МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Розробка веб-додатку обліку електроенергії підприємства з використанням asp.net**

Виконав: студент 5 курсу, групи 6.1219-пзс-з
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
В. В. Кулаков
(ініціали та прізвище)
Керівник доцент, к. т. н. Н. П. Полякова
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дискус»
Р.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____перший (бакалавський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Програмне забезпечення систем_____
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Тетяна КРИТСЬКА
“_01_” __березня_____2024_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

_____Кулакову Владиславу Валерійовичу_____
(прізвище, ім’я, по батькові)

1.Тема роботи Розробка веб-додатку обліку електроенергії підприємства з використанням asp.net
керівник роботи _____Полякова Наталія Петрівна, доцент, к.т.н __
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від ___26.12.2023 № 2212-с_____

2. Строк подання студентом кваліфікаційної роботи_____21.05.2024_____
3. Вихідні дані кваліфікаційної роботи бакалавра
- комплект нормативних документів;
- технічне завдання до роботи.

4. Зміст розрахунково – пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- аналіз сучасних технологій для створення клієнтської частини Веб застосунків;
- проєктування Веб-застосунку;
- програмна реалізація застосунку.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
_____слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	25.01 – 29.01.24	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	01.02 – 06.02.24	виконано
3	Аналіз існуючих методів рішення	07.02 – 11.02.24	виконано
4	Огляд та збір літератури стосовно теми кваліфікаційної роботи	12.02 – 26.02.24	виконано
5	Огляд та аналіз існуючих рішень та аналогів	27.02 – 29.02.24	виконано
6	Вивчення засобів asp.net для розробки Веб-застосунку	01.03 – 08.03.24	виконано
7	Визначення функціоналу застосунку, що розроблятиметься з використанням засобів asp.net	09.03 – 17.03.24	виконано
8	Узгодження подальших дій з науковим керівником	18.03.24	виконано
9	Проєктування бази даних та створення asp.net mvc-проєкту застосунку	19.03 – 27.03.24	виконано
10	Проєктування та конструювання застосунку із використанням засобів asp.net mvc	28.03 – 29.04.24	виконано
11	Програмна реалізація застосунку та його апробація	30.04 – 06.05.24	виконано
12	Оформлення звіту	07.05 – 24.05.24	виконано
13	Оформлення презентації. Отримання рецензій від опонентів	25.05 – 27.05.24	виконано

Студент _____ В.В. Кулаков
(підпис) (прізвище та ініціали)

Керівник роботи _____ Н.П. Полякова
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок – 74,

Рисунків – 10,

Джерел – 10.

Кулаков В. В. Розробка веб-додатку обліку електроенергії підприємства з використанням ASP.NET 121 «Програмне забезпечення систем» / наук. Керівник Н. П. Полякова. Запоріжжя : ЗНУ, 2024, 74 с.

Мета полягає у створенні простого у використанні та ергономічного, з точки зору тривалого користування на робочому місці, веб-додатку обліку електроенергії. У процесі розробки була розглянута проблема контролю процесів виробництва, споживання електроенергії та діагностики неполадок елементів системи обліку і вузлів виробничого процесу. А також проблема застарівання обладнання та програмного забезпечення. В результаті, засобами ASP.NET MVC був створений веб-додаток для системи обліку електроенергії замість застарілого десктопного додатку. Розроблений застосунок містить в собі три шари (шар доступу до даних, шар бізнес-логіки і шар представлення). Шар доступу до даних відповідає за обробку бінарних файлів із показниками споживання та за формування (методом Code First) і використання, в якості моделі, сутностей бази даних. Шар представлення відповідає за візуалізацію показників споживання у вигляді діаграм, графіків та таблиць. Шар бізнес-логіки відповідає за взаємодію між шаром представлення та шаром доступу до даних, а саме, за обробку запитів від шару представлення та передачу у нього даних від шару доступу до даних. Стилізація додатку, а саме: гама кольорів, розташування елементів управління, розмір і функціональність області діаграм і таблиць зроблені таким чином, щоб користувачу було зручно використовувати додаток тривалий час. Таким чином, результат виконання роботи реалізував положення, зазначені у меті.

Ключові слова: *веб-додаток, шар, сутність, code first, фідер, група.*

ABSTRACT

Pages - 74,

Figures - 10,

References - 10.

Kulakov V.V. Development of a web application for electricity metering of an enterprise using ASP.NET 121 ‘Systems software’ / supervisor. Supervisor N. P. Polyakova. Zaporizhzhia: ZNU, 2024, 74 p.

The aim is to create an easy-to-use and ergonomic, in terms of long-term use in the workplace, web-based electricity metering application. The development process addressed the problem of controlling production processes, electricity consumption and diagnosing malfunctions of metering system elements and production process units. We also addressed the problem of hardware and software obsolescence. As a result, a web application for the electricity metering system was created using ASP.NET MVC to replace the outdated desktop application. The developed application contains three layers (data access layer, business logic layer and presentation layer). The data access layer is responsible for processing binary files with consumption data and for creating (using the Code First method) and using database entities as a model. The presentation layer is responsible for visualising consumption indicators in the form of charts, graphs and tables. The business logic layer is responsible for the interaction between the view layer and the data access layer, namely, for processing requests from the view layer and transferring data to it from the data access layer. The styling of the application, namely: the colour scheme, the location of controls, the size and functionality of the diagrams and tables area are made in such a way that the user can use the application conveniently for a long time. Thus, the result of the work has realised the provisions specified in the objective.

Keywords: *web application, layer, entity, code first, feeder, group*.

ЗМІСТ

ВСТУП

Для підприємства, виробничій процес якого передбачає використання великих обсягів електроенергії, питання обліку є нагальним. Зазвичай такі підприємства мають декілька точок під’єднання до мережі, на яких встановлені лічильники комерційного обліку електроенергії. Показання таких лічильників збираються та обробляються сертифікованими системами комерційного обліку електроенергії.

Проте, комерційний облік електроенергії не передбачає збір даних про споживання електроенергії по конкретним підрозділам та агрегатам. Для цього підприємству потрібно встановити окреме обладнання для збору та обробки даних про споживання.

Для цих цілей існують системи технічного обліку електроенергії. Встановлення такої системи дозволяє збирати дані про споживання електроенергії конкретними підрозділами підприємства, частинами підрозділів та/або окремого обладнання.

Детальне спостереження за споживанням електроенергії окремими частинами підприємства, агрегатами та лініями надають підприємству можливість контролювати виробничі процеси, проводити діагностику неполадок, планувати та корегувати витрати електроенергії, що призводить до підвищення ефективності виробництва, зменшення фінансових витрат та зменшення необхідного обсягу часу на локалізацію та усунення неполадок обладнання, що неможливо уникнути в процесі виробництва.

Враховуючи кількість підрозділів та обладнання, кількість точок збору даних про споживання може сягати сотень одиниць. Для спостереження та обробки даних з такої кількості споживачів потрібен застосунок, який забезпечить зручне групування та візуалізацію зібраних даних.

Глосарій

Фідер — електричний ланцюг (лінія передачі), яка з’єднує конкретного споживача з певним каналом контролера.

Контролер збору даних — пристрій для збору показників споживання електроенергії певних споживачів, що знаходяться на іншому кінці фідеру.

Канал контролеру — місце з’єднання (клема) до якої під’єднується фідер. В контексті файлу бінарних даних з показниками — місце, де записані показники споживання конкретного фідеру.

Група — абстракція, створена для об’єднання фідерів у певні групи, в залежності від їх розташування або призначення.

Асоціація — абстракція, створена для об’єднання груп у певні групи груп. Конструкція використовується для зручного виводу наборів груп, які мають бути візуалізовані разом.

Наскрізний номер каналу — порядковий номер клеми, відносно клем всіх контролерів.

Миттєві значення (Миттєві дані) — дані за останню хвилину по певним споживачам чи групам споживачів.

Півгодинний файл — файл, що є результатом збору даних головним контролером за півгодини. Контролери опитують фідери кожну хвилину і передають дані головному контролеру, який записує їх похвилинно у файл. Коли сформовано 30 хвилинних наборів головний контролер починає запис нового файлу.

1.1 Опис проблеми

1.1.1 Підприємство

Замовником застосунку для системи обліку електроенергії є АТ «Нікопольський завод феросплавів». Основним продуктом є феросплави, для виплавки яких використовуються пічні агрегати, енергоносієм для яких є електроенергія. Також, завод має підрозділи, які забезпечують його матеріалами, що беруть участь у технічному процесі виробництва цільового продукту. Це забезпечує підприємству певний ступінь автономності. Проте, також, збільшує кількість підрозділів і, відповідно, електричного обладнання. Підприємство споживає колосальні об’єми електроенергії [1, с. 336].

1.1.2 Проблема

Велика кількість технічних процесів та обладнання, яке живиться від електромережі потребує контролю споживання електроенергії. Як наслідок, на підприємстві встановлено близько трьохсот лічильників електроенергії. Дані з яких збираються за допомогою автоматизованої системи збору даних, а обробляються і візуалізуються за допомогою десктопного застосунку, який має клієнтську та серверну частини. Автоматизована система збору даних була розроблена у 90-х роках 20-го століття, силами підрозділу «Лабораторія метрології» Центральної Лабораторії Автоматизації та Механізації, остання версія програмного забезпечення для цієї системи розроблена і впроваджена у 1999 році. Проте, на сьогоднішній день це обладнання технічно застаріло, а комплектуючі, які потрібні для обслуговування відсутні на ринку. Програмне забезпечення, яке використовується для обробки та візуалізації також морально і технічно застаріло і потребує оновлення. Якщо обладнання системи збору даних або застосунок до цієї системи вийдуть з ладу, стане неможливо вести облік по підрозділам і здійснювати контроль виробничих процесів.

1.1.3 Шляхи вирішення

Для вирішення вищевказаної проблеми є два шляхи:

1. Пошук підрядника який здійснить проектування, монтаж та запуск системи збору даних, розробить застосунок для візуалізації даних і буде супроводжувати систему збору даних та застосунок, виконуючи обслуговування та, за необхідності, ремонт.
2. Самостійна розробка оновленого обладнання для системи збору даних та застосунку для візуалізації зібраних даних, з використанням сучасних комплектуючих для системи збору та сучасних технологій для застосунку.

1.2 Постановка задачі

Силами підрозділу «Лабораторія метрології» Центральної Лабораторії Автоматизації і Механізації було виконано розробку нового обладнання системи збору даних та прикладного програмного забезпечення для нього, яке формує бінарні файли. Тому задача полягає у тому, щоб розробити застосунок для обробки та візуалізації зібраних даних.

Нижче представлено список неформальних вимог до застосунку, що відповідає за обробку та візуалізацію даних:

- Застосунок має реалізовувати модель існуючої системи збору даних і відповідні сутності.
- Виконувати читання та обробку файлів, що формує система збору даних нової версії.
- Мати функціонал розрахунків конкретних показників на основі вище згаданих файлів.
- Має працювати в браузері.
- Повинен працювати на наявному у підприємстві обладнанні.

- Налаштування клієнтського комп’ютеру для роботи з новим застосунком має бути простим і швидким.
- Зовнішній вигляд застосунку має враховувати досвід роботи з застосунком-попередником.
- Застосунок має надавати можливість візуалізації даних у різних зручних для сприйняття форматах (наприклад таблиці, графіки, діаграми).
- Застосунок має бути простим і вузькоспеціалізованим.

1.3 Огляд існуючих методів рішення

Аналогічні рішення, що конкурують

Було розглянуто наступні рішення і послуги з розробки та впровадження системи обліку електроенергії, які пропонують:

1. Компанія «ГРАНД ТЕСЛА» пропонує встановлення обладнання, а саме: лічильники, засоби збору та передачі даних. Та власне програмне забезпечення для візуалізації даних. Пропонується одночасне встановлення систем комерційного та технічного обліку, проведення монтажних та пусконалагоджувальних робіт. Особливості фірмового програмного забезпечення та його можливості не наводяться [2].
2. Кампанія «VIK-SOFT», автоматизована система обліку енергоресурсів «ЕнергоЦентр». Підтримує технічний та комерційний облік енергоресурсів, в тому числі й облік електроенергії. Підтримує різні типи приладів обліку, надає засоби для налаштування звітності та інструменти візуалізації даних [3].
3. Кампанія «Міротекс» пропонує програмний комплекс для побудови систем обліку та контролю споживання енергоресурсів "Електро облік". Комплекс може бути використаний для побудови інтегрованих систем обліку різних типів енергоресурсів (електроенергія, вода, газ, тепло та

ін.) за наявності в складі системи модулів опитування відповідних пристроїв [4].

Представлені вище системи не відповідають вимогам до застосунку. Хоча вони і надають широкий вибір функціоналу, але перевантажені непотрібними у контексті задачі можливостями, складні в розгортанні і налаштуванні.

2 ПРОЄКТ ПРОГРАМНОІ СИСТЕМИ

2.1 Вимоги до оточення

2.1.1 Вимоги до апаратного забезпечення

Основною задачею застосунку на стороні клієнту є відображення даних у вигляді графіків, діаграм та таблиць на наявному на підприємстві апаратному забезпеченні. Тому були визначені наступні вимоги до апаратного забезпечення користувача.

Мінімальне наявне апаратне забезпечення користувача:

- Одноядерний процесор з базовою тактовою частотою 1.8 GHz.
- Оперативна пам’ять ємністю 2 Gb.

Рекомендоване апаратне забезпечення користувача:

- Чотирьохядерний процесор з базовою тактовою частотою 2.5 GHz.
- Оперативна пам’ять ємністю 8 Gb.

Оскільки на сервері застосунок виконує аналіз даних різного обсягу та розрахунки, відправляє запроси до БД та отримує й обробляє відповіді, для різних користувачів одночасно, серверне апаратне забезпечення потребує кращої конфігурації, ніж клієнтське апаратне забезпечення.

Мінімальне наявне апаратне забезпечення серверу:

- Восьмиядерний процесор з базовою тактовою частотою 3.8 GHz.
- Оперативна пам’ять 16 Gb.
- Жорсткий диск ємністю 500 Gb.
- Інтернет з’єднання 100 Mbit.

Рекомендоване апаратне забезпечення серверу:

- Шістнадцятиядерний процесор з базовою тактовою частотою 4.5 GHz.
- Оперативна пам’ять ємністю 32 Gb.

- SSD ємністю 2 TB.
- Інтернет з’єднання 1 Gbit.

2.1.2 Вимоги до програмного забезпечення

Вимоги до програмного забезпечення користувача:

- Операційна система Windows XP SP2 і вище.
- Встановлений веб браузер

Вимоги до програмного забезпечення сервера:

- Операційна система Windows Server 2008 R2 і вище.
- MS SQL Express 2016 SP2.
- Веб-сервер IIS 7.0 і вище.

2.1.3 Вимоги до користувачів

Внутрішній устрій застосунку передбачає наявність тільки однієї ролі, тому вимоги до користувачів застосунку передбачають наступні два пункти:

- Рівень володіння ПК.
- Рівень володіння предметною областю, для потреб якої створено застосунок.

Рівень володіння ПК

Користувач має володіти навичками роботи з комп’ютером, як мінімум, на середньому рівні. Знатися у використанні маніпуляторів (миші, клавіатури). Бажано, щоб користувач вмів користуватися функціональними клавішами та знав комбінації функціональних клавіш клавіатури з функціями миші. Вміти користуватися браузером та його основними функціями.

Рівень володіння предметною областю

Застосунок не передбачає обов’язкової наявності певного фаху у користувача, але необхідно розумітися у структурі підприємства, виробничих

процесах, агрегатах, а також у структурі мережі лічильників. Розумітися у поняттях «фідер», «група», «миттєві значення», «хвилинні дані», «півгодинні дані», «годинні дані». Знати і розуміти скорочення, які використовуються у назвах фідерів та груп, абревіатури підрозділів.

Доскональне розуміння всіх процесів, пов'язаних з обліком електроенергії у всіх підрозділах підприємства не обов'язкове, але вітається. Обов'язковим є розуміння процесів у власній предметній області або зоні відповідальності.

2.2 Архітектура системи

Для реалізації веб-застосунку до автоматизованої системи технічного обліку електроенергії було створено три шари, які реалізують тришарову архітектуру. Перший — шар представлення, другий — шар бізнес-логіки, третій — шар доступу до даних. На рисунку 1 зображено шари тришарової архітектури.

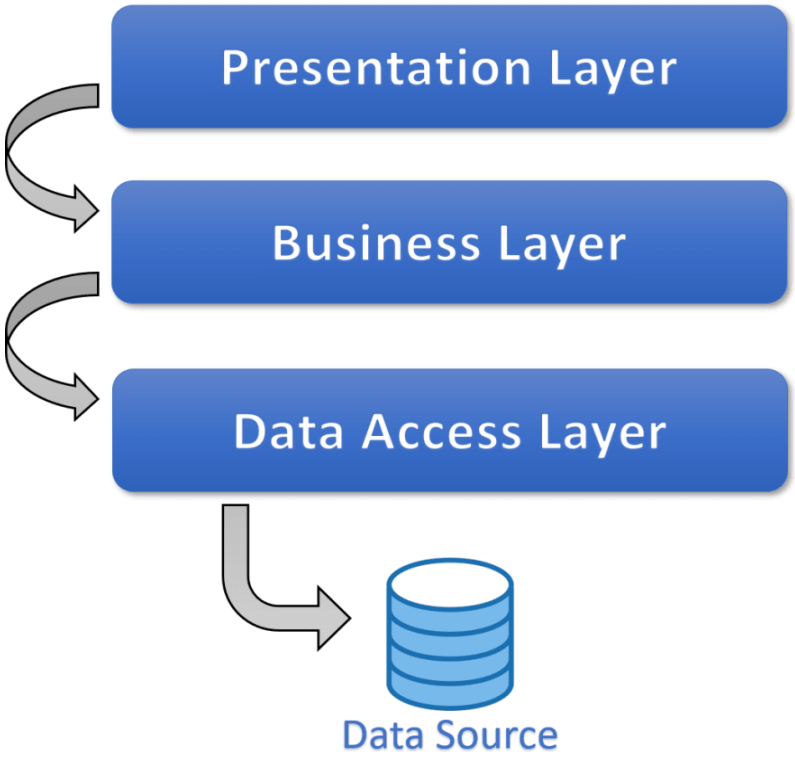

Рисунок *1 – Шари тришарової архітектури*

Шар представлення та шар бізнес-логіки реалізовані за допомогою ASP.NET MVC та винесені в окремий проєкт у рамках одного рішення. Другим проєктом у рішенні є бібліотека класів, що реалізує шар доступу до даних.

Проєкт, що є бібліотекою класів реалізує роботу з базою даних та з бінарними файлами, у яких зберігаються дані про споживання електроенергії на підприємстві.

Класи проєкту бібліотеки класів, які реалізують сутності для створення та роботи з БД також виступають і у ролі моделі для ASP.NET MVC проєкту [5, с.4].

2.3 Специфікація даних

2.3.1 Опис формату та/або структури даних

Реалізація певних методів контролерів передбачає передачу даних у представлення у форматі JSON.

Формат даних JSON (JavaScript Object Notation) — це текстовий формат обміну даними, який переважно використовується для обміну структурованими даними через мережу. Він базується на підмножині мови програмування JavaScript, яка визначена у стандарті «ECMA-262 3rd Edition - December 1999».

Даний текстовий формат зручний для написання та читання, як для людини, так і для комп’ютеру. Також він не залежить від мови реалізації. Ці властивості роблять його дуже зручним форматом для передачі даних.

JSON заснований на двох структурах даних:

- Колекція пар ключ/значення. У різних мовах, ця концепція реалізована як об'єкт, запис, структура, словник, хеш, іменований список або асоціативний масив.

- Упорядкований список значень. У більшості мов це реалізовано як масив, вектор, список або послідовність.

Використання даного формату обміну даними у застосунку обумовлено його виключною зручністю та універсальністю [6, с.6].

2.4 Функціональні вимоги

Додаток повинен дозволяти користувачу:

- Переглядати стовпчасті діаграми поточного споживання електроенергії (стовпчасті діаграми миттєвих даних) груп та фідерів.
- Переглядати стовпчасті діаграми півгодинних показників за добу по групам.
- Переглядати лінійний хвилинний графік за добу по обраній групі.
- Переглядати лінійний хвилинний графік за добу по набору обраних груп.
- Переглядати лінійний хвилинний графік за добу по набору фідерів обраної групи.
- Переглядати таблицю годинних та півгодинних даних за добу по обраній групі.
- Переглядати таблицю добових даних за місяць по наборам груп та фідерів.

На рисунку 2 зображено діаграму використання застосунку.

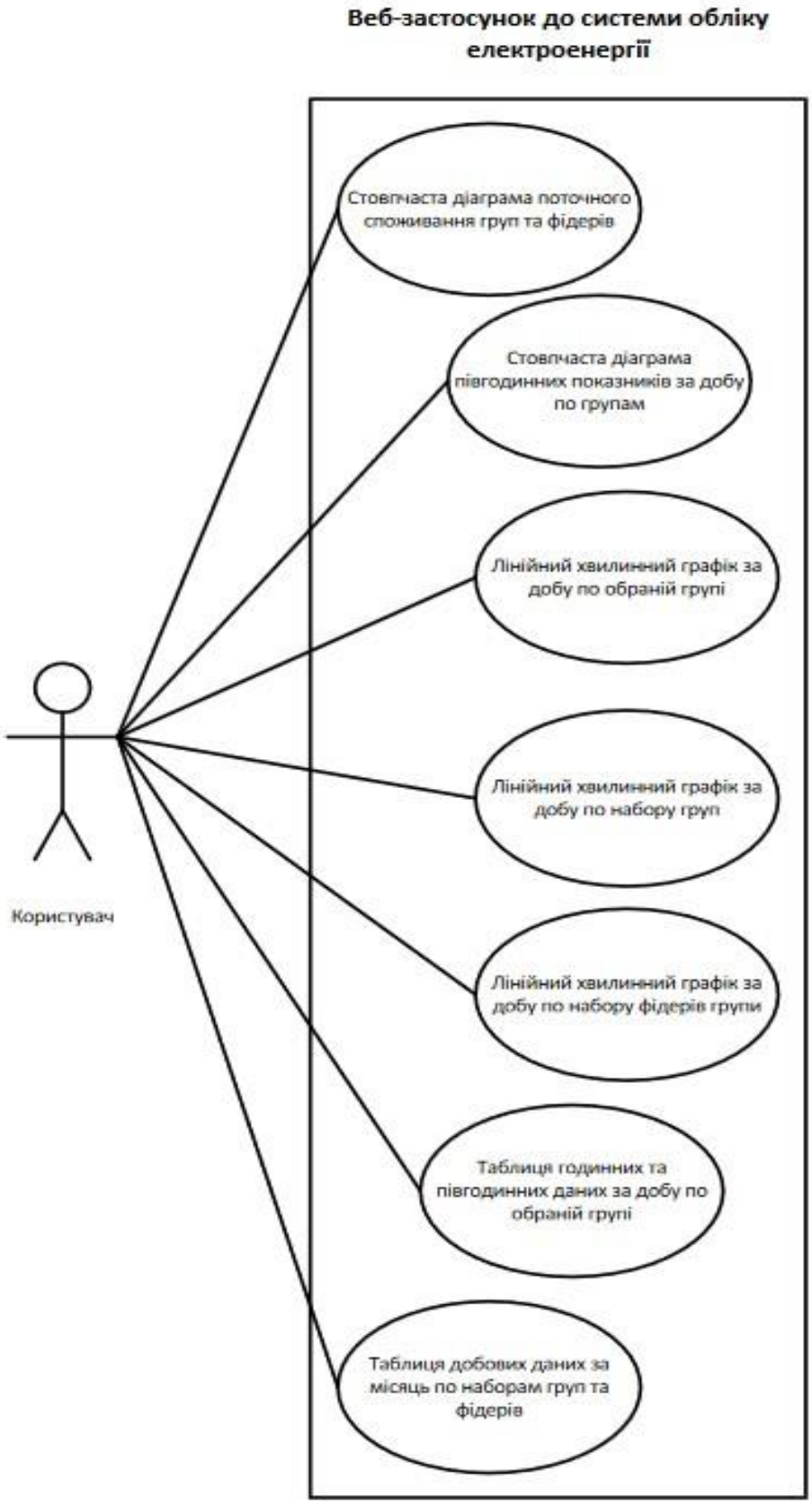

Рисунок *2* – *Діаграма використання застосунку*

2.5 Вимоги до інтерфейсу

Застосунок має реалізовувати Web-інтерфейс, що матиме зручний, ергономічний дизайн, який дозволить користувачам працювати з ним на протязі всієї робочої зміни без серйозного впливу на органи сприйняття.

Оскільки застосунок має попередній варіант, до інтерфейсу якого звикли користувачі, дизайн нового застосунку має бути реалізований з урахуванням необхідності адаптації інтерфейсу до досвіду роботи користувачів з минулою версією застосунку. Елементи управління (кнопки, пункти меню) мають бути розташовані схожим чином, як у застосунку минулої версії.

На рисунку 3 можна побачити макет головної сторінки застосунку.

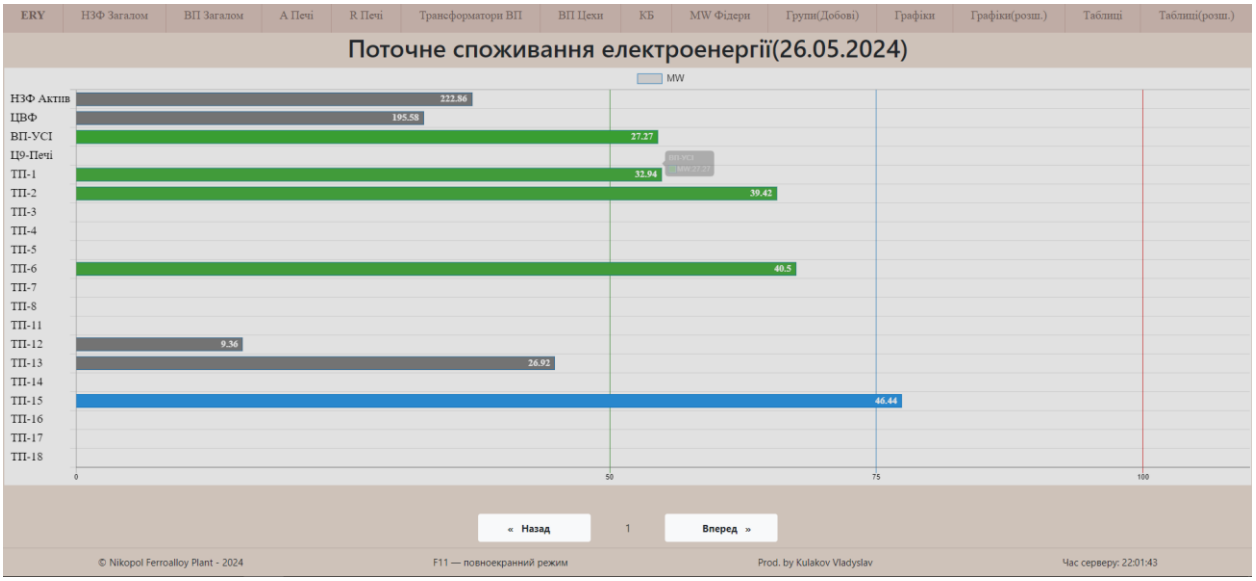

Рисунок *3 –Макет головної сторінки*

2.6 Проєкт програмної системи

2.6.1 Засоби реалізації

До засобів реалізації входять наступні складові:

- Мова програмування.
- Середовище розробки.
- Бібліотеки.
- СКБД.

Розглянемо кожну складову.

Мова програмування

Як потенціальні мови розробки серверної частини розглядалися мови Java та C#. Обидві мови мають свої сильні та слабкі сторони.

Мова Java

Переваги мови Java:

- Кросплатформність завдяки JVM.
- Великий вибір середовищ розробки.
- Велика кількість відкритих бібліотек та фреймворків.

Недоліки мови Java:

- Можлива нижча продуктивність.
- Вимоги до оперативної пам’яті.

Мова C#

Переваги мови C#:

- Великий набір бібліотек та інструментів платформи .NET.
- Легкість розробки для платформи Windows.
- Потужне середовище розробки з великою кількістю інструментів і плагінів.
- Вища продуктивність та оптимізація.

Недоліки мови C#:

- Портативність.
- Нижча популярність за межами екосистеми Microsoft.

Через потенційно вищу продуктивність та оптимізацію використання оперативної пам’яті, а також, враховуючи, що на всі комп’ютери підприємства встановлено ті чи інші випуски операційної системи Windows, задля використання можливостей екосистеми Microsoft, було обрано мову C#

Для клієнтської частини застосунку, в обох випадках розглядалася зв’язка наступних технологій:

- Мова розмітки HTML.
- Мова сценаріїв JavaScript [7].
- Каскадні таблиці стилів (CSS).

Середовище розробки

Як потенціальні середовища розробки розглядалися Microsoft Visual Studio для C# та IntelliJ IDEA для Java. Враховуючи вибір мови програмування та широкі можливості середовища розробки, було обрано Microsoft Visual Studio.

Бібліотеки

Для розробки серверної частини не планується використовувати бібліотеки, відмінні від бібліотек екосистеми .NET Framework. Проте, для побудови графіків і діаграм клієнтської частини було розглянуто дві відкриті JavaScript-бібліотеки для побудови графіків та діаграм, а саме:

- Бібліотека D3.js.
- Бібліотека Chart.js.

Перевагою бібліотеки D3.js є висока гнучкість та потужні можливості кастомізації, проте, через широкі можливості, ця бібліотека складна у використанні та потребує багато часу на вивчення.

Перевагою бібліотеки Chart.js є простота у вивчені та використанні, вбудована інтерактивність графіків та діаграм, підтримка популярних графіків і діаграм та проста й лаконічна документація.

Після огляду можливостей обох бібліотек, враховуючи час, який виділено на розробку застосунку та рівень складності вивчення та використання, а також достатній для заявлених задач рівень функціоналу, було обрано бібліотеку Chart.js [8].

СКБД

В якості СКБД для застосунку розглядалися дві реляційні СКБД, а саме:

- MySQL.
- MS SQL Express

Перевагами MySQL є кросплатформність, легкість у використанні та велика спільнота розробників, що забезпечує достатню кількість інформаційних ресурсів та підтримку, проте масштабування може бути складним без додаткових інструментів, а деякий функціонал не поставляється в рамках відкритої ліцензії.

Перевагами MS SQL Express є відмінна інтеграція з екосистемою Microsoft, потужні засоби забезпечення безпеки даних, наявність розширеного функціоналу навіть у безкоштовній версії та офіційна підтримка компанії Microsoft. Проте, безкоштовна версія має обмеження на розмір бази даних та використання ресурсів. Також MS SQL Express може бути складним у вивченні для тих, хто не мав досвіду роботи з вищезгаданою системою.

Враховуючи питання безпеки даних, наявність підтримки та документації, досвід роботи, а також інтеграцію з екосистемою Microsoft було обрано СКБД MS SQL Express.

3 МОДУЛІ І АЛГОРИТМИ

Архітектура, яка описана в розділі 3 визначає структуру та організацію коду, що є важливим аспектом застосунку. Тому опис модулів, програмної системи буде наводитись з розбиттям на шари, до яких ці модулі належать.

3.1 Шар доступу до даних

3.1.1 Класи для роботи з базою даних

Для формування бази даних засобами Entity Framework, методом Code First [9, с.37], було створено класи, які представляють модель бази даних та визначають її сутності. Нижче наведено назви та короткий опис сутностей та їх полів.

1. Клас Fider — представляє сутність «фідер» у базі даних, поля якої описують конкретний фідер у базі даних. На лістингу 1 зображено поля сутності «фідер».

Лістинг *1 Сутність «Fider»*

```
public class Fider
    {
        public int Id { get; set; }
        public int PassThroughChannelNumber { get; set; }
        public string Name { get; set; }
        public string ElectricityTransformationParams { get;
set; }
        public float ElectricityTransformationCoefficient {
get; set;}
        public string VoltageTransformationParams { get; set;
}
```

```
        public float VoltageTransformationCoefficient { get;
set; }
        public int ImpulsesPerKilowattPerHour { get; set; }
        public float FinalTransformationCoefficient { get;
set; }
        public int MaxPower { get; set; }
        public string LoadType { get; set; }
        public string CounterName { get; set; }
        public string CounterNumber { get; set; }
        public string Description { get; set; }
        public virtual List<FiderGroup> FiderGroups { get;
set; }
    }
```

2. Клас Group — представляє сутність «група» у базі даних, поля якої описують конкретну групу у базі даних. На лістингу 2 зображено поля сутності «група».

Лістинг 2 *Сутність «Group»*

```
public class Group
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string ShortName { get; set; }
        public float MaxPower { get; set; }
        public DateTime DateOfLastRevision { get; set; }
        public string Description { get; set; }
        public virtual List<FiderGroup> FiderGroups { get;
set; }
        public virtual List<AssociationsOfGroups>
              AssociationsOfGroups { get; set; }
```

```
}
```

3. Клас Association — представляє сутність «асоціація» (група груп) у базі даних, поля якої описують конкретну асоціацію у базі даних. На лістингу 3 зображено поля сутності «асоціація».

Лістинг *3 Сутність «Association»*

```
public class Association
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string ShortName { get; set; }
        public string DisplayName { get; set; }
        public float MaxPower { get; set; }
        public DateTime DateOfLastRevision { get; set; }
        public string Description { get; set; }
        public virtual List<AssociationsOfGroups>
               AssociationsOfGroups { get; set; }

    }
```

4. Клас FiderGroup — представляє сутність «фідер-група» у базі даних та реалізує зв’язок «багато до багатьох» між таблицями «Fider» та «Group». Сутність була сформована вручну, через необхідність привласнення різних знаків конкретному фідеру у конкретній групі, тому що під час розрахунків в одній групі фідер може відніматися від загальної суми, а у іншій, навпаки додаватися до неї. На лістингу 4 зображено поля сутності «фідер-група».

Лістинг 4 *Сутність «FiderGroup»*

```
public class FiderGroup
    {
        public int Id { get; set; }
```

```
        public virtual Fider Fider { get; set; }
        public virtual Group Group { get; set; }
        public string Sign { get; set; }
    }
```

5. Клас AssociationsOfGroups — представляє сутність «асоціація груп» у базі даних та реалізує зв’язок «багато до багатьох» між таблицями «Association» та «Group». Сутність була сформована вручну, через необхідність привласнення різних знаків конкретній групі у конкретній асоціації, тому що під час розрахунків в одній асоціації група може відніматися від загальної суми, а у іншій, навпаки додаватися до неї. На лістингу 5 зображено поля сутності «асоціація груп».

Лістинг *5 Сутність «AssociationsOfGroups»*

```
public class AssociationsOfGroups
    {
        public int Id { get; set; }
        public virtual Group Group { get; set; }
        public virtual Association Association { get; set; }
        public string Sign { get; set; }
    }
```

6. Клас ASUContect — визначає контекст даних, який використовується для взаємодії з базою даних. Поля класу визначають таблиці, що створюються на основі наведених вище сутностей. Конструктор класу контексту враховує міграції, якщо такі є. На лістингу 6 зображено поля сутності «ASUContect» та метод-конструктор.

Лістинг *6 Сутність «ASUContext»*

```
public class ASUContext : DbContext
    {
```

```
        public ASUContext() : base("ASUConnection") {
            Database.SetInitializer(
              new MigrateDatabaseToLatestVersion<ASUContext,
                          Migrations.Configuration>());
        }
        public DbSet<Group> Groups { get; set; }
        public DbSet<Fider> Fiders { get; set; }
        public DbSet<TimeZones> TimeZones { get; set; }
        public DbSet<FiderGroup> FiderGroups { get; set; }
        public DbSet<Association> Associations { get; set; }
    }
```

Ці класи відповідають за модель бази даних, яка створюється методом Code First. Ці сутності використовуються для взаємодії з базою даних та виступають моделлю даних у застосунку.

Для взаємодії з базою даних, а саме, для отримання даних з бази даних, було реалізовано клас «ASUAPI», який містить методи отримання даних. Методи класу використовують технологію LINQ to Entities, що спрощує формування запитів до бази даних, оскільки запити LINQ to Entities надає простий та інтуїтивно зрозумілий підхід для отримання даних, за допомогою виразів, які по формі близькі до виразів мови SQL.

На лістингу 7 зображено методи класу ASUAPI.

Лістинг *7 Клас «ASUAPI»*

```
public class ASUAPI
    {
        public static Group GetGroup(int id);
        public static int GetGroupIDByShortName(string
shortName);
        public static Dictionary<int, List<Fider>>
```

```
            GetFidersFromGroupsByIdList(int[] ids);
        public static List<Fider> GetFidersByIdList(int[]
ids);
        public static List<Fider> GetAllFiders();
        public static List<Group> GetAllGroups();
        public static List<Group>
            GetGroupsByAssociation(string associationName);
        public static List<Association> GetAllAssociations();
        public static List<Fider> GetFidersFromGroup(int id);
        public static Dictionary<Fider, string>
            GetFidersWithSignsFromGroup(int id);
    }
```

Далі розглянемо методи класу **ASUAPI**:

- **GetGroup()** — Цей метод повертає з бази даних групу, що відповідає переданому у параметрі ідентифікатору.
- **GetGroupIDByShortName()** — Метод повертає ідентифікатор групи, що відповідає переданому у параметрі короткому імені групи.
- **GetFidersFromGroupsByIdList()** — На основі переданого у пераметрі масиву ідентифікаторів груп повертає словник, кожний елемент якого ключем має ідентифікатор групи, а значенням колекцію фідерів, які входять до групи з даним ідентифікатором.
- **GetFidersByIdList()** — Повертає колекцію фідерів, що відповідають масиву ідентифікаторів, які передані у якості параметра даного метода.
- **GetAllFiders()** — Повертає колекцію усіх фідерів, які є у базі даних.
- **GetAllGroups()** — Повертає колекцію усіх груп, які є у базі даних.
- **GetGroupsByAssociation()** — Повертає колекцію груп, які входять до асоціації, що відповідає переданому у параметрі імені асоціації.

- **GetAllAssociations**() — Повертає колекцію всіх асоціацій, які містяться у базі даних.
- **GetFidersFromGroup()** — Повертає колекцію фідерів, які входять до групи, що відповідає переданому у параметрі ідентифікатору.
- **GetFidersWithSignsFromGroup()** — відповідно до переданого ідентифікатору групи, повертає словник, кожен елемент якого ключем має конкретний фідер цієї групи, а значення відповідає знаку цього фідера у даній групі. Знак вказує на те, буде значення показника по даному фідеру під час розрахунків додатнім чи від'ємним.

На рисунку 4 зображено діаграму класів, які відповідають за структуру та роботу з БД.

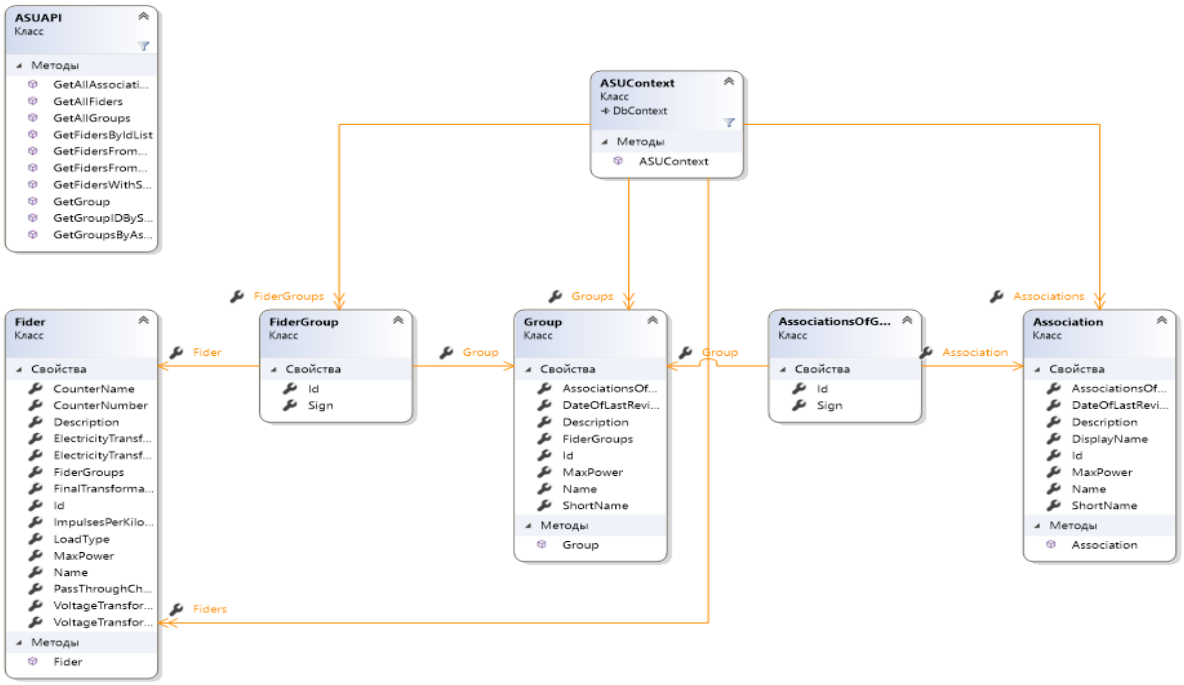

Рисунок 4 – *Діаграма класів для роботи з БД*

3.1.2 Обробка бінарних файлів з показниками

Для роботи з бінарними файлами було створено клас DataManager. Цей клас призначений для читання бінарних файлів, обробки отриманих даних, розрахунку на їх основі вихідних значень показників споживання та формування колекцій даних відповідно до потреб застосунку.

На лістингу 8 зображено поля та методи класу DataManager.

Лістинг 8 *Клас «DataManager»*

```
public class DataManager
    {
        private readonly DateTimeFormatInfo dtfi =
            CultureInfo.GetCultureInfo("ru-
ru").DateTimeFormat;
        private readonly List<string> fileNames = new
List<string>();
        public DateTimeFormatInfo GetDateTimeFormat();
        private List<ushort> ParseMomentData(string path);
        private List<List<float>> ParseHalfHourDataViaMinutes
            (string path);
        private List<float> ParseHalfHourData(string path);
        private Dictionary<int, float> ParseDayDataForFiders
            (string path, List<int> fiderThrowChannelNumbers);
        private float CalculateMomentByFiders
            (Dictionary<Fider, string> fiders,
                List<ushort> momentData);
        private List<float> CalculationForHalfHourEvidence
            (Dictionary<Fider, string> fiders,
                List<float> halfHourData);
        private List<List<float>>
            CalculationForHalfHourViaMinutesEvidence
```

```
                (Dictionary<Fider, string> fiders,
                    List<List<float>> halfHourDataViaMinutes);
        private Dictionary<int, float>
DaySumForFidersCalculation
            (Dictionary<int, float> channel_imp_pairs,
                Dictionary<int, float>
channel_coefficient_pairs);
        public List<List<float>> GetMonthByDaysForCustomers
            (int[] identifs, string date, string
customerCategory);
        public List<List<float>> GetMonthByDaysForGroups
            (string path, int daysInMonthCount, int[]
identifs,
                Dictionary<int, List<Fider>>
groupsFidersDict);
        public List<List<float>> GetMonthByDaysForFiders
            (int daysInMonthCount, string path, List<Fider>
                    fiderList);
        public Dictionary<string, float> GetMomentForGroups
            (List<Group> groups);
        public Dictionary<string, float> GetMomentForAllFiders
            (List<Fider> fiders);
        public List<float> GetMinuteChartPresentList
            (string date, Dictionary<Fider, string> fiders);
        public List<List<float>> GetGroupFidersMinuteData
            (string date, Dictionary<Fider, string> fiders);
        public List<List<float>> GetHalfHourPresentList
       (HttpRequestBase Request, Dictionary<Fider, string>
fiders);
        public List<List<float>> GetHourPresentList
           (List<List<float>> halfHourPresentList);
```

```
        private List<List<float>> UnsortedCalculatedData
            (string path, string date,
                Dictionary<Fider, string> fiders);
    public List<float> GetHalfHourBarList
             (Dictionary<Fider, string> fiders, DateTime date);
    public List<float> ErrorList();
}
```

Далі розглянемо поля та методи класу **DataManager**:

- Поле **dtfi** — Надає відомості про форматування значень дати та часу, пов'язані з мовою та регіональними параметрами.
- Поле **fileNames** — зберігає назви сталі назви півгодинних файлів.
- Метод **GetDateTimeFormat()** — повертає значення dtfi.
- Метод **ParseMomentData()** — отримує, як параметр, шлях до файлу миттєвих даних потужності, виконує читання та парсинг файлу та повертає колекцію прочитаних значень. Кожне значення — це кількість імпульсів по конкретному каналу.
- Метод **ParseHalfHourDataViaMinutes()** — отримує, як параметр, шлях до конкретного півгодинного файлу, виконує його читання та парсинг, повертає двовимірну колекцію даних, де кожен елемент колекції першого рівня є колекцією хвилинних значень імпульсів по конкретному каналу. Колекція формується для подальшої обробки, для виводу у вигляді лінійного хвилинного графіку.
- Метод **ParseHalfHourData()** — отримує, як параметр, шлях до конкретного півгодинного файлу, виконує його читання, парсинг, та формує для повернення колекцію, в кожен елемент якої є сумою хвилинних значень імпульсів за півгодини по конкретному каналу. Вихідна колекція

використовується для подальшої обробки і формування півгодинних таблиць, годинних таблиць та півгодинних гістограм.

- Метод **ParseDayDataForFiders()** — отримує, як параметр, шлях до директорії конкретної доби та колекцію наскрізних номерів каналів, виконує читання та парсинг всіх півгодинних файлів, які доступні, та повертає словник, ключем кожного елементу в якому є наскрізний номер каналу, а значенням — сума імпульсів по відповідному каналу за добу або наявну частину доби.
- Метод **CalculateMomentByFiders()** — отримує два параметри, перший — словник, ключем кожного елементу є фідер певної групи, а значенням — знак даного фідера у конкретній крупі, другий параметр — дані про миттєве споживання (кількість імпульсів), читання та парсинг яких був проведений з файлу миттєвий даних, за допомогою методу **ParseMomentData()**. Повертає значення типу float, яке є сумою показників потужності фідерів групи, переданих у першому параметрі, за останню хвилину. Показники розраховується з урахуванням знаку фідерів переданої групи, сладовими якої вони являються. Вихідне значення розраховане у мегаватах (MW), приведене до години.
- Метод **CalculationForHalfHourEvidence()** — отримує два параметри, перший — словник, ключем кожного елементу є фідер певної групи, а значенням — знак даного фідера у конкретній крупі, другий параметр — дані про споживання (кількість імпульсів), читання та парсинг яких був проведений з файлу півгодинних даних, за допомогою методу **ParseHalfHourData()**. Повертає колекцію значень, кожен елемент якої є розрахованим значенням потужності, по кожному з переданих фідерів певної групи, за півгодини. Порядок значень потужності у вихідній колекції співпадає з порядком фідерів у вхідній колекції. Значення розраховуються з урахуванням знаку фідерів переданої групи, сладовими якої вони являються. Значення показників потужності розраховані у мегаватах (MW), приведені до години. Вихідна

колекція використовується для подальшої обробки та формування півгодинних таблиць, годинних таблиць та півгодинних гістограм.

- Метод **CalculationForHalfHourViaMinutesEvidence()** — отримує два параметри, перший — словник, ключем кожного елементу є фідер певної групи, а значенням знак даного фідера у конкретній крупі, другий параметр — дані про споживання (кількість імпульсів), читання та парсинг яких був проведений з файлу півгодинних даних, за допомогою методу **ParseHalfHourDataViaMinutes()**. Повертає колекцію значень, кожен елемент якої є колекцією хвилинних показників потужності, по кожному з переданих фідерів певної групи, за півгодини. Порядок колекцій хвилинних показників потужності у вихідній колекції співпадає з порядком фідерів у вхідній колекції. Значення розраховуються з урахуванням знаку фідерів переданої групи, сладовими якої вони являються. Значення хвилинних показників, які містять колекції другого рівня розраховані у мегаватах (MW), приведені до години. Вихідна колекція використовується для подальшої обробки та формування хвилинних лінійних графіків.
- Метод **DaySumForFidersCalculation()** — отримує два параметри. Перший — словник, ключем елемента є наскрізний канал контролера, а значенням — сума імпульсів по відповідному каналу за добу або наявну частину доби. Другий — також словник, ключем є наскрізний номер каналу, а значенням — коефіцієнт трансформації, який є складовою формули розрахунку значення показника споживання. Повертає словник, ключем кожного елементу якого являється наскрізний номер каналу, а значенням — значення показника потужності за добу по відповідному каналу, розраховане у мегаватах (MW).
- Метод **GetMonthByDaysForCustomers()** — отримує три параметри. Перший — колекцію ідентифікаторів, які можуть бути ідентифікаторами груп або ідентифікаторами фідерів. Другий — строкове значення дати, а саме місяць та рік. Третій — строкове значення категорії споживачів, яке визначає, чи є набір

ідентифікаторів набором ідентифікаторів груп або набором ідентифікаторів фідерів. В залежності від параметру категорії, метод повертає двовимірну колекцію, кожний елемент якої являється колекцією розрахованих значень показників потужності подобово за місяць по набору фідерів, або набору груп споживачів. Порядковий номер елементів колекції першого рівня співпадає з порядковим номером ідентифікатора в колекції ідентифікаторів. Значення показників потужності розраховане у мегаватах (MW).

- Метод **GetMonthByDaysForGroups()** — отримує чотири параметри. Перший параметр — це шлях до директорії обраного місяця. Другий параметр — кількість днів у обраному місяці. Третій параметр — колекцію ідентифікаторів груп. Четвертий параметр — словник, кожний елемент якого ключем має ідентифікатор групи, а значенням колекцію фідерів, які входять до групи з даним ідентифікатором. Метод повертає двовимірну колекцію, кожний елемент якої є колекцією розрахованих значень показників потужності подобово за місяць по набору груп споживачів. Порядковий номер елементів колекції першого рівня співпадає з порядковим номером ідентифікатора в колекції ідентифікаторів. Значення показників потужності розраховане у мегаватах (MW).
- Метод **GetMonthByDaysForFiders()** — отримує три параметри. Перший параметр — це кількість днів у обраному місяці. Другий параметр — шлях до директорії обраного місяця. Третій параметр — колекцію фідерів. Метод повертає двовимірну колекцію, кожний елемент якої є колекцією розрахованих значень показників потужності подобово за місяць по набору фідерів. Порядковий номер елементів колекції першого рівня співпадає з порядковим номером фідерів у колекції фідерів, переданій у якості третього параметру. Значення показників потужності розраховане у мегаватах (MW).
- Метод **GetMomentForGroups()** — отримує один параметр, а саме колекцію груп. Повертає словник, ключем кожного елементу якого являється ім’я

групи, а значенням — показник потужності у відповідній групі, за останню хвилину. Значення показника потужності розраховане у мегаватах та приведене до години.

- Метод **GetMomentForAllFiders()** — отримує один параметр, а саме колекцію фідерів. Повертає словник, ключем кожного елементу якого являється ім’я фідеру, а значенням — показник потужності у відповідному фідері, за останню хвилину. Значення показника потужності розраховане у мегаватах та приведене до години.
- Метод **GetMinuteChartPresentList()** — отримує два параметри. Перший параметр — строкове значення дати у форматі «день-місяць-рік». Другий параметр — словник, кожен елемент якого ключем має конкретний фідер певної групи, а значення відповідає знаку цього фідера у даній групі. Метод повертає сформовану і готову до використання у побудові лінійного графіку колекцію, елементи якої являються сумами показників потужності переданих фідерів. Кожний елемент відповідає одній хвилині доби. Якщо це поточна доба, то кількість елементів колекції відповідатиме кількості хвилин, що минули з початку доби.
- Метод **GetGroupFidersMinuteData()** — отримує два параметри. Перший параметр — строкове значення дати у форматі «день-місяць-рік». Другий параметр — словник, кожен елемент якого ключем має конкретний фідер певної групи, а значення відповідає знаку цього фідера у даній групі. Метод повертає двовимірну колекцію, кожен елемент якої являється колекцією. Елементи колекції другого рівня являються значеннями показників потужності переданих фідерів. Кожний елемент колекції першого рівня відповідає конкретному переданому фідеру, а кожний елемент колекції другого рівня відповідає одній хвилині доби. Якщо це поточна доба, то кількість елементів колекції відповідатиме кількості хвилин, що минули з

початку доби. Вихідна колекція являє собою набір колекцій, які відповідають переданим фідерам і містять набір похвилинних значень за добу.

- Метод **GetHalfHourPresentList()** — метод отримує два параметри. Перший параметр — об’єкт запиту (HttpRequest). Другий параметр — словник, кожен елемент якого ключем має конкретний фідер певної групи, а значення відповідає знаку цього фідера у даній групі. Метод запускає процес читання і парсингу півгодинних та розрахунку значень показників потужності. Отримані дані сортуються для сумісності з табличним виводом. Вихідна колекція являється двовимірною. Порядковий номер елементів колекції першого рівня відповідає порядковому номеру елементів словника (другий параметр). Кожне значення елементу колекції другого рівня відповідає значенню показника потужності конкретної півгодини обраної доби. Якщо це поточна доба, то колекції другого рівня будуть містити всі наявні півгодинні значення. Якщо останній наявний півгодинний файл сформований не повністю, то останнє півгодинне значення колекцій другого рівня буде містити значення, розраховане на основі наявних хвилинних значень у цьому файлі. Значення показників потужності розраховані у мегаватах (MW) та приведені до години.
- Метод **GetHourPresentList()** — отримує один параметр, який являється вихідним значенням методу **GetHalfHourPresentList()**. Метод формує колекцію годинних значень на основі переданої колекції півгодинних значень. Значення разраховуються у мегаватах (MW).
- Метод **GetHalfHourBarList()** — отримує два параметри. Перший параметр — словник, кожен елемент якого ключем має конкретний фідер певної групи, а значення відповідає знаку цього фідера у даній групі. Другий параметр — об’єкт DateTime, що містить обрану користувачем дату. Вихідним значенням являється колекція, кожен елемент якої містить значення показника потужності, що є сумою показників потужності всіх фідерів, які були передані.

Кожне значення відповідає сумі півгодинних значень всіх фідерів групи. Кількість значень колекції відповідає кількості півгодин. Якщо це поточна доба, то кількість значень дорівнюватиме наявним півгодинам. У разі відсутності файла півгодини значення дорівнюватиме коду помилки.

- Метод **UnsortedCalculatedData()** — допоміжний для методів **GetHalfHourPresentList()** та **GetHalfHourBarList()**. Метод отримує три параметри. Перший параметр — шлях до директорії обраної доби. Другий — строковий параметр, що містить обрану користувачем дату. Третій параметр — словник, кожен елемент якого ключем має конкретний фідер певної групи, а значення відповідає знаку цього фідера у даній групі. Метод повертає несортовану двовимірну колекцію. Кожен елемент колекції першого рівня також є колекцією та відповідає конкретній півгодині, а кожен елемент колекції другого рівня відповідає конкретному фідеру, який був переданий третім параметром. Значення розраховуються у мегаватах (MW) та приведені до години. Якщо файли у директорії відсутні, замість відсутнього файла або файлів метод сформує колекцію з кодом помилки.
- Метод **ErrorList()** — допоміжний метод, який формує колекцію кожен елемент якої містить значення помилки.

3.2 Шар бізнес-логіки

Для реалізації механізмів взаємодії між шаром доступу до даних та шаром представлення були створені класи контролерів у проекті ASP.NET MVC. Контролери реалізують логіку взаємодії шару представлення з шаром доступу до даних. Було реалізовано:

1. **HomeController** — методи даного контролера відповідають за отримання та обробку даних, які передаються на головну сторінку застосунку. Також, вони викликають метод відтворення представлення.

2. **ChartController** — методи даного контролера відповідають за отримання та обробку даних з БД та бінарних файлів, формування та передачу їх у пов’язані представлення для формування лінійних графіків та стовпчастих діаграм (гістограм).
3. **TableController** — методи даного контролера відповідають за отримання та обробку даних з БД та бінарних файлів, формування та передачу їх у пов’язані представлення для формування таблиць різних форматів та вмісту.

На лістингу 9 зображено структуру класу «HomeController».

Лістинг 9 *Структура класу «HomeController»*

```
public class HomeController : Controller
    {
        public ActionResult Index();
        [HttpPost]
        public ActionResult Clock();
    }
```

Розглянемо методи цього класу:

- Метод **Index()** — даний метод визначає поточну дату та передає її у представлення. Повертає предаставлення Index.cshtml.
- Метод **Clock()** — метод, що викликається POST-запитом, та повертає об’єкт JSON, у якому інкапсульовано час серверу у момент виклику (години, хвилини, секунди).

На лістингу 10 зображено структуру класу «ChartController».

Лістинг 10 *Структура класу «ChartController»*

```
public class ChartController : Controller
    {
        public ActionResult Chart();
        public ActionResult ChartPartial();
```

```
        [HttpPost]
        public ActionResult AjaxChartPartial(
            string groupName, string date);
        public ActionResult MomentBarChart(string
associationName);
        [ActionName("MomentBarChart")]
        [HttpPost]
        public ActionResult MomentBarChartPost
            (string associationName);
        public ActionResult FidersMomentBarChart();
        [ActionName("FidersMomentBarChart")]
        [HttpPost]
        public ActionResult FidersMomentBarChartPost();
        public ActionResult HalfHourBarChart();
        [ActionName("HalfHourBarChart")]
        [HttpPost]
        public ActionResult HalfHourBarChartPost
            (string groupName, string date);
        [HttpGet]
        public ActionResult ChartExtended();
        [HttpPost]
        public ActionResult GetChartGroups(string date, int[]
ids);
        [HttpPost]
        public ActionResult GetChartFidersInGroup
            (string date, int id);
    }
```

Розглянемо методи цього класу:

- Метод **Chart()** — метод формує колекцію елементів SelectListItem з назвами груп та передає їх у представлення. Повертає представлення Chart.chtml.
- Метод **ChartPartial()** — метод викликає логіку взаємодії з базою даних та на основі відповіді запускає логіку читання, парсингу, розрахунку та обробки даних по обраній групі, формує об’єкт ViewBag, серед полів якого є отримані показники споживання у запитуваному форматі, колекція міток для графіку та показник максимальної потужності групи, на основі якого формується масштаб графіку. Повертає часткове представлення MinuteLineChartPartial.cshtml.
- Метод **AjaxChartPartial()** — метод, що викликається POST-запитом з представлення MinuteLineChartPartial. Отримує два параметри. Перший параметр — коротке ім’я групи (ShortName), другий параметр — обрана користувачем дата, або поточна дата, що була передана у представлення Chart.cshtml. Повертає JSON-об’єкт з оновленними згідно імені групи та дати даними. Використовується для оновлення даних лінійного графіку без перезавантаження веб-сторінки.
- Метод **MomentBarChart()** — метод отримує параметром ім’я ассоціації, визначає поточну дату. Формує об’єкт ViewBag, полями якого є ім’я ассоціації та поточна дата і передає його у представлення. Повертає представлення MomentBarChart.cshtml.
- Метод **MomentBarChartPost()** — метод, що викликається POST-запитом з представлення та отримує один параметр — ім’я ассоціації. Викликає логіку взаємодії з базою даних та на основі відповіді запускає логіку читання, парсингу, розрахунку та обробки даних по обраній асоціації. Повертає JSON-об’єкт, який містить колекцію міток (назви груп асоціації), колекцію значень показників споживання по групам обраної асоціації та колекцію значень максимальної потужності для кожної групи, на основі яких формується

масштаб елементів графіку. Використовується для первинного формування та оновлення даних стовпчастого графіку без перезавантаження веб-сторінки.

- Метод **FidersMomentBarChart()** — метод визначає поточну дату та формує об’єкт ViewBag з полем поточної дата і передає його у представлення. Повертає представлення FidersMomentBarChart.cshtml.
- Метод **FidersMomentBarChartPost()** — метод, що викликається POST-запитом з представлення. Викликає логіку взаємодії з базою даних та на основі відповіді запускає логіку читання, парсингу, розрахунку та обробки даних по всім наявним у БД фідерам. Повертає JSON-об’єкт, який містить колекцію міток (імена фідерів), колекцію значень показників споживання по фідерам та колекцію значень максимальної потужності для кожного фідера, на основі яких формується масштаб елементів графіку. Використовується для первинного формування та оновлення даних стовпчастого графіку без перезавантаження веб-сторінки.
- Метод **HalfHourBarChart()** — метод визначає поточну дату, викликає логіку взаємодії з базою даних і на основі відповіді формує колекцію елементів SelectListItem з назвами груп. Формує об’єкт ViewBag, поля якого містять поточну дату, колекцію елементів SelectListItem для формування списку вибору та елемет списку, що буде встановлено як обраний за замовчуванням. Повертає представлення HalfHourBarChart.cshtml.
- Метод **HalfHourBarChartPost()** — метод, що викликається POST-запитом з представлення та отримує два параметри: ім’я групи та обрану дату. Викликає логіку взаємодії з базою даних та на основі відповіді запускає логіку читання, парсингу, розрахунку та обробки даних по обраній групи. Повертає JSON-об’єкт, який містить колекцію міток, колекцію значень показників споживання по обраній групі та значення максимальної потужності групи, на основі якого формується масштаб елементів графіку. Використовується для

первинного формування та оновлення даних стовпчастого графіку без перезавантаження веб-сторінки.

- Метод **ChartExtended()** — метод викликає логіку взаємодії з базою даних і на основі відповіді формує колекцію об’єктів, які містять ідентифікатор групи, ім’я групи, показник максимальної потужності групи та колекцію фідерів, що містить група. Також формується колекція міток для лінійного графіку і колекція обєктів, що містять коротке ім’я групи та її ідентифікатор. На основі цих колекцій формується об’єкт ViewBag, який передається у представлення. Метод повертає представлення ExtandedChart.cshtml.
- Метод **GetChartGroups()** — метод, що викликається POST-запитом з представлення та отримує два параметри: обрану дату та набір ідентифікаторів груп. Викликає логіку взаємодії з базою даних та на основі відповіді запускає логіку читання, парсингу, розрахунку та обробки даних по набору обраних груп. Повертає JSON-об’єкт, який складається з трьох полів. Перше поле містить колекцію об’єктів, що складаються з двох полів, а саме ідентифікатора групи та відповідної колекції значень показників споживання. Друге поле містить найбільше значення показника потужності серед усіх показників усіх груп. Третє поле містить найбільший серед набору груп показник максимальної потужності. Перше поле використовується для формування датасетів лінійного графіку, друге та третє поля використовуються для визначення масштабу графіку. Метод використовується для первинного формування лінійного графіку груп та оновлення даних у графіку без перезавантаження веб-сторінки.
- Метод **GetChartFidersInGroup()** — метод, що викликається POST-запитом з представлення та отримує два параметри: обрану дату та ідентифікатор групи. Викликає логіку взаємодії з базою даних та на основі відповіді запускає логіку читання, парсингу, розрахунку та обробки даних по набору фідерів обраної групи. Повертає JSON-об’єкт, який складається з трьох полів. Перше поле

містить колекцію об’єктів, що складаються з двох полів, а саме наскрізного номера каналу фідера та його знак у поточній групі. Друге поле містить колекцію об’єктів, що складаються з двох полів, а саме наскрізного номеру каналу фідеру та колекції значень показників потужності, відповідного фідеру. Третє поле містить найбільше значення показнника потужності серед показників потужності усіх фідерів групи. Перше поле використовується для формування форми для лінійного графіку, друге для формування датасетів графіку, третє для визначення масштабу графику. Метод використовується для первинного формування лінійного графіку фідерів групи та оновлення даних у графіку без перезавантаження веб-сторінки.

На лістингу 11 зображено структуру класу «TableController».

Лістинг 11 *Структура класу «TableController»*

```
public class TableController : Controller
    {
        public ActionResult Table();
        public ActionResult HalfOrHourTablePartial();
        [HttpGet]
        public ActionResult ExtendedTable();
        [ActionName("ExtendedTable")]
        [HttpPost]
        public ActionResult ExtendedTablePost(int[] identifs,
            string category, string tableType, string date);
    }
```

Розглянемо методи цього класу:

- Метод **Table()** — метод викликає логіку взаємодії з базою даних та на основі відповіді формує колекцію елементів SelectListItem з назвами груп, визначає

групу по замовчуванню, розміщує їх у об’єкті ViewBag та передає у представлення. Повертає представлення Table.chtml

- Метод **HalfOrHourTablePartial()** — метод викликає логіку взаємодії з базою даних та на основі відповіді запускає логіку читання, парсингу, розрахунку та обробки даних по обраній групі, формує об’єкт ViewBag, який має чотири поля. Перше поле — ім’я групи, друге поле — колекція об’єктів, що мають два поля та містять ім’я конкретного фідеру та його знак у групі, третє поле — двовимірна колекція півгодинних значень за добу, порядковий номер елементів колекції першого рівня співпадає з порядковим номером елементів колекції фідерів (друге поле), четверте поле — обрана користувачем дата або поточна дата. Якщо користувач обрав пункт годинних даних у формі, тоді до ViewBag додається п’яте поле, яке містить двовимірну колекцію годинних значень за добу, порядковий номер елементів колекції першого рівня співпадає з порядковим номером елементів колекції фідерів (друге поле). Повертає часткове представлення HalfHourTablePartial.cshtml або HourTablePartial.cshtml, в залежності від вибору користувача.
- Метод **ExtendedTable()** — метод викликає логіку взаємодії з базою даних та на основі відповіді формує об’єкт ViewBag, який має три поля. Перше поле — колекція об’єктів, які мають два поля, а саме: ідентифікатор фідеру та ім’я фідеру. Друге поле — колекція об’єктів, які мають два поля, а саме ідентифікатор групи та ім’я групи. Третій параметр — колекція об’єктів, що мають два поля, а саме ідентифікатор асоціації та ім’я асоціації. Цей об’єкт передається у представлення. Метод повертає представлення ExtandedTable.cshtml.
- Метод **ExtendedTablePost()** — метод що викликається POST-запитом з представлення та отримує чотири параметри, а саме: колекцію ідентифікаторів, категорію споживачів, тип таблиці та дату. В залежності від типу таблиці метод викликає відповідну логіку і на основі ідентифікаторів,

дати та категорії споживачів і повертає JSON-об’єкт, який містить двовимірну колекцію даних, що форматовані для табличного виводу у представленні.

3.3 Шар представлення

Для реалізації шару представлення були створені представлення (View) в межах проекту ASP.NET MVC. Створені представлення для методів контролерів «**HomeController**», «**ChartController**» та «**TableController**». Також було створено представлення макету _Layout.cshtml.

Для методів контролера «**HomeController**» було створено представлення Index.cshtml. Представлення реалізує заставку за допомогою CSS стилів та JavaScript таймерів.

Для методів контролера «**ChartController**» було створено шість представлень, які описані далі.

Chart.cshtml

Chart.cshtml — реалізує макет для лінійного хвилинного графіку. У представлення підключаються бібліотеки для роботи з графіками, а саме:

- Moment.js — бібліотека реалізує формат даних, що необхідний для побудови графіку з віссю часу, засобами Chart.js.
- Hammer.js — бібліотека для розпізнавання жестів.
- Chart.js — бібліотека для роботи з графіками.
- chartjs-plugin-zoom.js — бібліотека, яка реалізує масштабування та панорамування.

Також реалізована AJAX-форма, яка складається з елементу «календар», завдяки якому реалізується вибір дати та спадаючого списку з назвами груп.

Завдяки AJAX-формі реалізовано механізм, який дозволяє робити POST-запити до серверу без перезавантаження сторінки загалом. Події модифікації

елементів «календар» та «спадаючий список» ініціюють перезавантаження часткового представлення у цільовому контейнері.

MinuteLineChartPartial.cshtml

MinuteLineChartPartial.cshtml — часткове представлення, використовується представленням **Chart.cshtml**. Відповідає за побудову графіку хвилинних значень показників потужності.

Представлення містить контейнер з елементом <**canvas**>, який використовується бібліотекою **Chart.js** для візуалізації графіку. За конфігурацію об’єкту графіку, надання та оновлення даних для наборів даних графіку, відповідає скрипт. Також в межах скрипту встановлюється таймер, для виклику функції оновлення даних один раз на хвилину.

За оновлення даних відповідає функція **ajaxChartDataUpdate()**. Засобами **AJAX**, асинхронно, один раз на хвилину, скрипт часткового представлення робить запит на сервер для оновлення набору даних у графіку. Після оновлення викликається функція **update()** об’єкту поточного графіку, після чого відбувається повторна візуалізація графіку з урахуванням оновлених наборів даних.

ExtandedChart.cshtml

ExtandedChart.cshtml — представлення реалізує побудову, наповнення даними та оновлення лінійного графіку з декількома наборами даних. У представлення підключаються JavaScript бібліотеки для побудови та налаштування графіків, які були наведені у описі представлення **Chart.cshtml**.

Дане представлення було реалізоване без елементу форми та без використання часткового представлення. Структура **ExtandedChart.cshtml** містить контейнер з елементом **<canvas>**, який використовується бібліотекою **Chart.js** для візуалізації графіку та контейнер з елементами, які імітують форму

і використовуються користувачем для конфігурування графіку за потребами, а саме:

- Дві кнопки типу «**Radio button**», які об’єднані в групу та реалізують можливість переключення між режимами «Групи» та «Фідери групи».
- Елемент «**Календар**», що надає функціонал для вибору дати.
- Кнопка «**Список**».

Переключення між режимами «Групи» та «Фідери групи» впливають на функціонал кнопки «Список». Якщо обрано режим «Групи», під час наведення курсора на кнопку «Список» відбувається динамічна візуалізація меню, яке містить елементи типу «Check box», що відповідають списку наявних у БД груп, які об’єднані в одну групу елементів, завдяки чому реалізована можливість множинного вибору груп. Якщо обрано режим «Фідери групи», під час наведення курсора на кнопку «Список» відбувається динамічна візуалізація меню, яке містить елементи типу «Radio button», що відповідають списку наявних у БД груп, під час наведення курсору на конкретний елемент типу «Radio button», відбувається візуалізація підменю з елементами типу «Check box», що об’єднані в одну групу елементів та відповідають списку наявних у групі фідерів, завдяки чому реалізована можливість вибору конкретних фідерів певної групи.

На рисунку 5 зображено елемент «Список» у режимі «Групи».

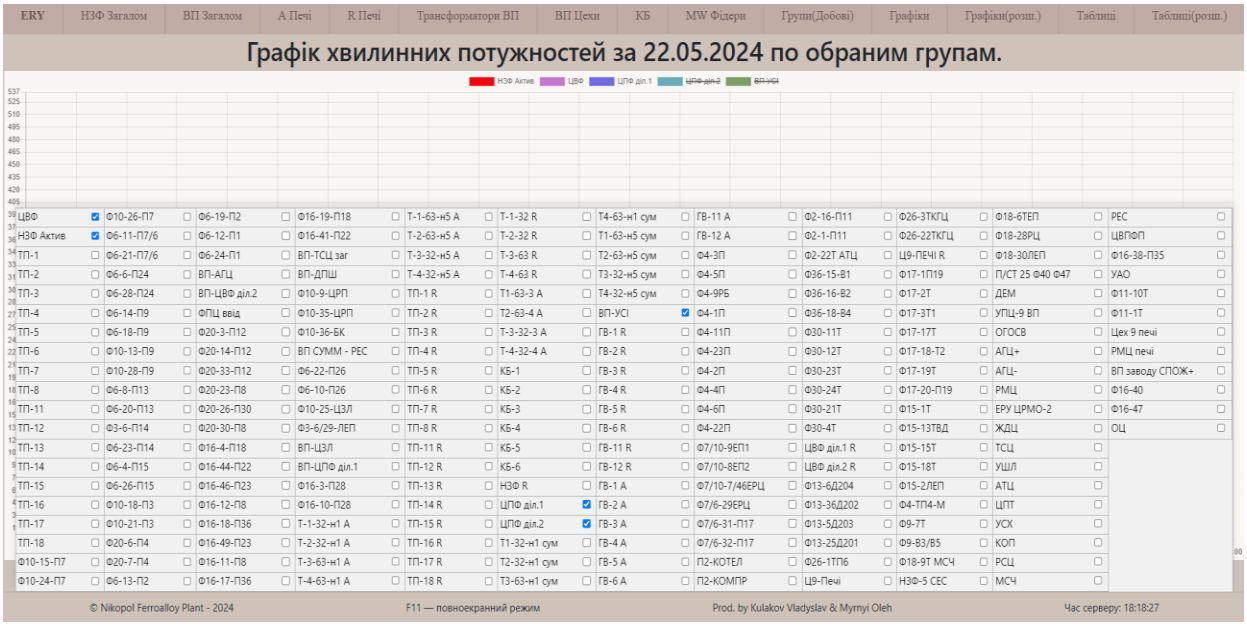

Рисунок 5 – *Елемент «Список» у режимі «Групи»*

На рисунку 6 зображено елемент «Список» у режимі «Фідери групи».

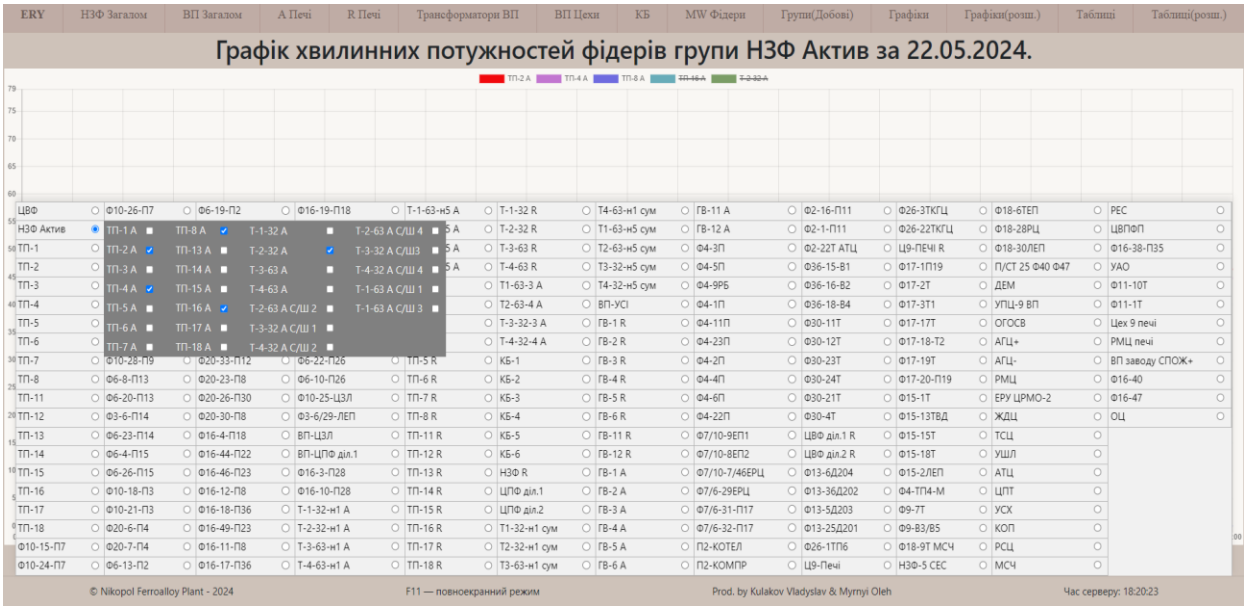

Рисунок 6 – *Елемент «Список» у режимі «Фідери групи»*

За обробку подій елементів, конфігурування графіку, а також його наповнення та оновлення відповідає скрипт, що реалізований у файлі «**ExtandedChartScript.js**». Даний файл додається до представлення окремо.

У файлі **ExtandedChartScript.js** реалізуються наступні функції:

- **setStartValues()** — функція встановлює значення по замовчуванню. Для графіку у режимі «Групи» ідентифікатори цільових груп. У режимі «Фідери групи» ідентифікатор цільової групи та ідентифікатори фідерів даної групи. Також призначає обробники подій для елементів інтерфейсу.
- **chartInit()** — функція виконує ініціалізацію об’єкта графіку та його полів, а також встановлює налаштування графіку, призначає обробники подій.
- **chartRefresh()** — функція отримує параметром об’єкт, що являється відповіддю на AJAX-запит. На основі відповіді на запит та значення режиму графіку, виконує налаштування параметрів масштабування та панорамування. Проводить переініціалізацію або оновлення наборів даних, в залежності від поточної конфігурації графіку. За необхідності оновлює вміст заголовків та міток, згідно даних про режим та конфігурацію графіку.
- **chartDataLoad()** — функція виконує POST-запит до серверу, засобами AJAX [10]. Вибір конкретного метода контролеру виконується відповідно до режиму графіку. Якщо запит успішний — викликається функція оновлення, в яку передається об’єкт, що є відповіддю від серверу. У разі, якщо запит неуспішний, у консоль виводиться повідомлення про помилку, а функція оновлення графіку не викликається.
- **buttonHover()** — функція являється обробником події наведення курсору миші на кнопку «Список». В залежності від обраного режиму графіку та значень, відповідно, колекцій груп або ідентифікатору групи та колекції номерів наскрізних каналів цієї групи (для режиму «Фідери групи»), а також наявних в БД груп та фідерів виконується побудова меню вибору для кожного з режимів графіку. Крім того, відмічаються раніше обрані

елементи або елементи по замовчуванню, якщо графік завантажується вперше.

- **radioDropdownContentMouseEnter()** — функція являється обробником події, що відбувається під час зміни конфігурації елементів підменю конкретної групи у режимі «Фідери групи». Встановлює відповідну змінну у значення «True». Змінна бере участь у конфігуруванні меню під час спрацьовування обробника «**radioItemCheckboxOnChange()**».
- **radioItemCheckboxOnChange()** — функція являється обробником події зміни стану елементів «Check box» у підменю конкретної групи у режимі «Фідери групи». Подія спрацьовує, якщо елемент типу «Check box» у підменю конкретної групи, в режимі «Фідери групи», змінив свій стан. Враховуючи оновлену конфігурацію, викликає функцію **chartDataLoad()**, що призводить до відправки POST-запиту на сервер та оновлення графіку.
- **radioItemDivMouseEnter()** — функція являється обробником події натискання клавіші миші на контейнер-огортку, в якому знаходяться елемент типу «Check box» та відповідний йому елемент типу «Label», у режимі «Фідери групи». Змінює значення «checked» елементу «Check box» на «True».
- **dateChanged()** — обробник події зміни дати у елементі «Календар». Викликає функцію **chartDataLoad()**, яка, відповідно до обраної дати, виконує запит на сервер, що призводить до оновлення графіку.
- **getCurrentDate()** — функція повертає поточну дату.
- **modeChanged()** — функція являється обробником події, що відбувається під час зміни режиму графіку з «Групи» на «Фідери групи» та навпаки. Зберігає назву обраного режиму графіку у змінній. Викликає функцію **chartDataLoad()**, що призводить до відправки POST-запиту на сервер та оновлення графіку.

- **groupCheckboxOnChange()** — функція являється обробником події змінення значення «checked» елементу «Check box», у меню груп в режимі «Групи». Викликає функцію **chartDataLoad()**, що призводить до відправки POST-запиту на сервер та оновлення графіку відповідно до нової конфігурації меню в режимі «Групи».
- Обробник події «**window.onload»**, реалізований за допомогою анонімної функції. Під час завантаження представлення, обробник події викликає функції: **setStartValues()**, **chartInit()**, та функцію **chartDataLoad()**, яка відповідає за POST-запит до серверу. Також, обробник встановлює таймер, який один раз на хвилину викликає даний метод. Таким чином реалізується оновлення діаграми, задля актуалізації даних, з дискретизацією в одну хвилину.

Також підключається файл стилів «**ExtandedChartStyleSheet.css**», у якому визначаються стилі для представлення та його елементів.

MomentBarChart.cshtml

MomentBarChart.cshtml — представлення реалізує побудову, наповнення даними та оновлення стовпчастої діаграми з декількома наборами даних. У представлення підключаються JavaScript бібліотеки для побудови та налаштування діаграми, які були наведені у описі представлення **Chart.cshtml**.

Структура **MomentBarChart.cshtml** містить контейнер з елементом **<canvas>**, який використовується бібліотекою **Chart.js** для візуалізації графіку та контейнер з елементами, які використовуються користувачем для навігації:

- Кнопка «Назад».
- Кнопка «Вперед».
- Мітка з номером сторінки.

Кожен набір даних стовпчастої діаграми відповідає конкретній групі в асоціації, яка представлена на діаграмі. У представленні реалізовано скрипт, який

відповідає за формування, налаштування та наповнення стовпчастої діаграми. Скрипт містить наступні функції:

- **maxPowerPercent()** — функція реалізує логіку розрахунку відсоткового відношення поточного показника потужності групи до показника максимальної потужності даної групи. Результат використовується для відображення наборів даних кольором, який передбачено для певного відсоткового показника (сірий, зелений, синій або червоний).
- **chartInit()** — функція виконує ініціалізацію об’єкта стовпчастої діаграми та його полів, а також встановлює налаштування діаграми, призначає обробники подій.
- **chartUpdate()** — відповідно до полів об’єкту відповіді, що був отриманий після виклику функції **loadMomentChartData()**, виконує налаштування обробників подій графіку, скриптових параметрів та оновлення наборів даних. Після виконання власної логіки викликає метод **update()** об’єкту діаграми.
- **clickBack()** — обробник події натискання кнопки «Назад». Бере участь в управлінні навігацією та відображенні коректного номера сторінки під час перегортання. Викликає метод **chartUpdate()**.
- **clickForward()** — обробник події натискання кнопки «Вперед». Бере участь в управлінні навігацією та відображенні коректного номера сторінки під час перегортання. Викликає метод **chartUpdate()**.
- **loadMomentChartData()** — функція виконує POST-запит до серверу засобами AJAX. Якщо запит здійснено вперше після завантаження — викликається функція **chartInit()**. Якщо запит здійснено повторно — функція **chartUpdate()**. У разі, якщо запит неуспішний, у консоль виводиться повідомлення про помилку, а функція оновлення графіку не викликається.

- Обробник події «**onwheel**» контейнеру, в якому знаходиться елемент **<canvas>** — обробник події прокручування коліщатка миші. Бере участь в управлінні навігацією та відображенні коректного номера сторінки під час перегортання. Дублює функціонал методів **clickBack()** та **clickForward()**, адаптований для події «**onwheel**». Викликає метод **chartUpdate()**.
- Обробник події «**window.onload**» реалізований за допомогою анонімної функції. Під час завантаження представлення, обробник встановлює таймер, який один раз на хвилину викликає метод **loadMomentChartData()**. Таким чином реалізується оновлення діаграми, задля актуалізації даних, з дискретизацією в одну хвилину.

Також, у представленні наявний блок «style» із набором CSS класів, які використовуються для стилізації.

HalfHourBarChart.cshtml

HalfHourBarChart.cshtml — представлення реалізує побудову, наповнення даними та оновлення стовпчастої діаграми, яка виконує візуалізацію добових даних споживання групи. Значення кожного стовпчика є розрахованим значенням споживання обраної групи за півгодини. У представлення підключаються JavaScript бібліотеки для побудови та налаштування діаграми, які були наведені у описі представлення **Chart.cshtml**.

Структура **HalfHourBarChart.cshtml** містить контейнер з елементом **<canvas>**, який використовується бібліотекою **Chart.js** для візуалізації діаграми, контейнер з елементами, які імітують форму та використовуються користувачем для конфігурування діаграми за потребами, а саме:

- Спадаючий список — містить набір груп.
- Елемент «Календар», для вибору дати.

Представлення містить скрипт, який реалізує логіку завантаження та оновлення наборів даних, що відображаються. Скрипт реалізує наступні функції:

- **chartInit()** — функція виконує ініціалізацію об’єкта графіку та його полів, а також визначає налаштування діаграми, призначає обробники подій та скриптові параметри. Викликає функцію **chartUpdate()**
- **chartUpdate()** — функція спирається на об’єкт, отриманий POST-запитом до серверу. Враховуючи півгодинні значення показників потужності групи, функція оновлює набори даних діаграми та, за допомогою скриптових параметрів, визначає стилі відображення для цих наборів даних. Після виконання власної логіки викликає метод update() об’єкту діаграми.
- **maxPowerPercents()** — функція реалізує логіку розрахунку відсоткового відношення поточного показника півгодинного значення потужності групи до показника максимальної потужності даної групи. Впливає на стиль відображення наборів даних.
- **selectListOnChange()** — обробник події змінення значення спадаючого списку. Викликає функцію **chartUpdate()**.
- **dateOnChange()** — обробник події змінення значення елемента «календар». Викликає функцію **chartUpdate()**.
- Обробник події **window.onload —** Під час завантаження представлення, обробник встановлює таймер, який один раз на хвилину викликає метод **LoadHalfHourChartData()**.

FidersMomentBarChart.cshtml

FidersMomentBarChart.cshtml — представлення реалізує побудову, наповнення даними та оновлення стовпчастої діаграми, яка виконує візуалізацію поточних даних споживання всіх наявних фідерів, незалежно від їх приналежності до тих чи інших груп. Значення кожного стовпчика є розрахованим значенням потужності певного фідеру. У представлення підключаються JavaScript бібліотеки для побудови та налаштування діаграми, які були наведені у описі представлення **Chart.cshtml**.

Структура **MomentBarChart.cshtml** містить контейнер з елементом **<canvas>**, який використовується бібліотекою **Chart.js** для візуалізації графіку та контейнер з елементами, які використовуються користувачем для навігації:

- Кнопка «Назад».
- Кнопка «Вперед».
- Мітка з номером сторінки.

У представленні реалізовано скрипт, який відповідає за формування, налаштування та наповнення стовпчастої діаграми. Скрипт містить наступні функції:

- **maxPowerPercent()** — функція реалізує логіку розрахунку відсоткового відношення поточного показника потужності групи до показника максимальної потужності даної групи. Результат впливає на стилізацію.
- **chartInit()** — функція виконує ініціалізацію об’єкта стовпчастої діаграми та його полів, а також встановлює налаштування діаграми, призначає обробники подій.
- **chartUpdate()** — виконує налаштування обробників подій графіку, скриптових параметрів та оновлення наборів даних. Після виконання власної логіки викликає метод **update()** об’єкту діаграми.
- **clickBack()** — обробник події натискання кнопки «Назад». Викликає метод **chartUpdate()**.
- **clickForward()** — обробник події натискання кнопки «Вперед». Викликає метод **chartUpdate()**.
- **loadFidersMomentChartData ()** — функція виконує POST-запит до серверу, засобами AJAX. Якщо запит здійснено вперше після завантаження — викликається функція **chartInit()**. Якщо запит здійснено повторно — функція **chartUpdate()**.
- Обробник події прокручування коліщатка миші («**onwheel**»). Бере участь в управлінні навігацією та відображенні коректного номера сторінки під час

перегортання. Дублює функціонал методів **clickBack()** та **clickForward()**. Викликає метод **chartUpdate()**.

- Обробник події **window.onload**. Під час завантаження представлення, обробник встановлює таймер, який один раз на хвилину викликає метод **loadMomentChartData()**.

Також, у представленні наявний блок із набором CSS класів, які використовуються для стилізації.

Для методів контролера «TableController» було створено чотири представлення:

1. **Table.cshtml** — представлення реалізує макет для таблиць півгодинних та годинних значень показників потужності по фідерам обраної групи. Також реалізована AJAX-форма, яка складається з елементу «календар», завдяки якому реалізується вибір дати та спадаючого списку з назвами груп. Завдяки ajax-формі реалізовано механізм, який дозволяє робити POST-запити до серверу без перезавантаження сторінки загалом. Події модифікації елементів «Календар» та «Спадаючий список» ініціюють перезавантаження часткових представлень у цільовому контейнері. Форма містить групу з двох кнопок типу «Radio button», зміна конфігурації яких призводить до завантаження відповідного часткового представлення. Представлення містить скрипт, в якому визначається обробник події «**window.onload**», у якому ініціалізується таймер, що один раз на хвилину активує функцію «**submit()**» AJAX-форми, завдяки чому реалізується механізм оновлення даних таблиці. Стилізація макету та часткових представлень відбувається завдяки набору стилів у секції «style» представлення-макету та стилів, які містять часткові представлення.
2. **HourTablePartial.cshtml** — часткове представлення, використовується представленням **Table.cshtml**. Відповідає за побудову таблиці годинних значень показників потужності по фідерах обраної групи. Представлення

отримує дані з показниками з відповідного методу контролеру та засобами технології «**Razor**» формує таблиці годинних значень за добу.

3. **HalfHourTablePartial.cshtml** — часткове представлення, використовується представленням **Table.cshtml**. Відповідає за побудову таблиці півгодинних значень показників потужності по фідерах обраної групи. Представлення отримує дані з показниками з відповідного методу контролеру та засобами технології «**Razor**» формує таблиці півгодинних значень за добу.
4. **ExtandedTable.cshtml** — представлення реалізує побудову, наповнення даними та оновлення добової таблиці по обраному місяцю за передбаченими категоріями.

Дане представлення реалізоване без AJAX-форми і використання часткового представлення. В лістингу 12 наведено структуру **ExtandedChart.cshtml**.

Лістинг 12 *Структура* «ExtandedTable.cshtml»

```
<h2 id="legend"></h2>

<div id="exTable">

    <div id="captionDiv">
        <table id="caption"></table>
    </div>

    <div id="contentDiv">
        <table id="content"></table>
    </div>

    <div id="totalDiv">
        <table id="total"></table>
    </div>

</div>

<div id="filds">
```

```
    <div id="category">
        <select id="catSelect">
            <option value="cat">Категорія:</option>
        </select>
        <select id="catItems" disabled></select>
    </div>

    <div id="type">
        <select id="typeSelect">
            <option value="0">Тип:</option>
        </select>
    </div>

    <div id="dispMode">
        <div id="nativeDatePicker">
            <label for="monthVisit" >Оберіть місяць та
рік:</label>
            <input type="month" id="monthVisit"
name="monthVisit"/>
        </div>
     </div>

    <div id="includeMath">
        <input id="sumCols" type="checkbox" />
        <label id="sumCols" for="sumCols">Сума у
стовпцях</label>
        <input id="sumRows" type="checkbox" />
        <label id="sumRows" for="sumRows">Сума у
рядках</label>
    </div>

    <div id="sbmtDiv" class="fildItem">
        <button id="submit">OK</button>
    </div>

</div>
```

Згідно структури, яка представлена у наведеному лістингу, HTML-макет має три основні групи елементів:

- Заголовок
- Групу елементів, що є складовими таблиці.

- Групу елементів, що є складовими блоку конфігурації.

Група елементів-складових таблиці складається з загального контейнеру, в якому міститься три підконтейнери, кожен з яких містить елемент <table>. Таким чином, таблиця є зіставною і складається з трьох різних таблиць, а саме:

- Таблиця заголовків, у ролі яких виступають імена конкретних групи або фідерів.
- Таблиця контенту, яка відображає безпосередньо дані та суму показників у стовпцях.
- Таблиця сум, елементи якої відображають суму добових значень (значень у рядках).

Завдяки даному принципу групування досягається гнучкість конфігурування та стилізації зіставної таблиці. Структура підтаблиць, їх стовпчики, рядки та конкретні значення комірок формуються динамічно, за допомогою засобів JavaScript.

Група елементів блоку конфігурації складається з п’яти наступних блоків:

- Блок вибору категорії таблиці та конкретних елементів категорії.
- Блок вибору типу таблиці.
- Блок вибору дати.
- Блоку управління розрахунками.
- Блок підтвердження.

Перший блок містить два спадаючих списки. Перший дозволяє обрати категорію таблиць («Готові таблиці», «Групи», «Фідери»), а другий, залежно від категорії, містить список конкретних елементів категорії, а саме: конкретні попередньо сформовані таблиці або список груп, з можливістю множинного вибору, або список фідерів, з можливістю множинного вибору.

Другий блок містить спадаючий список, який дозволяє обрати тип таблиці. Наразі реалізовані механізми формування місячних таблиць, але з часом будуть реалізовані й інші типи таблиць, такі як:

- Таблиці місячних даних за рік.
- Таблиці річних даних.

Також, дана структура та формат таблиць були розроблені з перспективою реалізації таких форматів як:

- Таблиці з можливістю вибору конкретних днів, місяців або років.
- Таблиці з можливістю вибору часового діапазону.

Третій блок містить елемент «Календар» з конфігурацією, яка дозволяє обрати місяць та рік. Проте сама структура розроблена з перспективою розширення функціоналу. Також, враховуючи, що не всі браузери або конкретні версії цих браузерів надають можливість користуватися елементом «Календар», до цього блоку були додані елементи, що дублюють його для таких браузерів. На лістингу даний функціонал не відображено задля уникнення перевантаження загальної структури.

Четвертий блок містить два елементи типу «Check box», обробники подій яких відповідають за розрахунок суми у рядках та у стовпцях.

П’ятий блок містить кнопку, натискання на котру призводить до POST-запиту до серверу засобами AJAX та, як наслідок, оновлення даних з поточною конфігурацією або завантаження даних відповідно обраної конфігурації.

За обробку подій, конфігурування, а також наповнення та оновлення таблиць відповідає скрипт, що реалізований у файлі «**ExtandedTableScript.js**». Даний файл додається до представлення окремо.

У файлі **ExtandedTableScript.js** реалізуються наступні функції:

- **setStartFildsValues()** — виконує стартову ініціалізацію полів блоку конфігурації, крім поля конкретних елементів категорії. Встановлює обробники подій елементів, значення по замовчуванню. Викликає функцію **setCatFilds()** для ініціалізації поля конкретними елементами встановленої категорії.

- **setCatFilds()** — функція являється обробником події модифікації поля вибору категорії. В залежності від обраної категорії, виконує ініціалізацію або оновлення вмісту поля конкретних елементів категорії. Призначає обробник події, який реалізує логіку множинного вибору для елементів категорій «Групи» та «Фідери».
- **chosenOption()** — обробник події модифікації конкретних елементів категорії, який реалізує логіку множинного вибору для елементів категорій «Групи» та «Фідери». Записує в колекцію всі обрані користувачем групи або фідери в відповідні колекції, змінює стан обраних на неактивний, для запобігання вибору одного елемента декілька разів.
- **chosenType()** — обробник події модифікації поля вибору типу таблиці. Встановлює значення типу таблиці, яке було обрано користувачем.
- **sumColsChange()** — обробник події зміни стану елементу типу «Check box», який відповідає за візуалізацію та приховування суми у стовпцях (сума значень показників потужності всіх обраних елементів таблиці за конкретну добу).
- **sumRowsChange()** — обробник події зміни стану елементу типу «Check box», який відповідає за візуалізацію та приховування суми у рядках (сума добових значень показників потужності конкретного елементу таблиці за місяць).
- **extTableAjaxQuery()** — функція виконує POST-запит до серверу, засобами AJAX. У разі успішного виконання запиту викликає функцію **drowTables()** оновлення даних таблиці або, у випадку зміни конфігурації, повного її перемальовування
- **drowTables()** — Виконує формування коректного заголовку для таблиці, послідовно викликає функції, які відповідають за формування та візуалізацію частин зіставної таблиці.

- **drowCaption()** — функція відповідає за динамічне формування та візуалізацію таблиці заголовків.
- **drowContent()** — функція відповідає за динамічне формування та візуалізацію таблиці добових значень показників потужності за місяць. Порядок допоміжних та основних рядків відповідає порядку у таблиці заголовків.
- **drowTotal()** — функція відповідає за динамічне формування та візуалізацію таблиці місячних сум (сум у рядках). Порядок допоміжних та основних рядків відповідає порядку у таблиці заголовків.
- **buttonClick()** — обробник події натискання кнопки підтвердження конфігурації таблиці. Викликає функцію **extTableAjaxQuery()** з певним набором параметрів, в залежності від значення категорії таблиці.
- **eraseCatItemSelect()** — допоміжна функція, що виконує видалення вмісту поля вибору елементів конкретної категорії.
- Обробник події «**window.onload»**. Під час завантаження представлення, обробник події викликає функцію **setStartFildsValues()**, яка встановлює значення по замовчуванню для полів конфігурації, та функцію **extTableAjaxQuery()**, яка відповідає за POST-запит до серверу.

Також підключається файл стилів «**ExtandedTableStyleSheet.css**», у якому визначаються стилі для представлення та його елементів.

Представлення _Layout.cshtml реалізує майстер-сторінку, завдяки якій реалізується єдиний уніфікований зовнішній вигляд застосунку. Містить пункти меню, футер та контейнер для основного контенту сторінок. Також у майстер-сторінку підключені всі скрипти та стилі, які є загальновживаними для усіх представлень у застосунку.

3.4 Структура даних

3.4.1 Опис сутностей БД

Для застосунку була реалізована база даних **ASUDB**. Сутності та їх поля, у базі даних, описують структуру системи обліку та абстракції, які необхідні для роботи застосунку.

База даних містить наступні п’ять сутностей:

1. **Fiders —** описує властивості фідерів системи збору даних підприємства. Поля, які містить сутність, зберігають інформацію про назву, розташування, наявне на даному фідері обладнання, його тип та властивості, а також коефіцієнти та параметри, що використовуються для розрахунку значень показників споживання
2. **Groups —** описує властивості груп, у які об’єднуються фідери. Має зв’язок з сутністю Fiders у форматі багато-до-багатьох. Групи об’єднують фідери за локаційними та/або логічними ознаками, що дозволяє проводити розрахунки споживання за окремими підрозділами, частинами виробничого процесу, агрегатами.
3. **Associations —** описує властивості асоціації (групи груп), у які об’єднуються групи. Має зв’язок з сутністю Groups у форматі багато-до-багатьох.
4. **FiderGroups —** реалізує зв’язок багато-до-багатьох між сутностями **Fiders** та **Groups**.
5. **AssociationsOfGroups —** реалізує зв’язок багато-до-багатьох між сутностями **Groups** та **Associations**

У таблицях 1-5 вказаний опис сутностей бази даних **ASUDB**.

Таблиця 1 — *Опис сутності* ***Fiders***

Сутність Fiders		
Назва поля	**Опис**	**Тип**
Id (PK) (NOT NULL)	Ідентифікатор	integer
PassThroughChannelNumber (NOT NULL)	Наскрізний номер каналу контролеру	integer
Name	Назва фідеру	nvarchar
ElectricityTransformationParams	Параметри трансформації по струму	nvarchar
ElectricityTransformationCoefficient (NOT NULL)	Коефіцієнт трансформації по струму	real
VoltageTransformationParams	Параметри трансформації по напрузі	nvarchar
VoltageTransformationCoefficient (NOT NULL)	Коефіцієнт трансформації по напрузі	real
ImpulsesPerKilowattPerHour (NOT NULL)	Кількість імпульсів на кіловат-годину	integer
FinalTransformationCoefficient (NOT NULL)	Остаточний коефіцієнт трансформації	real
MaxPower (NOT NULL)	Максимальна потужність фідеру	integer
LoadType	Тип електричного навантаження	nvarchar
CounterName	Назва лічильника	nvarchar
CounterNumber	Номер лічильника	nvarchar
Description	Опис	nvarchar

Таблиця 2 — *Опис сутності* ***Groups***

Сутність Groups		
Назва поля	**Опис**	**Тип**
Id (PK) (NOT NULL)	Ідентифікатор	integer
Name	Повна назва групи	nvarchar
ShortName	Коротка назва групи	nvarchar
MaxPower	Максимальна потужність групи	integer
DateOfLastRevision	Дата останньої ревізії групи	datetime
Description	Опис	nvarchar

Таблиця 3 — *Опис сутності* ***Associations***

Сутність Associations		
Назва поля	**Опис**	**Тип**
Id (PK) (NOT NULL)	Ідентифікатор	integer
Name	Повна назва асоціації	nvarchar
ShortName	Коротка назва асоціації	nvarchar
MaxPower	Максимальна потужність групи	integer

DateOfLastRevision	Дата останньої ревізії асоціації	datetime
Description	Опис	nvarchar
DisplayName	Назва для відображення асоціації у застосунку	nvarchar

Таблиця 4 — *Опис сутності* ***FiderGroups***

Сутність FiderGroups		
Назва поля	**Опис**	**Тип**
Id (PK) (NOT NULL)	Ідентифікатор	integer
Sign	Знак фідеру у групі	nvarchar
Fider_Id (FK)	Ідентифікатор фідеру	integer
Group_Id (FK)	Ідентифікатор групи	integer

Таблиця 5 — *Опис сутності* ***AssociationsOfGroups***

Сутність AssociationsOfGroups		
Назва поля	**Опис**	**Тип**
Id (PK) (NOT NULL)	Ідентифікатор	integer
Sign	Знак групи в асоціації	nvarchar
Association_Id (FK)	Ідентифікатор асоціації	integer
Group_Id (FK)	Ідентифікатор групи	integer

3.4.2 Опис бінарного файлу показників споживання

Дані про споживання містяться у 30-хвилинних (півгодинних) бінарних файлах, у яких записано 30 блоків по 1024 значення типу short. Порядковий номер кожного значення співпадає з наскрізним номером каналу фідеру.

На рисунку 7 зображено вміст півгодинного файлу.

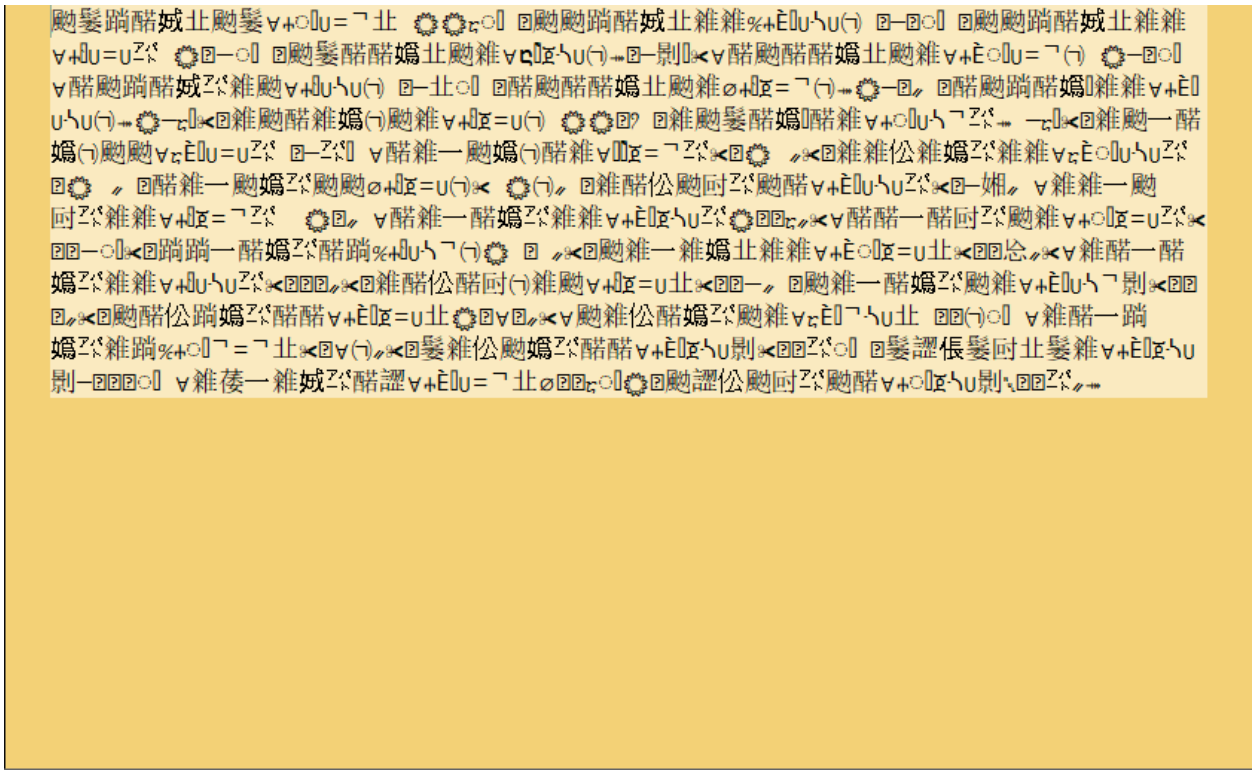

Рисунок *7 – Вміст 30-хвилинного файлу*

3.5 Проєкт інтерфейсу

В процесі розробки інтерфейсу, прийняті рішення, які стосуються колірної гами, шрифтів, розмірів, форми і розташування елементів інтерфейсу, обумовлені ергономічними вимогами. Користувачі мають взаємодіяти з додатком на протязі 8 годин або більше, тому інтерфейс застосунку розроблено таким чином, щоб чинити якомога менший вплив на органи сприйняття користувача.

Серед зручних особливостей інтерфейсу:

- Великий шрифт.

- Проста система навігації.
- Ергономічна колірна гама.
- Засоби відображення даних займають більшу частину робочого простору.

Інтерфейс застосунку розроблявся з урахуванням особливостей інтерфейсу попередника, що дозволить користувачу використовувати досвід роботи з попереднім застосунком і виключити необхідність перевчатися та довго звикати до нового інтерфейсу.

Нижче наведено скриншоти з прикладами інтерфейсу.

На рисунку 8 зображено діаграму поточного споживання електроенергії.

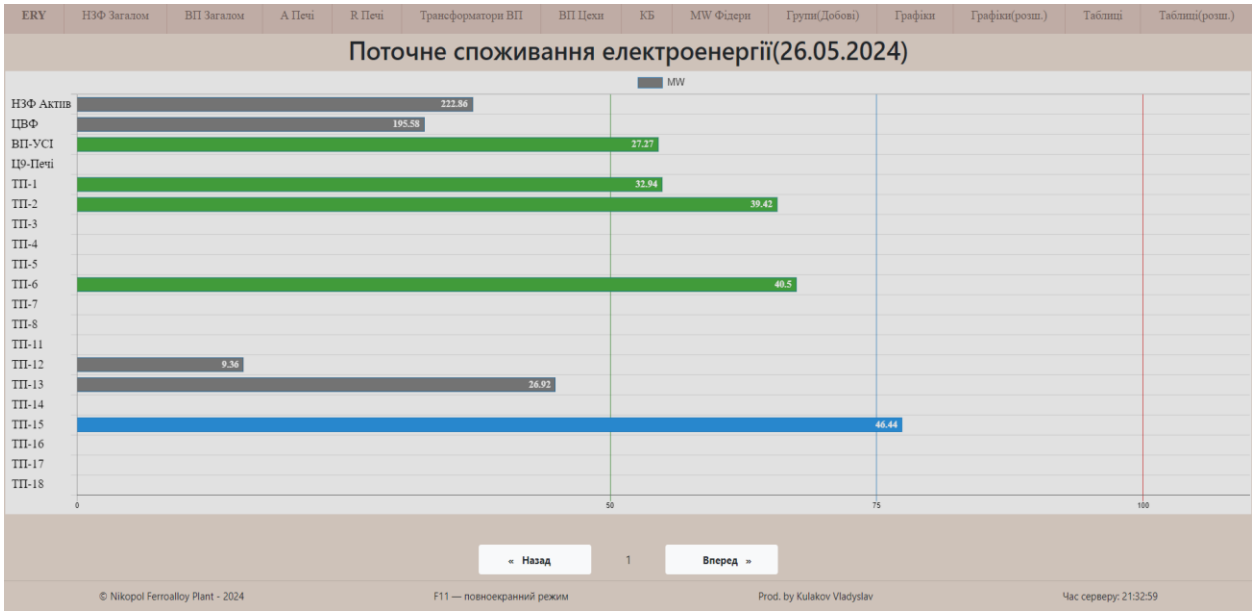

Рисунок *8 – Діаграма поточного споживання електроенергії*

На рисунку 9 зображено графік хвилинних значень потужності за добу по обраним групам.

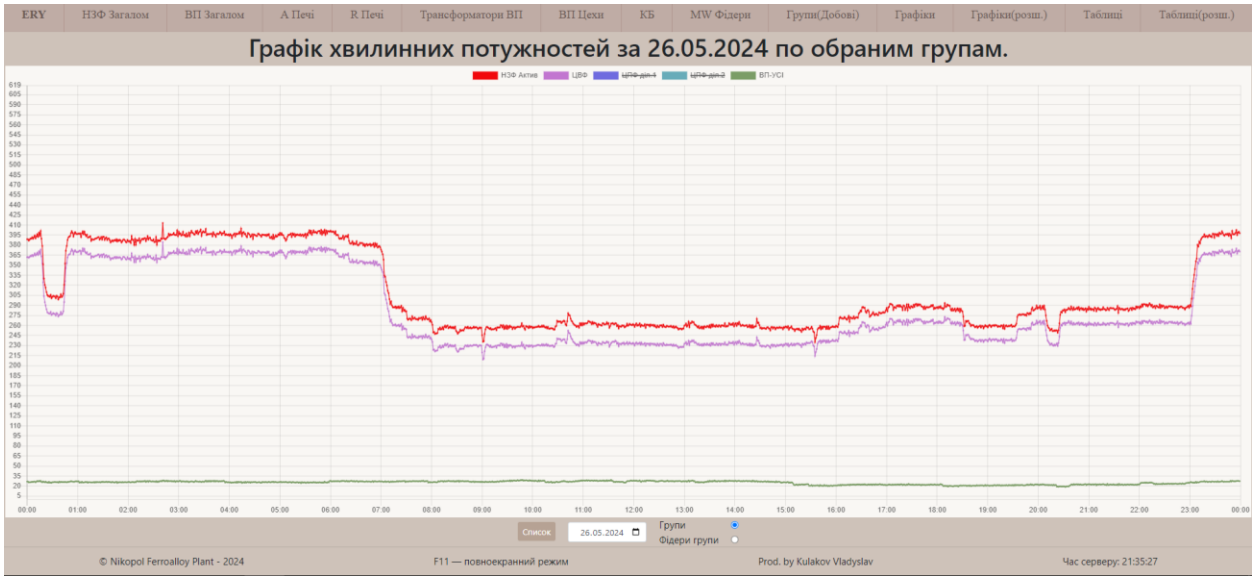

Рисунок 9– *Графік хвилинних значень потужності*

На рисунку10 зображено діаграму півгодинних значень потужності за добу, з можливістю вибору групи у спадаючому списку.

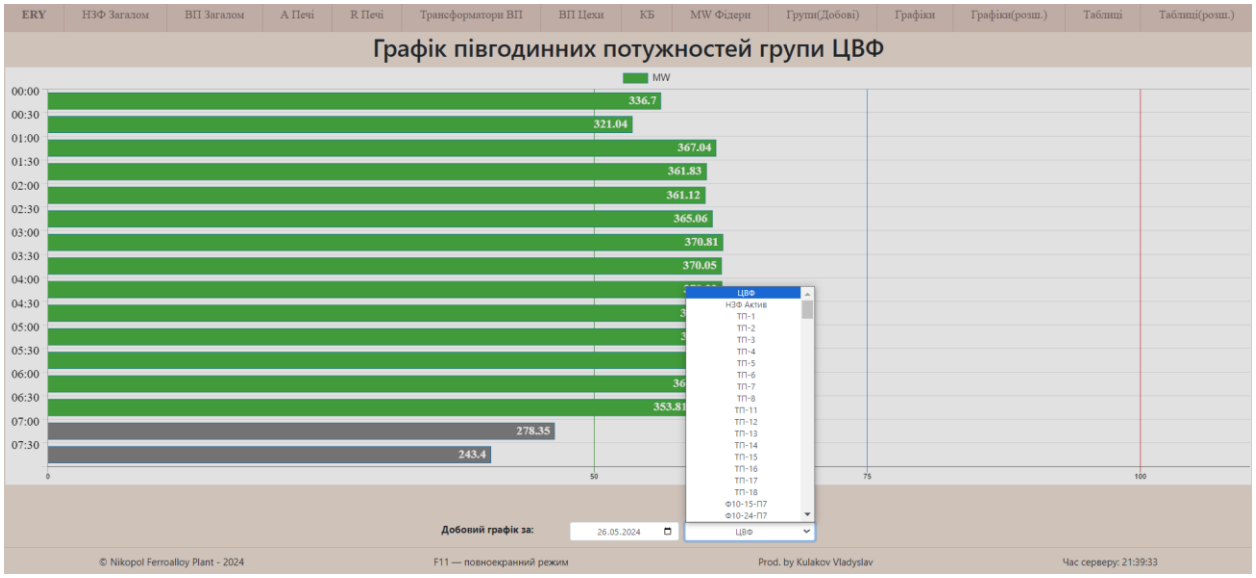

Рисунок *10 – Діаграма півгодинних значень потужності за добу*

3.6 Реалізація і тестування

3.6.1 Реалізація

Фізичні характеристики поточної версії системи:

Об’єм коду:

1. C# — 104 кілобайти, ~2270 рядків.
2. JavaScript — 68 кілобайт, ~2190 рядків

Кількість модулів: 19

Кількість і об’єм в кілобайтах:

1. Веб- застосунок: 1037300 кілобайт

Кількість користувачів: 65

3.6.1 Тестування

Було проведено ручне тестування застосунку, з метою виявлення відхилень від запланованих сценаріїв користування. Враховуючи об’єм програми та специфіку предметної області ручне тестування (Manual Testing) вбачається найбільш доцільним, тому що дозволяє пропрацювати справжні, очевидні та неочевидні сценарії користування та порівняти отримані результати з очікуваними. Тестування не виявило відхилень у роботі інтерфейсу користувача.

ВИСНОВКИ

1. Результати вивчення предметної області, потреб підприємства і наслідків можливого виходу з ладу наявного застосунку для системи обліку електроенергії свідчать про необхідність розробки нової версії застосунку.
2. Проведено аналіз можливих шляхів вирішення проблеми та порівняння комерційних аналогів застосунку на ринку. Вивчення свідчить, що наявні аналоги не відповідають вимогам та потребам підприємства.
3. Було визначено основні функціональні вимоги та вимоги до інтерфейсу користувача. Розроблено архітектуру додатку, що відповідає потребам та вимогам підприємства.
4. В результаті виконання кваліфікаційної роботи було створено веб-застосунок для системи обліку електроенергії на підприємстві, проведене ручне тестування (Manual Testing) застосунку, з метою перевірки сценаріїв взаємодії користувача з системою, яке не виявило відхилень у роботі інтерфейсу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мовчан В. П., Бережний М. М. Основи металургії. Дніпропетровськ : Пороги, 2001.

2. Автоматизована система комерційного обліку електроенергії (АСКОЕ). Енергетична компанія ГРАНД ТЕСЛА. URL: https://grandtesla.com.ua/service/askoe (дата звернення: 28.02.2024).

3. Все про продукт MDM/ACOE "ЕнергоЦентр". VIK SOFT. URL: https://www.asue.com.ua/page_info-2-1-0.html (дата звернення: 28.02.2024).

4. Програмне забезпечення АСКОЕ. ТОВ "МІРОТЕКС". URL: https://miroteks.com.ua/ua/p30745478-programmnoe-obespechenie-askueaskoe.html (дата звернення: 28.02.2024).

5. Snyder T. Programming ASP.NET MVC 4. USA : O'Reilly, 2012.

6. Marrs T. JSON at Work: Practical Data Integration for the Web. O'Reilly Media, 2017. 376 p.

7. Flanagan D. JavaScript: The Definitive Guide. 6th ed. Beijing : O’Reilly, 2011. 1078 p.

8. Usage · Chart.js documentation. Chart.js | Open source HTML5 Charts for your website. URL: https://www.chartjs.org/docs/2.9.4/getting-started/usage.html (date of access: 20.04.2024).

9. Lerman J. Programming entity framework: Code first. Sebastopol, CA : O'Reilly Media, 2011.

10. jQuery AJAX get() and post() Methods. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/jquery/jquery_ajax_get_post.asp (date of access: 23.04.2024).

Декларація
академічної доброчесності
здобувача ступеня вищої освіти ЗНУ

Я, Кулаков Владислав Валерійович, студент 4 курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти ipz20bd-214@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Розробка веб-додатку обліку електроенергії підприємства з використанням ASP.NET**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 14.06.2024 _____ Кулаков Владислав Валерійович
(підпис) (прізвище та ініціали) (студент)

Дата 15.06.2024 _____ Полякова Наталія Петрівна
(підпис) (прізвище та ініціали) керівник)