	id_user	INT(11)
builder	invoices	tariff	VARCHAR(255)
builder	invoices	status	TINYINT(1)
builder	invoices	order_id	VARCHAR(255)
builder	invoices	created_at	TIMESTAMP
builder	invoices	updated_at	TIMESTAMP
builder	invoices	sum	FLOAT
builder	migrations	id	INT(10)
builder	migrations	migration	VARCHAR(255)
builder	migrations	batch	INT(11)
builder	password_resets	email	VARCHAR(255)
builder	password_resets	token	VARCHAR(255)
builder	password_resets	created_at	TIMESTAMP
builder	roles	id	INT(10)
builder	roles	name	VARCHAR(255)
builder	roles	description	VARCHAR(255)
builder	roles	created_at	TIMESTAMP
builder	roles	updated_at	TIMESTAMP
builder	role_user	id	INT(10)
builder	role_user	role_id	INT(10)
builder	role_user	user_id	INT(10)
builder	users	id	INT(10)
builder	users	name	VARCHAR(255)
builder	users	email	VARCHAR(255)

builder	users	password	VARCHAR(255)
builder	users	remember_token	VARCHAR(100)
builder	users	created_at	TIMESTAMP
builder	users	updated_at	TIMESTAMP
builder	users	type	VARCHAR(100)
builder	users	referred_by	VARCHAR(255)
builder	users	affiliate_id	VARCHAR(255)

ДОДАТОК Б1. ПІДКЛЮЧЕННЯ ДОДАТКОВИХ КЛАСІВ І МЕТОД ВІДОБРАЖЕННЯ КАБІНЕТУ КОРИСТУВАЧА

```
namespace App\Http\Controllers\Builder;

use App\App;
use App\User;
use App\Invoices;
use Carbon\Carbon;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use LiqPay\LiqPay;

use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Session;



class HomeController extends Controller
{

…

public function index () {

    $query = App::where('id_user', Auth::id())->get();
    $count = count($query);
    $type = User::where('id', Auth::id())->first();

    $type_limit = FALSE;
    $tariff_array = [TRUE, TRUE, TRUE];

    switch ($type->type) {
       case 'mini':
          if ($count >= 7) {
             $type_limit = TRUE;
          }
          $tariff_array = [FALSE, TRUE, TRUE];
```

```
            break;
        case 'start':
            if ($count >= 20) {
                $type_limit = TRUE;
            }
            $tariff_array = [FALSE, FALSE, TRUE];
            break;
        case 'profi':
            if ($count >= 50) {
                $type_limit = TRUE;
            }
            $tariff_array = [FALSE, FALSE, FALSE];
            break;
    }

    return view('builder.profile.home', [
        'title' => 'TxtLife | Builder',
        'description' => '',
        'projects' => $query,
        'type_limit' => $type_limit,
        'tariff_array' => $tariff_array
    ]);
}
```

ДОДАТОК Б2. СТВОРЕННЯ НОВОГО ПРОЕКТУ І ЙОГО ЗАПИС У БД

```
public function store (Request $request)
{

    $validate = $request->validate([
        'title' => 'required'
    ]);

    if ($validate) {

        $id_app = sha1(time());

        $insert = App::insert([
            'id_user' => Auth::id(),
            'id_app' => $id_app,
            'title' => $request->input('title'),
            'code' => $request->input('code'),
            "created_at" =>  \Carbon\Carbon::now(),
            "updated_at" => \Carbon\Carbon::now()
        ]);

        if ($insert) {
            return response()->json([
                'status' => 'success',
```

```
                'id_app' => '<a href="/dashboard/app/publish/'.$id_app.'" class="btn btn-
success">Опубликовать</a>',
                'work_app' => '<a href="/dashboard/app/'.$id_app.'" class="btn btn-
secondary">Продолжить работу</a>',
            ]);
        }
    }

    return response()->json(['status' => 'error']);
  }
```

ДОДАТОК Б3. ПЕРЕВІРКА ПОДДОМЕНА НА ЗАЙНЯТІСТЬ

```
public function check_domain_parent (Request $request, $type = FALSE) {

    if ($type == 0) {
        $domain = 'https://'.$request->input('domain').'.txtlife.com';

        //проверка на валидность урла
        if(!filter_var($domain, FILTER_VALIDATE_URL)){
            return response()->json(['status' => 'error_url']);
        }
        //инициализация curl
        $curlInit = curl_init($domain);
        curl_setopt($curlInit,CURLOPT_CONNECTTIMEOUT,10);
        curl_setopt($curlInit,CURLOPT_HEADER,true);
        curl_setopt($curlInit,CURLOPT_NOBODY,true);
        curl_setopt($curlInit,CURLOPT_RETURNTRANSFER,true);

        //получение ответа
        $response = curl_exec($curlInit);
        curl_close($curlInit);
        if ($response)  return response()->json(['result' => 'domain_isset']);

        return response()->json(['result' => 'success']);
    }
    elseif ($type == 1) {
        $domain = $request->input('domain');

        $url =
'https://api.reg.ru/api/regru2/domain/check?input_data={%22domains%22:[{%22dname%22:
%22'.$domain.'%22}]}&input_format=json&password=test&username=test';


        $ch = curl_init($url); // such as http://example.com/example.xml
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($ch, CURLOPT_HEADER, 0);

        $data = curl_exec($ch);

        curl_close($ch);
```

```
        return response()->json($data);
    }
  }
```

ДОДАТОК Б4. ПЕРЕВІРКА ДОМЕНА НА ЗАЙНЯТІСТЬ

```
public function check_whois (Request $request) {

    $domain = $request->input('domain');

    $url =
'https://api.reg.ru/api/regru2/domain/check?input_data={%22domains%22:[{%22dname%22:
%22'.$domain.'%22}]}&input_format=json&password=test&username=test';


    $ch = curl_init($url); // such as http://example.com/example.xml
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HEADER, 0);

    $data = curl_exec($ch);

    curl_close($ch);

    return response()->json($data);
  }
```

ДОДАТОК Б5. ОБРОБКА ДАНИХ ПУБЛІКАЦІЇ

```
public function publish_project (Request $request) {

    $check = $request->validate([
      'domain' => 'required',
      'my_domain' => 'numeric',
      'password' => 'required',
      'type' => 'alpha'
    ]);

    if (!$check) return response()->json(['status' => 'error']);

    $user = User::findOrFail(Auth::id());

    if (!password_verify( $request->input('password'), $user->password )) return response()-
>json(['status' => 'auth_error']);

    if ($request->input('type') == 'trial') {

      $domain = 'https://'.$request->input('domain').'.txtlife.com';

      //проверка на валидность урла
```

```
        if(!filter_var($domain, FILTER_VALIDATE_URL)){
            return response()->json(['status' => 'error_url']);

        }
        //инициализация curl
        $curlInit = curl_init($domain);
        curl_setopt($curlInit,CURLOPT_CONNECTTIMEOUT,10);
        curl_setopt($curlInit,CURLOPT_HEADER,true);
        curl_setopt($curlInit,CURLOPT_NOBODY,true);
        curl_setopt($curlInit,CURLOPT_RETURNTRANSFER,true);

        //получение ответа
        $response = curl_exec($curlInit);
        curl_close($curlInit);

        if ($response) {

            return response()->json(['status' => 'domain_isset']);

        } else {

            $send_publish = App::where(['id_user' => Auth::id(), 'id_app' => $request-
>input('id_app')])->update([
                'status_publish' => 1,
                'url_publish' => $request->input('domain').'.txtlife.com',
                'type_publish' => 'trial'
            ]);

            if (!$send_publish) return response()->json(['status' => 'other_error']);

            Session::flash('success_publish', 'Проект будет опубликован после модерации');
            return response()->json(['status' => 'redirect']);
        }

    }

    elseif ($request->input('type') == 'publish') {

        $domain = $request->input('domain');

        if ($request->input('mydomain') == 0) {


            $url =
'https://api.reg.ru/api/regru2/domain/check?input_data={%22domains%22:[{%22dname%22:
%22'.$domain.'%22}]}&input_format=json&password=test&username=test';


            $ch = curl_init($url);
            curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
            curl_setopt($ch, CURLOPT_HEADER, 0);
```

```
        $data = curl_exec($ch);

        curl_close($ch);

        $array = json_decode($data);

        $response_status = $array->answer->domains[0];

        if (isset($response_status->error_code)) return response()->json(['status' =>
'domain_isset']);

        if (!isset($response_status->error_code) && isset($response_status->result) &&
$response_status->result == 'Available') {

            $order_id = sha1(time());

            /**
             * Обновляем запись
             */
            $send_publish = App::where(['id_user' => Auth::id(), 'id_app' => $request-
>input('id_app')])->update([
                'status_publish' => 4,
                'url_publish' => $request->input('domain'),
                'type_publish' => 'publish',
                'my_domain' => 0,
                'order_id' => $order_id
            ]);

            if (!$send_publish) return response()->json(['status' => 'error']);

            /**
             * Формируем оплату
             */
            $liqpay = new LiqPay(env('LIQPAY_PUBLIC', ''), env('LIQPAY_PRIVATE', ''));
            $sum = 500;

            if ($sum !== NULL) {

                //SQL запись
                Invoices::insert([
                    'id_user' => Auth::id(),
                    'tariff' => 'publish',
                    'sum' => $sum,
                    'order_id' => $order_id,
                    'created_at' =>  Carbon::now(),
                    'updated_at' => Carbon::now()
                ]);

                $html = $liqpay->cnb_form(array(
                    'version'       => '3',
```

```
                'action'        => 'pay',
                'amount'        => $sum,
                'currency'      => 'UAH',
                'description'   => 'Публикации сайта '.$domain,
                'order_id'      => $order_id,
                'result_url'    =>      env('LIQPAY_RESULT', ''),
                'server_url'    =>      env('LIQPAY_SERVER', ''),
                'language'              =>      'ru',
                'sandbox'       =>  '1'

            ));

            $res_arr = array("status"=> 1, 'form'=> $html, 'order_num'=> 1, 'error'=> 1);
            echo json_encode( $res_arr );

        }


//              Session::flash('success_publish_new', 'На ваш Email мы отправим дальнейшие
инструкции');
//              return response()->json(['status' => 'redirect']);

        } else {
            return response()->json(['status' => 'domain_isset']);
        }
    }
    elseif ($request->input('mydomain') == 1) {

        $order_id = sha1(time());

        /**
         * Обновляем запись
         */
        $send_publish = App::where(['id_user' => Auth::id(), 'id_app' => $request-
>input('id_app')])->update([
            'status_publish' => 4,
            'url_publish' => $request->input('domain'),
            'type_publish' => 'publish',
            'my_domain' => 1,
            'order_id' => $order_id
        ]);

        if (!$send_publish) return response()->json(['status' => 'error']);

        /**
         * Формируем оплату
         */
        $liqpay = new LiqPay(env('LIQPAY_PUBLIC', ''), env('LIQPAY_PRIVATE', ''));
        $sum = 500;

        if ($sum !== NULL) {
```

```
            //SQL запись
            Invoices::insert([
               'id_user' => Auth::id(),
               'tariff' => 'publish',
               'sum' => $sum,
               'order_id' => $order_id,
               'created_at' =>  Carbon::now(),
               'updated_at' => Carbon::now()
            ]);

            $html = $liqpay->cnb_form(array(
               'version'       => '3',
               'action'        => 'pay',
               'amount'        => $sum,
               'currency'      => 'UAH',
               'description'   => 'Публикации сайта '.$domain,
               'order_id'      => $order_id,
               'result_url'        =>        env('LIQPAY_RESULT', ''),
               'server_url'        =>        env('LIQPAY_SERVER', ''),
               'language'                    =>        'ru',
               'sandbox'      =>  '1'

            ));

            $res_arr = array("status"=> 1, 'form'=> $html, 'order_num'=> 1, 'error'=> 1);
            echo json_encode( $res_arr );

         }
      }
   }
}
```

ДОДАТОК Б6. МЕТОД СКАЧУВАННЯ ПРОЕКТУ НА КОМП'ЮТЕР

```
public function download ($id) {

    $file_uri = App::where(['id_user' => Auth::id(), 'id_app' => $id])->select('dropbox')->first();

    $file = storage_path()."/app/".$file_uri->dropbox;

    return response()->download($file);
}
```

ДОДАТОК Б7. ЗАВАНТАЖЕННЯ ИНТЕРФЕСА ПУБЛІКАЦІЇ ПРОЕКТУ

```
public function publish (App $app)
{

    $type = User::where('id', Auth::id())->select('type')->first();
```

```
        if ($type->type == NULL) {
            Session::flash('error_publish', 'У вас нет права на публикацию. Оплатите желаемый
тариф ниже.');
            return redirect()->route('app.index');
        }

        $check = App::where('id_app', $app->id_app)->select('status_publish')->first();

        if ($check->status_publish == 1) {
            Session::flash('error_publish', 'Данный проект еще не прошел модерацию.');
            return redirect()->route('app.index');
        } else if ($check->status_publish == 2) {
            Session::flash('error_publish', 'Данный проект был отклонен на публикацию.');
            return redirect()->route('app.index');
        } else if ($check->status_publish == 3) {
            Session::flash('error_publish', 'Данный проект уже опубликован.');
            return redirect()->route('app.index');
        }

        $query = App::where(['id_user' => Auth::id()])->whereIn('status_publish', [1, 3])->get();
        $count = count($query);

        $type = User::where('id', Auth::id())->select('type')->first();

        $tariff = '';

        switch ($type->type) {
            case 'mini':
                if ($count >= 1) {
                    $this->type_limit = TRUE;
                } else {
                    $this->left = 1 - $count;
                    $this->percent = 100 - (($this->left * 100) / 1);
                    $tariff = 'Мини';
                }
                break;
            case 'start':
                if ($count >= 5) {
                    $this->type_limit = TRUE;
                } else {
                    $this->left = 5 - $count;
                    $this->percent = 100 - (($this->left * 100) / 5);
                    $tariff = 'Старт';
                }
                break;
            case 'profi':
                if ($count >= 10) {
                    $this->type_limit = TRUE;
                } else {
                    $this->left = 10 - $count;
```

```
            $this->percent = 100 - (($this->left * 100) / 10);
            $tariff = 'Профи';
        }
        break;
    }

    if ($this->type_limit) {
        Session::flash('error_limit', 'Превышен тарифный лимит на публикацию.');
        return redirect()->route('app.index');
    }



    return view('builder.profile.publish', [
        'title' => 'Builder | Публикация проекта "'.$app->title.'"',
        'description' => '',
        'app' => $app,
        'left' => $this->left,
        'percent' => $this->percent,
        'tariff' => $tariff
    ]);
}
```

ДОДАТОК Б8. МЕТОД ОТРИМАННЯ POST ОПОВІЩЕННЯ ВІД LIQPAY

```
public function liqpay_status (Request $request) {

    /**
     * Получаем данные
    */
    $r = $request->input('data');

    $result= json_decode(base64_decode($r));

    $sign = base64_encode( sha1(env('LIQPAY_PRIVATE', '').$r.env('LIQPAY_PRIVATE', ''), 1 ));

    /**
     * Проверяем сигнатуры
     */
    if($sign === $request->input('signature')) {

        /**
         * Делаем обновление
         */
        if( $result->status == 'success' ) {

            Invoices::where('order_id', $result->order_id)->update([
                'status' => 1,
                'updated_at' => Carbon::now()
            ]);
```

```
            /**
             * Обновляем тариф
             */
            $query = Invoices::where('order_id', $result->order_id)->select('tariff', 'id_user',
'order_id')->first();

            /**
             * Обновляем статус, если это оплата публикации
             */
            if ($query->tariff == 'publish') {
               App::where(['id_user' => $query->id_user, 'order_id' => $query->order_id])-
>update([
                  'status_publish' => 1,
                  'updated_at' => Carbon::now()
               ]);
            } else {
               User::where('id', $query->id_user)->update([
                  'type' => $query->tariff
               ]);
            }

         }
      }

   }
```

ДОДАТОК Б9. МЕТОД ПУБЛІКАЦІЇ ПРОЕКТУ АДМІНІСТРАТОРОМ

```
public function update (Request $request, App $app, $id)
   {

      $check = $request->validate([
         'status' => 'numeric',
         'url_publish' => 'required'
      ]);

      if(!$check) return back();

      $path = $request->input('dropbox');

      /**
       * Проверка наличия файла
       */
      if($request->hasFile('dropbox_file')) {


         $path  = $request->file('dropbox_file')->store('projects');

         /**
          * Проверка загрузки файла
```

```
         */
        if (!$request->file('dropbox_file')->isValid()) {

            return back()->withErrors('Проект не импортирован!');

        } else {

            /**
             * Удаляем прошлый файл, если таков есть
             */
            $project = $app->where('id_app', $id)->first();

            if ($project->dropbox !== NULL) {
                Storage::delete($project->dropbox);
            }
        }
    }

    $message = $request->input('message_text');

    if (empty($message)) $message = NULL;

    $insert = $app->where('id_app', $id)->update([
        'status_publish' => $request->input('status'),
        'url_publish' => $request->input('url_publish'),
        'message_text' => $message,
        'published_at' => Carbon::now(),
        'dropbox' => $path
    ]);



    if ($insert) {
        Session::flash('success', 'Проект изменен!');
        return redirect()->route('admin.index');

    } else return back()->withErrors('Ошибка! Повторите попытку');

}
```