МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА

Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту

"_____" _____ 2024 р.

Зав. кафедри, к.т.н., доцент

_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Інформаційна система для аналізу споживання електроенергії у побутових умовах

Виконав – студент 2 курсу, групи 21м-кн
освітньої програми "Комп’ютерні науки"
спеціальності 122 Комп’ютерні науки
Пилипчук Артем Олегович
Науковий керівник – д.т.н., професор
Кулаков Павло Ігорович

Умань – 2024

Реферат

на кваліфікаційну роботу «Інформаційна система для аналізу споживання електроенергії у побутових умовах»

студента факультету економіки та підприємництва
Уманського національного університету садівництва
Пилипчука Артема Олеговича

Структура та обсяг роботи - пояснювальна записка до кваліфікаційної роботи магістра складається з 3 розділів, містить 79 сторінок, 50 рисунків, 1 таблицю, 2 додатки, використано 50 літературних джерел.

Кваліфікаційна робота магістра присвячена розробці інформаційної системи для аналізу споживання електроенергії у побутових умовах. Запропонована інформаціійна система дозволяє покращити процес розрахунку вартості спожитої електроенергії та подачі показників лічильників спожитої електроенергії, аналізувати тарифи на електроенергію, має калькулятор розрахунку вартості електроенергії, забезпечує подання заявки на підключення та зворотній зв'язок.

Наукова новизна отриманих результатів кваліфікаційної роботи полягає у тому, що розроблено нові рекомендації щодо впровадження веб-системи побутового споживання енергії. Протягом розробки використовувалися теоретичні методи та експериментальні дослідження, методи математичної статистики та теорії ймовірності. Практичне значення одержаних результатів полягає в тому, що розробка дозволяє удосконалити сервіси подачі показів, має калькулятор розрахунку вартості електроенергії, подання заявки на підключення. Результати дослідження забезпечують процес зберігання, опрацювання, передавання інформації шляхом створення та використання веб-сайту.

У першому розділі проведено обґрунтування доцільності розробки, досліджуються існуючі аналоги та формулюється завдання розробки. У другому розділі виконано аналіз сучасних технологій створення інформаційних систем та розроблено структуру системи. Третій розділ містить процес розробки програмного забезпечення.

Анотація

У кваліфікаційній роботі магістра розроблено аналітичну веб-систему для аналізу побутового споживання енергії. Запропонована інформаційна система дозволяє покращити процес розрахунку вартості спожитої електроенергії та подачі показників лічильників спожитої електроенергії. Проведено аналіз існуючих аналогічних програмних продуктів, розглянуто особливості побудови сучасних веб-додатків, обґрунтовано доцільність розробки інформаційної системи для аналізу споживання електроенергії у побутових умовах. Розроблена система дозволяє удосконалити сервіси подачі показників лічильника, аналізувати тарифи на електроенергію, має калькулятор розрахунку вартості електроенергії, забезпечує подання заявки на підключення та зворотній зв'язок. Новизна роботи полягає в тому, що розроблено нові рекомендації щодо впровадження розробленої системи.

Ключові слова: веб-додаток, інформаційна система, облік електроенергії, побутове споживання, аналіз споживання.

Annotation

The master's thesis developed an analytical web system for the analysis of household energy consumption. The proposed information system makes it possible to improve the process of calculating the cost of consumed electricity and submitting indicators of consumed electricity meters. An analysis of existing similar software products was carried out, the peculiarities of building modern web applications were considered, the feasibility of developing an information system for the analysis of electricity consumption in household conditions was substantiated. The developed system allows you to improve the services of submitting meter readings, analyze electricity tariffs, has a calculator for calculating the cost of electricity, provides connection applications and feedback. The novelty of the work is that new recommendations have been developed for the implementation of the developed system.

Keywords: web application, information system, electricity accounting, household consumption, consumption analysis.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА

Факультет економіки і підприємництва
Кафедра _____Інформаційних технологій_____
Освітня програма _____Комп'ютерні науки_____
Спеціальність_____122 Комп'ютерні науки_____
(код, назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ ___Ліщук Р.І._____
(підпис) (ініціали, прізвище)
"_____" _____ 20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студенту_____*Пилипчуку Артему Олеговичу* _____
(прізвище, ім'я, по батькові)
1. Тема проекту_*Інформаційна система для аналізу споживання електроенергії у побутових умовах*
керівник роботи_____*Кулаков Павло Ігорович, д.т.н., професор кафедри інформаційних технологій*_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від _"03" грудня 2024 року №01-06/320____
2. Строк подання студентом роботи_____01.12.2023_____
3. Вихідні дані до роботи _*Розробити інформаційну систему для аналізу споживання електроенергії у побутових умовах, забезпечити аналіз експлуатаційних даних з об'єкту дослідження, використати сучасну програмну архітектуру, для розробки використати мову програмування Python.*
4. Перелік питань, які потрібно розробити __*Аналітичний огляд предметної області, обґрунтування доцільності розробки, розробка структури програмного продукту, розробка і тестування програмного модуля для системи аналізу побутового споживання енергії*
5. Перелік графічного матеріалу _50 рисунків, 1 таблиця, 2 додатки, 50 джерел___
6. Консультанти розділів роботи

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

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1.	Вибір теми кваліфікаційної роботи		
2.	Одержання індивідуального завдання		
3.	Складання календарного плану написання кваліфікаційної роботи		
4.	Підготовка до виконання кваліфікаційної роботи - підбір та вивчення літератури - участь у виконанні науково-дослідних робіт - виконання курсових робіт - інші заходи		
5.	Уточнення теми кваліфікаційної роботи та календарного плану-графіка, виходячи зі специфіки базового підприємства, установи		
6.	Підготовка: - першого розділу - другого розділу - третього розділу - висновків		
7.	Доопрацювання кваліфікаційної роботи з урахуванням зауважень наукового керівника		
8.	Написання та оформлення роботи в остаточному варіанті		
9.	Перевірка роботи на академічний плагіат		
10.	Одержання відгуку наукового керівника		
11.	Попередній захист роботи на кафедрі		
12.	Затвердження роботи завідувачем кафедри		
13.	Одержання рецензії		
14.	Захист кваліфікаційної роботи		

Студент_____ **Пилипчук А.О.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Кулаков П.І.**
(підпис) (прізвище та ініціали)

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ…….. 7
ВСТУП…… 8
РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ…….. 11
1.1. Аналіз існуючих інформаційних систем для аналізу споживання електроенергії…… 11
1.2. Інструменти та засоби реалізації інформаціійної системи …….. 18
1.3. Постановка задачі…… 22
РОЗДІЛ 2. РОЗРОБКА МОДЕЛЕЙ, ДИЗАЙНУ, ТА АРХІТЕКТУРИ ВЕБСАЙТУ…… 23
2.1. Розробка ER-моделі та UML діаграми класів ..……. 23
2.2. Дизайн вебсайту системи побутового споживання енергії……. 25
2.3. Архітектура вебсайту ……..... 36
РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ……. 40
3.1. Верстка вебсайту ……..... 40
3.2. Програмування серверної частини вебсайту …… 46
3.3. Технологічний аудит розробки…… 58
ВИСНОВОК…… 61
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ…… 63
ДОДАТОК А Лістинг програми - верстка …….. 67
ДОДАТОК Б Лістинг основних методів та класів..……..... 76

Перелік умовних позначень, скорочень і термінів

ІТ	Інформаційні технології.
ІС	Інформаційна система.
МП	Мікропроцесор.
БД	База даних.
ПЗ	Програмне забезпечення.
ПД	Програмний додаток.
МД	Мобільний додаток

Вступ

Важливість програмного забезпечення для обліку побутового споживання електроенергії не можна переоцінити, оскільки воно відіграє ключову роль у модернізації енергетичного сектору та підвищенні енергоефективності. Сучасні програми для моніторингу споживання електроенергії дозволяють споживачам отримувати детальну інформацію про свої витрати, що сприяє більш усвідомленому використанню ресурсів. Вони надають можливість аналізувати споживання за різні періоди, виявляти піки навантаження та навіть прогнозувати витрати в залежності від змін у споживанні.

Використання такого програмного забезпечення допомагає споживачам зменшити витрати на електроенергію, завдяки більш точному плануванню енергетичних потреб. Це також сприяє зменшенню негативного впливу на навколишнє середовище, оскільки більш раціональне використання електроенергії зменшує попит на її виробництво, що, в свою чергу, може знизити викиди забруднюючих речовин.

Окрім того, програмні рішення для обліку електроспоживання забезпечують зручний інтерфейс для взаємодії з надавачами електроенергії, спрощуючи процес отримання рахунків та сплати, а також підвищують інформованість населення про актуальність енергоефективних рішень. Таким чином, технології обліку споживання електроенергії стають важливим інструментом для сталого розвитку та енергетичної незалежності споживачів.

Актуальність теми дослідження. В усьому світі соціально-економічні умови проживання населення значною мірою визначаються рівнем електроспоживання. Електрична енергія забезпечує комфортність у побуті (освітлення, тепло, телебачення тощо) і є ключовим чинником під час реалізації життєво необхідних умов існування населення (водопровід, центральне опалення, ліфти, електричний транспорт тощо). Без електричної енергії неможлива робота промисловості та сфери обслуговування, які забезпечують зайнятість, оплату праці та оздоровлення населення. Стабільне, якісне постачання енергією населення та

промисловості – невід'ємна умова економічного розвитку країни. А для покращення стану та розвитку енергетичної системи України необхідним кроком є аналітика побутового споживання енергії. У сучасному світі найкраще це робити за допомогою аналітичних веб-систем побутового споживання енергії. Тому тематика даного дослідження є актуальною.

Об’єкт дослідження кваліфікаційної роботи магістра - процес аналізу побутових витрат енергії, процеси розробки методів та програмних засобів для аналізу побутових витрат енергії.

Предмет дослідження кваліфікаційної роботи магістра - програмне забезпечення власної аналітичної веб-системи аналізу побутового споживання енергії.

Мета дослідження кваліфікаційної роботи магістра - створення аналітичної веб-системи побутового споживання електроенергії, як інструменту отримання інформації та аналітики електроенергії.

Задачі дослідження кваліфікаційної роботи магістра:

- проаналізувати існуючі веб-системи побутового споживання електроенергії;

- визначити кращі сервіси і взяти їх за зразок;

- розробити макет (прототип) вебсайту та дизайн користувацького інтерфейсу, зробити верстку вебсайту по дизайну;

- провести аналіз необхідних для створення вебсайту мов програмування, систем керування базами даних та фреймворків;.

- створити власну веб-систему побутового споживання енергії.

Для досягнення поставлених в роботі завдань використовуються сучасні методи створення програмного забезпечення та баз даних.

Наукова новизна отриманих результатів:

У роботі використовувались теоретичні методи та експериментальні дослідження, методи математичної статистики та теорії ймовірності. Новизна

одержаних результатів полягає в тому, що розроблено нові рекомендації щодо впровадження веб-системи побутового споживання енергії.

Практичне значення одержаних результатів кваліфікаційної роботи магістра полягає в тому, що розроблений веб-сайт дозволяє удосконалити сервіси подачі показів лічильника, тарифи за електроенергію, калькулятор розрахунку вартості електроенергії, подання заявки на підключення та зворотній зв'язок. Результати дослідження впроваджено в процес організації зберігання, опрацювання, передавання інформації шляхом створення та використання веб-сайту побутового споживання енергії.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз існуючих інформаційних систем для аналізу споживання електроенергії

Інформаційна система підприємства "Київська обласна енергопостачальна компанія" надає можливість [1] сплачувати рахунки за електроенергію без комісії, передавати показники лічильника, контролювати стан особового рахунку, додавати до п’яти особових рахунків, дізнатися про діючі тарифи, поширює інформацію про краще використання енергії протягом дня у зв’язку з військовими діями.

Після детального дослідження та аналізу системи виявлено, що інформаційна система повністю задовільняє потреби споживачів. Особливо вразила відкритість та своєчасність цін на тарифи для побутових та малих непобутових споживачів, які при зміні одразу оновлюються. Було прийняте рішення взяти за зразок елементи такого виводу тарифів у розробці.

Також високої оцінки заслуговує support - зворотній зв’язок компанії. Окрім стандартної інформації «Про компанію» та «Режим роботи» є й посилання на «КОЛ-ЦЕНТР», «СТОП КОРУПЦІЯ. Телефон довіри» та «Оператори систем розподілу».

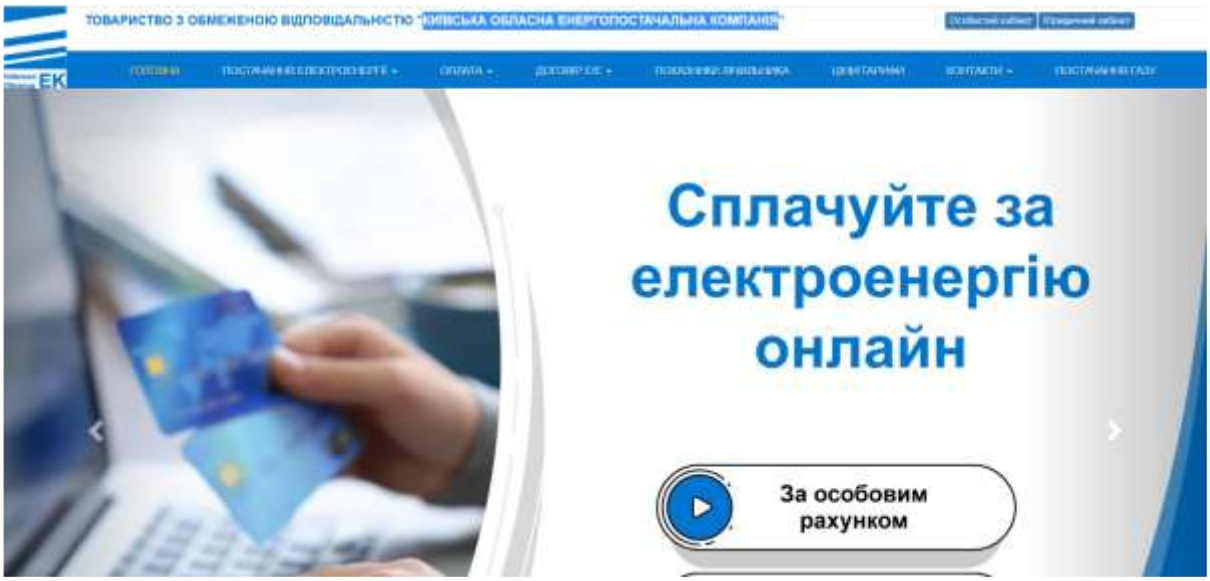


Рисунок 1.1 - Київська обласна ЕК – вигляд головної сторінки вебсайту



Рисунок 1.2 - Розумне споживання електроенергії

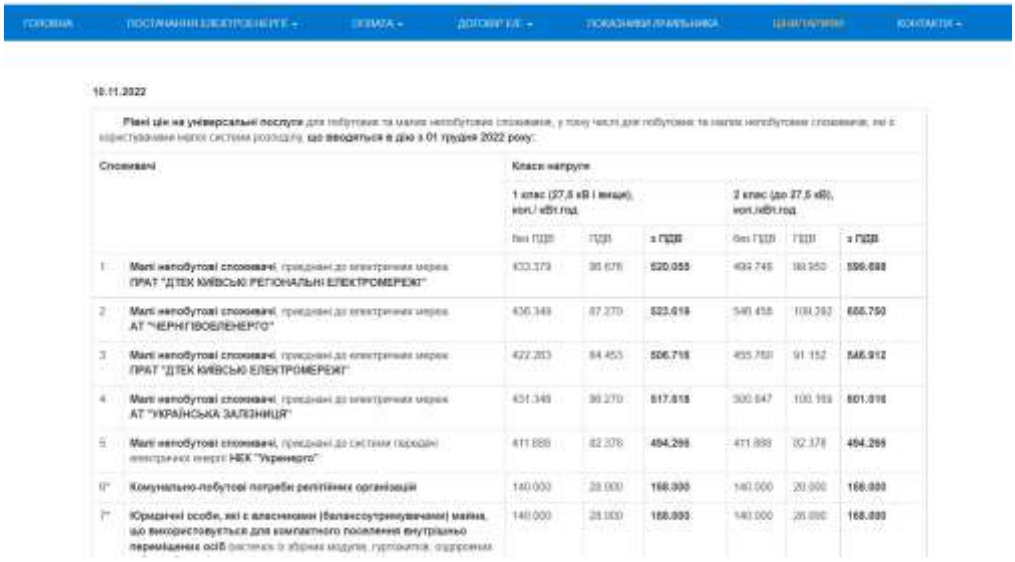


Рисунок 1.3 - Ціни на тарифи для побутових споживачів

Акціонерне товариство Полтаваобленерго [2] є спеціалізованою мережевою організацією, основний вид діяльності якої – надання послуг із передачі електричної енергії за допомогою здійснення комплексу організаційно та технологічно пов'язаних дій, що забезпечують передачу електричної енергії через технічні пристрої.

Лише тільки передати показники споживання, інформаційна система цього підприємства дозволяє сімома способами: через сайт АТ

«ПОЛТАВАОБЛЕНЕРГО» без реєстрації, у меню «Передати покази»; на e-mail pokaz@poe.pl.ua; через онлайн-сервіс «Ваша абонентська книжка»; через чат-бот «ПолтаваОблЕнерго»; через мобільний додаток, перейшовши за посиланням; через месенджер Viber; у месенджері Telegram @PltOblEnergo_bot.

У онлайн-чаті можна поділитися своєю думкою про роботу, позначити проблеми, які потребують вирішення, отримати відповіді на питання, які стосуються діяльності організації. Інформація про персональні дані громадян і компаній, що направили запит в електронному вигляді, зберігається та обробляється з дотриманням вимог українського законодавства про персональні дані і не публікується у відкритому доступі. Великим плюсом є розвинута функціональність особистого кабінету, дуже зручний спосіб передачі показників лічильників. Візьмемо його за зразок в веб-систему побутового споживання енергії, що розробляється. Значним мінусом є мала швидкість завантаження данних. При перому запуску веб-сайта необхідно чекати декілька секунд.

Проаналізувавши даний веб-ресурс, зроблено висновок, що на цьому сайті є вся необхідна для побутового споживача інформація, але сайт створено так, що якщо користуватися ним щодня, тоді працювати з ним не дуже зручно.



Рисунок 1.4 - Полтаваобленерго – вигляд головної сторінки вебсайту

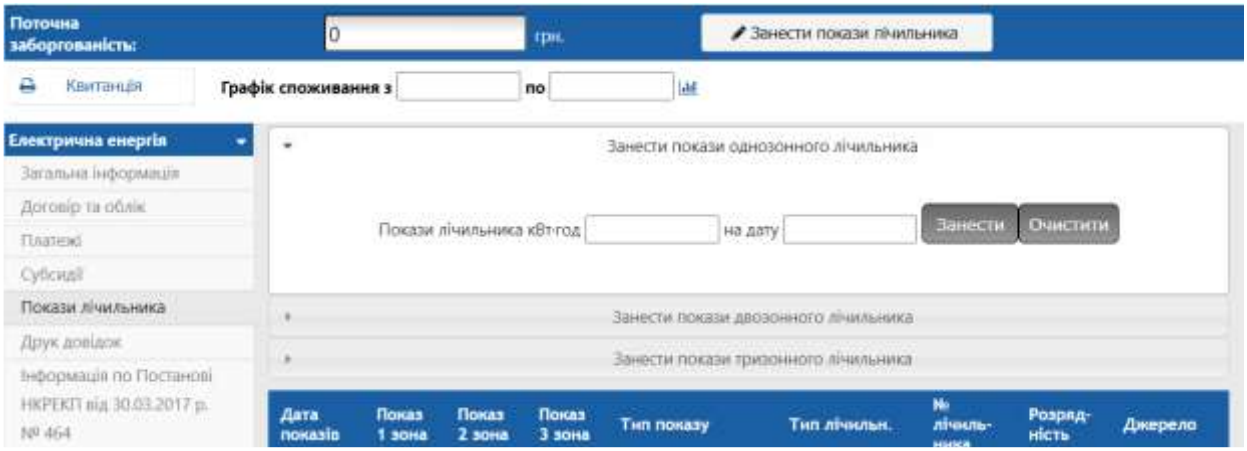


Рисунок 1.5 - Внесення показів лічильника в особистому кабінеті

Витрата електроенергії в кожній сім'ї залежить від потужності використовуваних приладів та часу їх дії. За електроенергію необхідно платити за встановленим тарифом. Щоб самостійно визначити кількість спожитої за місяць електроенергії або виконану струмом роботу, необхідно: визначити показання лічильника на початку та наприкінці місяця; знайти різницю показань лічильника на початку та наприкінці місяця (кількість спожитої електроенергії протягом місяця в кіловат-годинах); отриману кількість електроенергії помножити на тариф. Споживання електроенергії в кіловат-годинах показують лічильники електроенергії. Вони бувають різними на вигляд і типу конструкції.



Рисунок 1.6 - Види лічильників електроенергії

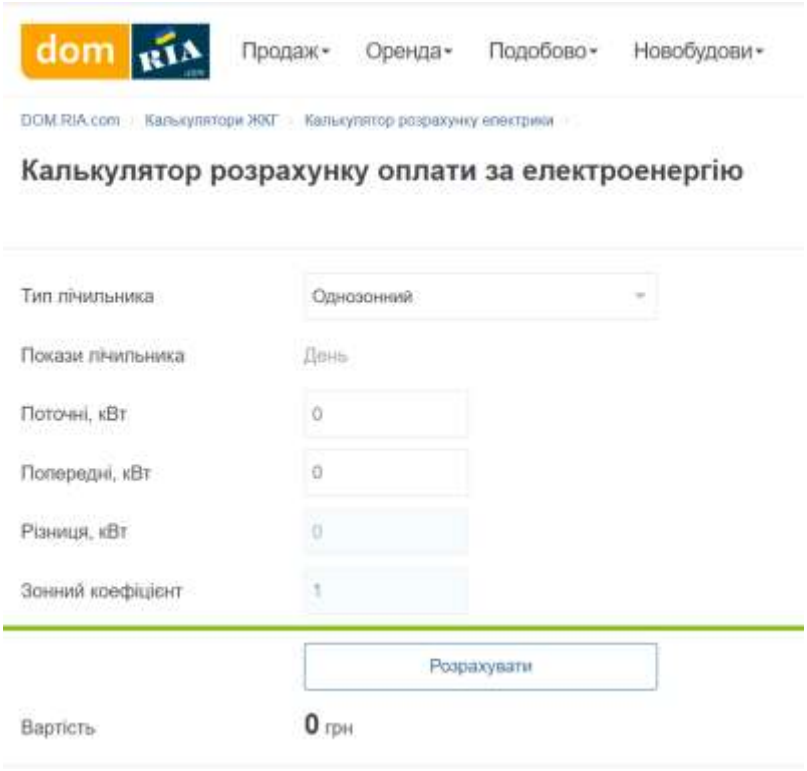


Рисунок 1.7 - Калькулятор розрахунку оплати за електроенергію

На веб-ресурсі [3] розрахунку оплати за електроенергію розрахунок електроенергії за лічильником проводиться шляхом множення спожитої кількості електроенергії (кВт · год) на тариф (грн / кВт · год).

Для двухзонного лічильника розрахунок проводиться окремо для денного і нічного часу. У нічний час (з 23.00 до 7.00) тариф розраховується зі зниженим коефіцієнтом 0,5, а в денний (з 7.00 до 23.00) - з коефіцієнтом 1. Щоб розрахувати електроенергію по двозонному лічильнику, спочатку потрібно визначити частку електроенергії, спожитої вдень і вночі.

Для трьохзонного лічильника виділяють періоди: пік – (з 8.00 до 11.00 і з 20.00 до 22.00) з коефіцієнтом 1,5; напівпік – (з 7.00 до 8.00, з 11.00 до 20.00 і з 22.00 до 23.00) – коефіцієнт 1; ніч – (з 23.00 до 7.00) – коефіцієнт 0,4.

Розраховуючи вартість обирається тип лічильника (однозонний, двозонний, трьохзонний), поточні покази (кВт), попередні покази (кВт). Після цього автоматично розраховується різниця (кВт), та зонний коефіцієнт.

Натискаючи на кнопку «Розрахувати» отримуємо «Вартість». На цьому ресурсі дуже високі показники швидкісті та зручності, тому будемо використовувати цей досвід при додаванні розрахунку енергії на створюваний нами веб-ресурс.

Полтаваенергозбут показує завжди актуальну інформацію щодо цін на електричну енергію [4]. Дуже вдалий дизайн в поєднанні з ергономікою об’єктів. Лише з головної сторінки сайту можна побачити всі основні розділи ресурсу, а також діючі тарифи та архів з неактуальними тарифами. Також є фахівці інформаційно-консультаційного центру надають консультації з питань застосування чинного законодавства України у сфері енергетики; прав і обов’язків електропостачальника та споживача; роботи структурних підрозділів ТОВ «ПОЛТАВАЕНЕРГОЗБУТ».

Великий інтерес викликає викликають підсторінки сторінки «Споживачам» де в п’яти пунктах надається вся необхідна інформація споживачам. Перелік цих пунктів: «Споживачам універсальної послуги», «Споживачам ринку вільних цін», «Побутовим споживачам», «Нормативні документи», «Корисна інформація». Візьмемо деякі з цих пунктів за основу для нашого веб-ресурсу.

В інформаційній системі ПрАТ «Закарпаттяобленерго» клієнтам надаються наступні послуги: «Тарифи», «Договір споживача про надання послуг з розподілу», «Учасникам ринку електроенергії», «Інформація для населення», «Інформація для юридичних осіб», «Перерви у електропостачанні», «Інформаційно-консультаційний центр», «Нормативні документи», «Прийом громадян». Дизайн цього веб-ресурсу дуже вдалий та зручний, тому його елементи будуть використанні як зразок для розробки.

Сайт має велику базу інформації для споживачів: «Питання та відповіді стосовно державної реформи ринку електричної енергії», «Цільові програми», «Річна, квартальна та особлива інформація», «Інформація товариства, що підлягає оприлюдненню», «Геодезичні інформаційно-технічні дані об'єктів електромереж», «Охорона праці».

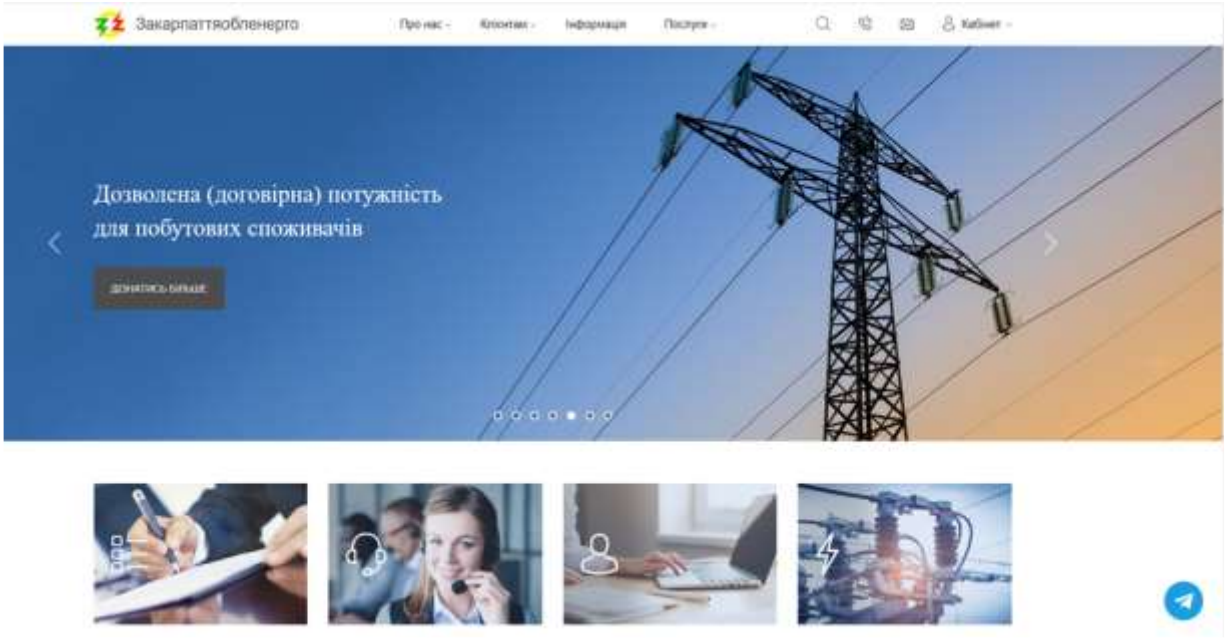


Рисунок 1.8 Закарпаттяобленерго – вигляд головної сторінки вебсайту

За функціональністю наданих послуг веб-ресурс трохи поступається Київській обласній ЕК та Полтаваобленерго, також судячи з відгуків зворотній зв’язок від компанії бажає кращого [5, 6].

Дослідивши зазначені веб-сервіси, можна зробити висновок, що всі вони є аналітичними веб-системами побутового споживання електроенергії. Кожен веб-сайт дозволяє отримувати актуальну інформацію щодо тарифів, внесення показників, новин, розрахунку вартості електроенергії. Певні з досліджених веб-ресурсів мають всі зазначені категорії, а інші лише деякі з них.

У результаті аналізу даних сервісів робимо висновок: серед наведених прикладів не існує оптимального програмного продукту, що володів би абсолютно повним функціоналом.

1.2 Інструменти та засоби реалізації інформаційної системи

Мову програмування JavaScript розробили спеціально для того, щоб створювати інтерактивні сайти. Так як Javascript – це клієнтська мова програмування, то це означає, що вона працює на стороні клієнта. Коли ми переходимо на будь-який сайт в Інтернеті, ми робимо це за протоколом HTTP. Зі свого домашнього комп'ютера ми надсилаємо запит на віддалений сервер, на якому розташовується сайт. І віддалений сервер нам вже надсилає відповідь (html-сторінку, яка буде відображена на домашньому комп'ютері у браузері). Клієнтом для протоколу HTTP є браузер. Це може бути Google Chrome, Firefox, Opera та інші. Можна спрощено сказати, що мова програмування Javascript вбудована у можливості браузера. У разі встановлення браузера ви вже маєте можливості для роботи з мовою Javascript. Звідси найголовніший плюс мови Javascript - це те, що для того, щоб ним користуватися не потрібно встановлювати будь-яке додаткове програмне забезпечення. Все вбудоване в браузер і маючи його, вже можна працювати з Javascript. Головне завдання JavaScript – внести можливості автоматизації, на вебсторінки сайтів. Насамперед ми будемо створювати інтерактивні елементи, з якими користувач може взаємодіяти. Можна взаємодіяти з цими елементами та отримувати якийсь оброблений результат [7, 8]. Крім того, за допомогою Javascript і її бібліотеки JQuery [9 -11] можна робити різні слайдери, каруселі, картинки, які змінюються самі собою. Програма JavaScript — це послідовність інструкцій, тобто вказівок браузеру виконати якісь дії. Інструкції виконуються послідовно, зверху донизу. Щоб сказати JavaScript, що інструкція закінчена, потрібно поставити крапку з комою або перейти на новий рядок. Новий рядок правильно працює в більшості випадків, а крапка з комою завжди. Тому краще ставити крапку з комою наприкінці кожної інструкції. JavaScript не змінює вихідний файл із розміткою, але, виконуючи інструкції, змінює сторінку прямо у браузері користувача.

Наступна в нашому списку HTML – це мова розмітки гіпертекстових документів. Вона потрібна, щоб відображати у браузері спеціальним чином

відформатований документ з безліччю вкладених елементів: заголовками, абзацами, списками, гіперпосиланнями, медіаджерелами, розташуванням зображень, відео та аудіо. HTML-документ це текстовий файл із розширенням .html або .htm. У браузері він перетворюється на веб-сторінку і складається з набору тегів. Вони таки допомагають представляти текст на екрані: завдяки їм браузер розуміє, що він читає не просто текст, а структуровану інформацію, розбиту на блоки. Тег виглядає як набір символів, укладений у кутові дужки. Символи у дужках позначають ім'я тега, яке описує його функції. Тег - це складовий елемент, що визначає розмітку структурних блоків. Він відкривається, і це починає свою дію; та закривається, позначаючи завершення команди. Закриті та відкриті теги розрізняються лише слішем перед ім'ям тега. Ці теги створюють оболонку, в яку міститься текст. Саме незакриті теги призводять до частих помилок та некоректних відображень сторінки. Всередині тега можуть бути атрибути – додаткова інформація, яку потрібно приховати з тексту. Вони ставляться тільки в тег, що відкриває, між ним і ім'ям тега повинен бути пробіл, а після нього йде знак рівності. Значення атрибуту полягає у лапки. За допомогою їх можна розширити можливості тегів і звернутися до них, щоб дізнатися детальну інформацію.

Далі розглянемо CSS – це мова опису зовнішнього вигляду документа, тобто вона відповідає за те, як виглядають веб-сторінки: колір фону та декоративних елементів, розмір та стиль шрифтів. Термін розшифровується як Cascading Style Sheets (каскадні таблиці стилів). CSS взаємодіє з іншою мовою розмітки - HTML, яка відповідає за розміщення елементів на сторінці. CSS, як і будь-яку мову, має синтаксис. У ньому є правила - значення, що визначають зовнішній вигляд елементів. CSS-правило складається з селектора, CSS-властивостей та їх значень. Селектори — це мітки, які допомагають браузеру зрозуміти, до якої частини HTML коду потрібно застосувати задані параметри. CSS-властивості — це певні параметри оформлення, наприклад, колір елемента або тексту (color) або колір фону (background). Значення це просто значення, воно виражається текстом або числом, наприклад чорний (black). CSS-правила в коді розташовуються у фігурних

дужках {…}. Перед відкриттям дужки обов'язково потрібно вказати селектор, до якого належить це правило.

Python – це об'єктно-орієнтована мова загального призначення, її застосовують у різних галузях. Наприклад, на ній пишуть програми, програмують системи машинного навчання, аналізують дані. Розробникам вона подобається за вбудовані структури даних, зручні функції кодингу та динамічні посилання. Це допомагає писати код на Python швидко і знижує ймовірність помилок [12]. Python має наступні переваги:

- Простий код. Програми на Python [13] легко писати та читати. Якщо розробнику передадуть чужий код, він не витрачає багато часу, щоб у ньому розібратися. Також у простому синтаксисі легше знайти помилки та вразливості.

- Мінімум зайвих завдань. Розробнику не потрібно думати про технічні моменти, пов'язані з пам'яттю. Наприклад, Python автоматично видаляє об'єкти, до яких немає доступу.

- Кросфункціональність. Підтримка Python вбудована в різні програмні платформи та операційні системи, мову можна інтегрувати з Java, C та C++. Тому написаний на Python додаток не потрібно щоразу переписувати.

- Широкі можливості. Для Python існує багато різних бібліотек наборів готових функцій. Замість писати код з нуля, їх можна розгорнути буквально двома рядками коду та користуватися. Наприклад, маніпулювати математичними операціями, будувати нейромережі, автоматизувати процеси.

SQLite - це вбудована система, яка реалізує автономний, безсерверний, нульовий конфігурації, механізм транзакції СУБД SQL. Це база даних, яка налаштована на нуль, що означає, як інші бази даних, які вам не потрібно налаштовувати у вашій системі. Особливості SQLite:

- Не вимагає окремого процесу сервера або системи для роботи.

- Постачається з нульовою конфігурацією, що означає відсутність необхідності в налаштуванні або адмініструванні.

- Повна база даних SQLite зберігається в одному крос-платформному диску.

- Дуже маленький розмір, менше 400KB повністю налаштований з додатковими функціями.

- Є автономною, що означає відсутність зовнішніх залежностей.

- Транзакції повністю сумісні з ACID, забезпечуючи безпечний доступ до кількох процесів або потоків.

- Підтримує більшість функцій мови запитів, знайдених у стандарті SQL92 (SQL2).

- Написана на ANSI-C і надає простий та простий у використанні API

- доступна у UNIX (Linux, Mac OS-X, Android, iOS) та Windows (Win32, WinCE, WinRT).

Фреймворки - це програмні продукти, які спрощують створення та підтримку технічно складних чи навантажених проєктів. Фреймворк, зазвичай, містить лише базові програмні модулі, проте специфічні для проєкту компоненти реалізуються розробником на їхній основі. Тим самим досягається не тільки висока швидкість розробки, а й велика продуктивність та надійність рішень. Вебфреймворк – це платформа для створення сайтів та вебдодатків, що полегшує розробку та об'єднання різних компонентів великого програмного проєкту. За рахунок широких можливостей у реалізації бізнес-логіки та високої продуктивності ця платформа особливо добре підходить для створення складних сайтів, бізнес-додатків та вебсервісів.

Django — безкоштовний веб-фреймворк із відкритим вихідним кодом. Він використовує архітектурний патерн Модель-Представлення-Шаблон (MVT, Model-View-Template). Особливості Django:

- Маршрутизація URL-адреси

- Міграція схем бази даних

- Робота з базами даних: PostgreSQL, MySQL, SQLite та Oracle.

- Підтримка веб-серверів

- Використання об'єктно-реляційного роутера (ORM)

- Свій двигун шаблонів.

Django має чітку структуру. Фреймворк простий у використанні, тому у програмістів-початківців є всі шанси зрозуміти його і оволодіти ним. Багато особливостей фреймворку дозволяють йому значно спростити та прискорити роботу з кодом. Регулярно оновлюючись він враховує останні версії мови Python.

В підсумку маємо обраний стек технологій для розробки нашої веб- системи, а саме:

- HTML, CSS, JavaScript – для створення фронтенду
- Python – для розробки бекенду
- SQLite – для створення бази даних
- Django – безкоштовний веб-фреймворк.

1.3 Постановка задачі

Ціллю магістерської кваліфікаційної роботи є створення аналітичної веб-системи побутового споживання електроенергії, як інструменту отримання інформації та аналітики спожитої в побуті електроенергії. Для її реалізації небхідно проаналізувати існуючі веб-системи побутового споживання електроенергії, визначити кращі сервіси і взяти їх за зразок, розробити макет (прототип) вебсайту та дизайн користувацького інтерфейсу, зробити верстку вебсайту по дизайну, провести аналіз необхідних для створення вебсайту мов програмування, систем керування базами даних та фреймворків, створити власну веб-систему побутового споживання енергії.

Завдання розробки полягає у створенні інформаційна системи для аналізу споживання електроенергії у побутових умовах, яка дозволяє удосконалити сервіси подачі показів лічильника, тарифи за електроенергію, калькулятор розрахунку вартості електроенергії, подання заявки на підключення та зворотній зв'язок.

Для реалізації завдань розробки необхідно забезпечити процес організації зберігання, опрацювання, передавання інформації шляхом створення та використання веб-сайту для аналізу побутового споживання електроенергії.

РОЗДІЛ 2. РОЗРОБКА МОДЕЛЕЙ, ДИЗАЙНУ, ТА АРХІТЕКТУРИ ВЕБСАЙТУ

2.1 Розробка ER-моделі та UML діаграми класів

Модель «сутність-зв'язок» (ER-модель) (англ. Entity-relationship model або entity-relationship diagram) - модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. ER-модель - це мета-модель даних, тобто засіб опису моделей даних. Існує ряд моделей для представлення знань, але одним з найзручніших інструментів уніфікованого представлення даних, незалежного від програмного забезпечення, що його реалізує, є модель «сутність-зв'язок». Важливим є той факт, що з моделі «сутність-зв'язок» можуть бути породжені всі існуючі моделі даних (ієрархічна, мережева, реляційна, об'єктна), тому вона є найзагальнішою [14 - 16]. Розроблена ER-модель наведена на рис. 2.1.

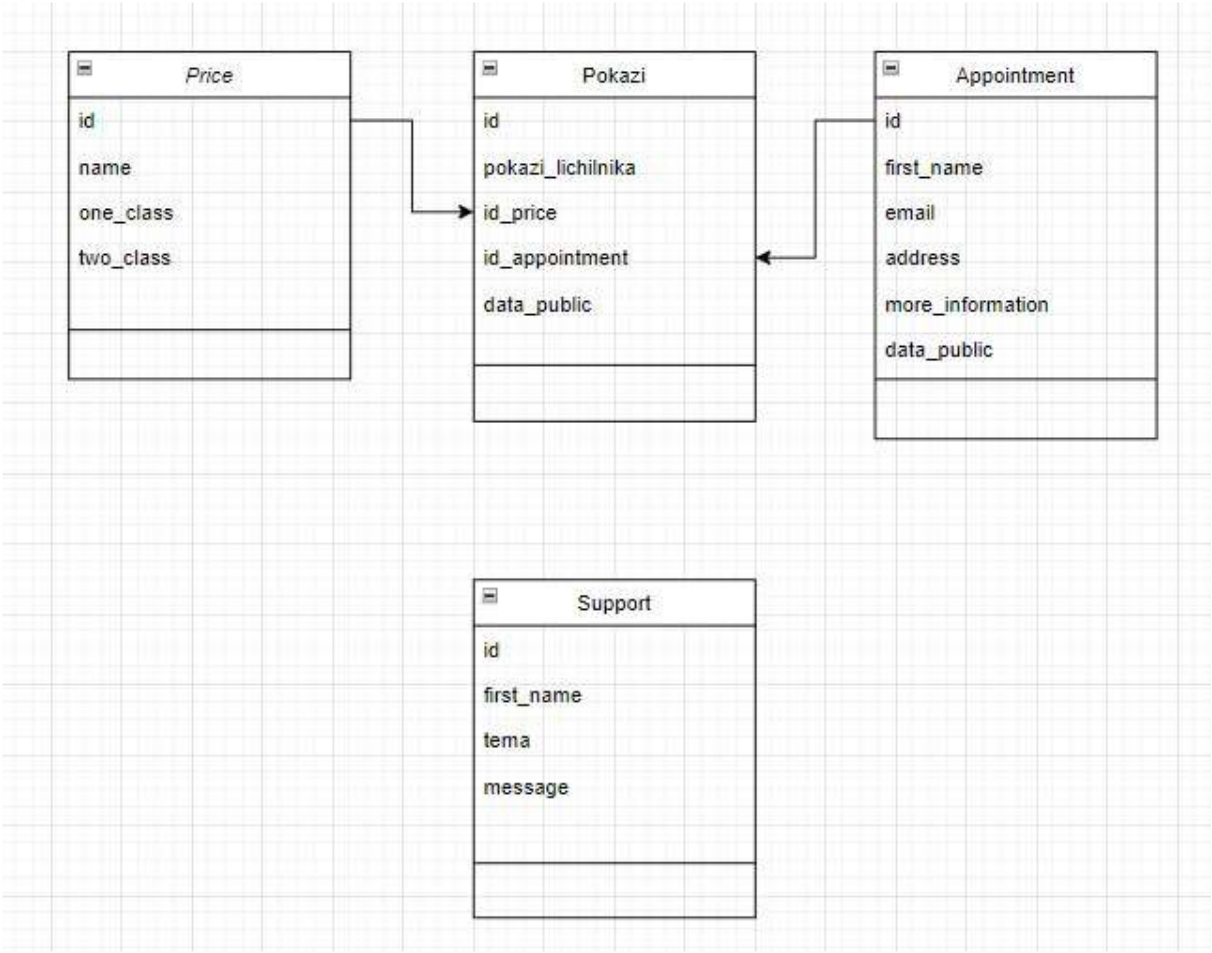


Рис. 2.1 - Модель «сутність-зв'язок» (ER-модель)

Для розробки та візуалізації артефактів програмної системи стандартною мовою є UML. Взаємозв'язок між різними об'єктами описується діаграмою класів, яка забезпечує дизайн та аналіз програми та розглядає її у статичній формі. Будучи найважливішою діаграмою UML, діаграма класів складається з класу, атрибутів та зв’язків, які є її істотними елементами. Щоб отримати уявлення про структуру програми, використовується діаграма класів, яка допомагає скоротити час ознайомлення [17 - 19]. Верхня секція (рис. 2.3) містить назву класу, середня секція описує стан, тому містить перелік полів класу, імена та типи даних, разом із указанням області видимості (“-“ – private, “+” – public, “#” – protected), нижня секція описує поведінку, тому містить містить перелік методів - імена методів, їх тип (статичні чи нестатичні), тип повернення.

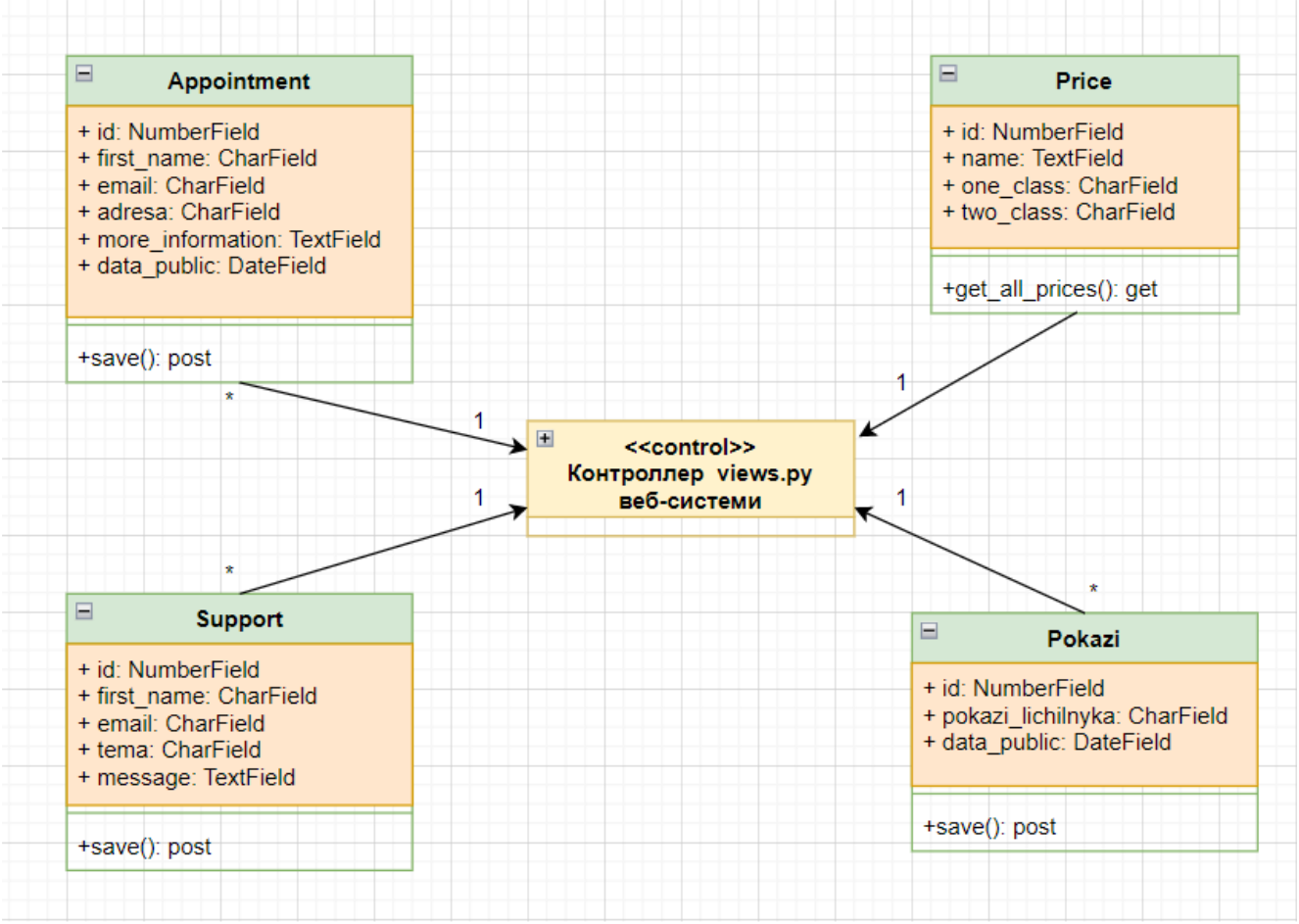


Рис. 2.2 - UML діаграма класів

2.2 Дизайн вебсайту системи побутового споживання енергії

Розробка дизайну сайту починається з прототипування. Це схематичне розташування блоків. Прототипування робиться для того, щоб можна було побачити головні елементи сторінок, структуру сайту [20 - 22]. Багато помилок, які виникають на початковому етапі розробки того чи іншого проєкту, пов'язані з неповним розумінням проєкту. Упускаються багато деталей з уваги, які згодом можуть дорогого коштувати. Для того щоб уникнути або мінімізувати ці помилки, і проводиться прототипування. Прототипування сайтів дає можливість швидко побудувати чітку структуру, проаналізувати її. Виявити всі недоліки і в найкоротші терміни усунути, з мінімальними втратами часу та фінансових коштів. Дехто вважає, що прототипування не так важливо, оскільки багато проєктів схожі і схема розташування блоків та елементів однакова. Насправді це частково не так, оскільки споживачі відрізняються один від одного, і мислять вони по-різному. Потрібно враховувати цільову аудиторію, як і що вона сприймає, на чому звернути увагу, а на чому ні. На етапі прототипування сайту деталізація інтерфейсу та інших дрібниць не дуже важливі, важливо побудувати правильну структуру та зрозуміти, як вона взаємодіятиме з користувачами. Також не варто забувати про те, що дизайнеру буде набагато простіше розробити дизайн сайту по прототипу. До прототипів зазвичай застосовують такі вимоги як:

- висока швидкість створення прототипу, швидке прототипування дозволяє заощадити час на розробку проекту;
- деталізація;
- легкість та швидкість внесення змін до прототипу;.
- інтерактивність (можливість прототипу реагувати на дію користувача);
- доступність для всіх учасників процесу (замовник, дизайнер, менеджер, програміст тощо).

Існують декілька різних методів прототипування сайтів, з яких виділяють основні:

- паперове прототипування;

- прототипування за допомогою спеціальних програм (Axure Pro, WireframeSketcher, SketchFlow тощо);

- прототипування за допомогою графічних програм (Photoshop, Illustrator, InDesign тощо).

Один з найпростіших і найшвидших методів створення прототипів сайтів, зі своїми плюсами та мінусами, є паперове прототипування. Для того щоб створити прототип достатньо використати аркуш паперу та ручку. З плюсів даного методу можна відзначити:

- високу швидкість створення прототипу;

- можливість створювати досить детальні прототипи;

- не потрібні спеціальні знання (програмування, володіння графічними редакторами);

- оптимальна швидкість зміни прототипу за рахунок перемальовки;

- можливість залишати коментарі у будь-яких місцях прототипу будь-яким із учасників розробки проекту.

З мінусів даного методу можна відзначити:

- відсутня інтерактивність у прототипу, може виявити ряд проблем в інтерфейсі;

- внесення змін не завжди можливе, доводиться перемальовувати;

- недостовірність розмірів елементів на прототипі, можливі наслідки створення дизайну сайту;

- не завжди має естетичний вигляд, за рахунок чого може не сподобатися замовнику.

Прототипування за допомогою спеціальних програм найбільш розповсюджений метод. Розглянемо його з використанням однією з найпопулярніших програм для прототипування сайтів Axure Pro. Досить проста програма, в якій сторінки організовані у деревоподібній формі. Швидкість роботи значно збільшена завдяки можливості повторного використання об'єктів (форми, картинки, текст тощо.). Кожен об'єкт має властивості, які можна змінювати.

Прототипи виходять естетичними та інтерактивними. Швидкість зміни досить висока, за рахунок можливості створення своїх об'єктів конкретних областей, будь то меню, хедер, футер, які легко можна повторно використовувати. З плюсів даного методу можна відзначити:

- високу швидкість створення прототипу;
- висока деталізація прототипу;
- прототипи мають естетичний вигляд;
- прототипи із середньою інтерактивністю;
- висока швидкість внесення змін без повторного відтворення прототипу;
- можливість створення прототипу у вигляді Html чи картинки;
- повна доступність всім учасників розробки проекту.

Якщо прототип влаштовує, тоді розробляється повноцінний дизайн. І вже після прототипування вебдизайнер змальовує макети.

Також можливе прототипування за допомогою графічних програм (Photoshop [23]). Розглянемо його на прикладі програми Adobe Photoshop [24]. Прототип можна створити досить швидко і зрозуміло, естетично привабливо. Більше такий метод підійде для створення простих та нескладних прототипів. Для об'ємних і складних проектів все ж таки більш прийнятно використовувати спеціалізовані програми. З плюсів даного методу можна відзначити:

- середня швидкість створення прототипу;
- висока деталізація прототипу;
- прототипи мають естетичний вигляд;
- висока швидкість внесення змін без повторного відтворення прототипу;
- повна доступність всім учасників розробки проекту.

З мінусів даного методу можна відзначити:

- відсутня інтерактивність у прототипу, створеного у фотошопі, може виявити ряд проблем в інтерфейсі. У деяких програмах, таких як InDesign, можна створити низьку інтерактивність;
- потрібні знання з користування графічним редактором.

Отже, прототипування нашого вебсайту ми будемо робити за допомогою Adobe Photoshop.

Детально розглянемо всі етапи макетування головної сторінки. Розіб’ємо макетування index.html на три етапи - блок «Header», блок «Наша спеціалізація», та об’єднаний блок «Наші сервіси» з «підвалом» сайту. Спочатку визначаємо всі основні елементи блоку «Header» - почнемо зі створення білого прямокутника, щоб намітити межі. Тут використовується інструмент Rectangle (1900 x 900 пікселів).



Рис. 2.3 - Використання інструменту Rectangle

Далі темним прямокутником визначимо панель навігації. Слід врахувати, що згодом на місці великого білого прямокутника буде блок зі змінним зображенням.



Рис. 2.4 - Прототипування панелі навігації

Наступним кроком додаємо заголовок та елементи навігації в макет сторінки.



Рис. 2.5 - Прототипування заголовку та елементів навігації

Далі темними прямокутниками додаємо всі місця для написання тексту в нашій «шапці» сайту.

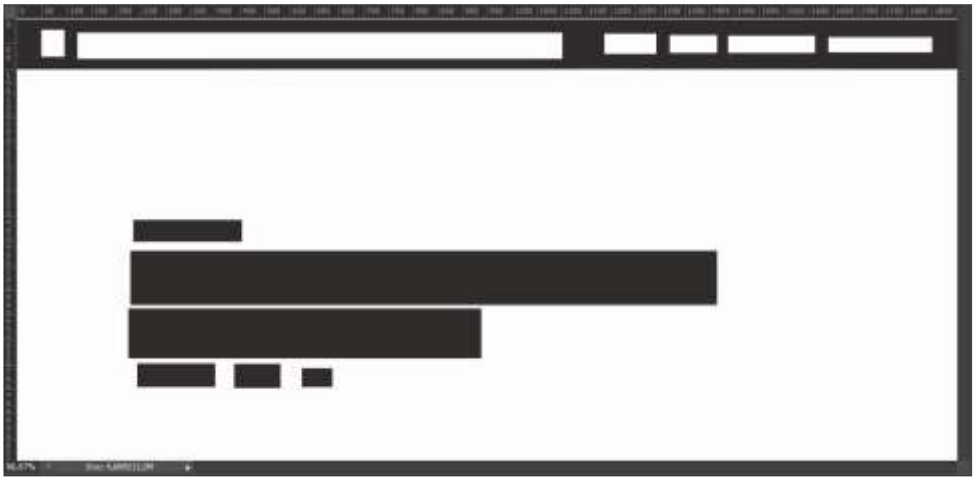


Рис. 2.6 - Додавання місць для написання тексту

Додаючи місце для трьох кругів та двох перемикачів зображень, використаємо інструмент Rounded Rectangle. Даний інструмент дозволяє змінити радіус кута фігури будь-якої миті. Це означає, що ви можете повернутися і змінити кути пізніше, якщо ваш дизайн вимагатиме цього. Також додаємо місце для кнопки з текстом.

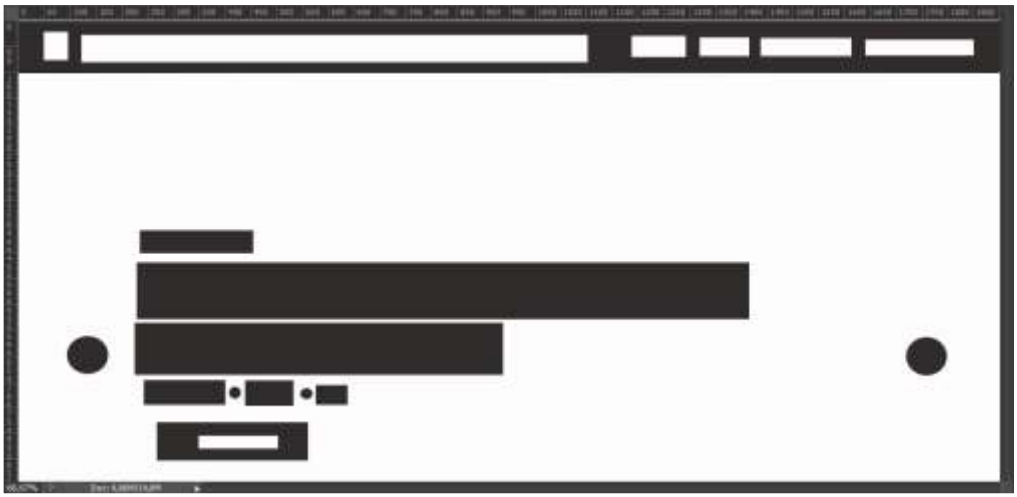


Рис. 2.7 - Використання інструменту Rounded Rectangle

Далі за аналогією створюємо блок «Наша спеціалізація», враховуючи, що навігаційна панель рухається при скролі миші, тому також відображаємо її при протопипуванні блоку.

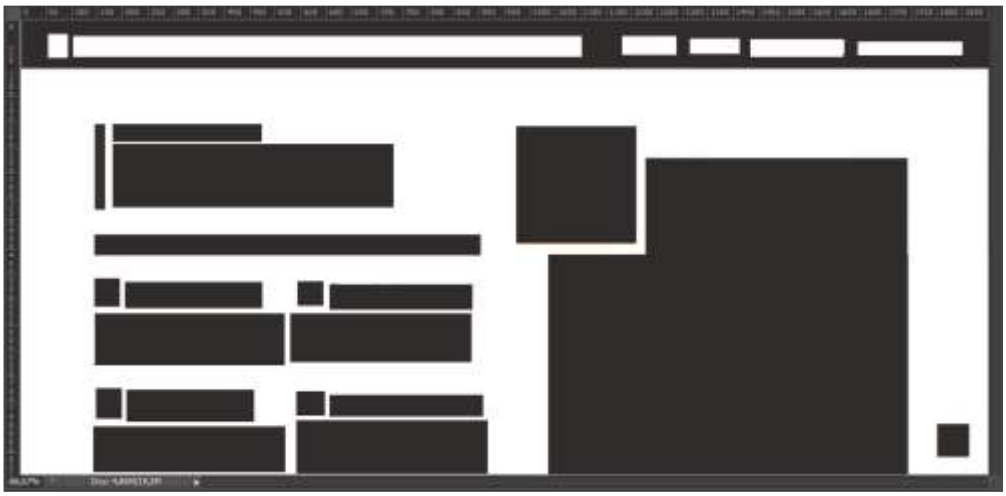


Рис. 2.8 - Створення прототипу блоку «Наша спеціалізація»

І наостанок прототипуємо об’єднаний блок «Наші сервіси» з «підвалом» сайту.

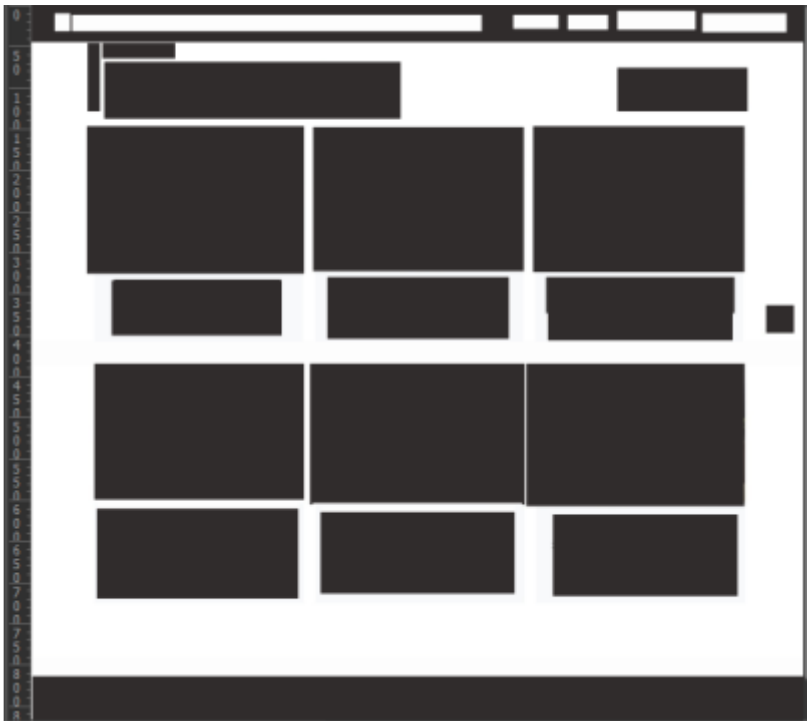


Рис. 2.9 - Створення прототипу блоку «Наші сервіси»

Прототипування всі інших сторінок вебсайту робимо за аналогією. Після того, як усі макети готові переходимо безпосередньо до дизайну по створеним прототипам вебсайта. Етапи створення дизайну по макету виглядають аналогічно прототипуванню, знову ж розглянемо на прикладі нашого сайту. При створенні дизайну вебсайту необхідно враховувати психологію кольору в дизайні основні правила його використання [25, 26].

Колір не менш важливий, ніж якість верстки та ілюстрацій, вибраний шрифт або зміст написаного повідомлення. Більше того, він виступає частиною цього повідомлення, і тому важливо правильно підбирати колір, виходячи з ідей, цілей та послання бренду. Далі ми розглянемо психологічне значення кольору і наведемо приклади того, як дизайнери та маркетологи їх використовують.

Правила кольору у дизайні [27] наступні. Вибір кольору залежить від продукту чи послуги. Люди звертають увагу на колір перед придбанням продукту чи послуги, і дизайнер має це використовувати. Наприклад, наголошувати на

головній перевагі товару відповідним кольором. Також слід враховувати вплив на психологічний стан людини теплих та холодних кольорів.

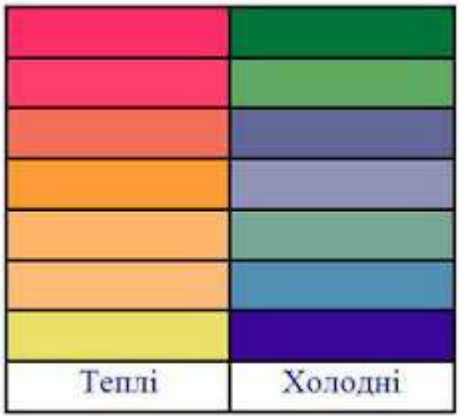


Рис. 2.10 - Теплі та холодні кольори

Колір робить продукт пізнаваним - іноді більше, ніж інші елементи дизайну. Наш мозок фокусується на відомих брендах, тому дизайнер повинен приділяти особливу увагу домінуючому кольору, який довго використовуватиметься в айдентиці компанії. Це твердження легко довести. Коли ми перебуваємо в незнайомому районі чи чужій країні і шукаємо, де поїсти чи випити каву, ми можемо побачити «Макдональдс», KFC або, наприклад, Starbucks здалеку - задовго до того, як розглянемо назву чи логотип закладу. Все тому, що брендам вдалося "закріпити за собою" кольори, які міцно з ними асоціюються. Нижче наведені основні ассоціації які викликає у людини певний колір.

Червоний - позитивні асоціації: сила, збудження, пристрасть, енергія, молодість, можливі негативні асоціації: гнів, небезпека. Помаранчевий - позитивні асоціації: довіра, тепло, інновації, доброзичливість, енергія, хоробрість, можливі негативні асоціації: розчарування, невігластво, незрілість. Жовтий - позитивні асоціації: оптимізм, тепло, щастя, креативність, доброзичливість, можливі негативні асоціації: обережність, неспокій, страх. Зелений - позитивні асоціації: здоров'я, надія, природа, зростання, свіжість, добробут, можливі негативні асоціації: заздрість, хвороба, нудьга. Фіолетовий - позитивні асоціації: мудрість,

розкіш, добробут, духовність, витонченість, велич, можливі негативні асоціації: зосередженість собі, занепадництво, пригніченість. Рожевий - позитивні асоціації: уява, запал, перетворення, баланс, креативність, можливі негативні асоціації: надмірна жіночність, імпульсивність. Білий - позитивні асоціації: чистота, ясність, непорочність, простота, свіжість, можливі негативні асоціації: стерильність, холод, ізоляція.

Враховуючи вплив кольорів на людину, спочатку створимо навігаційну панель вибравши колір тіла білим, активний текст та зображення помаранчеві, а неактивний текст чорний.

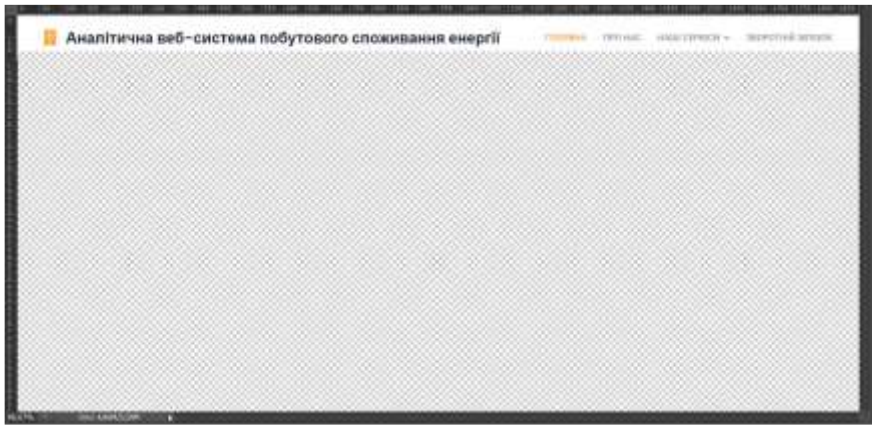


Рис. 2.11 - Створення дизайну навігаційної панелі

Далі додамо до зображення, не забуваємо, що воно буде змінним.



Рис. 2.12 - Додавання до дизайну змінного зображення

І наостанок додаємо текст та інші елементи.



Рис. 2.13 - Додавання до дизайну тексту

Далі, аналогічно, робимо дизайн блоку «Наша спеціалізація».



Рис. 2.14 - Дизайн блоку «Наша спеціалізація»

А також блоку «Наші сервіси» разом з футером, «підвалом» вебсайту.

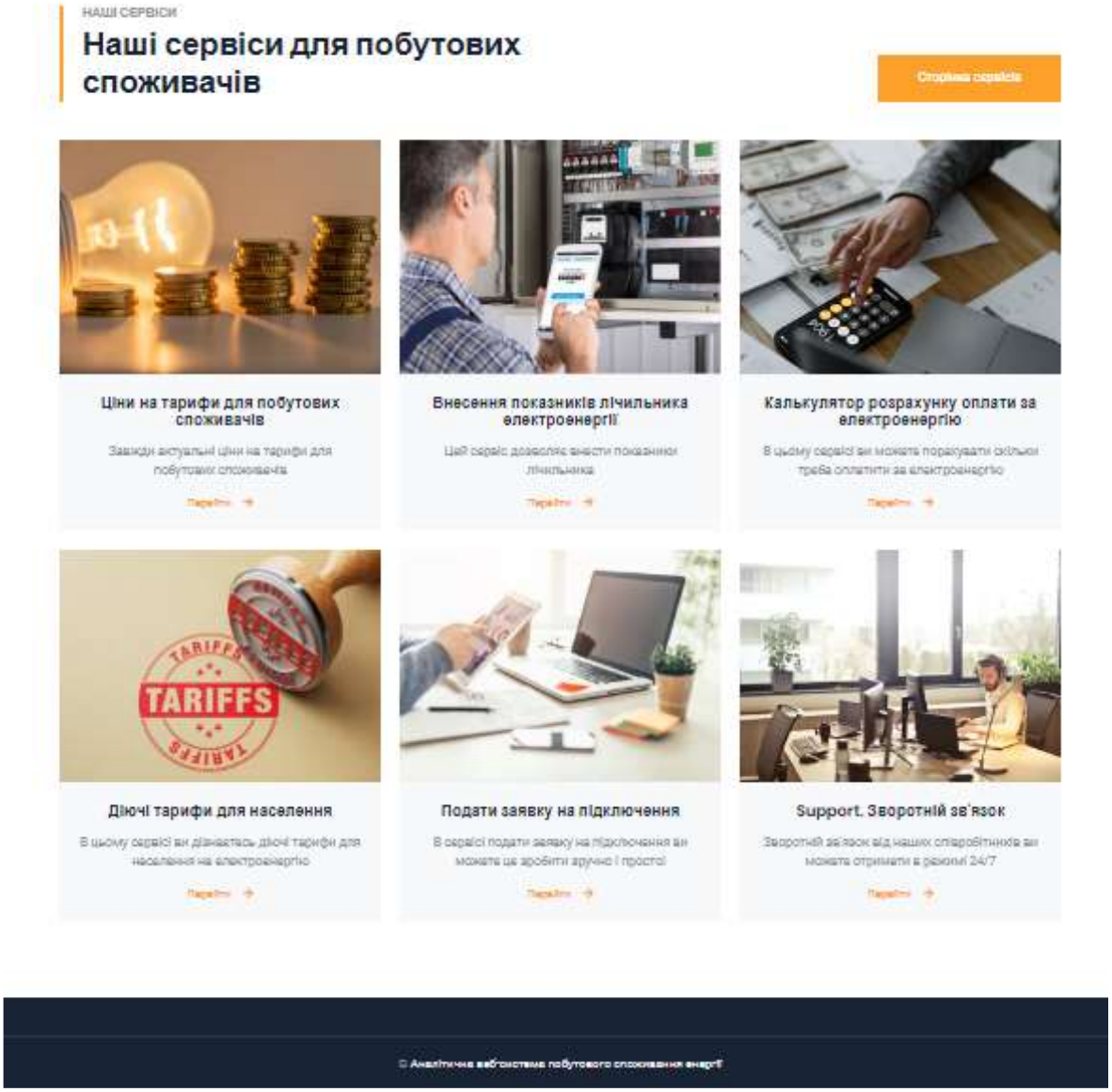


Рис. 2.15 - Дизайн блоку «Наші сервіси»

За аналогією, створюємо дизайн всіх сторінок сайту. Створивши дизайн для всіх сторінок, верстка сайту, буде менш складним процесом. Дизайн всієї головної сторінки буде мати наступний вигляд.

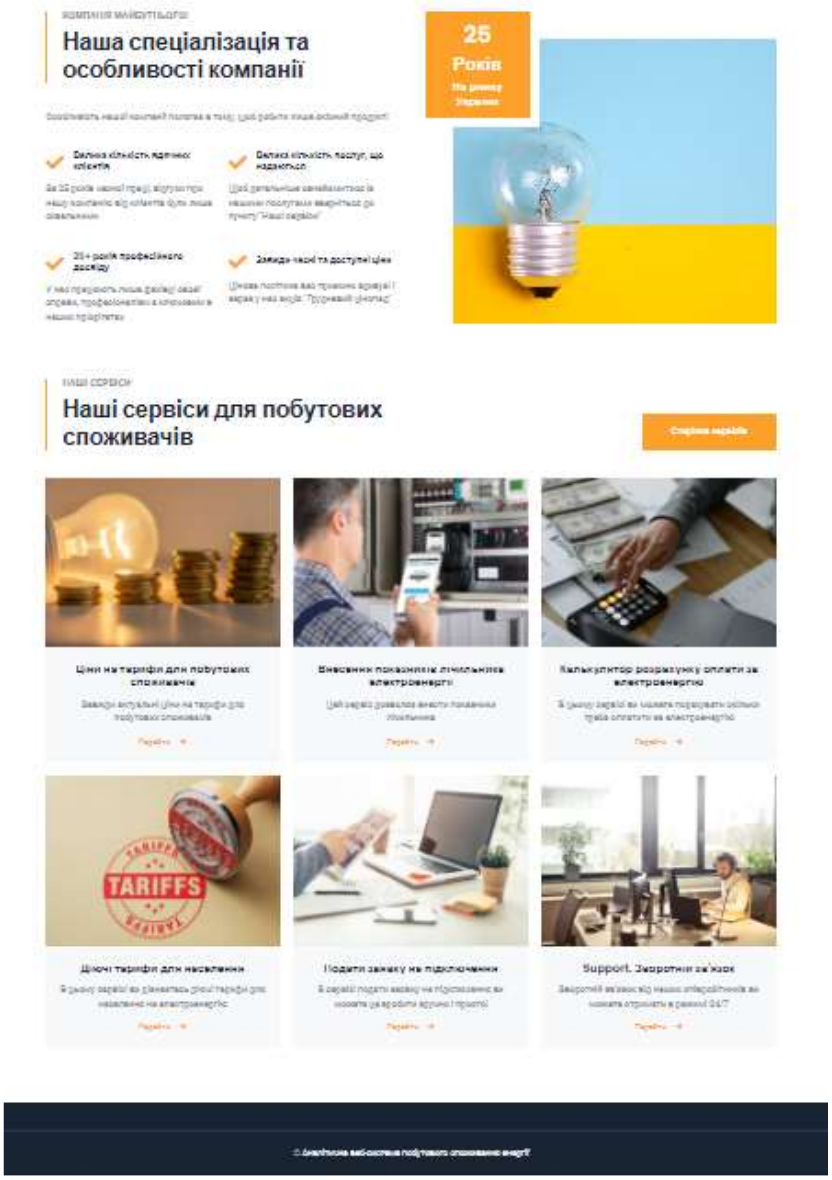


Рис. 2.16 - Дизайн головної сторінки сайту

2.3 Архітектура вебсайту

Архітектура сайту є структурою його сторінок і програмної частини. Це визначена, чітко оформлена система, що організовує різні файли, гіперпосилання, заголовки, пошук, навігаційну панель і просто інформацію, яка міститься на сайті. Саме логічна побудова всіх цих елементів, їхнє місце розташування та взаємозв'язок визначають, наскільки буде зручно і просто користуватися цим ресурсом. Така структура дає можливість користувачеві бачити всі розділи сайту,

дозволяє ознайомитися з різною інформацією, представленою на ньому, тобто значно підвищує зручність користування. Створення архітектури сайту є першочерговим етапом у його проектуванні [28 - 31]. Архітектура сайту потрібна для створення ресурсу, який допоміг би абсолютно будь-якому користувачеві знаходити потрібну йому інформацію, не витрачаючи на це великої кількості часу. Сучасна людина малоймовірно вирішить витрачати надто багато часу, відкриваючи нескінченну кількість посилань у пошуках потрібної інформації. У цій ситуації користувачеві зручніше залишити сайт і просто ввести запит у пошукову систему.

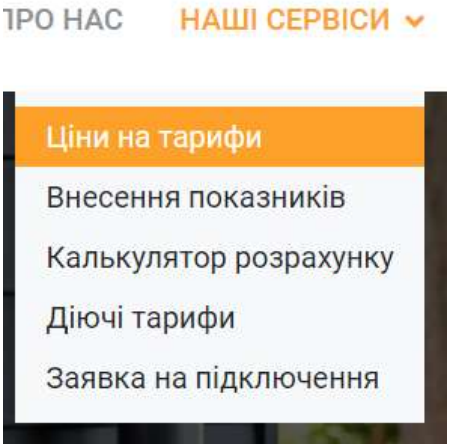


Рис. 2.17 - Архітектура блоку «Наші сервіси»

Добре продумана та прописана архітектура як великого, так і маленького веб-сайтів здатна значно спростити індексацію сторінок. Проте реалізовувати цю концепцію набагато зручніше на початковому етапі проектування ресурсу, коли вносити зміни ще не є чимось складним. Будь-якій людині знайома ситуація, коли пошук потрібної інформації на створеному сайті є досить складним і довготривалим процесом, особливо, якщо ресурс багаторазово збільшив кількість своїх сторінок. Частковим виходом із ситуації може стати пошук за ключовими словами, проте таке рішення ніяк не впливає на індексацію, та й є оптимальним далеко не для всіх. Тому оптимізація архітектури сайту є в цьому випадку найзручнішим рішенням. Це забере багато часу, але зробить сайт найбільш зрозумілим простому користувачеві, йому буде легше орієнтуватися і шукати необхідну інформацію. Процес оптимізації структури являє собою систематизація

всіх сторінок, грамотний розподіл внутрішніх посилань, робота з необхідними тегами, які є незамінними при пошуковому просуванні. Основна мета оптимізації в цьому випадку - зробити сайт максимально корисним та привабливим для користувача, допомогти йому знаходити потрібну інформацію, докладаючи мінімум зусиль. Якщо ж справа стосується комерційного ресурсу, то можуть з'явитися інші вимоги, наприклад, зробити так, щоб для відвідувачів стали найбільш доступними ключові сторінки, які приносять основний прибуток. Добре розроблена архітектура сайту допомагає виділити найважливіші сторінки для пошукових систем, наприклад, «Сервіси» або «Послуги». Такі посилання зазвичай розміщені на головній. На рисунку 2.18 зображена структура проєкту.

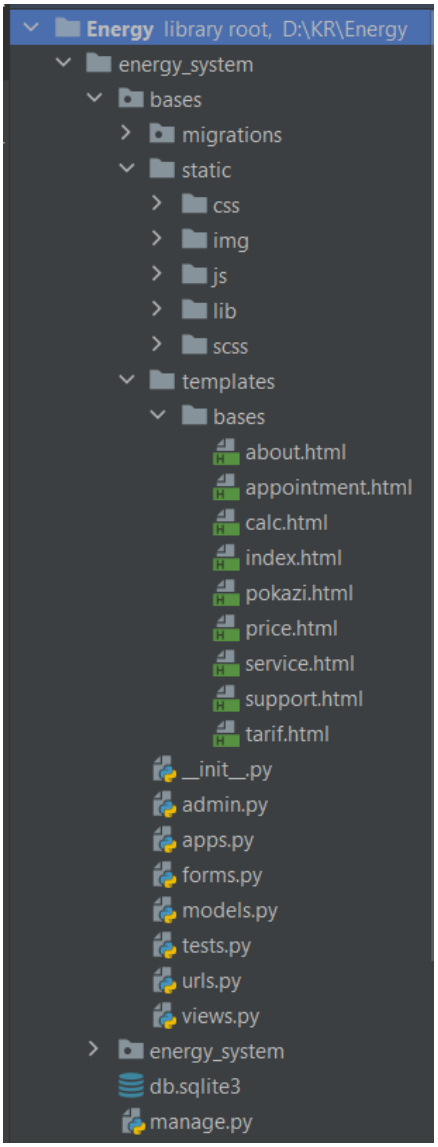


Рисунок 2.18 - Структура проєкту

В таблиці 2.1 наведено опис та призначення головних файлів в структурі проєкту.

Таблиця 2.1 - Опис та призначення головних файлів в структурі проєкту

Файл або Папка	Опис
Energy_system	Папка з основним проєктом
Bases	Папка з базовим проєктом
Migrations	Папка з міграціями
Static	Папка із статичними файлами і папками. Каскадними таблицями стилів css, зображеннями img, javascript скриптами js, бібліотеками lib, та препроцесами scss.
Templates	Папка із html-файлами
Admin.py	Файл із налаштуваннями адмін-панелі
Forms.py	Файл із налаштуваннями html-форм
Models.py	Файл із налаштуваннями моделей бази даних
Urls.py	Файл із налаштуваннями роутів
Views.py	Файл-контроллер

Як можна побачити з таблиці 2.1, структуру проєкту можна поділити на дві основні частини - вихідні файли, які групуються завдяки реалізації шаблону MTV (Model-Templates-Views), та веб-частини, яка дозволяє нам мати візуальний інтерфейс веб-системи в будь-якому браузері. Всі ці файли необхідні для розгортання сайту на сервері Django (або веб-хостингу) та її запуску.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Верстка вебсайту

Термін «верстка» прийшов у веб із видавничого ремесла. Там під версткою розуміється монтаж макета газети чи книги з підготовленого матеріалу - тексту, таблиць, зображень, що потім спрямовується до друку. В обох випадках головне завдання верстки - зробити так, щоб сайт чи друковане видання було зручно переглядати та взаємодіяти з ними. Верстка - це створення структури гіпертекстового документа і розробка графічного інтерфейсу користувача за допомогою HTML, скриптів і стилів. Головне завдання верстальника - точно і без помилок відправити прототип до реального коду. Простими словами, верстка - це перенесення попереднього макету сайту в HTML. Англомовний еквівалент терміна «верстка веб-сторінок» - Frontend web development, що можна перекласти як «фронтенд веб-розробки». Фронтенд сайту це те, що бачить звичайний користувач - текст, зображення, інші графічні елементи. Бекенд сайту - це внутрішня, невидима оку користувача частина сайту - як підкапотний простір автомобіля. Верстка розташовує всі елементи на сторінці і робить так, щоб з ними було зручно працювати. Тому верстка сайту - це відповідальне завдання, яке вимагає уважності, терпіння та постійного тестування. Верстку веб-сторінок неможливо уявити без HTML [32]. Якщо говорити простими словами, HTML це єдиний стандарт відображення всіх елементів веб-сторінки. Це мова розмітки, за допомогою якої браузери показують порядок, розмір, форми та шрифт тексту. З його тегами знайомі всі, хто займався створенням сайтів, наприклад:

<body> </body> - весь вміст сторінки;

<h1> </h1> - це позначення заголовка;

<h2> </h2> - підзаголовок;

<img> - зображення;

 - жирний шрифт;

Залежно від типу завдання та його складності, етапи верстки сайту можуть відрізнятися. Стандартний алгоритм роботи верстальника має такий вигляд:

- перенесення структури попереднього макету в HTML-документ;

- робота зі «стилями», завдання глобальних налаштувань CSS згідно з параметрами прототипу;

- налаштування та завдання властивостей для інтерактивних компонентів сторінки, при цьому можуть використовуватися різноманітні технології - наприклад, JQuery та JS;

- перенесення попереднього шаблону до системи керування контентом;

- тестування, на цьому етапі розробник намагається виявити типові проблеми: наприклад, некоректне відображення сторінок у різних браузерах або зміщення блоків у різних мобільних пристроях.

Сьогодні найпоширенішими є два типи верстки: блокова та таблична. Блокова верстка цілком відповідає основним принципам адаптивності і ним не суперечить. Сучасний сайт краще верстати саме за її допомогою. Більш сучасна, добре поєднується зі стилями вже «з коробки». Це означає, що за необхідності можна швидко змінити розташування об'єктів, задати потрібні відступи, інтервали, змінити рішення кольору будь-якого елемента сторінки. Крім того, блокові елементи сторінки краще з точки зору швидкодії та чуйності. Схематично уявити блоковий тип верстки можна так:

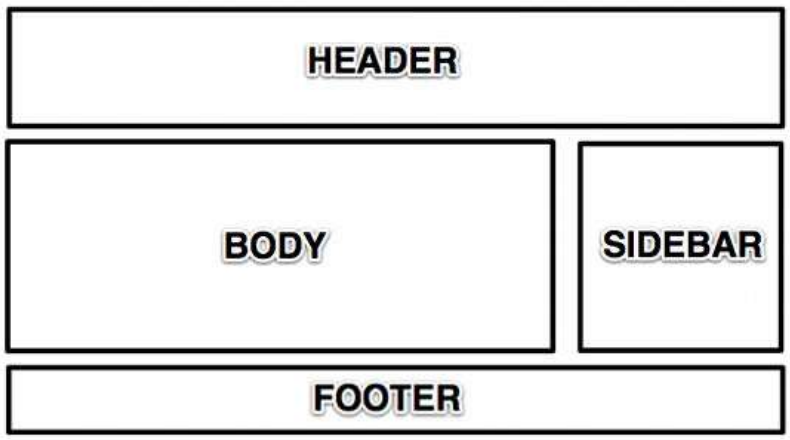


Рисунок 3.1 - Блоковий тип верстки

Таблична верстка, це застаріле рішення, яке активно використовувалося в інтернеті десятки років тому. Сьогодні цей підхід розмітки під час створення сайтів практично не зустрічається.

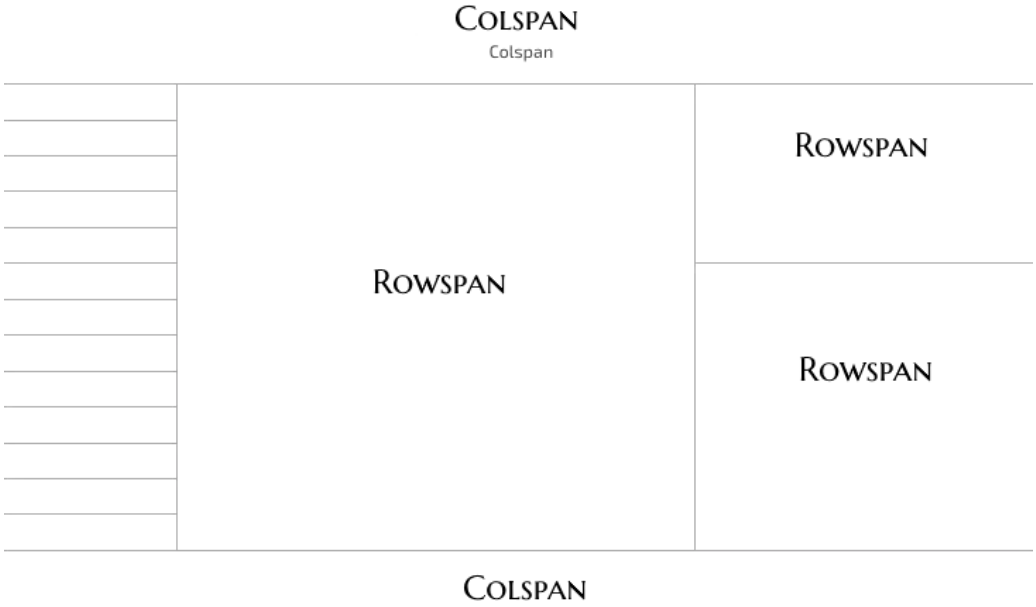


Рис. 3.2 - Табличний тип верстки

На відміну від блочного підходу, в табличному, розмітка задається тегом <table>. Після нього в ієрархічному порядку слідує <tr> і <td> - це рядки та стовпці, відповідно. Розмістити елемент на сторінці можна тільки створюючи порожні комірки, які будуть подібними до скелета. Такий підхід є досить трудомістким, окрім того, він збільшує фізичний розмір сторінки, що завжди погано. Таблична верстка не підходить для сучасних сайтів, що проектуються на основі адаптивного дизайну. Мабуть, єдине вдале застосування табличної верстки - це коли сторінка відкривається на весь екран незалежно від дозволу.

Для верстки вебсайту необхідно володіти наступними технологіями це мова гіпертекстової розмітки HTML [33], каскадні таблиці стилів CSS [34], та мова програмування Javascript [35]. Треба чітко розуміти що HTML - це каркас сайту [36 - 38]. По ньому браузер малює веб-сторінку, форматує за допомогою CSS і додає інтерактив через JavaScript.

Отже, для того щоб зручно використовувати наведені технології потрібен програмний продукт, який буде легко співпрацювати у їх поєднанні. Тому, для створення верстки по дизайну обираємо Bootstrap [39], найпопулярніший у світі HTML, CSS и JS фреймворк для створення адаптивних сайтів, а також програму-редактор Notepad++. Bootstrap з'явився в компанії Twitter, спочатку це була невелика бібліотека для внутрішніх потреб компанії, проте він швидко завоював популярність і вийшов далеко за її межі. На сьогоднішній день Bootstrap є визнаним фаворитом серед розробників адаптивних сайтів. Це фреймворк, що складається із шаблонів HTML, CSS та розширень JavaScript із бібліотекою JQuery [40 - 42]. Він дотримується принципу "mobile-first", що передбачає початкову розробку сайту для невеликого екрана мобільного телефону, потім контент розміщується згідно з роздільною здатністю планшета і, нарешті, верстальник працює над версією для ноутбука та комп'ютера. Завдяки такому підходу кінцевий користувач може переглядати сайт на будь-яких пристроях, не відчуваючи найменшої незручності.

Існує два способи підключення фреймворку до проекту. Його можна завантажити з офіційного сайту http://getbootstrap.com або використовувати посилання на версію CDN. Використання CDN має свої плюси та мінуси. Безперечною перевагою є швидке завантаження. Однак, якщо мережа недоступна, фреймворк не завантажиться. Переваги сайту на Bootstrap:

- швидкість розробки;
- адаптивність;
- популярність;
- безкоштовність;
- гнучкі налаштування;
- документованість;
- підтримка.

Bootstrap активно розвивається, виходять нові версії, виправляються помилки та з'являється новий функціонал. Коротко розглянемо принцип роботи фреймворка на прикладі компонента Dropdowns [43]:

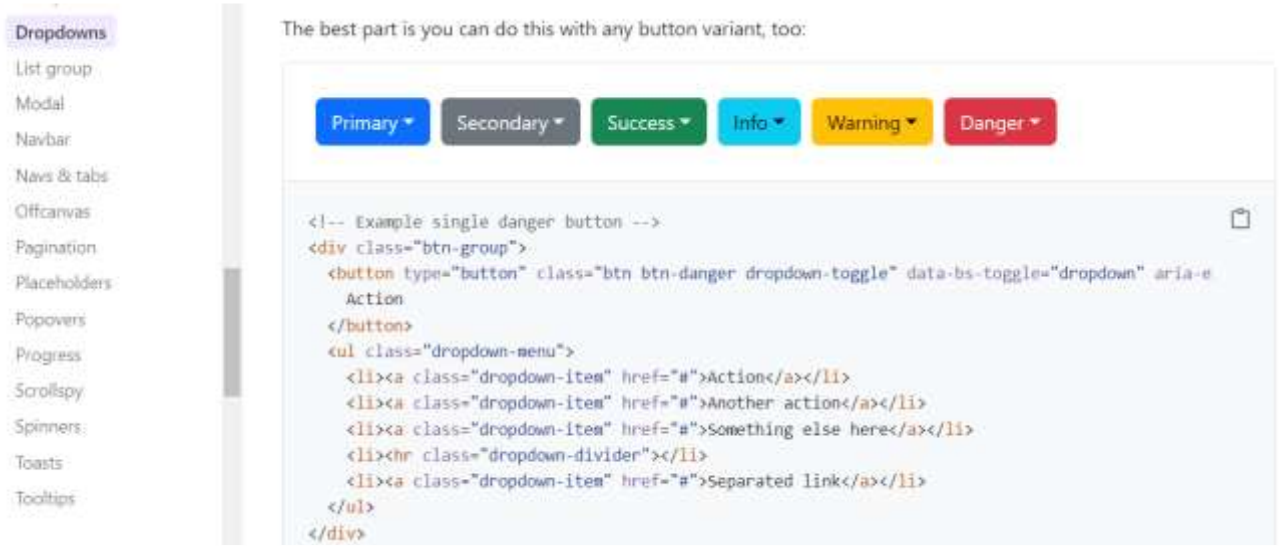


Рис. 3.3 - Компонента Dropdowns Bootstrap

Вся система стилів побудована на класах, тобто якщо необхідно, щоб елемент сприймався бібліотекою як кнопка, тоді потрібно прописати клас btn. Якщо хочемо, щоб кнопка була червоного кольору, тоді додаєм ще один клас btn-danger, якщо жовтого btn-warning. Сайти створені за допомогою Bootstrap мають однакову навігацію, структуру, кнопки. Щоб вирішити проблему, можна змінювати шаблон в залежності від ідей дизайнерів та побажань замовника. Приведемо приклад навігаційної панелі файлу index.html:

```
<!-- Navbar Start -->
<nav class="navbar navbar-expand-lg bg-white navbar-light
sticky-top px-4 px-lg-5 py-lg-0">
<a href="index.html" class="navbar-brand d-flex alignitems-
center">
<h3 class="m-0"><i class="fa fa-building text-primary
me-3"></i> Аналітична веб-система побутового споживання
енергії</h3>
</a>
```

```
<button type="button" class="navbar-toggler" data-bstoggle="
collapse" data-bs-target="#navbarCollapse">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarCollapse">
<div class="navbar-nav ms-auto py-3 py-lg-0">
<a href="index.html" class="nav-item nav-link
active">Головна</a>
<a href="about.html" class="nav-item nav-link">Про
нас</a>
<div class="nav-item dropdown">
<a href="service.html" class="nav-link dropdown-toggle"
data-bs-toggle="dropdown">Наші сервіси</a>
<div class="dropdown-menu bg-light m-0">
<a href="price.html" class="dropdown-item">Ціни на
тарифи</a>
<a href="pokazi.html" class="dropdown-item">Внесення
показників</a>
<a href="calc.html" class="dropdown-item">Калькулятор
розрахунку</a>
<a href="tarif.html" class="dropdown-item">Діючі
тарифи</a>
<a href="appointment.html" class="dropdownitem">
Заявка на підключення</a>
</div>
</div>
<a href="support.html" class="nav-item nav-link">Зворотній
зв'язок</a>
```

```
</div>
</div>
</nav>
<!-- Navbar End -->
```

В наступному пункті, при програмуванні серверної частини вебсайту нам необхідно буде змінити частину верстки під вимоги фреймворка Django.

3.2 Програмування серверної частини вебсайту

Для програмування бекенду нашої веб-системи ми обрали фреймворк Django та мову Python. Встановлюємо Python [44] і також встановлюємо безкоштовну (Community) програму PyCharm [45], тому що вона працює як з python так і з Django [46]. Спочатку створимо проєкт в Pycharm.

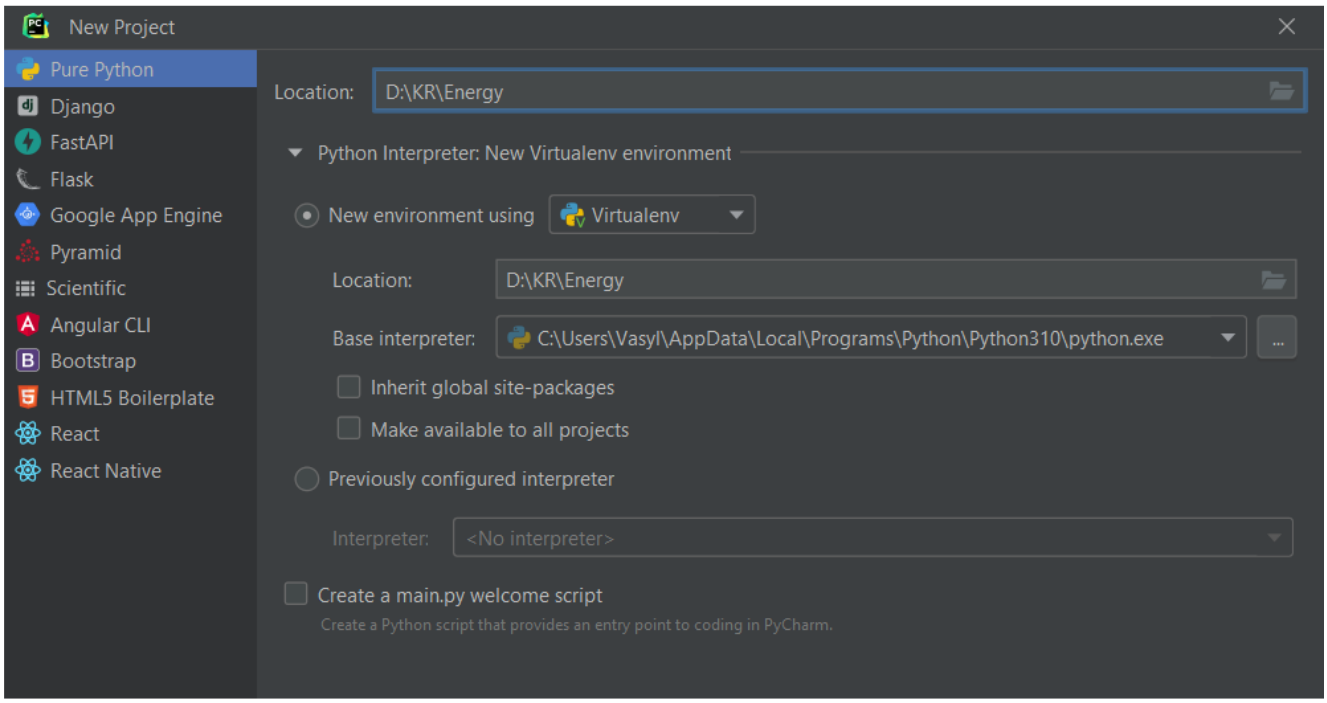


Рис. 3.4 - Створення проєкту в Pycharm

Далі створюємо віртуальне середовище, для того, щоб залежності встановлювалися локально в проєкт, а не глобально в операційну систему.

```
(Energy) PS D:\KR\Energy>  python -m venv venv
```

Рис. 3.5 - Створення віртуального середовища

Після цього, щоб увійти в віртуальне середовище потрібно активувати скрипт.

```
(Energy) PS D:\KR\Energy> D:\KR\Energy\venv\Scripts\Activate.ps1
```

Рис. 3.6 - Активація скрипту віртуального середовища

Після описаних дій ми опинимося в віртуальному середовищі.

```
(venv) PS D:\KR\Energy>
```

Рис. 3.7 - Активне віртуальне середовище

Далі по черзі запускаємо команди:

- pip install django – встановлюємо фреймворк Django;
- django-admin startproject energy_system - створюємо проєкт сайта;
- cd .\energy_system\ - переходимо в папку з проєктом;
- python manage.py runserver – запускаємо сервер.

Після чого переходимо за посиланням http://127.0.0.1:8000/ і бачимо стартову сторінку Django-проєкта.

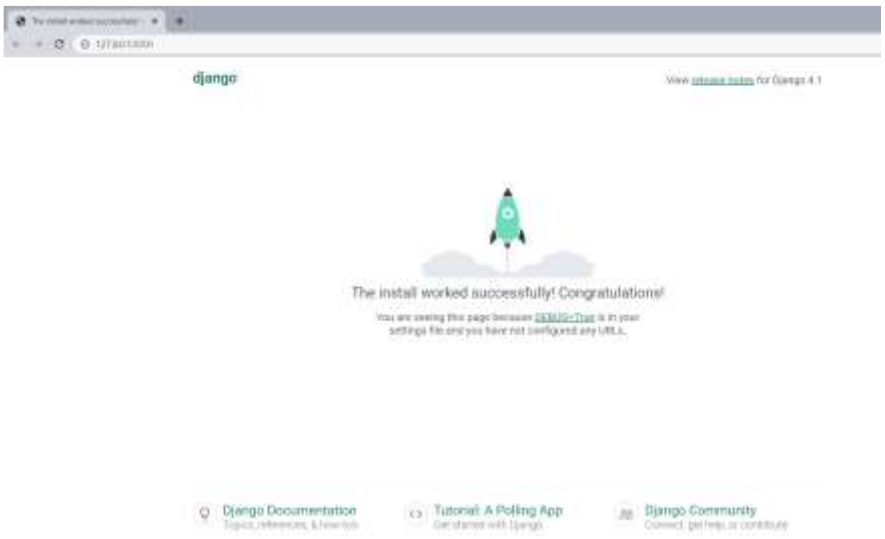


Рис. 3.8 - Стартова сторінка Django-проєкта

Для того, щоб зупинити сервер Django необхідно в терміналі натиснути сполучення клавіш CTRL+C. Далі проводимо міграції: python manage.py migrate. Ця команда створить в базі даних вже готові по замовчуванню таблиці, наприклад таблиці для адмін-панелі. У фреймворку Django admin-панель створюється разом із створенням проєкту. Доступна панель за посиланням http://127.0.0.1:8000/admin. Але перш ніж в адмін-панель увійти, потрібно створити за допомогою терміналу суперюзера: python manage.py createsuperuser; Username: admin; Email: admin@examples.com; Password: Vinny.

Запускаємо сервер, вводимо створені логін та пароль і потрапляємо в адмін-панель:



Рис. 3.9 - Адмін-панель Django

Для того, щоб змінити інтерфейс з англійської на українську мову потрібно в settings.py знайти LANGUAGE_CODE = 'en-en' Змінити її на LANGUAGE_CODE = 'uk', і адмін-панель стає українською.

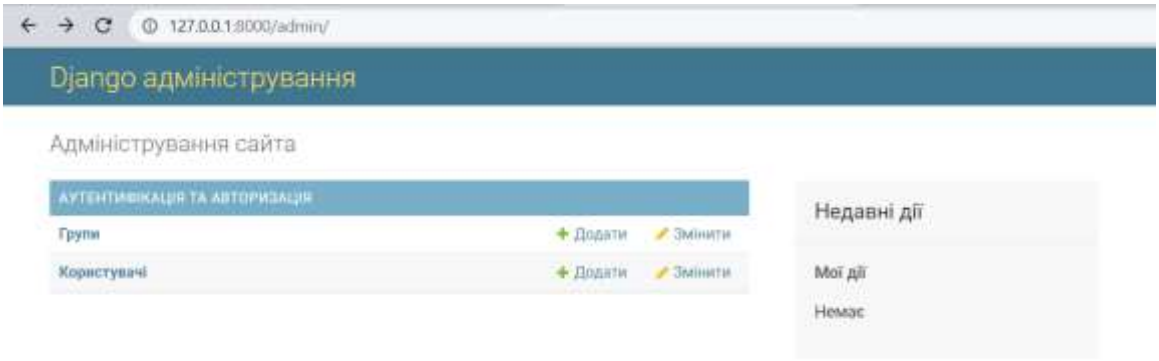


Рис. 3.10 - Українізована адмін-панель Django

Наступним кроком створюємо базовий додаток в проєкті django:

```
(venv) PS D:\KR\Energy\energy_system> python manage.py startapp bases
```

Рис. 3.11 - Створення базового додатку в проєкті django

Реєструємо створений додаток в settings.py

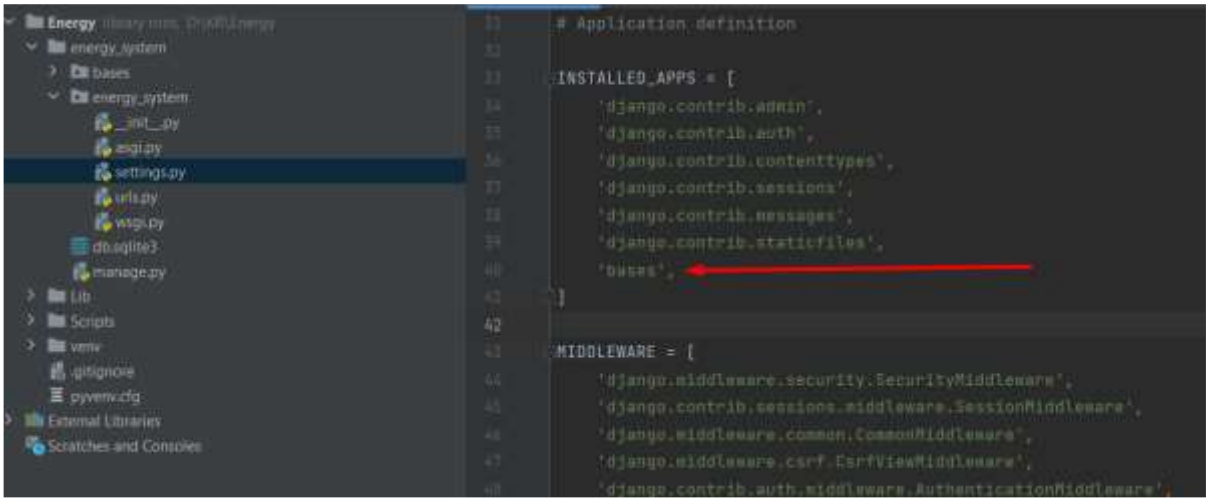


Рис. 3.12 - Реєстрація додатку bases в settings.py

Створюємо головну сторінку сайта:

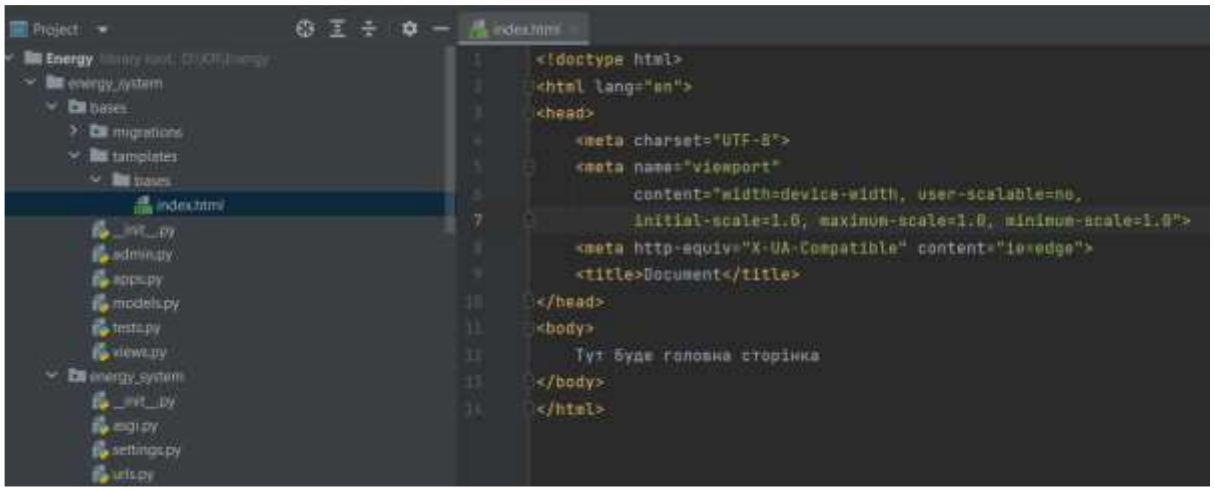


Рис. 3.13 - Створення головної сторінки сайта

В представленні bases/views.py описуємо функцію яка буде викликати файл index.html from django.shortcuts import render

def index(requests):

```
return render(requests, 'bases/index.html')
```

Зв’язуємо в energy_system/urls.py створену функцію з url адресою:

```
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
path('admin/', admin.site.urls),
path('', include('bases.urls')),
]
```

Запускаємо сервер і бачимо що підключився файл index.html

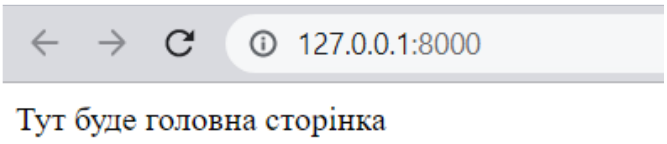


Рис. 3.13 - Запуск головної сторінки на сервері Django

Для того, щоб Django [47] коректно відображав зображення та підключав css стилі та js скрипти, необхідно провести деякі налаштування. Для цього в файлі setting.py необхідно дописати код:

```
STATIC_URL = 'static/'
STATICFILES_DIRS = [
BASE_DIR / "static",
"/var/www/static/",
]
```

В файли html потрібно буде дописувати код

```
{% load static %}
```

Потім, щоб вставити зображення, потрібно буде на писати код наприклад ось так [48]:

```
<img src="{% static 'images/logo.png' %}">
```

Аналогічно підключимо файли CSS:

```
<link href="{% static 'css/style.css' %}" rel="stylesheet">
```

та JS:

<script src="{% static 'js/all.js' %}"></script>

Також в папці static створюємо папки css, lib, js та img і переносимо всі файли в них із файлів нашої верстки.

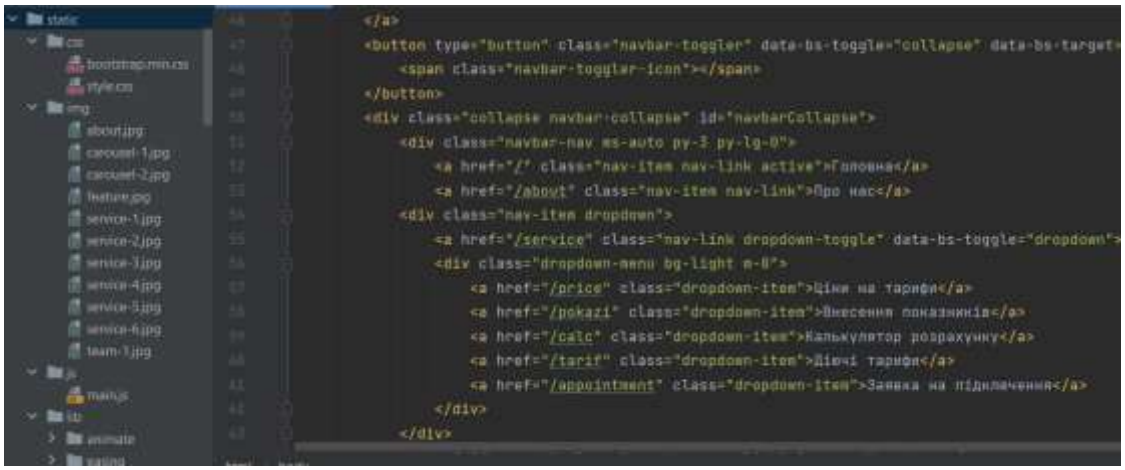


Рис. 3.14 - Архітектура папки static

В підсумку сайт виглядає так само як при верстці, але завдяки Django-серверу він функціонує так само, як сайт розміщений на хостингу.

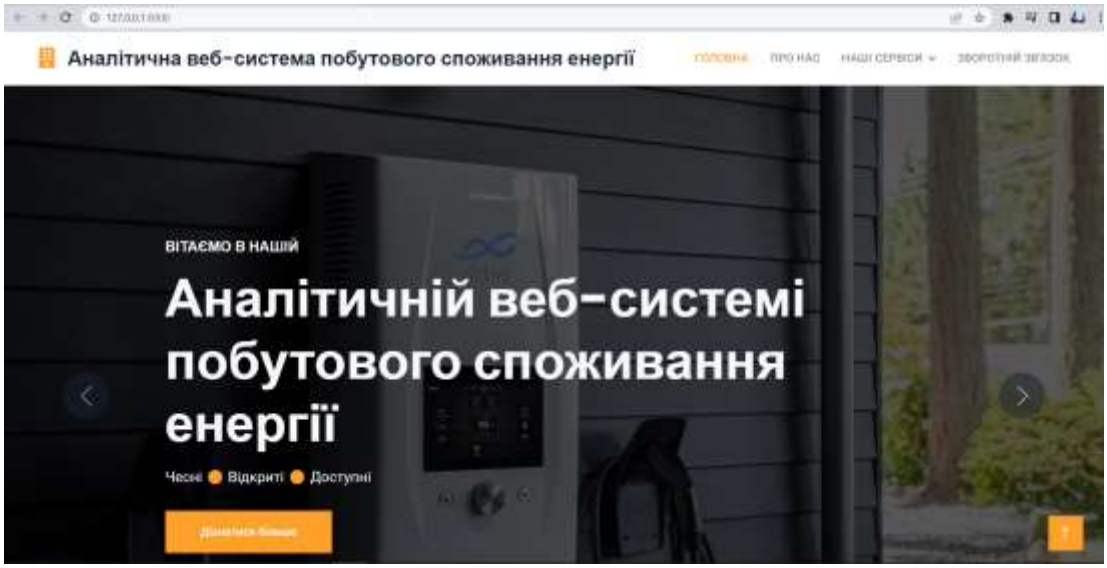


Рис. 3.15 - Вигляд головної сторінки на сервері Django

Далі створюються моделі (таблиці) в базі даних. Почнемо із моделі для сторінки price - цін на тарифи. Для цього в bases/models.py створимо клас Price, from django.db import models

```
class Price(models.Model):
name= models.TextField()
one_class = models.CharField(max_length=30)
two_class = models.CharField(max_length=30)
@staticmethod
def get_all_prices():
return Price.objects.all()
def __str__(self):
return self.name
```

В якому створимо три поля з назвою споживачів та цінами для першого та другого класів. Після чого створимо міграцію: python manage.py makemigrations

Та виконаємо її: python manage.py migrate

Для того, щоб клас відобразився в адмін-панелі реєструємо його в admin.py

```
from django.contrib import admin
from bases.models import Price
admin.site.register(Price)
```

Після чого наша модель відобразиться в панелі адміністратора

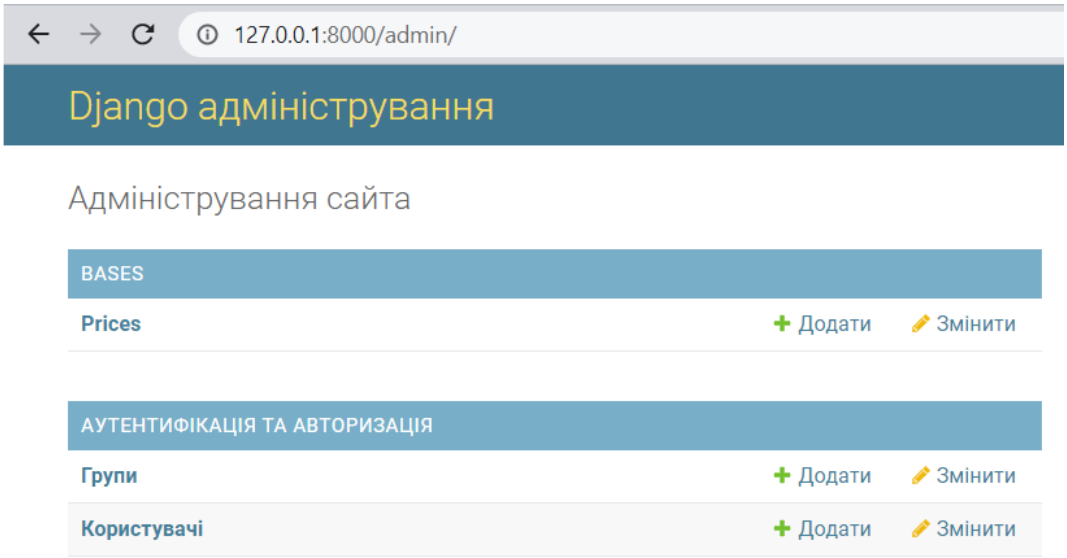


Рис. 3.16 - Відображення моделі Price в панелі адміністратора

Вносимо дані через адмін-панель

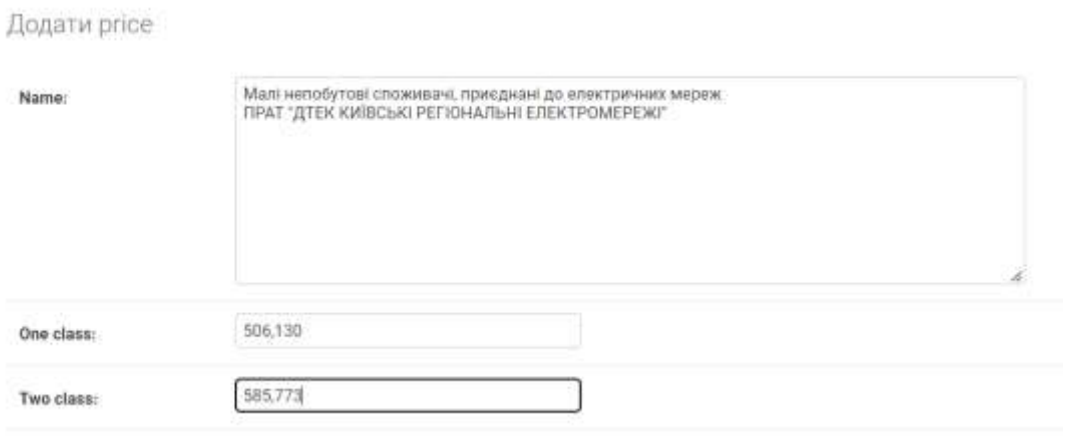


Рис. 3.17 - Додавання даних в модель Price в адмін-панелі

Далі в bases/views.py додаємо метод price

```
def price(request):
price = Price.objects.all()
return render(request, 'bases/price.html', {'price' : price})
```

і в файл price.html додаємо код

```
<table width="1200" class="table-bordered">
<tr>
<th>№</th>
<th class="text-center">Споживачі</th>
<th>1 клас</th>
<th>2 клас</th>
</tr>
{% for el in price_data%}
<tr>
<td>{{ el.id }}</td>
<td>{{ el.name }}</td>
<td>{{ el.one_class }}</td>
<td>{{ el.two_class }}</td>
</tr>
```

{% endfor %}

</table>

В результаті на сторінці з цінами на тарифи http://127.0.0.1:8000/price/ виведеться результат з бази даних.

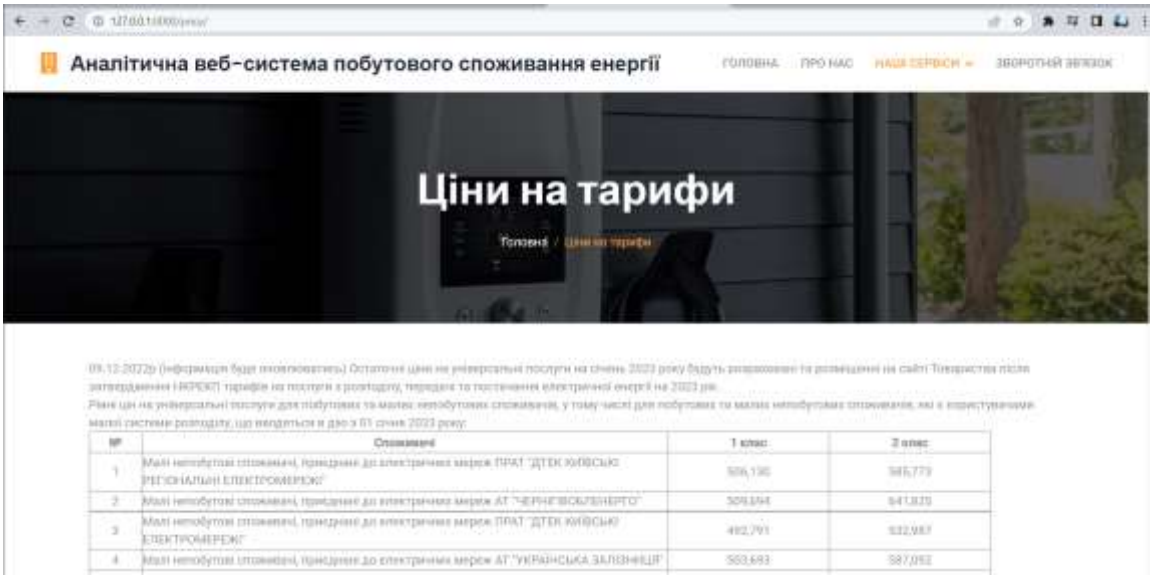


Рис. 3.18 - Вигляд сторінки з цінами на тарифи

До цього моменту ми розглядали приклад із введенням данних із адмін-панелі і виведенням цих даних на сторінці сайта. Далі розглянемо інший приклад із введенням даних із сторінок сайту, щоб дані записувалися в базу даних і відображалися в адмін-панелі. Зробимо це із сторінкою подачі заявки на підключення. Для цього в models.py створимо клас Appointment

class Appointment(models.Model):

first_name = models.CharField('Прізвище та ім\'я',

max_length=50)

email = models.CharField('email', max_length=50)

adresa = models.CharField('Ваша адреса', max_length=50)

more_information = models.TextField('Вкажіть бажаний час або

ще якусь додаткова інформація')

data_public = models.DateField('Дата подачі заявки')

def __str__(self):

```
return self.first_name
```

В bases створимо forms.py і заповнимо його даними для відображення нашої форми подачі заявки

```
from .models import Appointment
from django.forms import ModelForm, TextInput, Textarea, DateInput
class AppointmentForm(ModelForm):
class Meta:
model = Appointment
fields = ['first_name', 'email', 'adresa',
'more_information', 'data_public']
widgets = {
'first_name': TextInput(attrs={
'class': 'form-control border-0 col-sm-3 formfloating',
'placeholder': 'Прізвище та ім\'я',
'id': "gname",
}),
'email': TextInput(attrs={
'class': 'form-control border-0 col-sm-3 formfloating',
'placeholder': "Email",
'id': "gmail",
}),
'adresa': TextInput(attrs={
'class': 'form-control border-0 col-sm-3 formfloating',
'placeholder': 'Адреса підключення',
}),
'more_information': Textarea(attrs={
'class': 'form-control border-0 col-sm-3 formfloating',
'placeholder': 'Додаткові відомості',
'id': "message",
```

```
}),
'data_public': DateInput(attrs={
'class': 'form-control border-0 col-sm-3 formfloating',
'placeholder': 'Бажана дата підключення у форматі
01.01.2024',
'id': "date_appointment",
}),
}
```

В bases/views.py створюємо функцію appointment

```
def appointment(requests):
error = ''
if requests.method == 'POST':
form = AppointmentForm(requests.POST)
if form.is_valid():
form.save()
else:
error = 'Форма заповнена неправильно'
form = AppointmentForm()
data = {
'error': error,
}
return render(requests, 'bases/appointment.html', data)
```

Для відображення введених даних в адмін-панелі додамо код в admin.py

```
from django.contrib import admin
from bases.models import Price, Appointment
admin.site.register(Price)
@admin.register(Appointment)
class AppointmentAdmin(admin.ModelAdmin):
pass
```

Також змінемо appointment.html наступним чином

```
<div class="h-100 d-flex flex-column justify-content-center p-5">
<h1 class="mb-4">Подати заявку</h1>
<form method="post">
{% csrf_token %}<br>
{{form.first_name}}<br>
{{form.more_information}}<br>
{{form.data_public}}<br>
<span>{{error}}</span>
<div class="col-12">
<button class="btn btn-primary w-100 py-3"
type="submit">Подати заявку</button>
</div>
</form>
</div>
```

Після створення та проведення міграцій отримаємо форму подачі заявки за адресою http://127.0.0.1:8000/appointment/

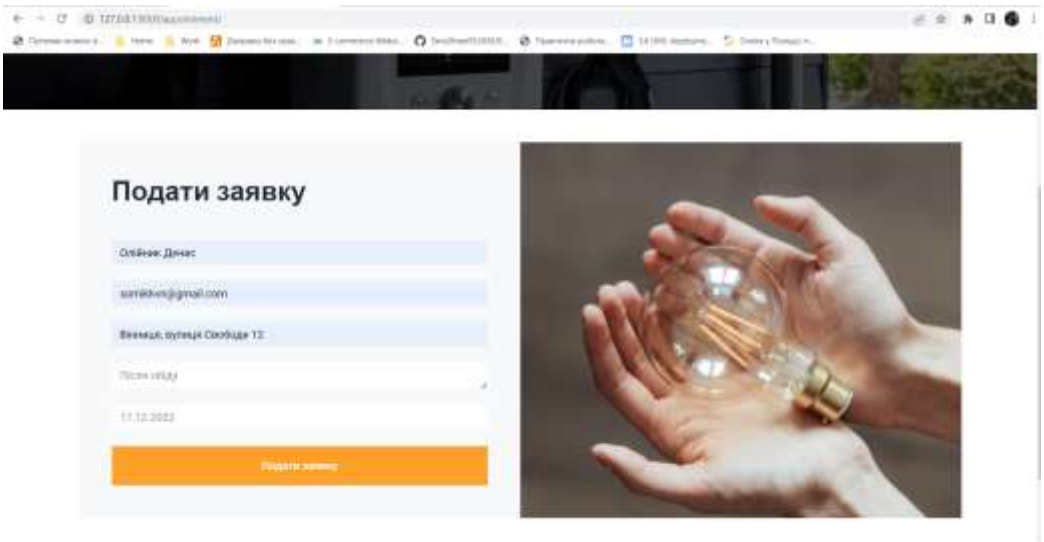


Рис. 3.19 - Вигляд сторінки «Подати заявку»

Всі інші сервіси створюються за аналогією розглянутих прикладів.

3.3 Технологічний аудит розробки

Для тестування верстки нашої веб-системи використаємо декілька інструментів. Спочатку перевіремо верстку нашого вебсайта за допомогою сервісу Markup Validation Service [49]. Розглянемо це на прикладі головної сторінки файлу index.html. Для цього перейдемо на сайт і оберемо вкладку “Validate by File Upload” після чого завантажимо зазначений файл і натиснемо кнопку “Check”.

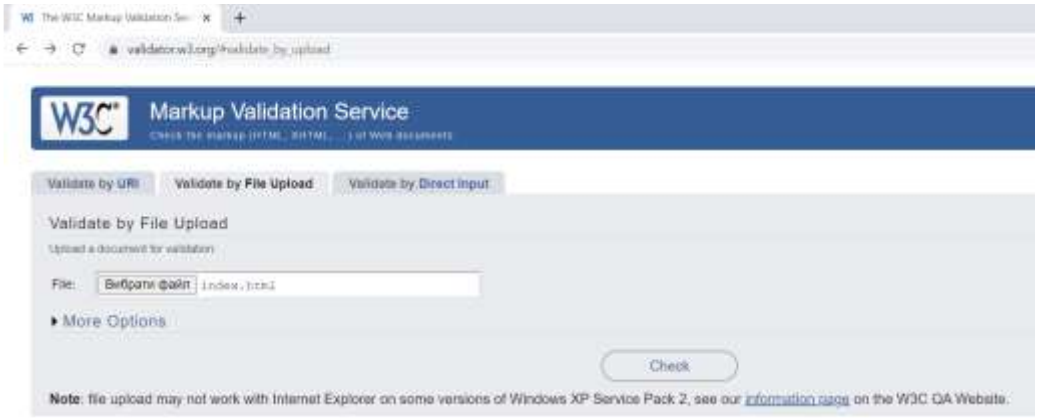


Рис. 3.20 - Перевірка верстки головної сторінки

В результаті отримуємо повідомлення «Перевірку документів завершено. Немає помилок або попереджень для показу».

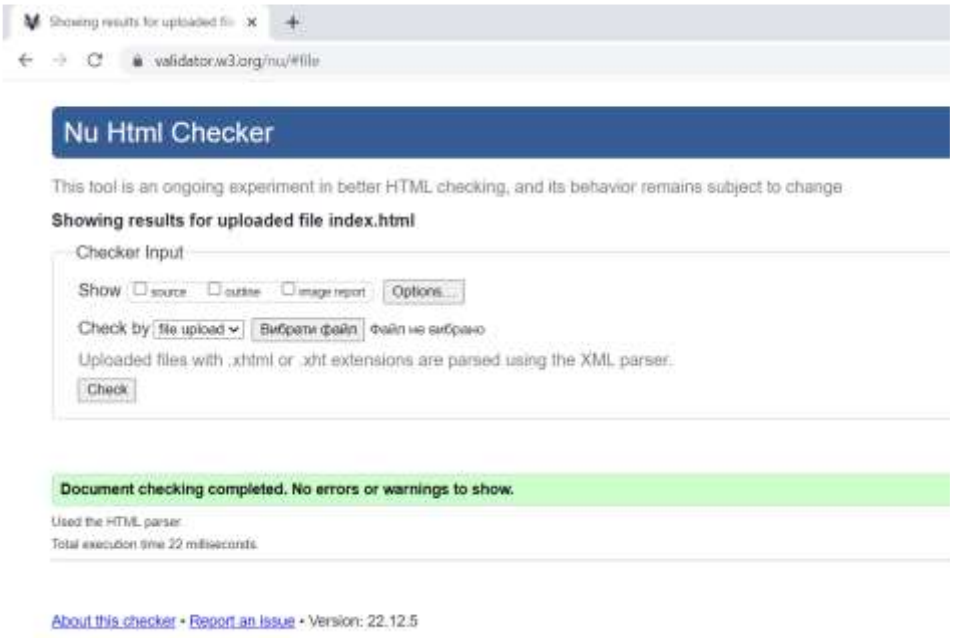


Рис. 3.21 - Результат перевірки верстки головної сторінки

Кожен вебсервіс ми перевірили і протестували, тому переходимо до тестування вебсайту в панелі розробника. Знову для прикладу покажемо це на прикладі головної сторінки.

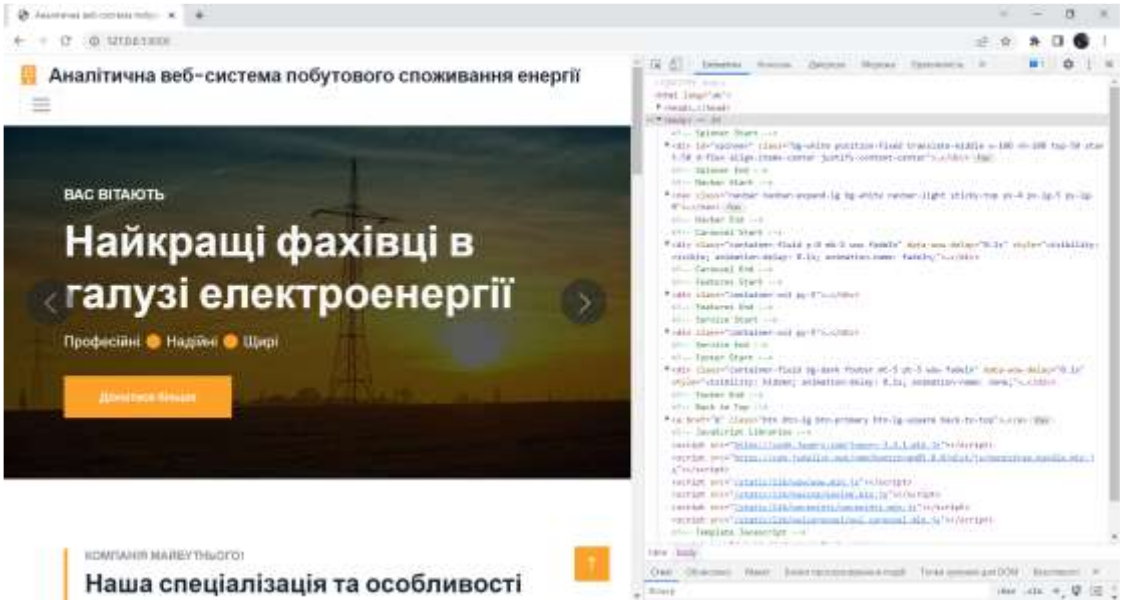


Рис. 3.22 - Результат перевірки на помилки в панелі розробника

Як бачимо, помилок знову не виявлено. Перевіримо також вебсайт на адаптивність, тобто його вигляд на різних мобільних пристроях. Спочатку перевіримо у режимі планшету.

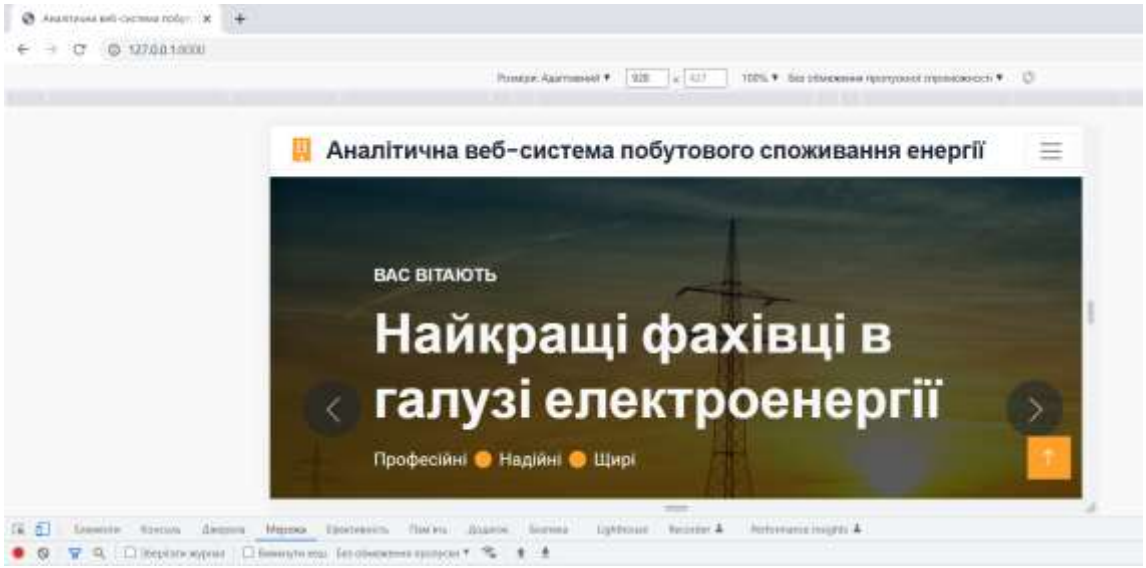


Рис. 3.23 - Результат перевірки вебсайту на адаптивність для планшету

Також перевіримо для мобільного пристрою, наприклад, для iPhone 12 Pro.

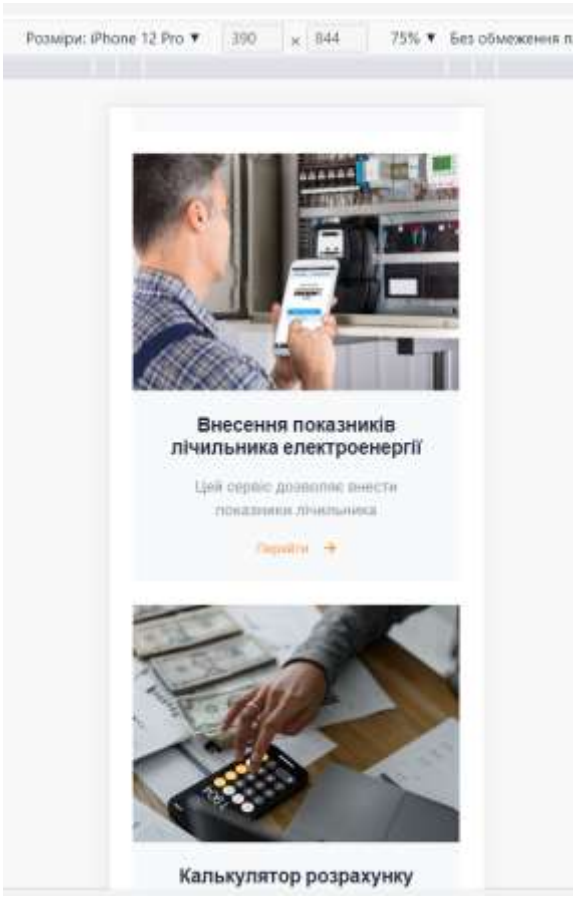


Рис. 3.24 - Результат перевірки вебсайту на адаптивність для iPhone 12 Pro

Знову впевнюємось, що сайт виглядає правильно. Отже, веб-система створена і перевірена, завантажуємо її на Github [50].

ВИСНОВОК

У кваліфікаційній роботі магістра розроблено інформаційну систему для аналізу споживання електроенергії у побутових умовах, яка являє собою аналітичну веб-систему, що виконує функцію отримання інформації та забезпечує аналіз побутового споживання електроенергії клієнтами.

Протягом виконання кваліфікаційної роботи було проаналізовано існуючі веб-системи побутового споживання електроенергії, визначено кращі сервіси і обрано їх за зразок, розроблено макет (прототип) веб-сайту та дизайн користувацького інтерфейсу, зроблено верстку вебсайту на основі розробленого дизайну, проведено аналіз необхідних для створення веб-сайту мов програмування, систем керування базами даних та фреймворків, створено власну веб-систему побутового споживання енергії. У роботі використовувались теоретичні методи та експериментальні дослідження, методи математичної статистики та статистичні вибірки, також були використані статистичні методи прогнозування. Новизна одержаних результатів полягає в розробці рекомендацій щодо впровадження веб-системи побутового споживання енергії.

Розроблена інформаційна система дозволяє удосконалити сервіси подачі показів лічильника, аналізувати тарифи за електроенергію, має калькулятор розрахунку вартості електроенергії, подання заявки на підключення та зворотній зв'язок. Результати дослідження впроваджено в процес організації зберігання, опрацювання, передавання інформації шляхом створення та використання вебсайту побутового споживання енергії.

Дослідивши різноманітні веб-сервіси аналогічного призначення зроблено висновок, що на даний момент не існує програмного продукту, що володів би повним набором необхідних функцій з максимальними якісними характеристиками, тому було поставлене завдання вдосконалити це у створюваному веб-сервісі. В процесі розробки веб-системи побутового споживання енергії, виконано огляд існуючих вебсистем, огляд існуючих мов програмування, огляд баз даних, огляд вебфреймворків, розроблено ER-діаграми, розроблено

схеми класів UML, обрано локальний веб-сервер для розгортання системи, розроблено дизайн інтерфейсу користувача, створено та протестовано код.

Після розгортання веб-сайту на реальному хостингу, можна виконувати інші налаштування, наприклад задіяти сервіс перевірки швидкості завантаження сайту, сервіс для перевірки навантаження на сайт, автоматичне кросбраузерне тестування сайта, сервіс для перевірки відкритих портів на будь-якому хостингу по IP, а також сервіси показу Ping, Traceroute, Whois.

Створену веб-систему можна використовувати в реальних проєктах, для інформування побутових споживачів електроенергії та аналітики даних.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. ТОВ "Київська обласна енергопостачальна компанія" [Електронний ресурс]. – Режим доступу: https://koec.com.ua/

2. АТ "Полтаваобленерго" [Електронний ресурс]. – Режим доступу: https://www.poe.pl.ua/

3. Калькулятор розрахунку оплати за електроенергію [Електронний ресурс]. – Режим доступу: https://dom.ria.com/uk/kalkulyatory-zkh/kalkulyator-tsenielektroenergii/

4. Діючі тарифи для населення ТОВ – Полтаваенергозбут [Електронний ресурс]. – Режим доступу: https://www.energo.pl.ua/

5. ПраТ «Закарпаттяобленерго» [Електронний ресурс]. – Режим доступу: https://zakarpat.energy/

6. Google відгуки про ПраТ«Закарпаттяобленерго» [Електронний ресурс]. –Режим доступу: https://www.google.com/search?q=Закарпаттяобленерго

7. Programming, scripting, and markup languages [Electronic resource] – https://insights.stackoverflow.com/survey/2021#key-territories-country

8. Django Підручник [Електронний ресурс] – Режим доступу до ресурса: https://w3schoolsua.github.io/django/index.html

9. IDE для професійної розробки на Python [Електронний ресурс] – Режим доступу до ресурса: https://www.jetbrains.com/pycharm/

10. JavaScript [Електронний ресурс] – Режим доступу до ресурса: https://developer.mozilla.org/en-US/docs/Web/JavaScript

11. Photoshop - Довідка й навчальні посібники [Електронний ресурс] – Режим доступу до ресурса: https://helpx.adobe.com/ua/pdf/photoshop_reference.pdf

12. The Python Tutorial [Електронний ресурс] – Режим доступу до ресурса: https://docs.python.org/uk/3/tutorial/index.html.

13. Query бібліотека JavaScript [Електронний ресурс] – Режим доступу до ресурса: https://jquery.com/

14. Query Підручник [Електронний ресурс] – Режим доступу до ресурса: https://w3schoolsua.github.io/jquery/index.html#gsc.tab=0

15. Вебсервіс для спільної розробки програмного забезпечення [Електронний ресурс] – Режим доступу до ресурса: https://github.com/

16. Завантажити Python. Офіційний сайт [Електронний ресурс] – Режим доступу до ресурса: https://www.python.org/

17. Інформатика (рівень стандарту): підручник для закладів загальної середньої освіти України / Й.Я. Ривкінд, Т.І. Лисенко, Л.А. Чернікова, В.В. Шакотько.- Київ: “Генеза”, 2018.- 144 с.

18. Кольори в HTML [Електронний ресурс] – Режим доступу до ресурса: https://html-css.co.ua/html/color-in-html/

19. Мова розмітки гіпертексту HTML [Електронний ресурс] – Режим доступу до ресурса: https://uk.wikipedia.org/wiki/HTML

20. Навчальні матеріали: Python [Електронний ресурс] – Режим доступу до ресурса: http://www.matfiz.univ.kiev.ua/pages/13.

21. Основи CSS [Електронний ресурс] – Режим доступу до ресурса: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics

22. SQL Підручник [Електронний ресурс] – Режим доступу до ресурса: https://w3schoolsua.github.io/sql/index.html#gsc.tab=0

23. Основні відомості про шари Photoshop [Електронний ресурс] – Режим доступу до ресурса: https://helpx.adobe.com/ua/photoshop/using/layerbasics.html

24. Підручник з HTML та CSS [Електронний ресурс] – Режим доступу до ресурса: https://htmlbook.at.ua/news/tutorial_css/1-0-2

25. Плугар І.В. Python з нуля [Електронний ресурс] – Режим доступу до ресурса: https://www.youtube.com/c/IoanPlugar_inf

26. Путівник мовою програмування Python. [Електронний ресурс] – Режим доступу до ресурса: https://pythonguide.rozh2sch.org.ua/

27. Фреймворк Bootstrap для створення адаптивних сайтів [Електронний

ресурс] – Режим доступу до ресурса: https://getbootstrap.com/docs/5.0/getting-started/introduction/

28. Фреймворк Django [Електронний ресурс] – Режим доступу до ресурса: https://www.djangoproject.com/

29. Вікіпедія [Електронний ресурс] – Режим доступу до ресурса: https://uk.wikipedia.org/wiki/модель_сутність-зв'язок

30. Елементи UML [Електронний ресурс] – Режим доступу до ресурса: https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-elements.html

31. Автоматизація прототипування інформаційних систем [Електронний ресурс] – Режим доступу до ресурса: https://posibniki.com.ua/postavtomatizaciya-prototipuvannya-informaciynih-sistem

32. Що таке архітектура веб-сайтів? [Електронний ресурс] – Режим доступу до ресурса: https://uk.theastrologypage.com/website-architecture

33. Markup Validation Service [Електронний ресурс] – Режим доступу https://validator.w3.org/

34. PHP: Hypertext Preprocessor [Електронний ресурс] – Режим доступу https://www.php.net/

35. Розділ мови PHP [Електронний ресурс] – Режим доступу https://metanit.com/php/

36. MySQL [Електронний ресурс] – Режим доступу https://www.mysql.com/

37. MySQL Матеріал з Вікіпедії — вільної енциклопедії [Електронний ресурс] – Режим доступу https://uk.wikipedia.org/wiki/MySQL

38. PostgreSQL [Електронний ресурс] – Режим доступу https://www.postgresql.org/

39. What is PostgreSQL? [Електронний ресурс] – Режим доступу https://aws.amazon.com/rds/postgresql/what-is-postgresql/

40. SQLite [Електронний ресурс] – Режим доступу https://www.sqlite.org/index.html

41. SQLite tutorial [Електронний ресурс] – Режим доступу https://www.sqlitetutorial.net/

42. MongoDB: The Developer Data Platform [Електронний ресурс] – Режим доступу https://www.mongodb.com/home

43. MongoDB Tutorial [Електронний ресурс] – Режим доступу https://www.w3schools.com/mongodb/

44. SQL Server у локальному середовищі [Електронний ресурс] – Режим доступу https://www.microsoft.com/ru-ru/sql-server/sql-server-downloads

45. Introducing Microsoft SQL Server 2019 [Електронний ресурс] – Режим доступу https://clouddamcdnprodep.azureedge.net/gdc/gdcJivzXl/original

46. CherryPy — A Minimalist Python Web Framework [Електронний ресурс] –Режим доступу https://docs.cherrypy.dev/en/latest/

47. Flask’s documentation [Електронний ресурс] – Режим доступу https://flask.palletsprojects.com/en/2.2.x/

48. Welcome to Pyramid, a Python Web Framework [Електронний ресурс] – Режим доступу https://trypyramid.com/

49. Web2py Web Framework [Електронний ресурс] – Режим доступу http://www.web2py.com/

50. Use Chrome Developer Tools to check tags [Електронний ресурс] – Режим доступу https://support.google.com/campaignmanager/answer/2828688?hl=en