МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Веб-система активного управління енергозабезпеченням від альтернативних джерел енергії»

за напрямом підготовки 6.050101 «Комп'ютерні науки»

Виконавець роботи: студент групи ІТ-52 Портяной Максим Дмитрович

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____«__» _____2019 р.

Науковий керівник _____ к.т.н., доц., Шендрик В.В.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ Шифрін Д. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2019

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Напрям підготовки – 6.050101 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«__» _____ 2019 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Портяной Максим Дмитрович

1 Тема роботи *Веб-система активного управління енергозабезпеченням від альтернативних джерел енергії*

керівник роботи *Шендрик Віра Вікторівна, к.т.н., доцент*,

затверджені наказом по університету від « 17 » травня 2019 р. № 0834-III

2 Строк подання студентом роботи «28» травня 2019 р.

3 Вхідні дані до роботи Перелік систем аналогів, список рекомендованої літератури, обрис технічного завдання.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз предметної області, постановка задачі, проектування веб системи та розробка веб системи.

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

6. Консультанти розділів роботи:

Розділ	**Консультант**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка

Студент _____ Портяной М.Д.
(підпис)

Керівник роботи _____ к.т.н., доц. Шендрик В.В.
(підпис)

РЕФЕРАТ

Тема роботи: «Веб-система активного управління енергозабезпеченням від альтернативних джерел енергії».

Пояснювальна записка містить вступ, 4 розділи, висновки, додатки та список літератури, включає 83 сторінки, 4 таблиці, 32 ілюстрації, 15 джерел.

В першому розділі визначається актуальність напряму дослідження. Наводиться огляд стану сучасних інформаційних технологій, що можуть бути використані у процесі розробки продукту, проводиться аналіз переваг та недоліків систем-аналогів та формується перелік вимог, яким має відповідати кінцевий продукт.

Другий розділ включає в себе формулювання мети проекту, його задач та методів дослідження. Також у цьому розділі проводиться планування робіт по розробці продукту.

У третьому розділі проводиться детальний опис поетапного проектування веб системи та розробки архітектури додатку. Цей розділ також включає в себе структурно-функціональний аналіз проекту, описані основні принципи роботи розробленої веб системи.

У четвертому розділі описується розробка веб системи, що є практичним застосуванням запропонованих методологій дослідження. Детально наводяться етапи розробки та приклади використання функцій системи.

Результатом роботи є реалізована веб система для активного управління енергозабезпеченням від альтернативних джерел енергії.

Ключові слова: веб система, сайт, додаток, енергозабезпечення, альтернативні джерела енергії.

ЗМІСТ

ВСТУП

Кожного року людство використовує приблизно 35 мільярдів барелів нафти. Безумовно, нафтопродукти є одним з найбільших забруднювачів планети: пластик, поліетилен та пальне – це лише маленька частина нафтового вкладу у забруднення усіх без виключення шарів біосфери. Але найважливішим є те, що корисні копалини не відновлюються. За підрахунками вчених, людство вже використало приблизно 40% світового запасу нафти [1].

Сьогодні усі без виключення розвинені країни розвивають індустрії з переробки відходів. Серед світових лідерів за показником відсотку перероблених відходів такі країни, як Німеччина (65%), Південна Корея (59%) та Словенія (58%). Отже, людство має приблизне уявлення про те, як обійтися без корисних копалин для виробництва продуктів життєдіяльності. Однак енергію, яку люди виробляють, використовуючи вугілля та нафту переробити неможливо. Для того, аби досягти незалежності від вичерпних ресурсів, ми маємо використовувати альтернативні джерела енергії. Вчені давно винайшли способи добувати енергію, використовуючи сонячне світло, вітер та рух води, проте методи впровадження цих технологій у повсякденне життя досі потребують обговорення.

На сьогоднішній день людство використовує відновлювальні джерела енергії лише на 13% від своїх загальних потреб. Для того, аби досягти 100% ефективності використання альтернативних джерел, потрібно вирішити питання політики, логістики, комерційні проблеми і так далі. Повний перехід на альтернативні джерела енергії – задача дуже комплексна та, нажаль, абстрактна, і потребує поетапного підходу. Визначення етапів для такого масштабного процесу та підготовка ресурсів, необхідних для їх реалізації це те, що ми можемо зробити вже сьогодні.

Масштабне виробництво енергії за допомогою відновлювальних джерел є предметом дослідження міжнародних корпорацій та світових лідерів енергетичної індустрії. Однак виробництво енергії у малих масштабах для

потреб населення – це питання іншого масштабу. Сьогодні сонячні панелі та вітряки набувають популярності у населення. «Зелена» енергія є досить ефективним рішенням для забезпечення електроенергією приватних будинків і, навіть, квартирних комплексів. Проте засоби переходу на екологічно-чисті джерела енергії є досі незрозумілими для більшості населення. Наприклад, управління енергозабезпеченням від приватних сонячних панелей або вітряків, контроль показників ефективності та потенціал – це лише деякі характеристики, які потрібно враховувати при переході на альтернативні джерела для приватного користування. На сьогодні в Україні немає вітчизняних аналогів таким системам, як Amazon Alexa або Google Assistant, які забезпечують контроль споживання енергії та показників ефективності. Але жодна з перелічених систем не є специфічною для споживачів «зеленої» енергії. Тому метою цього проекту є створення веб-системи активного управління енергозабезпеченням від альтернативних джерел енергії для приватного використання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Актуальність проекту

Справедливо припустити, що, витрачаючи великі кошти на встановлення енергоефективних екологічно-чистих систем, приватний власник зацікавлений у тому, аби побачити результат та реальні зміни. Системи енергозабезпечення від альтернативних джерел, як правило, обіцяють значні покращення, економію та високу ефективність, отже сприймаються населенням як інвестиція у власне майбутнє. Аксіомою бізнесу є те, що інвестор завжди очікує, що кошти, вкладені ним будуть працювати на нього. Отже приватний власник, інвестувавши кошти у свій благоустрій, встановивши сонячну панель або вітряк, очікує побачити, як ці інвестиції повертаються. І, хоча встановлення сонячних панелей з метою забезпечення енергією приватного будинку буде окупатися від 7 до 15 років, приватний власник все ж таки зацікавлений у тому, аби побачити зміни на власні очі.

Найпростішим прикладом може бути можливість контролювати витрати електроенергії, аналіз та статистика використання порівняно з минулим місяцем, співвідношення між кількістю виробленої та спожитої енергії. Добре відомо, що альтернативні джерела енергії залежать від погодних умов. Тому моніторинг статистики вироблення енергії залежно від погодних умов за конкретний проміжок часу (день, тиждень, місяць) зміг би вплинути на прийняття рішень щодо використання енергії. Наприклад, якщо навіть най похмурішого дня власник не використав і 50% від виробленої енергії, було б актуальним задуматися про продаж частини сонячних панелей, або розширення споживання - наприклад, продаж електроенергії сусідньому домогосподарству (звісно, останній варіант має бути пропрацьований з точки зору закону, але це лише припущення).

Крім того, відмова від використання централізованих систем енергозабезпечення населення на користь приватних «зелених» систем означає

можливість управління показниками та гнучкість налаштувань. Власник може бути зацікавлений у тому, аби мати змогу встановити ліміт на використання енергії у певний період часу, або взагалі обмежити енергопостачання на час своєї відсутності. Крім того, власник гібридної системи енергозабезпечення (сонячні панелі та вітряки) може захотіти контролювати джерело енергопостачання залежно від погоди. Сонячного та безвітряного літнього дня, наприклад, було б логічно використовувати сонячну енергію та відключити вітряк. Хмарного осіннього дня, навантаження на джерела можна було б розподілити рівномірно. Це дало б змогу використовувати джерела енергії з максимальною ефективністю.

Останнім фактором, що впливає на актуальність проекту можна визначити безпеку. Відсутність підключення до централізованих систем електропостачання повністю виключає вірогідність перепадів напруги у мережі, оскільки мережа складається лише з одного користувача. Крім того, така система управління енергозабезпеченням дає можливість попередження про надмірне споживання напруги (може бути актуальним взимку, при використанні нагрівальних засобів опалення, що можуть надмірно навантажувати мережу). Не кажучи вже про те, що при пожежі енергозабезпечення всього будинку може бути вимкнено дистанційно за допомогою комп’ютера.

1.2. Визначення цілей проекту

Тепер, визначивши актуальність проекту, можна встановити його цілі. Веб-система – це програмний продукт, що базується на веб-сервері і доступ до якого здійснюється за допомогою веб-браузеру. Для того, аби максимізувати ефективність системи при роботі з певною платформою або типом продукту, потрібно враховувати особливості саме тієї платформи. У даному випадку, при розробці веб-системи потрібно враховувати стандарти та особливості, які характерні саме до веб-індустрії, а саме:

- дизайн та інтерфейс має відповідати найсучаснішим стандартам та тенденціям веб-дизайну;
- швидкість завантаження веб-сторінок повинна бути максимальною;
- веб-додаток має підтримуватися більшістю сучасних браузерів;
- веб-сторінки мають бути оптимізовані для користувачів з вадами зору або повною його відсутністю;
- швидкість обробки запитів користувача сервером повинна бути оптимізована;
- з метою полегшення використання веб-додатку, деяка частина запитів до серверу повинна оброблятися без перезавантаження сторінки;
- веб-додаток повинен відповідати останнім стандартам веб-безпеки та захисту приватної інформації.

Таким чином, базуючись на платформі, яку було обрано для реалізації, можна розбити цілі проекту на дві категорії: візуальна складова проекту та технічна. Для кожної з цих категорій цілі та засоби будуть різними, залежно від потреб та вимог проекту.

1.3. Візуальна складова

Для того, аби створити веб-додаток, потрібно спочатку створити дизайн інтерфейсу. Крім того, продукт повинен мати логотип та відповідати стилістиці компанії або стартапу, що буде займатися його розвитком та впровадженням. Дизайн інтерфейсу повинен відповідати наступним стандартам:

- бути візуально привабливим;
- важливі елементи інтерфейсу повинні виділятися стилістично (наприклад, кнопка «Придбати», «Зареєструватися» або «Дізнатися більше»);
- шрифти, використані у дизайні повинні бути векторними та підтримуватися веб-браузерами;

- зручний та інтуїтивний інтерфейс користувача;
- графічні матеріали такі, як логотип та іконки за можливістю повинні бути векторними зображеннями для кращого відображення на різних екранах;
- при розробці дизайну слід використовувати мобільно-орієнтований підхід (mobile first).

1.4. Технічна складова

Про розробці веб-системи слід враховувати останні тенденції в індустрії веб-розробки. Для даного веб-додатку наявність бази даних є необхідним параметром. Крім того, при розробці веб-додатку слід враховувати, що у майбутньому система може бути розвинута як програмний продукт або мобільний додаток (або і те, і інше), тому, слід дотримуватись найсучасніших тенденцій у сфері розробки крос-платформних веб-додатків (REST API) [2]. Вимоги до технічної частини:

- максимальна швидкість завантаження веб-сторінок;
- веб-додаток має підтримуватися більшістю сучасних браузерів;
- веб-сторінки мають бути оптимізовані для користувачів з вадами зору або повною його відсутністю;
- швидкість обробки запитів користувача сервером повинна бути оптимізована;
- з метою полегшення використання веб-додатку, деяка частина запитів до серверу повинна оброблятися без перезавантаження сторінки [3];
- веб-додаток повинен відповідати останнім стандартам веб-безпеки та захисту приватної інформації;
- чистий, задокументований код;
- передбачити можливість багаторівневої авторизації: гість, користувач, абонент та адміністратор.

2 ПОСТАНОВКА ЗАДАЧІ

2.1. Мета та задачі

Створення веб-системи слід розпочати зі створення веб-сайту, присвяченому безпосередньо продукту. Сайт повинен ознайомлювати відвідувачів з веб-системою, описувати її можливості та функціонал з метою зацікавити та привабити майбутніх користувачів. Зацікавлений відвідувач повинен мати змогу зареєструватися у системі з метою отримувати новини та приймати участь у промо-акціях засобами електронної пошти. Відповідно, зареєстрований користувач повинен мати змогу увійти у свій особистий кабінет. В особистому кабінеті користувач матиме змогу ввести свою особисту інформацію, таку як адреса, повне ім’я, номер телефону та інше. Це потрібно для того, аби при підключенні альтернативних джерел енергії, провайдер сервісу мав точну інформацію про власника. Після того, як користувач запровадить особисту інформацію про себе, він переходить до категорії абонентів. Користувач сервісу – це фізична (або юридична) особа, яка зареєструвалась у сервісі, має особистий акаунт, але не запровадила особисту інформацію про себе. Користувач не повинен мати доступ до частини функціоналу сервісу, яка відповідає за налаштування сонячних панелей або вітряків. Абонент – це користувач сервісу, який запровадив свою особисту інформацію і, таким чином, матиме доступ до тієї частини сервісу, яка відповідає за налаштування та перегляд альтернативних джерел енергії. Однак, якщо абонент не має зареєстрованих джерел інформації, секція з функціоналом керування ними буде у нього пустою. Реєстрація джерел енергії для конкретного користувача буде здійснюватися адміністратором. Тільки після того, як адміністратор зареєструє у системі сонячну панель, яка належить певному абоненту, її власник матиме змогу побачити налаштування для неї. Таким чином, адміністратор повинен мати змогу переглядати інформацію усіх абонентів, зареєстрованих в системі та реєструвати і видаляти джерела енергії для конкретних абонентів. Адміністратор також повинен мати

смогу редагувати особисту інформацію абонента (на випадок, якщо останній не матиме змоги оновити її самостійно).

2.2. Вибір засобів реалізації

Оскільки створення такої веб-системи неможливе без бази даних, з метою кращої структуризації програмного коду, підвищення швидкодії та вирішення частини питань безпеки, доцільно обрати фреймворк для обробки серверної частини додатку. Для цього було обрано MVC PHP-фреймворк Laravel.

MVC (Model, View, Controller) – це архітектурна структура, що широко використовується для розробки інтерфейсу користувача, методом розділення додатку на три взаємопов’язані частини: модель, вид та контролер. При розробці веб-додатків часто використовується однаковий шаблон комунікації між сервером та клієнтом (браузером): сервер оброблює надісланий клієнтом запит, робить запит до бази даних, далі збирає інформацію, отриману з бази даних з інформацією про клієнта, та відповідає на запит клієнта. MVC-фреймворк виступає у даному випадку провайдером шаблонів для комунікацій між клієнтом (вид), сервером (контролер) та базою даних (модель) [4]. Переваги використання MVC-фреймворку при розробці веб-додатку:

- швидший процес розробки;
- підтримка асинхронних запитів;
- структуризація коду;
- зручний для користування формат відповіді на запит;
- вбудований інтерфейс захисту інформації;
- можливість модифікації конкретних частин без впливу на інші.

Laravel – популярний у спільноті веб-розробників фреймворк, побудований з підтримкою архітектури MVC та є зручним у використанні та підтримці. Переваги використання Laravel:

- вбудована системи аутентифікації та авторизації;

- проста інтеграція з поштовими сервісами;
- проста інтеграція з засобами веб-маркетингу;
- зручний формат обробки помилок та виключень;
- конфігурація маршрутизації URL-адрес;
- дуже детальна документація та широка підтримка у спільноті.

Крім фреймворку для обробки серверної частини додатку, доцільно використовувати JavaScript-фреймворк для покращення інтерфейсу користувача. За JavaScript-фреймворк було обрано Vue.js. Переваги використання JavaScript-фреймворків:

- ефективність та швидкість розробки;
- велика кількість вбудованого функціоналу;
- зручний інтерфейс для створення асинхронних запитів [5];
- зручний інтерфейс обробки результатів запитів;
- підтримка усіма сучасними браузерами.

Vue.js – це фреймворк що набирає широку популярність у спільноті front-end веб розробників. Дає переваги у швидкості розробки.

З метою покращення інтерфейсу веб-додатку, було прийняте рішення використовувати CSS-бібліотеку Bulma. Серед переваг використання CSS-бібліотек можна виділити такі, як:

- швидкість розробки;
- велика кількість вбудованих UI-компонентів;
- простота використання;
- гнучкість та простота модифікації.

У якості програмного забезпечення для керування версіями програмного коду було обрано GitHub. Програмне забезпечення для керування версіями (version control software) – це програмний інструмент для контролю над поступовими змінами в програмному коді, додаткових файлах та залежностях проекту. Переваги використання ПЗ для керування версіями:

- зберігання вихідного коду проекту в хмарі;

- простота контролю версій вихідного коду: за потреби, в будь-який момент можна відновити останню збережену версію проекту, незалежно від того, скільки змін було зроблено після збереження;
- зручний інтерфейс для документації коду.

3 ПРОЕКТУВАННЯ ВЕБ СИСТЕМИ

3.1. Структурно-функціональне моделювання діяльності веб системи

Функціональна модель представляє з необхідним ступенем деталізації систему функцій, які в свою чергу відображають свої взаємовідносини через об’єкти системи. Вона являє собою ієрархію взаємопов’язаних діаграм, кожна з яких представляє підсистему або її окрему компоненту. Вершина цієї структури містить загальний опис системи, який деталізується на наступних рівнях декомпозиції.

Для створення функціональної моделі використано програмний продукт Draw.io.

Розглянемо головний напрямок діяльності веб системи – обробка та представлення інформації про систему енергозабезпечення веб системою. Контекстну діаграму даного процесу представлено на рис. 3.1.

Рисунок 3.1 – Контекстна діаграма

Кожна діаграма складається з блоків діяльності, що описують одну чи декілька функцій, та стрілок, що відображають ресурсі, потрібні для забезпечення діяльності. Діяльність є поіменованим процесом, функцією чи задачею, що виконується впродовж певного інтервалу часу та формує результати своєї роботи; відображається прямокутником з назвою у формі дієслова у середині. Блоки на діаграмі розташовуються у порядку домінування: найбільш домінуючий – у верхньому лівому куті, найменш домінуючий – у правому нижньому. Таким чином, структура діаграми описує вплив одних функцій на інші. Всі блоки нумеруються в порядку домінування. Стрілки або дуги зображують об’єкти, зв’язують блоки та відображають їх взаємодію; зображуються лініями зі стрілками на кінцях. Імена стрілок можуть приєднуватись до них за допомогою зиґзаґів. Між об’єктами (дугами) та функціями (діяльностями) можливі відношення: вхід (input), контроль (control), вихід (output) та механізм (mechanism). Кожне з названих відношень зображується дугою, зв’язаною з певною стороною блока: ліва сторона призначена для вхідних дуг, верхня - для дуг контролю, права - для вихідних дуг, нижня -для дуг механізмів.

Вхідними стрілками до функції «Обробка та представлення інформації веб системою» є «Дані про користувача» та «Дані про систему енергозабезпечення». Вихідними стрілками є «Інформація про стан системи енергозабезпечення» та «Статистичні дані про систему енергозабезпечення». Механізмом є «Адміністратор».

Будь-яка діяльність батьківської діаграми може бути деталізована на діаграмі декомпозиції, яка є дочірньою. На діаграмі декомпозиції зображуються блоки діяльності, що представляють функції, які є складовими батьківської діяльності. Всі дуги батьківської діяльності переносяться на дочірню. Для зв’язку батьківської та дочірньої діаграм використовують С-номери, що дозволяють уникнути неоднозначності зв’язку між діаграмами.

На першому рівні декомпозиції моделі головна батьківська діаграма розбивається на наступні блоки (представлено на рис. 3.2):

- Реєстрація користувача;
- Переведення користувача до категорії абонентів;
- Система управління енергозабезпеченням від альтернативних джерел електроенергії абонента.

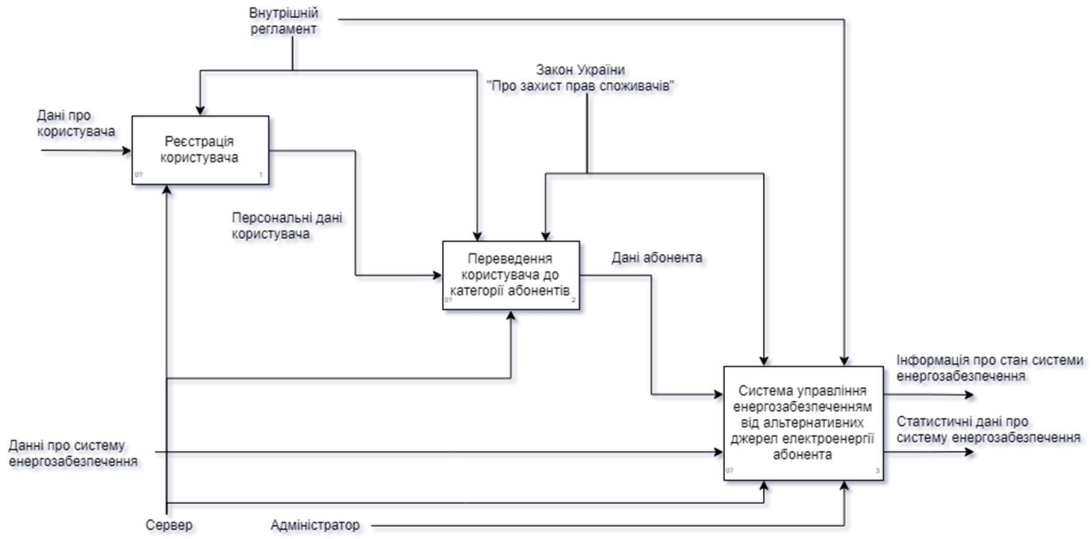

Рисунок 3.2 – Діаграма декомпозиції IDEF0

Вхідною стрілкою до діяльності «Реєстрація користувача» є «Дані про користувача»; вихідною – «Персональні дані користувача»; стрілкою контролю – «Внутрішній регламент», вихідною стрілкою є «Запровадження користувачем персональних даних».

Вхідною стрілкою до діяльності «Переведення користувача до категорії абонентів» є «Персональні данні користувача»; вихідною – «Дані абонента»; стрілками контролю є «Внутрішній регламент» та «Закон України “Про захист прав споживачів”».

Вхідними стрілками до діяльності «Система управління енергозабезпеченням від альтернативних джерел електроенергії абонента» є «Дані абонента» та «Дані про систему енергозабезпечення»; вихідними –

«Інформація про стан системи енергозабезпечення» та «Статистичні дані про систему енергозабезпечення»; стрілками контролю – «Внутрішній регламент» та «Закон України “Про захист прав споживачів”»; стрілкою механізмів – «Адміністратор».

Всього у моделі два рівні декомпозиції. Діаграму декомпозиції процесу «Система управління енергозабезпеченням від альтернативних джерел електроенергії абонента» зображено на рис. 3.3.

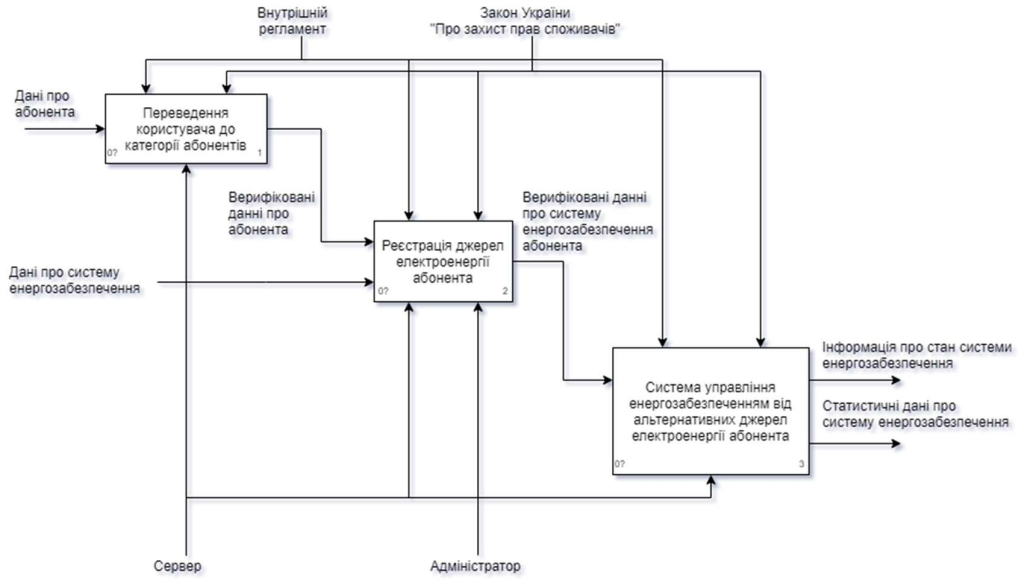

Рисунок 3.3 - Діаграма декомпозиції IDEF0 «Система управління енергозабезпеченням від альтернативних джерел електроенергії абонента»

Вхідною стрілкою до діяльності «Переведення користувача до категорії абонентів» є «Дані про абонента»; вихідною – «Верифіковані дані про абонента»; стрілками контролю є «Внутрішній регламент» та «Закон України “Про захист прав споживачів”».

Вхідними стрілками до діяльності «Реєстрація джерел електроенергії абонента» є «Верифіковані дані про абонента» та «Дані про систему енергозабезпечення»; вихідною стрілкою є «Верифіковані дані про систему енергозабезпечення абонента»; стрілками контролю є «Внутрішній регламент»

та «Закон України “Про захист прав споживачів”»; стрілкою механізмів – «Адміністратор».

Вхідною стрілкою до діяльності «Система управління енергозабезпеченням від альтернативних джерел електроенергії абонента» є «Верифіковані дані про систему енергозабезпечення абонента»; вихідними - «Інформація про стан системи енергозабезпечення» та «Статистичні дані про систему енергозабезпечення»; стрілками контролю є «Внутрішній регламент» та «Закон України “Про захист прав споживачів”».

3.2. Моделювання бази даних веб системи

Реляційна модель даних – це модель даних, де текстова чи числова інформація, що зображується за допомогою таблиць. Кожна таблиця, яка називається відношенням, складається з рядків (кортежей), та стовпчиків (атрибутів). Реляційна модель визначає структуру представлення даних (структура), захист від некоректних змін (цілісність) та операції, що можуть бути виконані з даними (операції з даними). ER-діаграми зручні тим, що процес виділення сутностей, атрибутів і зав’язків є ітераційним. Розробивши перший наближений варіант діаграм, вони уточняються, опитуючи експертів предметної області.

При розробленні інформаційної моделі в термінах ER-моделей ми повинні мати таку інформацію про предметну область:

− список сутностей предметної області;

− список атрибутів сутностей;

− опис взаємозв'язків між сутностями.

У результаті аналізу предметної області, було визначено перелік функцій, що описує модель бази даних веб системи:

- користувач повинен мати змогу зареєструватися у системі;

- запровадивши особисту інформацію, користувач класифікується як абонент;
- усі джерела енергії мають міститися у базі даних;
- за кожним абонентом може бути зареєстровано декілька джерел енергії;
- кожне з джерел енергії матиме свій власний перелік характеристик (залежно від технічних особливостей), але всі джерела повинні мати опцію відключення від мережі, або підключення до неї;
- адміністратор веб системи не повинен відноситися до категорії користувачів або абонентів;
- необхідна додаткова інформація про країну, де абонент буде використовувати систему енергозабезпечення.

Таким чином, можна виділити список сутностей та атрибутів:

- Користувачі (users) - сутність;
- Абоненти (subscribers) - сутність;
- Усі джерела енергії (надалі - системи) (systems) - сутність;
- Системи абонентів (subscribers' systems) - сутність;
- Адміністратори (admins) - сутність;
- Країни (countries) - сутність;
- Стан системи (system state) – атрибут сутності системи абонентів (subscribers' systems).

Очевидно, що користувач, запровадивши особисту інформацію та перейшовши до категорії абонентів, користувач буде відноситися тільки до одного абонента:

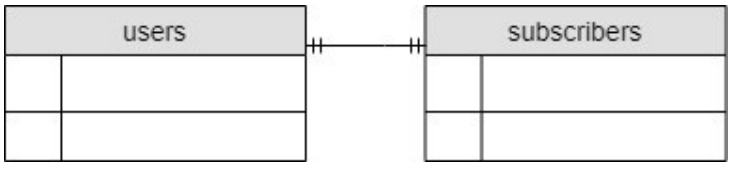

Рисунок 3.4 – Зв'язок між сутностями Users та Subscribers

Крім того, користувач може мати декілька систем. Будь-яка система, в свою чергу, може відноситися до декількох абонентів. Даний вид зв’язку називається «багато до багатьох»:

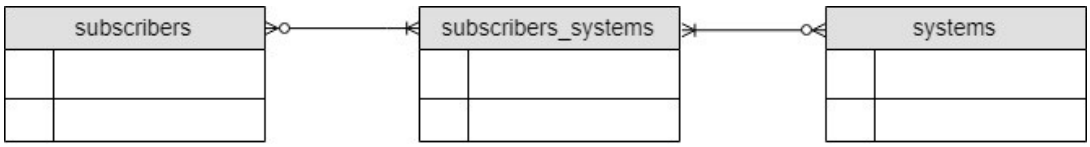

Рисунок 3.5 – Зв’язок між сутностями Subscribers та Systems з використанням сутності Subscribers_Systems

Крім того, кожен абонент проживає у певній країні. І в кожній із країн може проживати багато абонентів:

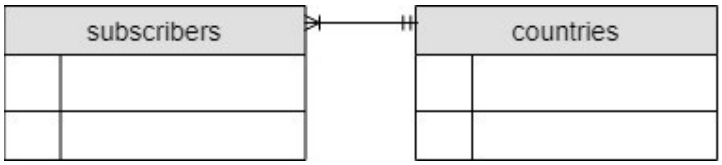

Рисунок 3.6 – Зв’язок між сутностями Subscribers та Countries

Розглянемо властивості сутностей з метою визначення атрибутів:

- **Користувачі:** ім’я та прізвище, електронна адреса (обов’язково унікальна), пароль для авторизації та ідентифікатор, який визначав би, чи є користувач абонентом;
- **Абоненти:** номер телефону, тип користувача (приватний або бізнес), адреса, місто, поштовий індекс та країна;
- **Системи:** тип системи (вітряк чи сонячна панель), код системи та потужність системи – це найменший перелік технічних особливостей системи, що дозволив би ідентифікувати систему;
- **Системи абонентів:** стан системи;
- **Країни:** ім’я країни, її координати та півкуля, у якій знаходиться країна (для підрахунку рекомендованого кута нахилу сонячних панелей);

- **Адміністратори:** ім’я та прізвище, електронна адреса та пароль.

Доповнимо ER-діаграму атрибутами сутностей та необхідними зв’язками. Для ідентифікації кожної сутності буде використовуватися унікальний індекс (primary key), для встановлення взаємозв’язків між сутностями слугуватиме зовнішній ключ (foreign key).

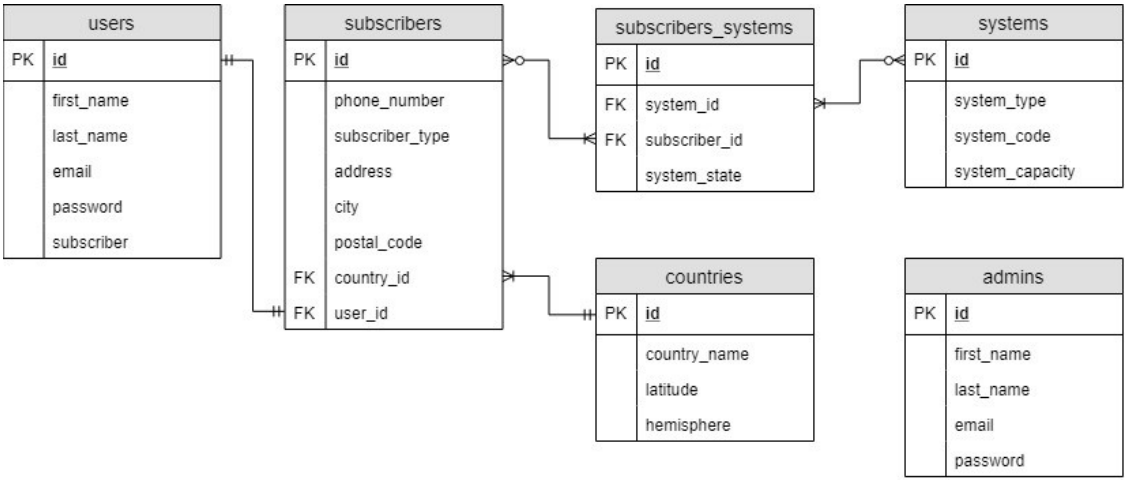

Рисунок 3.7 – Логічна модель бази даних

Всі відношення відповідають вимогам реляційної БД до нормальних форм, що дозволяє перейти до наступного етапу – розроблення фізичної моделі БД.

3.3. Моделювання варіантів використання веб системи

Для кращого розуміння логіки та функціоналу системи, було описано принципи роботи додатку через варіанти використання (Use Case або прецеденти).

Варіанти використання – це представлення послідовностей дій, які система може здійснювати у відповідь на дії користувачів або модулів додатку. Варіанти використання відображають функціональність системи.

Для опису функціонального призначення системи (того, що система повинна робити) використовуються діаграми варіантів використання. Склад моделі варіантів використання включає в себе опис акторів, варіантів використання (ВВ) програмного продукту (ПП) та діаграми варіантів використання (рис. 3.8).

Актори

Користувач – особа, яка зареєструвалась у сервісі, але не запровадила особисті дані;

Абонент – користувач, котрий запровадив свої особисті дані і може мати доступ до управління своєю системою, за наявності останньої.

Адміністратор – співробітник, котрий може переглядати особисту інформацію користувачів, реєструвати системи абонентів.

Варіанти використання

Авторизація (користувача) – слугує для ідентифікації користувачів, абонентів та адміністраторів, розмежування їх прав та рівнів доступу;

Запровадження особистої інформації – необхідна для отримання користувачем статусу «Абонента» процедура;

Перегляд/редагування особистої інформації – дозволяє абонентам та адміністратору переглядати та вносити зміни у особисту інформацію абонента;

Перегляд/управління системою та перегляд статистичних даних – дозволяє абонентам та адміністратору переглядати/вносити зміни у налаштування системи та переглядати статистичні дані;

Реєстрація системи – дозволяє адміністратору зареєструвати систему енергозабезпечення абонента.

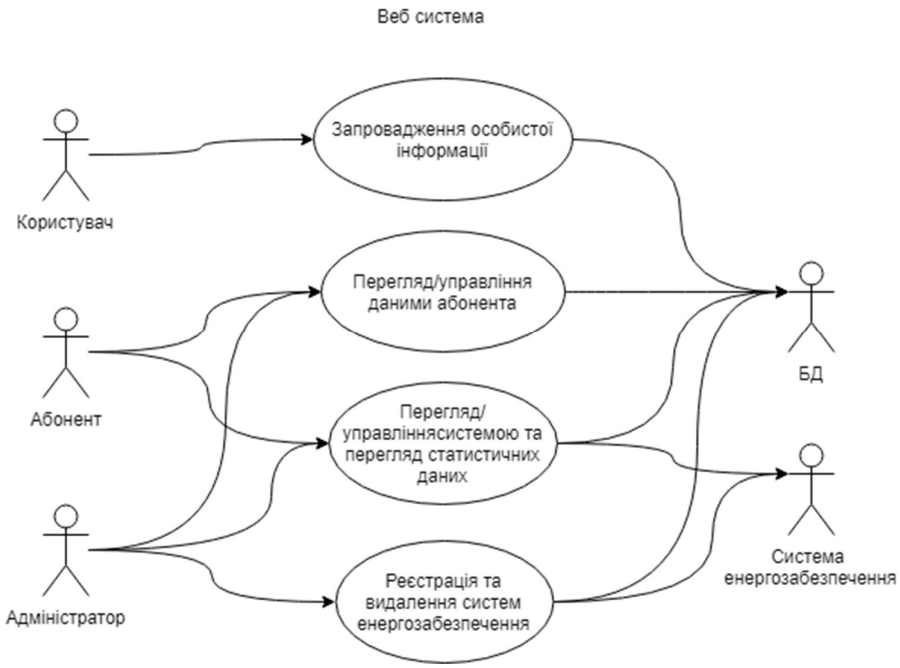

Рисунок 3.8 – Діаграма варіантів використання

4 РОЗРОБКА ВЕБ СИСТЕМИ

4.1. Установка та налаштування середовища

Для розробки веб системи було обрано фреймворк Laravel. Laravel – безкоштовний популярний PHP-фреймворк з підтримкою MVC-архітектури. Він є порівняно простим у використанні та включає в себе дуже багато необхідних модулів (наприклад, спрощена процедура створення системи авторизації, вбудований маршрутизатор веб-запитів та ін.), що значно прискорює процес розробки веб додатку. Для того, аби створити пустий проект на Laravel потрібно спочатку встановити Composer. Composer – це менеджер пакетів прикладного рівня для PHP. З його допомогою можна встановлювати та управляти залежностями проекту (бібліотеками, фреймворками та ін.). Крім того, він реалізує автозавантажувач класів для встановлених бібліотек, що полегшує використання сторонніх бібліотек та пакетів. Composer працює з командного рядка, тому для того, аби встановити пустий проект на Laravel достатньо застосувати команду: composer create-project --prefer-dist laravel/laravel orchestra.

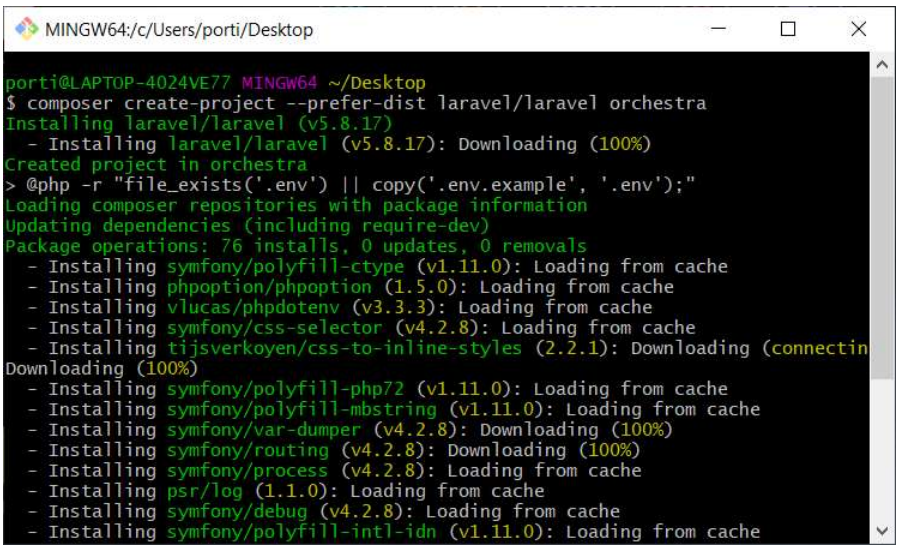

Рисунок 4.1 – Процес встановлення пустого Laravel проекту з використанням Composer

У процесі розробки можуть знадобитися додаткові JavaScript-залежності, тому доцільно використовувати npm (Node Package Manager) у якості менеджера пакетів. Він виконує приблизно схожі функції, що й Composer, але, на відмінну від Composer, який є менеджером пакетів мови PHP, npm відповідає за JavaScript-залежності. Для того, аби створити файл package.json, який буде містити всю необхідну інформацію про проект та залежності, які використовуються в ньому, достатньо перейти до корінного каталогу проекту та застосувати команду: npm init -y.

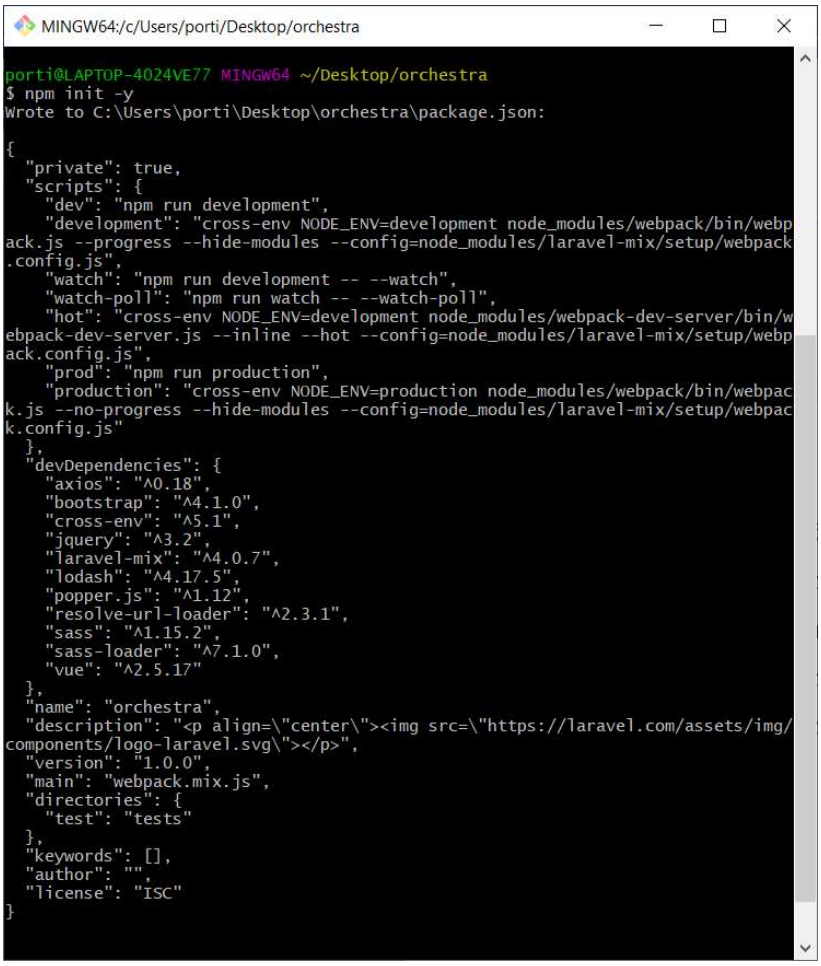

Рисунок 4.2 – Створення файлу package.json для управління JavaScript-залежностями з використанням npm

Хоча Laravel має власний вбудований веб-сервер для розробки на локальному рівні що називається artisan, для створення бази даних все одно буде необхідно встановити локальний сервер (наприклад, WAMP). Для створення бази даних можна використовувати панель управління phpMyAdmin, яка є стандартною для більшості локальних серверів, що використовують Apache. Назва бази даних буде відповідною до назви проекту db_orchestra.

Після того, як базу даних було створено необхідно налаштувати підключення проекту до цієї бази даних. Для цього у кореневому каталозі проекту потрібно створити файл «.env» та скопіювати до нього вміст файлу «.env.example». У новому файлі «.env» потрібно змінити поля «DB_DATABASE», «DB_USERNAME» та «DB_PASSWORD». Після цього, за допомогою командного рядка потрібно виконати наступну команду: php artisan key:generate.

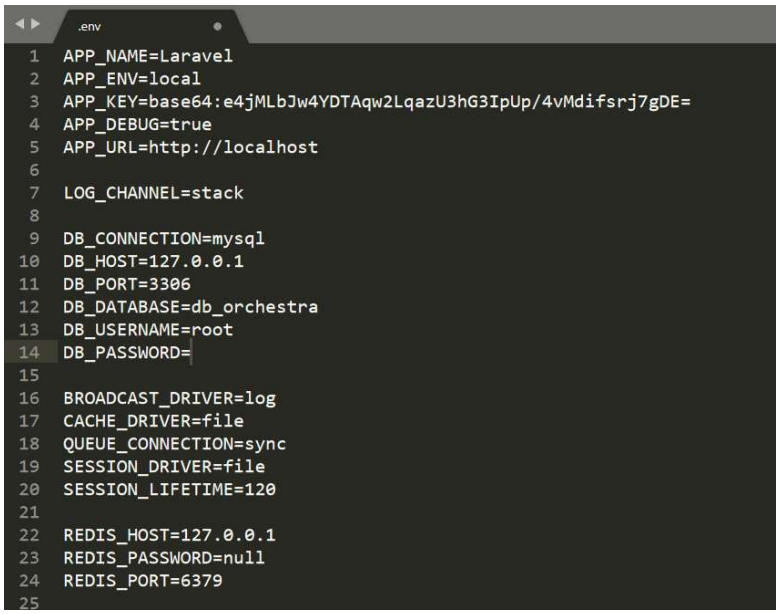

Рисунок 4.3 – Налаштування підключення до бази даних

Після цього, додаток має бути повністю готовий до наступного етапу – розробки функціоналу користувача. Аби перевірити це, достатньо виконати команду: php artisan serve. Якщо усі дії було виконано вірно, командний рядок

має повернути повідомлення наступного характеру: Laravel development server started: <http://127.0.0.1:8000>. Якщо відкрити цю адресу у веб браузері (або просто перейти за адресою http://localhost), додаток виглядатиме наступним чином:

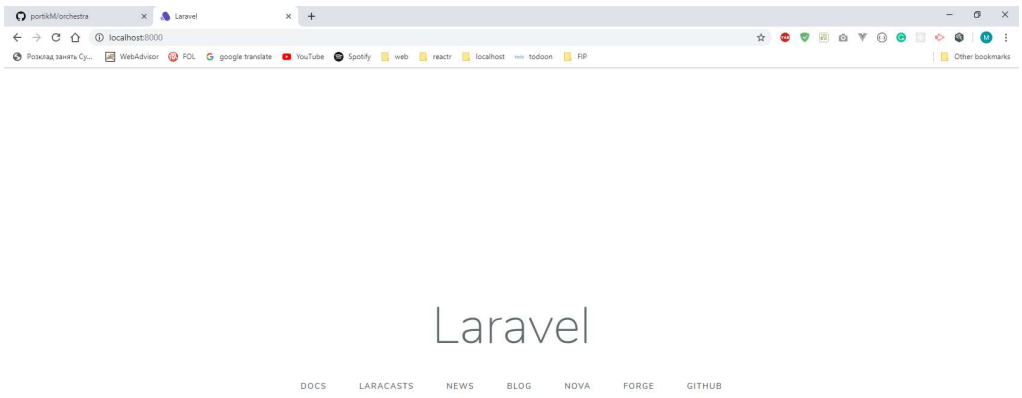

Рисунок 4.4 – Стандартна сторінка з привітанням пустого Laravel-проекту

4.2. Структура проекту

Через те, що Laravel має велику кількість вбудованого функціоналу, навість пустий проект є досить об’ємним. Проект важить більше, ніж 33 з половиною мегабайти. На щастя, 99% цього об’єму завжди залишається на серверній стороні додатку і майже ніяк не впливає на швидкість завантаження сторінок додатку.

Основною задачею будь-якого фреймворку є структуризація та організація файлів вихідного коду, функціоналу користувача та робочого процесу. Метою цієї задачі є: підвищення ефективності роботи програміста, поліпшення швидкодії, спрощення обслуговування додатку та можливість

використання сторонніх залежностей. Тому усі функції інтерфейсу користувача та серверної логіки у проекті з використанням будь-якого фреймворку слід створювати саме у призначених для цього директоріях та, у даному випадку, класах.

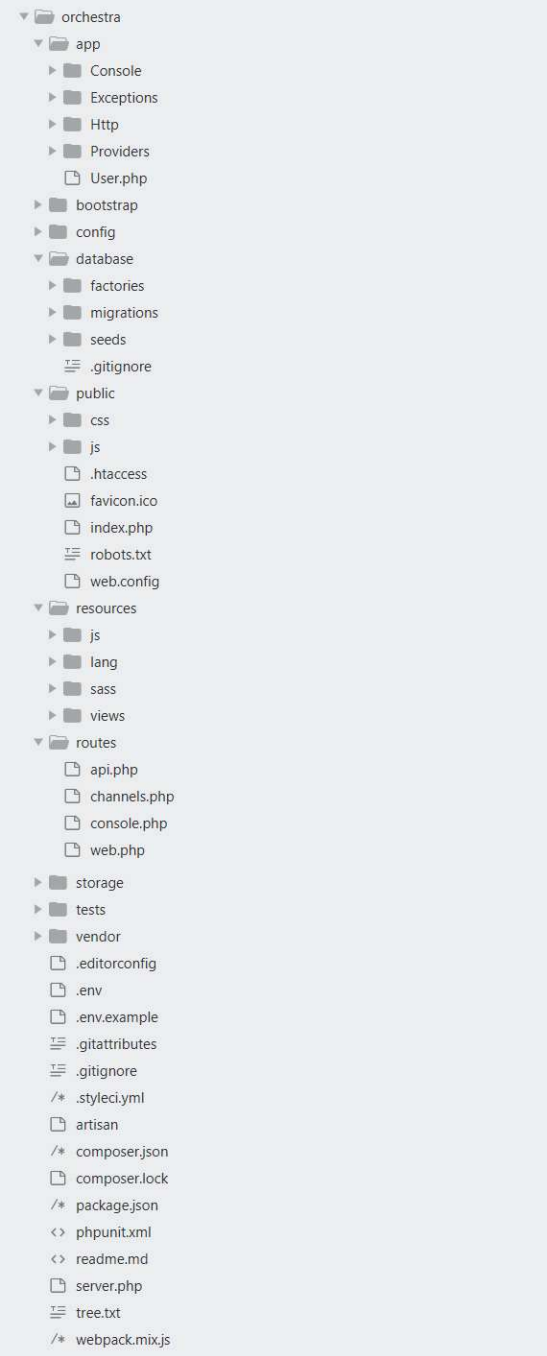

Рисунок 4.5 – Структура пустого проекту

З усіх директорій та файлів проекту, при розробці даного додатку найчастіше будуть використовуватися файли та класи директорій: app, app/Http/Controllers, app/Http/Middleware, database/migrations, database/seeds, public, resources та routes.

Директорія app містить файли класів моделей (model), каталоги Http/Controllers та Http/Middleware. Модель (model) – це клас, атрибути та методи якого відповідають за зв'язок з базою даних. При створенні нової моделі створюється клас, який є продовженням класу Model та є його наслідувачем. При цьому, він може мати свій набір атрибутів та методів. Таким чином, для того, аби отримати дані з конкретної сутності у базі даних достатньо створити модель для цієї сутності та зазначити назву сутності у вигляді атрибуту цього класу. Для встановлення зав'язків цієї сутності з іншими (наприклад, один до багатьох, багато до багатьох і т.д.), необхідно створити метод, який відповідав би за зв'язок з моделлю необхідної сутності.

У директорії app/Http/Controllers містяться файли класів контролерів (controller). Контролер (controller) – це клас, атрибути та методи якого відповідають за серверну логіку додатку. При створенні нової моделі створюється клас, який є продовженням класу Controller та є його наслідувачем. При цьому, він може мати свій набір атрибутів та методів. Наприклад, для того, аби отримати інформацію від певної моделі та відобразити її на певній сторінці додатку, необхідно створити контролер, у методі якого створити змінну, значенням якої буде результат, який повертає певна модель. Далі цю змінну необхідно відправити до певному виду (view) або повернути у вирі JSON-масиву (для API-додатків). Функціонал контролера дуже широкий та дозволяє здійснювати контроль над системою авторизації, обробкою запитів, контентом та базою даних.

Директорія app/Http/Middleware містить файли класів Middleware. Middleware - це потужний інструмент для обробки запитів, який представляє з себе систему класів, атрибути та методи яких відповідають логіку обробки

кожного запиту, що надходить до серверу. Кожний новий клас буде продовжувати клас Middleware і являтися його наслідувачем. Кожен запит, що надходить до серверу спочатку оброблюється усіма класами Middleware, і тільки у випадку, якщо жоден з класів не виявив помилки у запиті, відбувається обробка запиту класами серверної логіки. За допомогою Middleware можна здійснювати контроль над авторизацією користувача (клас Middleware, що відповідає за перевірку права користувача бачити інформацію, яку може бачити лише авторизований користувач, не дозволить обробку запита, що надійшов від гостя), захист від крос-доменних атак (клас Middleware, який перевіряє чи автором запиту є додаток, а не інший ресурс, не дозволить обробку запиту, якщо запит цього типу має бути створений додатком, але надійшов з іншого місця) та інше.

Директорія database містить каталоги migrations та seeds. Це потужні інструменти для роботи з базою даних. Наприклад, для того, аби створити нову сутність достатньо створити міграцію з назвою цієї сутності. Це створить клас, який буде продовженням класу Migration. Атрибути сутності у такому випадку достатньо вказати як атрибути цього класу. Для того, аби заповнити цю сутність інформацією, достатньо аналогічним способом створити Seeder для цієї сутності.

У директорії public містяться усі скомпільовані css- та js-файли, а також зображення.

У каталозі resources містяться файли виду (view), директорії контенту різних мов (для додатків з більш ніж однією мовою інтерфейсу), некомпільовані файли css-та js-препроцесорів.

Каталог routes містить файли, що відповідають за маршрутизацію запитів. Вони слугують зв’язком між маршрутом запиту, функцією контролера, що відповідає за обробку цього запиту та клас (або групу класів) Middleware, що перевіряють цей запит, перш ніж він дійде до функції контролера.

4.3. Результат реалізації веб системи

У результаті було розроблено веб систему з повністю адаптивним інтерфейсом. Домашньою сторінкою є сторінка сайту, що описує можливості та функції системи (рис. 4.6 та 4.7).

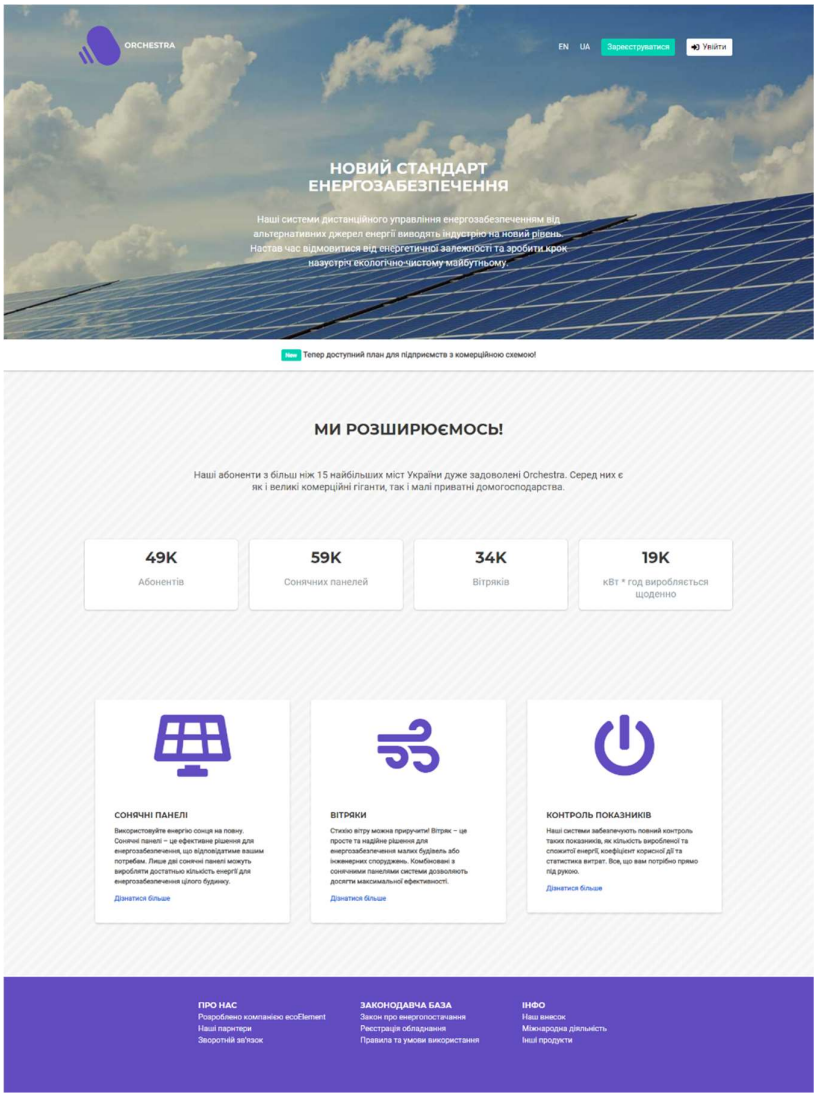

Рисунок 4.6 – Головна сторінка веб системи (вид на екрані комп’ютера)

Рисунок 4.7 - Головна сторінка веб системи (вид на екрані мобільного телефона)

Користувач має змогу обрати між українською та англійською мовами інтерфейсу. Аби зареєструватись у системі достатньо натиснути на кнопку «Зареєструватися».

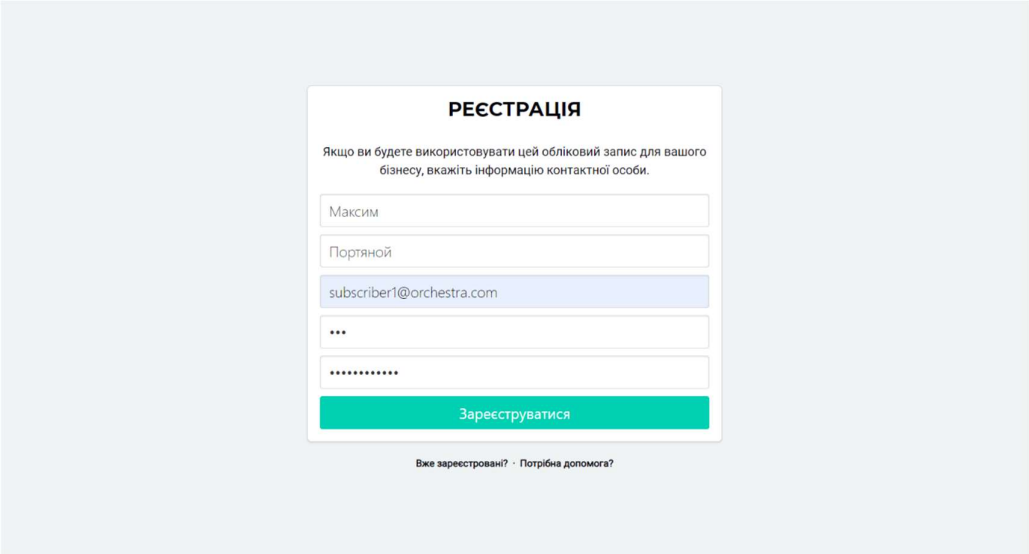

Рисунок 4.8 – Сторінка реєстрації

Система не дозволить створити акаунт, якщо користувач з такою електронною адресою вже існує. Крім того, система перевіряє рівень безпеки пароля та повертає помилку, якщо підтвердження пароля було виконане невірно.

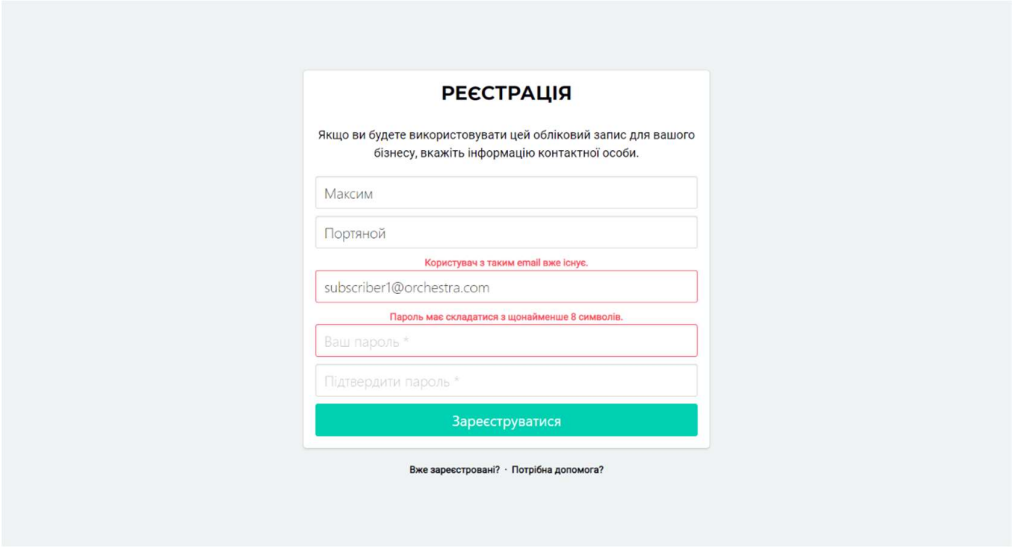

Рисунок 4.9 – Обробка виключень у процесі реєстрації користувача

Після успішної реєстрації користувач отримує доступ до панелі управління. На цій сторінці користувачу бачить інформацію про свій статус у системі (чи є користувач абонентом) та погоду у регіоні, де знаходиться користувач (система визначає ip-адресу користувача та відображає погоду для тієї місцевості). Якщо користувач не є абонентом, то у блоці швидкого доступу до системи він бачитиме повідомлення про те, що йому необхідно запровадити особисту інформацію.

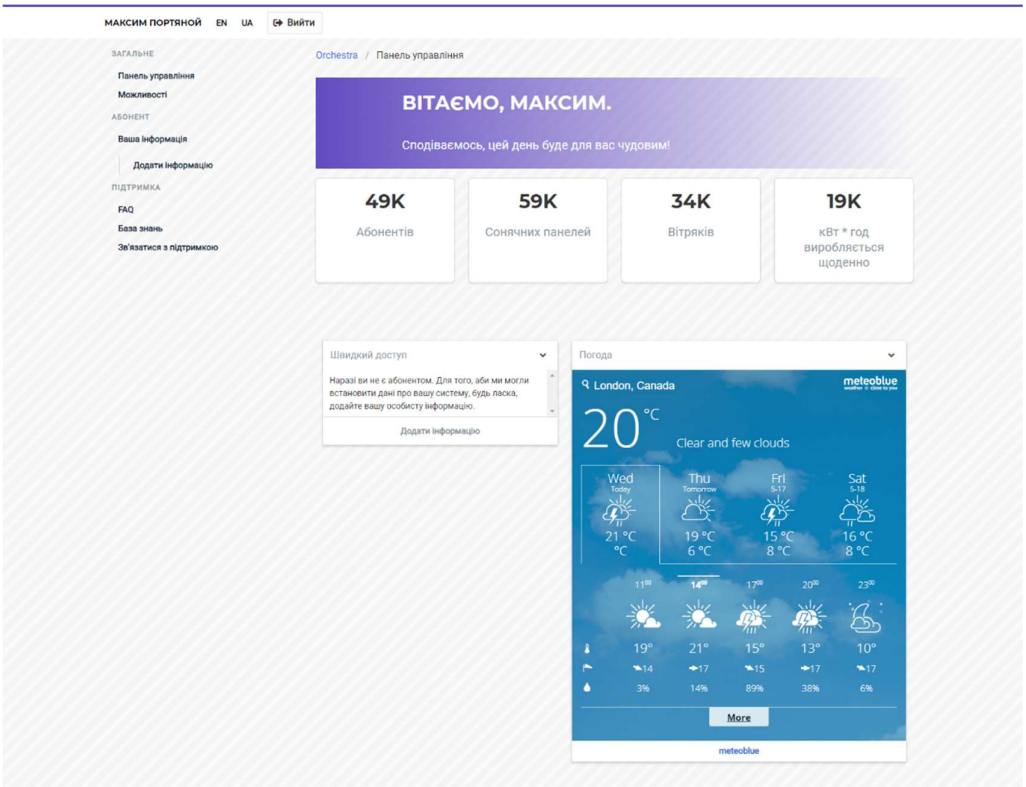

Рисунок 4.10 – Панель управління користувача

Оскільки користувач, який не є абонентом не має доступу до функціоналу управління електромережою та перегляду статистичних даних, єдиною доступною опцією для такого користувача є лише додання особистої інформації.

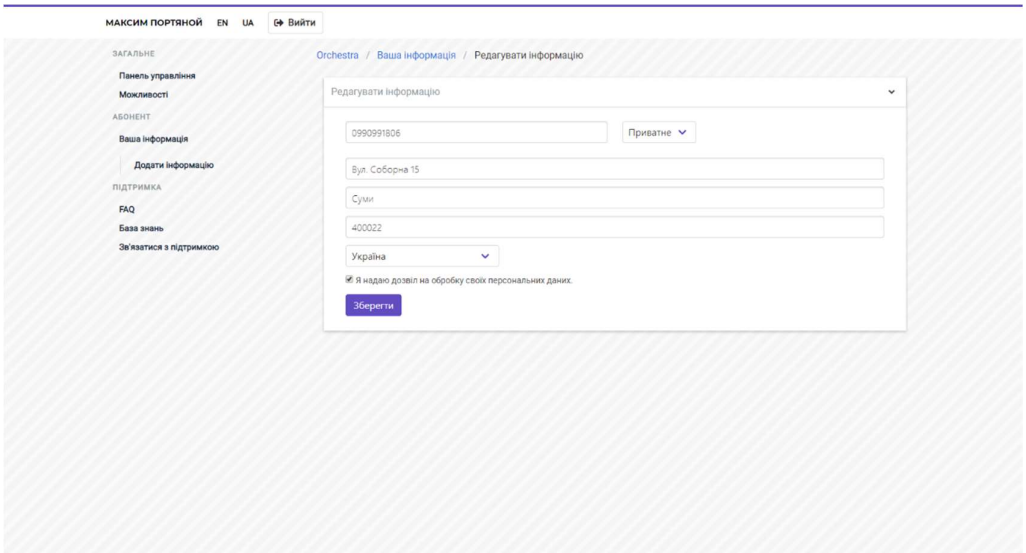

Рисунок 4.11 – Функціонал запровадження особистої інформації користувача

Після того, як користувач запровадить свою особисту інформацію, він потрапляє на сторінку редагування своєї інформації. Оскільки з цього моменту цей користувач є абонентом, він має доступ до додаткового функціоналу (редагування електронної адреси та пароля). Крім того, у меню зліва з'являються додаткові пункти меню, які були недоступними раніше.

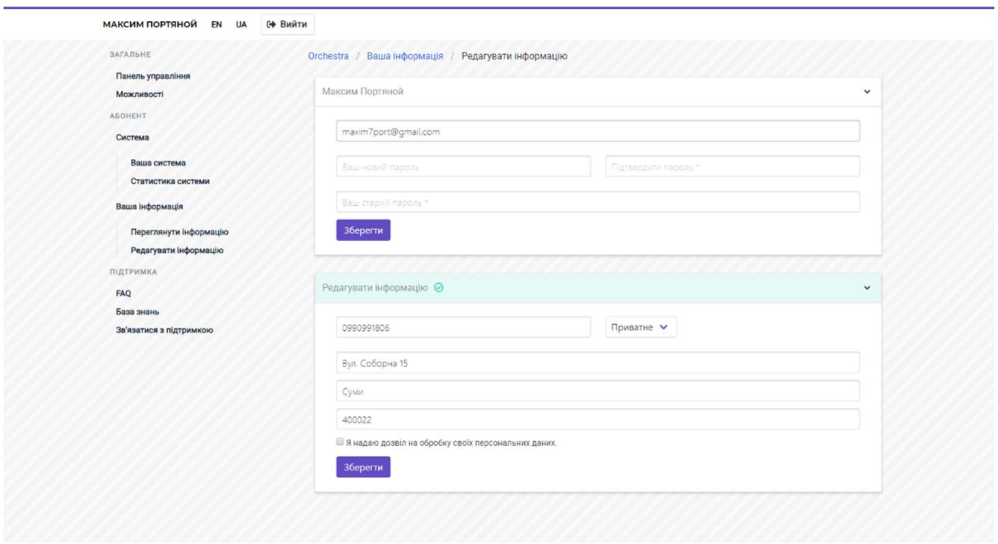

Рисунок 4.12 – Функціонал редагування особистої інформації користувача.

Оскільки абонент є новим, за замовчуванням для нього не зареєстровано жодної системи. Наприклад, хоча пункт меню «Ваша система», який відповідає за відображення системи абонента є доступним для нього, абонент побачить повідомлення, що систем, зареєстрованих за ним не знайдено. Для того, аби зареєструвати систему для цього абонента, потрібно авторизуватися як адміністратор. Система авторизації адміністратора – це окремий та незалежний модуль додатку.

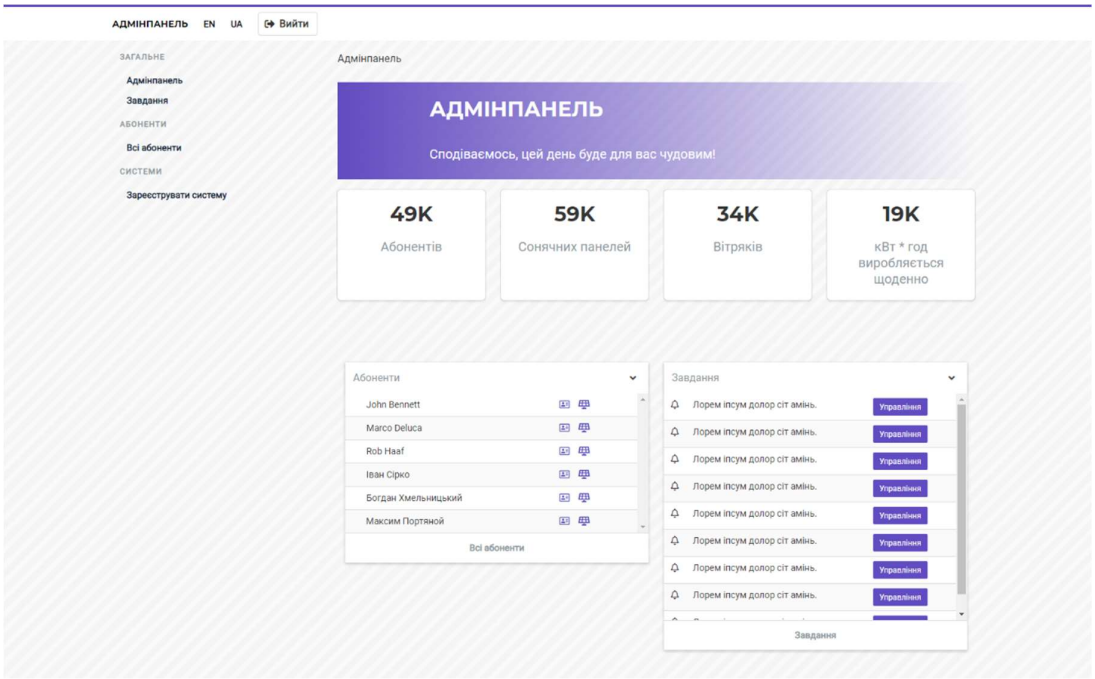

Рисунок 4.13 – Панель управління адміністратора

Панель управління адміністратора схожа на панель управління користувача, але має зовсім іншу систему навігації та функціонал. Пункт меню «Всі абоненти» виводить список всіх абонентів сервісу з можливістю переглянути та редагувати особисту інформацію кожного, побачити систему абонента та статистичні дані. Пункт меню «Зареєструвати систему» відповідає за реєстрацію систем абонентів.

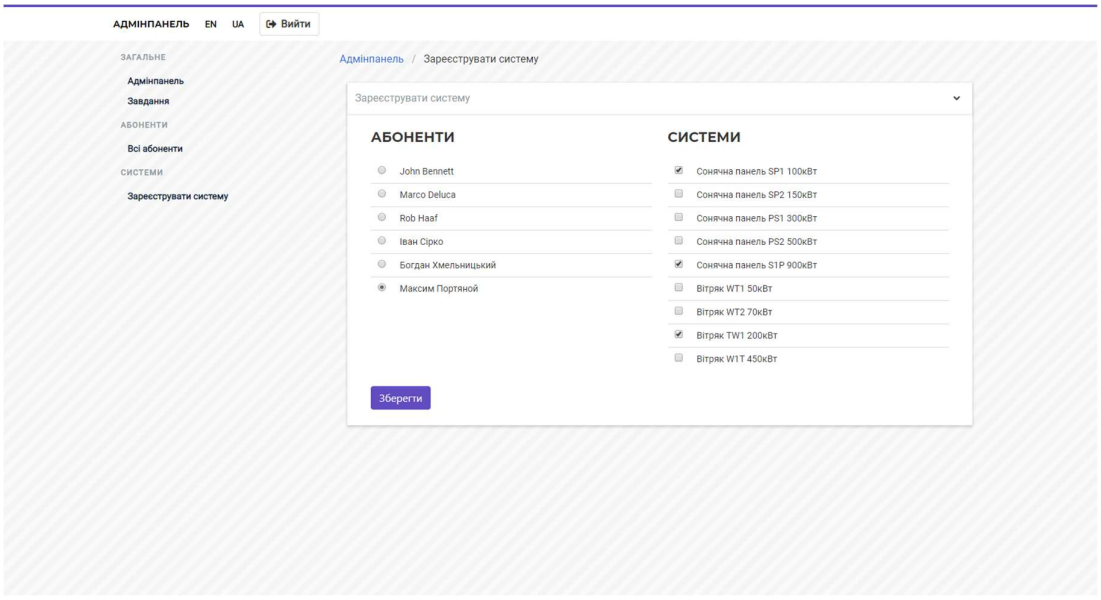

Рисунок 4.14 – Сторінка реєстрації системи абонента

Для того, аби зареєструвати систему достатньо вибрати абонента з списку абонентів (що зліва) та його систему зі списку усіх доступних систем (що справа) та зберегти зміни.

Тепер користувач бачитиме ці системи на своїй панелі управління.

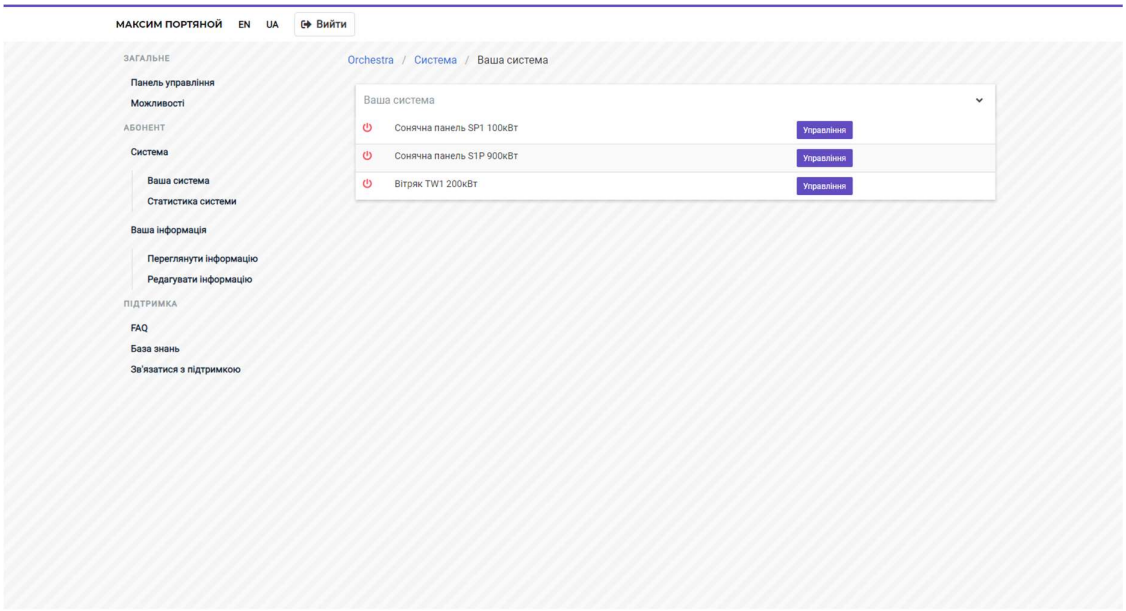

Рисунок 4.15 – Сторінка системи абонента

Пункт меню «Статистика системи», що відповідає за статистичні дані системи абонента, містить дані про регіон абонента. Система підраховує оптимальний кут нахилу сонячних панелей за спеціальною формулою, враховуючи координати країни абонента, сезон та півкулю Земного шару, у якій країна знаходиться. Також на цій сторінці можна побачити статистичні дані системи у вигляді графіків та діаграм.

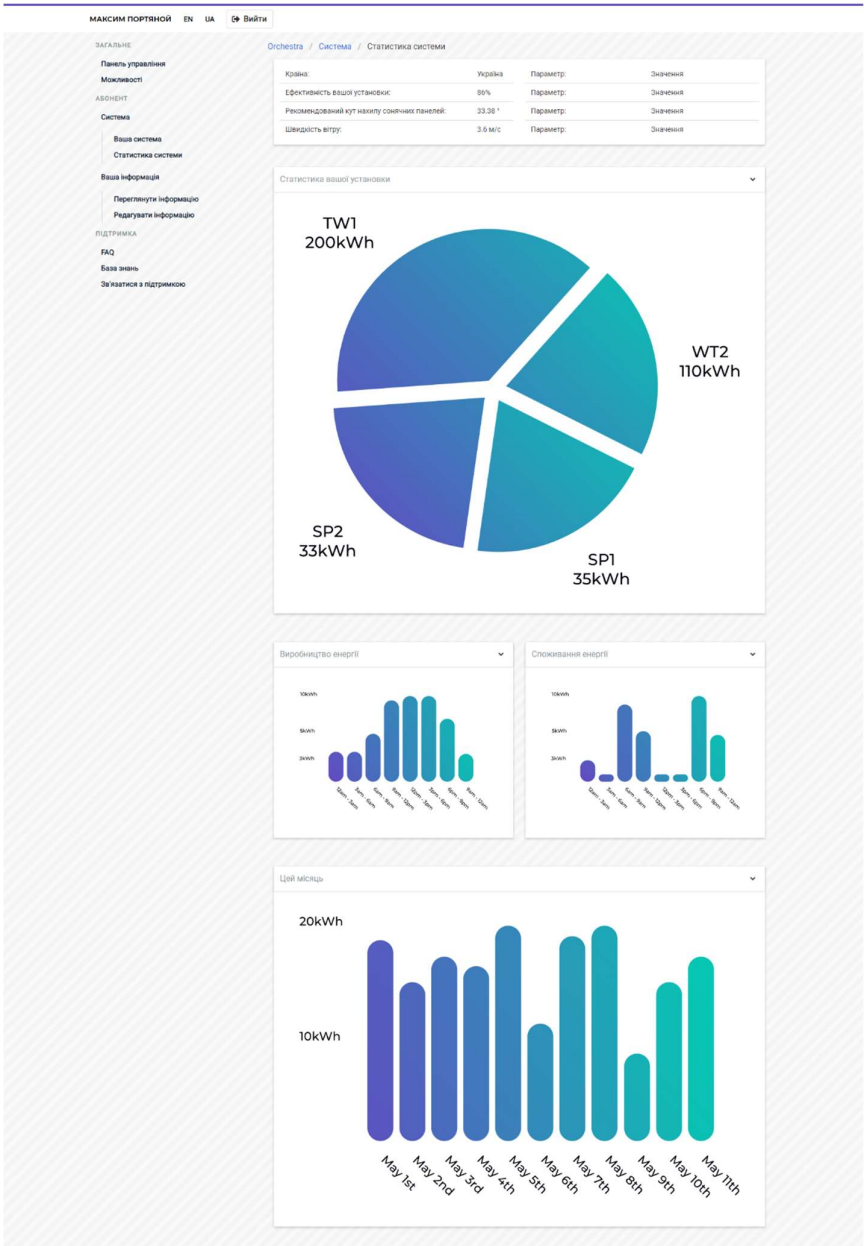

Рисунок 4.16 – Статистичні дані системи абонента

4.4. Система аутентифікації та маршрутизація запитів

За замовчуванням у Laravel існує лише один тип користувача. Відповідно, уся логіка системи аутентифікації та маршрутизації запитів побудована з розрахунку на те, що у системі буде лише один тип користувача. При потребі створити функціонал та інтерфейс для різних типів користувачів, зазвичай, слід спочатку вирішити, чи є різниця у ролях цих користувачів принциповою. Якщо роль першого типу користувача не суттєво відрізняється від ролі другого, то користувачів можна розбити на два (або більше) типи за допомогою додаткового атрибуту у сутності користувача. Такий атрибут зазвичай називається «рівень доступу». Прикладом розділення функціоналу таким чином може бути адміністратор та модератор блогу: обидва використовують спільну систему аутентифікації, але модератор не має змоги бачити частину функціоналу, яка доступна лише адміністраторам. Логіка обробки запитів від користувачів з різним рівнем доступу, зазвичай, є спільною для усіх рівнів, але деякі функції виконують перевірку рівня доступу користувача, перш ніж обробити запит. Якщо ж ролі користувачів відрізняються суттєво, краще створити окрему систему аутентифікації для кожного з користувачів. Наприклад, користувач інтернет магазину може авторизуватись у системі, у той час, як адміністратор може авторизуватися у панелі управління адміністратора, використовуючи для цього окрему систему аутентифікації. Як правило, у такому випадку логіка обробки запитів користувачів є різною та знаходиться у різних частинах додатку. Проте, деякі функції можуть бути спільними. Наприклад, адміністратор інтернет магазину часто має змогу додати або видалити певний товар до корзини певного користувача (на випадок, якщо користувач замовляє товар по телефону) та ін.

Для даної веб системи було обрано 4 типи користувачів: гість, користувач, абонент та адміністратор. Логіка обробки запитів від гостя не є необхідною (неавторизований користувач не матиме доступу до функціоналу жодного з рівнів доступу користувача). Що стосується користувача та абонента,

логіку обробки запитів можна реалізувати, створивши додаткову сутність для кожного з них. Адміністратор ресурсу – це зовсім інший тип користувача, тому для нього було б логічним створити окрему систему авторизації з власною логікою обробки запитів.

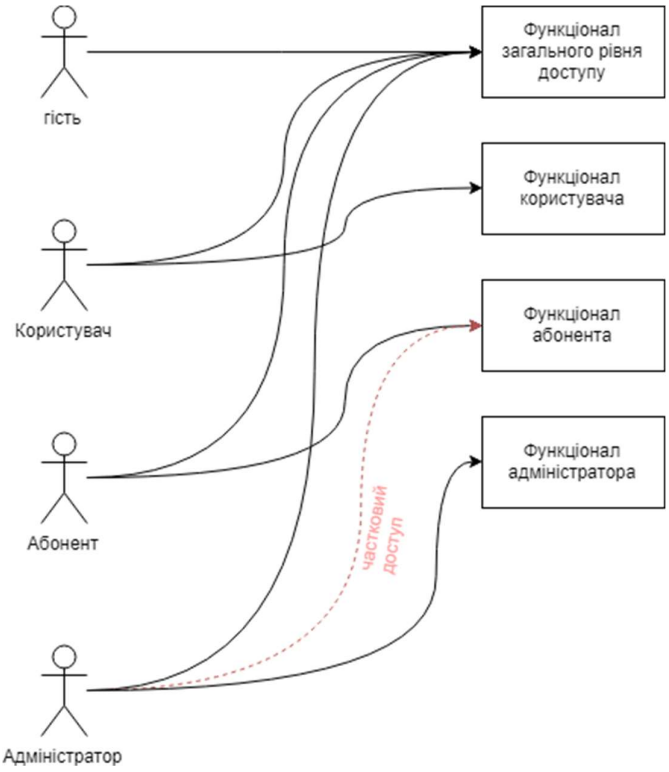

Рисунок 4.17 – Схема розподілення рівнів доступу до функціоналу

Схему розподілення рівнів доступу до функціоналу представлено на рис. 4.17. Функціонал загального рівня доступу – це головна сторінка додатку, сторінки реєстрації та авторизації користувача, а також сторінка відновлення пароля. Під функціоналом користувача мається на увазі сторінка панелі управління користувача та сторінка додання особистої інформації користувача, оскільки це є єдина доступна опція для користувача, що не є абонентом. Доступ до сторінки додання особистої інформації може мати лише користувач, тому що абонент це користувач, який вже запровадив свою особисту інформацію і може редагувати її, але не може її видалити. Адміністратор також не має доступу до цієї сторінки, адже користувач повинен обирати сам, яку інформацію

запровадити. Функціонал абонента є набагато ширшим: він може переглядати та редагувати свою особисту інформацію, змінювати свій логін та пароль, переглядати свою систему енергозабезпечення, змінювати її налаштування та переглядати її статистичні дані. Адміністратор може мати доступ до більшості цього функціоналу для кожного окремого абонента, але не може змінювати логін та пароль абонента. Функціонал адміністратора включає в себе перегляд усіх абонентів системи та реєстрацію системи енергозабезпечення за конкретним абонентом. Маніпуляції з існуючою системою енергозабезпечення та особистою інформацією абонента вважаються як спільний для абонента та адміністратора функціонал.

Маршрутизатор (router) Laravel забезпечує маршрутизацію запитів до функцій необхідного контролера. Крім того, саме маршрутизатор відповідає за уточнення рівня доступу, який має мати користувач, аби запит було оброблено. Побачити перелік усіх маршрутизаторів додатку, рівні доступу та функції контролерів, що відповідають за обробку цих запитів можна, скориставшись командою: php artisan route:list.

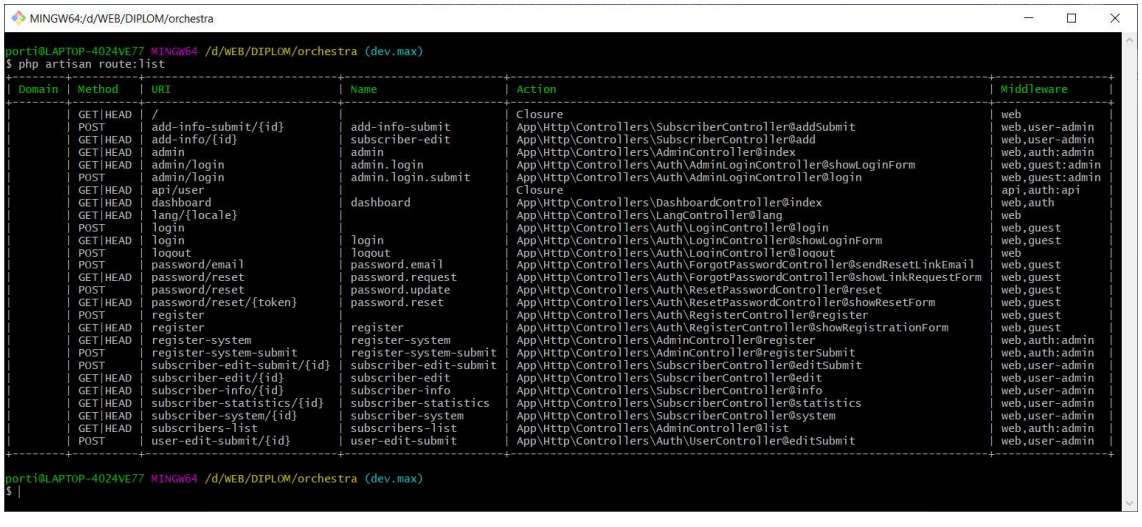

Рисунок 4.18 – Перелік маршрутизаторів додатку

Таблиця, зображена на рисунку 4.18 має 6 стовбців: Domain (домен), Method (метод), URI (адреса), Name (ім’я), Action (дія) та Middleware.

Домен – це унікальна адреса сайту у мережі інтернет. Оскільки розробка ведеться у локальному середовищі, ця колонка є пустою.

Колонка метод відображає метод запиту. У веб додатках найбільш широко використовуються наступні 4 методи запитів: GET, POST, PUT та DELETE. Метод GET використовується для читання інформації з серверу, або для передачі її на сервер. Наприклад, якщо потрібно створити запит, що повертав би інформацію про користувача, якого звуть Іван Сіріко, то слід використати метод GET. Також на різних сайтах часто можна побачити у адресній строчці вираз, схожий на: ?category=”Sport”. У даному випадку майже точно можна сказати, що метод GET використовується для фільтру за категорією. Проте, не є безпечним передавати за допомогою методу GET чуттєву інформацію, адже вона відображається у адресній строчці браузера та може бути викрадена, або змінена. Для цього є метод POST. Він використовується для поміщення в базу даних отриманої інформацію. Метод PUT є схожим на метод POST. Відмінність у тому, що POST може використовуватися як для редагування існуючих записів у БД, так і для створення нових, у той час, як PUT може лише редагувати існуючі і не може додавати нові записи. DELETE використовується для видалення інформації з бази даних. Інколи серед методів запиту також вказується HEAD – це означає, що даний запит включає в себе заголовок запиту, який зазвичай містить інформацію про сесію користувача (cookies-файли, JWT-файли та інше).

У колонці адреси можна побачити унікальний локальний шлях до певної сторінки. У випадку, коли додаток знаходиться на сервері у мережі інтернет та має власний домен, повною адресою сторінки (URL) була б назва домену разом з адресою URI.

Колонка дії відображає інформацію про функцію контролера, який відповідає за обробку запиту у форматі: шлях до контролера/ім’я контролера@назва функції.

У колонці Middleware можна побачити, який рівень доступу має мати користувач, аби мати доступ до певної сторінки або функціоналу. Middleware –

це інструмент для перевірки запитів на предмет виконання певної умови. Наприклад, за допомогою Middleware під назвою auth здійснюється перевірка, чи користувач, який хоче бачити певну сторінку, авторизований у системі. Якщо ця умова не виконується – користувач не може отримати доступ до цієї сторінки. У даному випадку у системі є п’ять видів Middleware: web, guest, auth, auth:admin та user-admin. Middleware під назвою web виконує перевірку, чи запит надійшов від додатку, або з будь-якого іншого місця в інтернеті. Це є одним із рівнів захисту від крос-доменних атак. Ті сторінки, які можуть бачити тільки неавторизовані користувачі використовують guest Middleware. Middleware під назвою auth навпаки, захищає сторінки та функціонал від неавторизованих користувачів. Кастомними Middleware є auth:admin та user-admin. Перше відповідає за функціонал, доступний лише адміністратору, друге – за спільний функціонал для адміністратора та авторизованого користувача.

4.5. Тестування

Обов’язковим етапом у розробці будь-якого продукту є тестування та виправлення помилок. Тестування продуту можна розбити на тестування інтерфейсу та тестування логіки. Тестування інтерфейсу включає в себе перевірку адаптивності усіх сторінок сервісу, тестування функціоналу зміни мови інтерфейсу та оцінку інтуїтивності інтерфейсу. Тестування логіки передбачає перевірку правильності усіх функцій, доступних користувачам, тестування безпеки та обробки виключень.

Інтерфейс користувача є повністю адаптивним та виглядає добре на девайсах з різним розміром екрану. Функціонал зміни мови інтерфейсу працює добре на усіх сторінках системи.

Рисунок 4.19 – Вид головної сторінки системи на мобільному телефоні з англійською мовою інтерфейсу

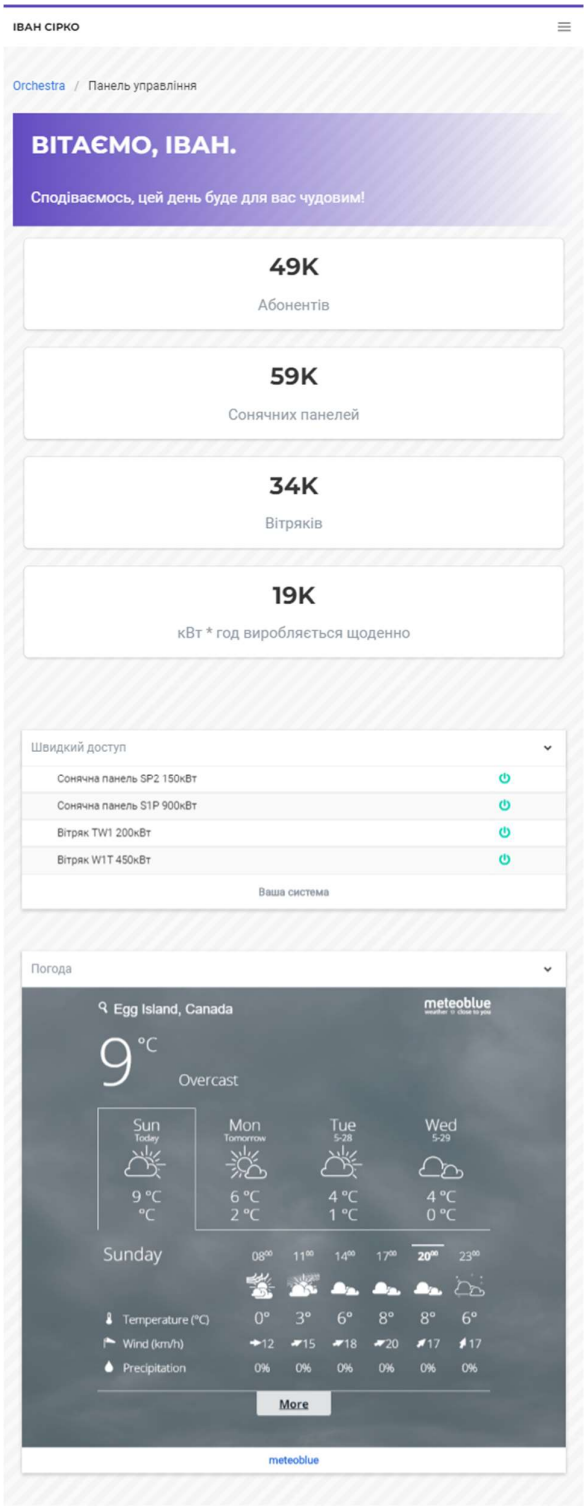

Рисунок 4.20 – Вид панелі управління користувача на планшеті з українською мовою інтерфейсу

Під час тестування логіки додатку було виявлено декілька потенційно вразливих особливостей додатку: функції перегляду і редагування особистої інформації та систем енергозабезпечення абонентів. У попередньому розділі можна побачити, що систему маршрутизації влаштовано таким чином, що посилання на всі сторінки, які містять інформацію абонента (система енергозабезпечення абонента, перегляд та редагування особистої інформації) складається з назви сторінки та цифри, що відповідає номеру абонента у базі даних. Тобто, якщо абонент змінить цифру у адресній строчці, він може побачити інформацію іншого користувача (за умови, що користувач під таким номером існує). Тому було розроблено та протестовано функцію, що не дозволяє би користувачам переглядати інформацію інших користувачів.

Редагування особистої інформації абонента – це ще одна вразлива функція системи. Крім того, що користувач може просто забути ввести значення в одне з полів, він також може допустити помилку. Тому було реалізовано та протестовану систему функцій обробки виключень.

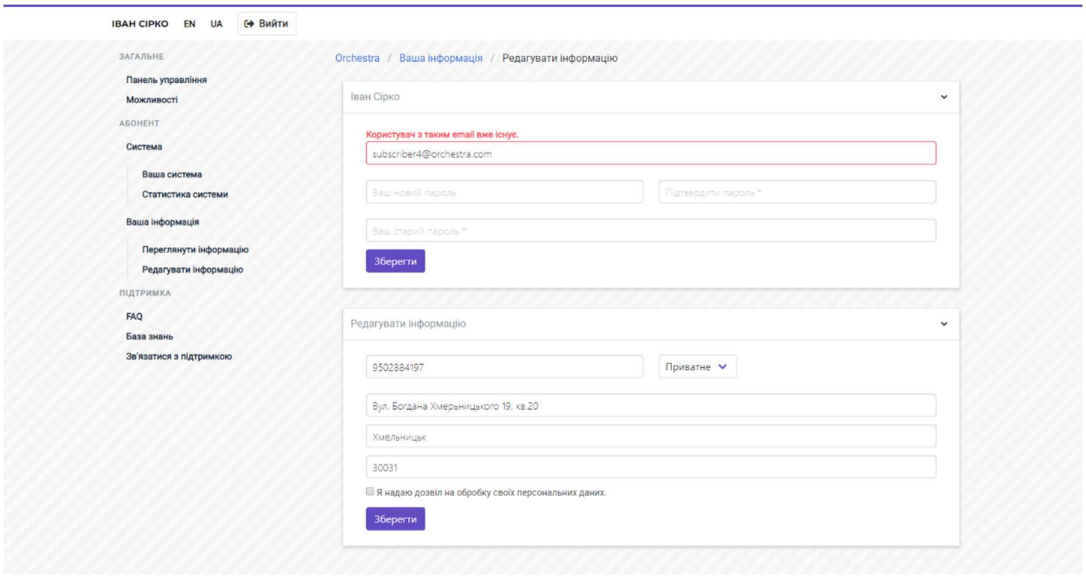

Рисунок 4.21 – Обробка подій виключень при зміні електронної адреси

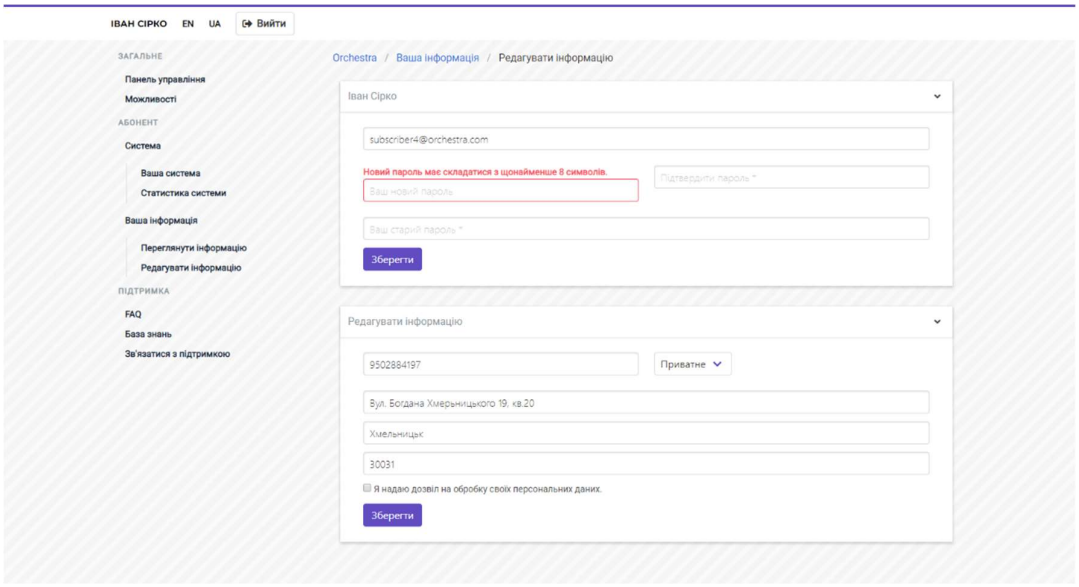

Рисунок 4.22 – Обробка виключень при зміні пароля

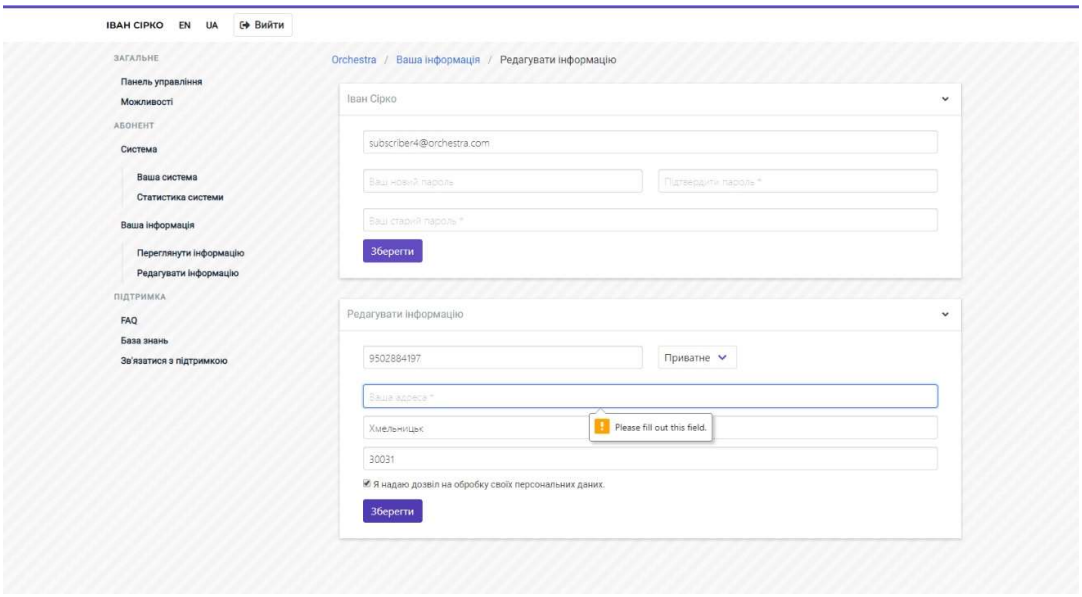

Рисунок 4.23 – Обробка виключень при редагуванні особистої інформації

Ще однією вразливою функцією є сама форма редагування інформації користувача. Аналогічно адреси посилання на сторінку з інформацією користувача працює і форма відправки даних: посилання, за яким відправляються дані складається з назви маршруту та цифри, яка відповідає

номеру користувача . Якщо у будь-якому місці екрану натиснути ПКМ, обрати з контекстного меню варіант «Інспектувати елемент», знайти заголовок форми та змінити цифру у параметрі action, то можна змінити особисту інформацію іншого користувача. Тому, було розроблено на протестовано функцію, що забороняє обробку запиту, якщо він надійшов від користувача, номер якого не співпадає з тим, що було отримано у запиті.

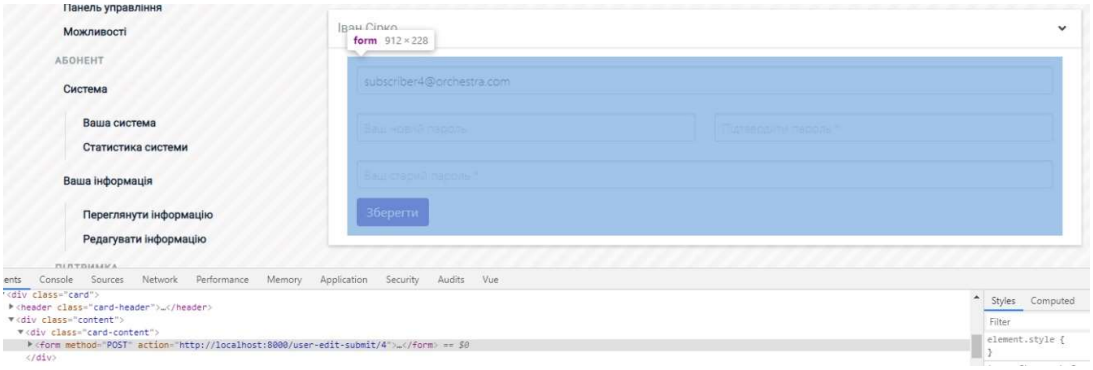

Рисунок 4.24 – Елемент форми редагування даних

ВИСНОВКИ

У ході розробки додатку було проведено детальний аналіз предметної області у результаті якого було визначено вимоги до веб системи та способи її застосування.

В аналітичній частині дипломної роботи було обґрунтовано вимоги до системи та обрано засоби реалізації.

У проектній частині дипломної роботи для веб системи було розроблено модель процесів додатку, логічна модель бази даних та побудовано діаграму способів використання.

На основі розробленої логічної структури і програмного алгоритму, з урахуванням вимог до додатку, перерахованих у ТЗ, було реалізовано фізичну модель реляційної бази даних.

Після цього було розроблено веб систему з використанням найсучасніших технологій, фреймворків та бібліотек, таких, як Laravel та Bulma. Розроблену систему було протестовано, а помилки, виявлені у процесі тестування - усунуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бейкерс Бр. Сонячна енергія і масштабах урбанізації : Корпорація Джон Віллей енд Сонс : Сомерсет, 2012. 48 с URL: https://ebookcentral-proquest-com.ezpxy.fanshawec.ca (дата звернення 16.04.2019).
2. Пуллман Г., Баотонг Ж. Розробка веб додатків у 21-му столітті : Роутледж : Амітвіл, 2016. 135 с. URL: https://ebookcentral-proquest-com.ezpxy.fanshawec.ca (дата звернення 11.04.2019).
3. Відія В., Джейрам Дж., Ішварая К. Системи управління базами даних : Альфа Сайенс Інтерншнал : Нью Делі, 2016. 98 с. URL: https://ebookcentral-proquest-com.ezpxy.fanshawec.ca (дата звернення 16.04.2019).
4. Бін М. Основи Laravel 5 : ТОВ Пакт Паблішінг : Олтон Біргхам, 2015. 212 с. URL: https://ebookcentral-proquest-com.ezpxy.fanshawec.ca (дата звернення 11.04.2019).
5. Рааш Дж. Програмування мовою JavaScript : Корпорація Джон Віллей енд Сонс : Нью Йорк, 2013. 115 с. URL: https://ebookcentral-proquest-com.ezpxy.fanshawec.ca (дата звернення 11.04.2019).
6. Гадорн Дж. Сонячна та нагрівна системи для приватних будинків : Вільгельм Ернст & Сонс : Берлін, 2015. 29 с. URL: https://ebookcentral-proquest-com.ezpxy.fanshawec.ca (дата звернення 15.04.2019).
7. Сіошансі Ф. Розумна сітка: впровадження ефективної та відновлюваної енергії : Елсвайер Саенс Техно : Сан Дієго, 2011. 209 с. URL: https://ebookcentral-proquest-com.ezpxy.fanshawec.ca (дата звернення 15.04.2019).
8. Дінсер І. Оптимізація енергетичних систем : Корпорація Джон Віллей енд Сонс : Нью Йорк, 2017. 94 с. URL: https://ebookcentral-proquest-com.ezpxy.fanshawec.ca (дата звернення 15.04.2019).

9. Стефанов С., Шарма К. Об’єктно-орієнтований JavaScript : ТОВ Пакт Паблішінг : Олтон Біргхам, 2013. 78 с. URL: https://ebookcentral-proquest-com.ezpxy.fanshawec.ca (дата звернення 19.04.2019).
10. Валейд Дж. PHP 5 для початківців : Корпорація Джон Віллей енд Сонс : Хобокен, 2004. 47 с. URL: https://ebookcentral-proquest-com.ezpxy.fanshawec.ca (дата звернення 19.04.2019).
11. Лавін П. Об’єктно-орієнтований PHP: концепції, техніки та код : Ноу-Стерч Пресс : Сан-Франциско, 2006. 69 с. URL: https://ebookcentral-proquest-com.ezpxy.fanshawec.ca (дата звернення 19.04.2019).
12. Конверс Т., Парк Дж., Морган С. Біблія PHP5 та MySQL : Корпорація Джон Віллей енд Сонс : Хобокен, 2004. 111 с. URL: https://ebookcentral-proquest-com.ezpxy.fanshawec.ca (дата звернення 19.04.2019).
13. Валейд Дж., Сухрінг С. PHP, MySQL, JavaScript та HTML5 в одному : Корпорація Джон Віллей енд Сонс : Сомерсет, 2013. 132 с. URL: https://ebookcentral-proquest-com.ezpxy.fanshawec.ca (дата звернення 19.04.2019).
14. Харрінгтон Дж. Застосування реляційних баз даних : Елсвайер Саенс Техно : Сан-Франциско, 2009. 100 с. URL: https://ebookcentral-proquest-com.ezpxy.fanshawec.ca (дата звернення 19.05.2019).
15. Аллен Г. SQL для початківців : Корпорація Джон Віллей енд Сонс : Сомерсет, 2013. 34 с. URL: https://ebookcentral-proquest-com.ezpxy.fanshawec.ca (дата звернення 19.05.2019).

ДОДАТОК А. ТЕХНІЧНЕ ЗАВДАННЯ

ТЕХНІЧНЕ ЗАВДАННЯ
на розробку веб-системи активного управління енергозабезпеченням від альтернативних джерел енергії

Суми 2018

Призначення й мета створення веб-системи

Призначення веб-системи

Веб-система призначена спростити процеси управління енергозабезпеченням від альтернативних джерел енергії для великої кількості власників методом створення єдиної платформи з індивідуальним функціоналом для кожного користувача.

Мета створення інформаційної системи –

Метою проекту є створення веб-системи що дозволяла би здійснювати контроль, моніторинг та управління енергозабезпеченням від альтернативних джерел енергії широкому колу користувачів.

Цільова аудиторія

У цільовій аудиторії веб-системи можна виділити наступні групи:

- компанія (стартап), що є ініціатором проекту;
- компанії, що займаються продажем, установкою, налаштуванням та ремонтом сонячних панелей та вітряків;
- дійсні користувачі альтернативних джерел енергії;
- відсоток населення, зацікавлений у переході на альтернативні джерела енергії;
- інші зацікавлені сторони.

Вимоги до веб-системи

Вимоги до веб-системи в цілому

Вимоги до структури й функціонування інформаційної системи

Веб-система повинна бути реалізована у вигляді веб-додатку, доступного в мережі інтернет. Сайт повинен складатися із взаємозалежних розділів із чітко розділеними функціями.

Вимоги до збереженні інформації

Адміністратор веб-системи повинен мати можливість створення резервних копій бази даних. Бажано, щоб процедура проводилась автоматично після завершення кожного робочого дня. Обираючи веб-хостинг рекомендовано звернути увагу на сервіси з найменшим відсотком часу поза доступом (downtime).

Вимоги до розмежування доступу

Доступ до персональних даних абонентів має бути обмежений і не може бути переданий третім особам. Користувачів веб-системи можна поділити на 3 групи відповідно до прав доступу:

1. Користувач;
2. Абонент;
3. Адміністратор.

Користувач має право:

- переглядати та змінювати свою власну інформацію (не особисту, наприклад електронна адреса, пароль та ім’я);
- додати особисту інформацію про себе.

Абонент має право:

- здійснювати ті самі дії, що й користувач, але за замовчуванням користувач відноситься до категорії абонентів тільки після того, як запровадить особисту інформацію – тобто абонент не може додати її повторно, але може редагувати її;

- переглядати зареєстровані сонячні панелі і вітряки та вносити зміни в їх налаштування (за наявності зареєстрованих джерел).

Адміністратор має право:

- переглядати особисту інформацію зареєстрованих абонентів;
- реєструвати джерела енергії та закріплювати їх за користувачами;
- переглядати зареєстровані сонячні панелі і вітряки для кожного абонента;
- вносити зміни в їх налаштування (за наявності).

Доступ до особистого кабінету користувача будь-якого рівня доступу частини повинен здійснюватися з використанням унікального логіна й пароля.

Навігація

Користувацький інтерфейс веб-системи повинен забезпечувати повне та інтуїтивно зрозуміле представлення елементів функціоналу. Навігаційні елементи повинні забезпечувати однозначне розуміння користувачем їх змісту: посилання на сторінки повинні бути мати заголовок, умовні позначки відповідати загальноприйнятим стандартам. Графічні елементи навігації повинні бути мати альтернативний підпис.

Веб-система повинна забезпечувати навігацію по всіх доступних користувачеві розділах та відображати відповідну інформацію. Для навігації повинна використовуватися система контент-меню. Меню повинне являти собою текстовий блок (список гіперпосилань) у лівій колонці або у верхній частині сторінки (залежно від затвердженого дизайну).

Функціональні можливості розділів

Сторінка авторизації дозволяє увійти до веб-системи користувачеві з його правами доступу. Для успішної авторизації треба ввести логін та пароль, обрані при реєстрації.

Після успішної авторизації, в залежності від рівня доступу користувача, він перенаправляється в особистий кабінет користувача-абонента, або адміністратора.

Функціонал особистого кабінету користувача та абонента має бути однаковий. Але, враховуючи те, що користувач не має змоги бачити секцію перегляду та управління енергозабезпеченням, елементи інтерфейсу, які відповідають за вищеперераховані секції мають бути неактивними.

Особистий кабінет адміністратора повинен мати опції перегляду всіх користувачів а також зареєстрованих за ними джерел енергії та їх налаштувань, а також реєстрації джерела енергії.

Загальні вимоги

Система не повинна бути перенавантаженою надлишковою інформацією. Текст повинен бути зручним для читання. Кольори та елементи на сайті не повинні відволікати увагу.

Вимоги до видів забезпечення

Вимоги до інформаційного забезпечення

Інформаційне забезпечення - це сукупність єдиної системи класифікації і кодування інформації, уніфікованих систем комунікації, схем інформаційних потоків, що циркулюють в організації та методологія побудови баз даних. Його призначення - це своєчасне формування і видача достовірної інформації для прийняття управлінських рішень.

Для реалізації веб-системи буде використана реляційна СУБД. Тому повинна бути розроблена логічна структура реляційної бази даних, на основі якої буде здійснюватися рішення задачі.

Вимоги до лінгвістичного забезпечення

Сайт повинен бути виконаний українською або російською мовою.

Вимоги до програмного забезпечення

ІС складається з серверної та клієнтської частин. Для їх правильного функціонування програмне забезпечення (ПЗ) повинно швидко та ефективно обробляти інформаційні потоки системи.

Реалізація серверної частини відбувається з використанням:

- Apache 2.4.33;

- MySQL 5.7.21;
- PHP 7.0+;
- Laravel 5.3.

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

– Операційна система Windows 7 або вища;

– Веб браузер Chrome 73+, Firefox 66+, Edge 18+, Safari 12.1+, Android Browser 67+;

– Включена підтримка JavaScript і Cookies.

Вимоги до апаратного забезпечення

Апаратне забезпечення серверної частини повинне задовольняти наступним вимогам:

- 500 МБ вільної RAM;
- 3 ГБ вільного місця на HDD;
- Підтримка серверів Apache, MySQL.

ДОДАТОК Б. ПЛАНУВАННЯ РОБІТ

Планування змісту структури робіт IT-проекту (WBS). Структура декомпозиції робіт (WBS) у проектному менеджменті є орієнтованою на доконане виконання проекту декомпозицією проекту на менші частки. Структура декомпозиції робіт є ключовою часткою робіт по проекту, яка організовує командну роботу по проекту у керовані частини.

WBS є ієрархічною декомпозицією проекту у фази, кінцеві результати та пакети робіт. Вона є ієрархічною структурою, що показує подальший розподіл необхідних для виконання мети зусиль; наприклад, програма, проект чи договір. У проекті чи договорі, розробка WBS відбувається, починаючи з кінцевих цілей та успішного розподілу її у керовані частини, що можуть бути оцінені за критеріями розміру, тривалості та відповідальностей (наприклад, системи, підсистеми, компоненти, задачі, підзадачі та пакети робіт) та включають усі необхідні для досягнення мети проекту кроки.

Система декомпозиції робіт надає загальний каркас для природнього розвитку загального планування та контролю договору і є базисом для розподілу роботи на частини, що можуть бути визначеними, та з яких може бути зроблене Технічне Завдання і установлені звіти по технічним даним, графікам, вартостям, робочим годинам .

Структура декомпозиції робіт дозволяє зібрати докупи підлеглі витрати по задачах, матеріалах тощо на вищий рівень "батьківських" задач, матеріалів тощо. Для кожного елементу структури декомпозиції робіт генерується опис задачі, що має бути виконаною. Ця техніка (іноді називається структурою декомпозиції системи використовується для визначення і налагодження сумарних рамок проекту.

WBS організовується навколо ключових продуктів проекту (чи запланованих результатів), а не необхідних робіт для випуску продукту

(заплановані дії). Так як заплановані результати є бажаним завершенням проекту, вони формують відносно стабільний набір категорій, у яких ціни запланованих для їх досягнення необхідних дій можуть бути зібрані докупи. Добре розроблена WBS робить легко досяжним призначення кожної діяльності проекту до виключно однієї термінальної події у WBS. Додатково до її функцій у обліку витрат WBS також допомагає співвіднести вимоги одного рівня системних специфікацій до іншого, наприклад, відповідність матриці вимог перехресних посилань до функціональних вимог на вищий чи нижчий рівні документації.

Розробка WBS зазвичай має відбуватися на початку проекту і перед детальним плануванням проекту і задач. WBS є попереднім етапом, основою для розробки мережевих і календарних планів, які потребують повного переліку всіх робіт за проектом, які можна отримати, маючи пакети робіт. WBS наочно демонструє весь обсяг робіт і місце окремих виконавців.

Основні етапи розробки WBS:

- визначення ступеня деталізації проектних робіт (так, щоб вони піддавалися оцінці);
- визначення кількості рівнів (як правило, три-чотири, для сучасних компаній чотири – оптимально);
- розробка структури кожного рівня (формуються горизонтальні рівні);
- підготовка опису елементів WBS (коротка назва кожної складової WBS);
- формування системи кодування (кодуються всі блоки);
- проведення зворотних обчислень (витрати знизу догори за принципом: відділ локалізації – субпідрядник).

WBS може застосовуватися для об'єднання робіт, які необхідно виконати, організаційних структур і відповідальності за роботу з підсистемами планування, оцінки, розподілу витрат і ресурсів, аналізу, контролю та обліку в єдину

взаємопов'язану інтегровану систему управління проектом. Будуємо таблицю WBS (рис. Б.1)

Організаційна структура проекту (OBS). Ця структура стосується тільки внутрішньої організаційної структури проекту і не зачіпає відносин проектних груп або учасників з батьківськими організаціями. Будується OBS аналогічно робочої структури, а саме:

- на першому рівні відображається організаційна структура як єдиний елемент;
- на другому і нижчих рівнях ділиться нижча частина структури на основні організаційні елементи.

Цей процес повторюється до найбільш низького рівня - базових робочих груп (змішаних цільових або функціональних), а при реалізації малих проектів - до окремих виконавців.

Обсяг робіт для цих найнижчих організаційних рівнів являє собою найбільш низькі елементи WBS, кожен з яких можна планувати і контролювати як окремі одиниці. Саме таке правило діє для створення OBS. Кількість рівнів залежить від розміру проекту.

Об'єднання робочої та організаційної структури дає можливість інтегрувати, планувати і контролювати роботу та порівнювати її виконання по підрозділах і організації взагалі. Кожен менеджер в цій ієрархії має свій набір планів і звітів за своїми сферами відповідальності. Як вже підкреслювалося, розподіл WBS здійснюється до робочого пакету, який виконується окремою групою. OBS, в свою чергу, розбивається до рівня груп, які виконують найнижчий рівень робіт в WBS. Таким чином, роботи найнижчого рівня WBS притаманні як WBS, так і OBS, тобто це - фундаментальні блоки обох структур. Будуємо таблицю OBS (рис. Б.2).

Побудова календарного графіка виконання ІТ-проекту. Для того, щоб мати реальне уявлення про тривалість виконання робіт з урахуванням обмеженості у використанні ресурсів, на підставі часткових мережевих моделей,

а також, проекту в цілому з урахуванням вихідних і святкових днів, будують календарний графік робіт.

Він є реальним розподілом робіт по пакету по календарними датами, тобто своєрідним розкладом виконання робіт. Діаграма Ганта є досить зручним для користування. Діаграма Ганта представляє собою відрізки, які розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, як складова плану, розміщуються по вертикалі. Початок, кінець і довжина відрізків на шкалі часу відповідають початку, кінця і тривалості завдання.

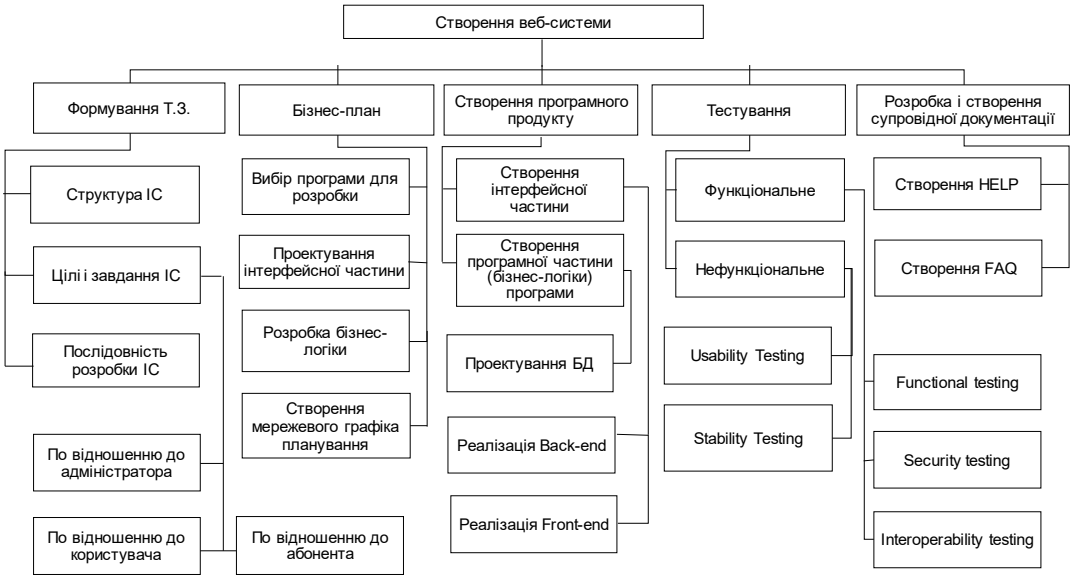

Рисунок Б.1 – Таблиця WBS (work breakdown structure)

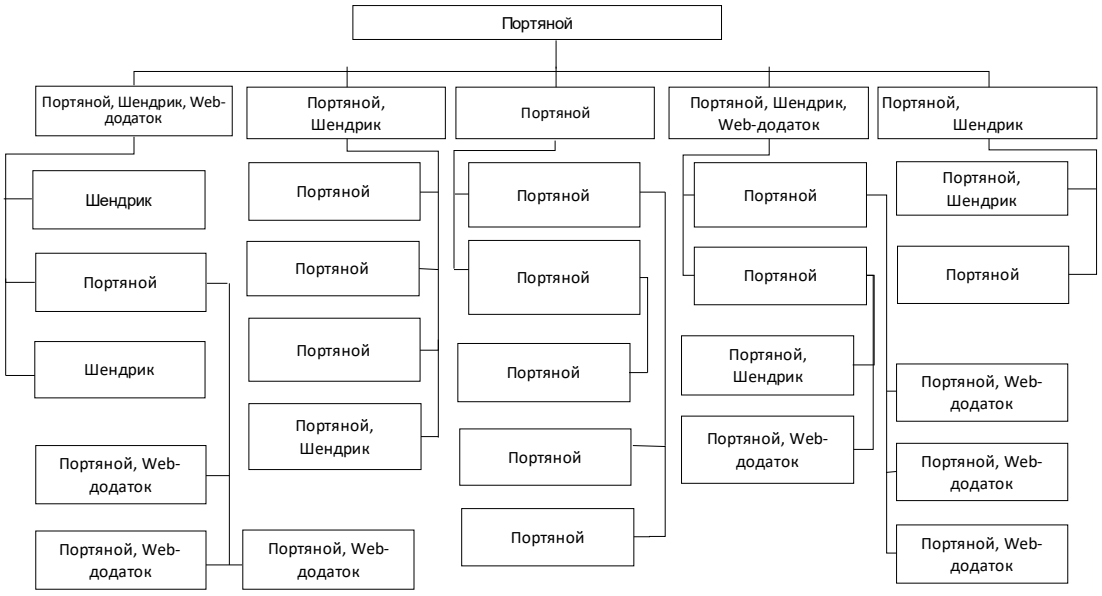

Рисунок Б.2 – Таблиця OBS (Organization Breakdown Structure)

Управління ризиками. До основних ризиків розробки Web-додатку для інформаційної системи підтримки діяльності сервісного центру «Сігма-сервіс» є:

- неглибоке вивчення предметної області;
- збільшення навантаження під час реалізації проекту;
- зміна цілей у ході реалізації проекту;
- людський фактор;
- відмова обладнання;
- відсутність кваліфікованого програміста;
- зміна строків виконання роботи;
- незнаходження спільної мови між керівником і виконавцем;
- зростання вимог до проекту.

№	Ризики	Виникнення	Втрати
1	Відсутність досвіду	3	2
2	Збільшення навантаження під час реалізації проекту	3	2
3	Зміна цілей у ході реалізації проекту	4	4
4	Людський фактор	2	3
5	Відмова обладнання	2	4
6	Відсутність кваліфікованого програміста	2	5
7	Зміна строків виконання роботи	2	4
8	Складнощі при впровадженні на місці	3	3
9	Зростання вимог до проекту	3	3

Таблиця Б.3 - Ймовірність виникнення і величина ризику

Ймовірність						
-	**5**	5	10	15	20	25
Зміна цілей у ході реалізації проекту	**4**	4	8	12	16	20
Збільшення навантаження під час реалізації проекту Зростання вимог до проекту	**3**	3	6	9	12	15
Відсутність досвіду Людський фактор Відмова обладнання Відсутність кваліфікованого програміста Зміна строків виконання роботи Складнощі при впровадженні на місці	**2**	2	4	6	8	10
-	**1**	1	2	3	4	5
		1	**2**	**3**	**4**	**5**
		-	Збільшення навантаження під час реалізації проекту	Людський фактор Складнощі при впровадженні на місці Зростання вимог до проекту	Відсутність досвіду Зміна цілей у ході реалізації проекту Відмова обладнання Зміна строків виконання роботи	Відсутність кваліфікованого програміста
				Втрати		

Таблиця Б.4 - Матриця «Ймовірність – Втрати»

Аналізуючи ризики за ймовірністю їх виникнення, можемо їх розділити на:

- Ігноровані
    - Відсутні
- Незначні
    - Збільшення навантаження під час реалізації проекту
    - Людський фактор
    - Відмова обладнання
    - Зміна строків виконання роботи
- Помірні

  - Відсутність кваліфікованого програміста
  - Зростання вимог до проекту
  - Відсутність досвіду
  - Складнощі при впровадженні на місці
- Істотні
  - Зміна цілей у ході реалізації проекту
- Критичні
  - Відсутні

Класифікація ризиків за рівнем впливу:

- Прийнятні
  - Відсутні
- Виправдані
  - Неглибоке вивчення предметної області
  - Збільшення навантаження під час реалізації проекту
  - Людський фактор
  - Відмова обладнання
  - Відсутність кваліфікованого програміста
  - Зміна строків виконання роботи
  - Не знаходження спільної мови між керівником і виконавцем
  - Зростання вимог до проекту
- Неприпустимі
  - Зміна цілей у ході реалізації проекту.

ДОДАТОК В. ВИХІДНИЙ КОД ДОДАТКУ

Код файлу web.php, що відповідає за визначення усіх маршрутизаторів додатку:

```
<?php

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('lang/{locale}', 'LangController@lang');

Route::get('/', function () {
    return view('landing');
});

Auth::routes();

// USER ROUTES

Route::get('/dashboard', 'DashboardController@index')->name('dashboard');
```

```
Route::middleware('user-admin')->group(function () {
   Route::get('/subscriber-system/{id}', 'SubscriberController@system')-
>name('subscriber-system');
   Route::get('/subscriber-statistics/{id}', 'SubscriberController@statistics')-
>name('subscriber-statistics');
   Route::get('/subscriber-edit/{id}', 'SubscriberController@edit')->name('subscriber-
edit');
   Route::get('/add-info/{id}', 'SubscriberController@add')->name('subscriber-edit');
   Route::post('/subscriber-edit-submit/{id}', 'SubscriberController@editSubmit')-
>name('subscriber-edit-submit');
   Route::post('/user-edit-submit/{id}', 'Auth\UserController@editSubmit')-
>name('user-edit-submit');
   Route::post('/add-info-submit/{id}', 'SubscriberController@addSubmit')-
>name('add-info-submit');
   Route::get('/subscriber-info/{id}', 'SubscriberController@info')->name('subscriber-
info');
});

// ADMIN ROUTES

Route::prefix('admin')->group(function() {
   Route::get('/login', 'Auth\AdminLoginController@showLoginForm')-
>name('admin.login');
   Route::post('/login', 'Auth\AdminLoginController@login')-
>name('admin.login.submit');
   Route::get('/', 'AdminController@index')->name('admin');
});

Route::get('/subscribers-list', 'AdminController@list')->name('subscribers-list');
```

```
Route::get('/register-system', 'AdminController@register')->name('register-system');
Route::post('/register-system-submit', 'AdminController@registerSubmit')-
>name('register-system-submit');
```

Код файлу 2019_05_05_030718_create_subscribers_systems_table.php, що відповідає за створення сутності систем енергозабезпечення абонентів:

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateSubscribersSystemsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('subscribers_systems', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->integer('subscriber_id');
            $table->foreign('subscriber_id')->references('id')->on('subscribers');
            $table->integer('system_id');
            $table->foreign('system_id')->references('id')->on('systems');
            $table->integer('system_state')->default(0); // 1 is for on, 0 is for off
            $table->timestamps();
```

```
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('subscribers_systems');
    }
}
```

Код файлу Subscriber.php що відповідає моделі бази даних сутності абонента:

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Subscriber extends Model
{
    protected $table = 'subscribers';

    protected $fillable = [
        'id',
        'phone_number',
        'subscriber_type',
```

```
        'address',
        'city',
        'postal_code',
        'country_id',
        'user_id',
    ];

    public function systems()
    {
        return $this->belongsToMany('App\System', 'subscribers_systems')->withPivot('system_state');
    }
}
```

Код файлу контролера SubscriberController.php, що містить усі функції серверної логіки обробки запитів абонента:

```
<?php

namespace App\Http\Controllers;

use App\Country;
use App\Subscriber;
use App\System;
use App\User;
use Auth;
use Carbon\Carbon;
use Illuminate\Http\Request;
use View;

class SubscriberController extends Controller
```

```
{
    public function system($id)
    {
        if (!Auth::guard('admin')->check()) {
            if (Auth::user()->id != $id || !Auth::user()->subscriber) {
                return redirect()->back();
            }
        } else {
            if (!User::find($id)->subscriber) {
                return redirect()->back();
            }
        }

        $currentUser = User::find($id);

        $subscriber = Subscriber::where('user_id', $id)->first();

        $systems = Subscriber::find($subscriber->user_id)->systems;

        return View::make('user.subscriber-system')->with(['currentUser' =>
$currentUser])->with(['systems' => $systems]);
    }

    public function statistics($id)
    {
        if (!Auth::guard('admin')->check()) {
            if (Auth::user()->id != $id || !Auth::user()->subscriber) {
                return redirect()->back();
            }
```

```
} else {
    if (!User::find($id)->subscriber) {
        return redirect()->back();
    }
}

$currentUser = User::find($id);

// calculating optimal angle for solar panels

$subscriber = Subscriber::where('user_id', $id)->first();
$country = Country::find($subscriber->country_id);
$currentMonth = Carbon::now()->month;

if ($currentMonth >= 5 && $currentMonth <= 10) {
    // formula for tilt angle for summer
    if ($country->hemisphere == "N") {
        $angle = $country->latitude - 15;
    } else {
        $angle = $country->latitude + 15;
    }
} else {
    // formula for tilt angle for winter
    if ($country->hemisphere == "N") {
        $angle = $country->latitude + 15;
    } else {
        $angle = $country->latitude - 15;
    }
}
```

```
        return View::make('user.subscriber-statistics')->with(['currentUser' =>
$currentUser])->with(['country' => $country])->with(['angle' => $angle]);
    }

    public function edit($id)
    {
        if (!Auth::guard('admin')->check()) {
            if (Auth::user()->id != $id || !Auth::user()->subscriber) {
                return redirect()->back();
            }
        } else {
            if (!User::find($id)->subscriber) {
                return redirect()->back();
            }
        }

        $currentUser = User::find($id);

        $subscriber = Subscriber::where('user_id', $id)->first();

        return View::make('user.subscriber-edit')->with(['currentUser' =>
$currentUser])->with(['subscriber' => $subscriber]);
    }

    public function editSubmit(Request $request, $id)
    {
        if (!Auth::guard('admin')->check()) {
            if (Auth::user()->id != $id) {
```

```
            return redirect()->back();
        }
    }

    $subscriber = Subscriber::where('user_id', $id)->first();

    $subscriber->update($request->all());

    return redirect()->back()->with(['subscriber_submit_success' => '1']);
}

public function add($id)
{
    if (Auth::guard('admin')->check()) {
        return redirect()->back();
    }

    if (Auth::user()->id != $id || Auth::user()->subscriber) {
        return redirect()->back();
    }

    $currentUser = User::find($id);

    $countries = Country::all();

    return View::make('user.subscriber-edit')->with(['currentUser' =>
$currentUser])->with(['countries' => $countries]);
}
```

```
    public function addSubmit(Request $request, $id)
    {
        if (Auth::user()->id != $id || Auth::user()->id != $request->id || Auth::user()->id
!= $request->user_id) {
            return redirect()->back();
        }

        $user = User::find(Auth::user()->id);

        Subscriber::create($request->all());

        $user->subscriber = '1';
        $user->save();

        return redirect('subscriber-edit/' . $id)->with(['subscriber_submit_success' =>
'1']);
    }

    public function info($id)
    {
        if (!Auth::guard('admin')->check()) {
            if (Auth::user()->id != $id || !Auth::user()->subscriber) {
                return redirect()->back();
            }
        } else {
            if (!User::find($id)->subscriber) {
                return redirect()->back();
            }
        }
```

```
        $currentUser = User::find($id);

        $subscriber = Subscriber::where('user_id', $id)->first();

        return View::make('user.subscriber-info')->with(['currentUser' =>
$currentUser])->with(['subscriber' => $subscriber]);
    }
}
```

Код файлу dashboard-aside-nav.blade.php, що відповідає з серверну реалізацію меню панелі управління користувача:

```
<aside class="menu is-hidden-mobile">
    <p class="menu-label">@lang('content/user/dashboard-aside-
nav.cat_general')</p>
    <ul class="menu-list">
        <li><a href="{{ url('/dashboard') }}">@lang('content/user/dashboard-aside-
nav.cat_general_dashboard')</a></li>
        <li><a href="#">@lang('content/user/dashboard-aside-
nav.cat_general_about')</a></li>
    </ul>
    <p class="menu-label">@lang('content/user/dashboard-aside-
nav.cat_subscriber')</p>
    <ul class="menu-list">
        @if(Auth::user()->subscriber)
        <li>
            <a>@lang('content/user/dashboard-aside-
nav.cat_subscriber_subcat_system')</a>
            <ul>
```

```
            <li><a href="/subscriber-system/{{Auth::user()-
>id}}">@lang('content/user/dashboard-aside-
nav.cat_subscriber_subcat_system_system')</a></li>
            <li><a href="/subscriber-statistics/{{Auth::user()-
>id}}">@lang('content/user/dashboard-aside-
nav.cat_subscriber_subcat_system_statistics')</a></li>
          </ul>
        </li>
        @endif
        <li>
          <a>@lang('content/user/dashboard-aside-
nav.cat_subscriber_subcat_info')</a>
          <ul>
            @if(Auth::user()->subscriber)
              <li><a href="/subscriber-info/{{Auth::user()-
>id}}">@lang('content/user/dashboard-aside-
nav.cat_subscriber_subcat_info_view')</a></li>
              <li><a href="/subscriber-edit/{{Auth::user()-
>id}}">@lang('content/user/dashboard-aside-
nav.cat_subscriber_subcat_info_edit')</a></li>
            @else
              <li><a href="/add-info/{{Auth::user()-
>id}}">@lang('content/user/dashboard-aside-
nav.cat_subscriber_subcat_info_add')</a></li>
            @endif
          </ul>
        </li>
    </ul>
```

```
    <p class="menu-label">@lang('content/user/dashboard-aside-
nav.cat_support')</p>
    <ul class="menu-list">
        <li><a href="#">@lang('content/user/dashboard-aside-
nav.cat_support_faq')</a></li>
        <li><a href="#">@lang('content/user/dashboard-aside-
nav.cat_support_knowledge')</a></li>
        <li><a href="#">@lang('content/user/dashboard-aside-
nav.cat_support_contact')</a></li>
    </ul>
</aside>
```