

**Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej**

Praca dyplomowa

*System Web do zarządzania danymi z pomiarów parametrów jakości energii elektrycznej*

*A web system for managing data from power quality parameters measurements*

Autor: *Szymon Lis*

Kierunek studiów: Automatyka i Robotyka

Opiekun pracy: dr inż. Krzysztof Piątek

Kraków, 2023

Spis treści

[1. Wstęp 3](#_Toc143014360)

[1.1. Opis problemu 3](#_Toc143014361)

[1.2. Przegląd istniejących rozwiązań 4](#_Toc143014362)

[1.3. Cel i zakres pracy 4](#_Toc143014363)

[2. Projekt 6](#_Toc143014364)

[2.1. Wymagania funkcjonalne i niefunkcjonalne 6](#_Toc143014365)

[2.2. Wybrane technologie 7](#_Toc143014366)

[2.3.1. ASP.NET Core 7](#_Toc143014367)

[2.3.2. Razor 9](#_Toc143014368)

[2.3.3. JavaScript 10](#_Toc143014369)

[2.3.4. SQL Server, MongoDB 10](#_Toc143014370)

[2.3. Architektura systemu 11](#_Toc143014371)

[2.3.1. MVC 11](#_Toc143014372)

[2.3.2. Wzorzec Repozytorium/Serwis 12](#_Toc143014373)

[2.4. Schemat bazy danych 14](#_Toc143014374)

[3. Implementacja proponowanego rozwiązania 15](#_Toc143014375)

[3.1. Import danych 15](#_Toc143014376)

[3.2. Dostęp do danych 17](#_Toc143014377)

[3.3. Szablony 19](#_Toc143014378)

[3.4. Raporty 20](#_Toc143014379)

[4. Interfejs użytkownika 22](#_Toc143014380)

[4.1. Argon Dashboard 23](#_Toc143014381)

[4.2. Opis funkcjonalności 24](#_Toc143014382)

[4.3. Raport 31](#_Toc143014383)

[5. Porównanie modeli bazy danych 35](#_Toc143014384)

[5.1. Założenia teoretyczne 35](#_Toc143014385)

[5.2. Badane podejścia 35](#_Toc143014386)

[5.3. Porównanie wyników 35](#_Toc143014387)

[6. Podsumowanie 35](#_Toc143014388)

[6.1. Wnioski 35](#_Toc143014389)

[6.2. Możliwości rozwoju 35](#_Toc143014390)

[Bibliografia 36](#_Toc143014391)

**Spis ilustracji**

[Rysunek 1. Stack technologiczny budowanej aplikacji. 7](#_Toc143014392)

[Rysunek 2. Fragment składni silnika Razor. 9](#_Toc143014393)

[Rysunek 3. Model wzorca MVC. 11](#_Toc143014394)

[Rysunek 4. Model wzorcu Repozytorium/Serwis. 13](#_Toc143014395)

[Rysunek 5. Schemat modelu relacyjnego bazy danych. 14](#_Toc143014396)

[Rysunek 6. Podział parametru na kategorie. 16](#_Toc143014397)

[Rysunek 7. Schemat kontrolera importu danych. 17](#_Toc143014398)

[Rysunek 8. Schemat kontrolera podglądu danych. 19](#_Toc143014399)

[Rysunek 9. Schemat kontrolera szablonów. 20](#_Toc143014400)

[Rysunek 10. Schemat kontrolera Raportów. 21](#_Toc143014401)

[Rysunek 11. Przykładowy interfejs zbudowany z wykorzystaniem Argon Dashboard 23](#_Toc143014402)

[Rysunek 12. Interfejs użytkownik – Strona główna 24](#_Toc143014403)

[Rysunek 13. Interfejs użytkownika - import danych. 25](#_Toc143014404)

[Rysunek 14. Interfejs użytkownika - okno wyboru importowanego pliku 25](#_Toc143014405)

[Rysunek 15. Interfejs użytkownika - odczytane nagłówki 25](#_Toc143014406)

[Rysunek 16. Interfejs użytkownika - edytor nagłówków 26](#_Toc143014407)

[Rysunek 17. Interfejs użytkownika - zapis do bazy 26](#_Toc143014408)

[Rysunek 18. Interfejs użytkownika – komunikat 27](#_Toc143014409)

[Rysunek 19. Interfejs użytkownika – podgląd 27](#_Toc143014410)

[Rysunek 20. Interfejs użytkownika – szablony 28](#_Toc143014411)

[Rysunek 21. Interfejs użytkownika - formularz dodawania/edycji szablonu 29](#_Toc143014412)

[Rysunek 22. Interfejs użytkownika - Raport 29](#_Toc143014413)

[Rysunek 23. Interfejs użytkownika - formularz generowania nowych raportów 30](#_Toc143014414)

[Rysunek 24. Interfejs użytkownika - formularz wysyłki wiadomości e-mail 30](#_Toc143014415)

[Rysunek 25. Raport - strona tytułowa 31](#_Toc143014416)

[Rysunek 26. Raport - opis norm 32](#_Toc143014417)

[Rysunek 27. Raport - wyniki zgodności z normą 33](#_Toc143014418)

[Rysunek 28. Raport - przykładowy przebieg 34](#_Toc143014419)

# 1. Wstęp

## 1.1. Opis problemu

Energia elektryczna jest kluczowym elementem nowoczesnej cywilizacji. Jakość dostarczanej energii elektrycznej ma bezpośredni wpływ na funkcjonowanie oraz wydajność urządzeń elektrycznych. Monitorowanie i zarządzanie danymi z pomiarów parametrów jakości energii elektrycznej jest zatem ważną kwestią szczególnie w czasach wszechobecnej rewolucji energetycznej. Rosnąca ilość alternatywnych źródeł energii takich jak instalacje fotowoltaiczne przynosi wyzwania dla stabilności i jakości energii elektrycznej. Te zdecentralizowane i zmiennokierunkowe źródła mogą wprowadzać zmienność w dostawach energii i generować zjawiska harmoniczne, co potencjalnie wpływa na jakość energii i wymaga zaawansowanych systemów monitorowania i zarządzania.

Aktualnie wykorzystywane urządzenia do monitorowania jakości energii elektrycznej, takie jak analizatory z serii PQBox lub PQM, generują duże ilości danych. Te dane, można wyeksportować do plików CSV, XSLX, jednak manualna analiza takich wyników wydaje się zadaniem bardzo czasochłonnym, gdyż dane te są rozproszone, brakuje również jednolitego ustandaryzowania tych danych. W zależności od analizatorów można uzyskać różne parametry. Dlatego naturalnym rozwiązaniem takiego problemu wydaje się ustandaryzowanie uzyskiwanych danych oraz import do środowiska które pozwoli na łatwą analizę, sprawdzenie warunków stawianych przez normy.

## 1.2. Przegląd istniejących rozwiązań

Obecnie na rynku istnieje wiele różnych producentów analizatorów energii elektrycznej, często można się spotkać z oprogramowaniem które pozwala przygotowywać wykresy, sprawdzać zadane warunki, jednak są to oprogramowania dedykowane pod poszczególny model analizatora. Przykładem takie oprogramowania może być oprogramowanie WinPQ, który umożliwia ocenę podstawowych danych pomiarowych, przygotowanie przebiegów poszczególnych wartości w czasie jednak obsługuje on tylko analizatory z serii PQI DE oraz PQI DA. Innym przykładem takiego rozwiązania może być oprogramowanie PQView, oprogramowanie to nie jest dedykowane dla konkretnych modeli urządzeń, a stworzone zostało do obsługi standardu *PQDIF* opracowanego przez Electric Power Research Institute oraz Electrotec Concepts [TODO przycytować coś i przypis o PQVIEW]. Jednak rynek analizatorów jest dość zróżnicowany i istnieje wiele wiodących producentów którzy nie obsługują tego standardu, takim przykładem jest firm Sonel z serią analizatorów PQM.

Ponadto znacząca część takich rozwiązań opiera się na aplikacjach desktopowych, co wiąże się z wieloma trudnościami, jak sprawy licencyjne na większej ilości maszyn, potrzeba fizycznej instalacji oprogramowania na maszynie

## 1.3. Cel i zakres pracy

Celem niniejszej pracy magisterskiej jest zaprojektowanie i implementacja systemu internetowego do zarządzania danymi z pomiarów parametrów jakości energii elektrycznej.

W pracy zostanie przedstawione działanie systemu, który umożliwi ładowanie danych z plików CSV, XLSX, tworzenie szablonów, oraz generowanie raportów, które umożliwią sprawdzenie zgodności wskazanych danych z normami wskazanymi w *Rozporządzeniu Ministra Gospodarki z dnia 4 maja 2007 r. w sprawie szczegółowych warunków funkcjonowania systemu elektroenergetycznego, Dziennik Ustaw Nr 93, poz. 623* **[TODO przypis]**

Jednym z kluczowych elementów pracy jest przeprowadzenie analizy wymagań dla proponowanego systemu. Wymagania te obejmują między innymi, wybór odpowiedniej technologii, proces doboru oraz analiza odpowiedniej bazy danych, proces rozpoznawania poszczególnych parametrów z nagłówków generowanych przez analizator, zaproponowanie generatora szablonów.

W trakcie pracy przeprowadzone zostanie porównanie różnych propozycji modeli bazodanowych. Porównane zostaną podejścia oparte o relacyjną, nierelacyjną bazę danych. Jak również podejście hybrydowe wykorzystujące obie technologie.

# 2. Projekt

Zważywszy na problem opisany we wstępie i celu pracy, pierwszym etapem realizacji takiego systemu jest postawienie wymagań, a następnie na ich podstawie dobór odpowiednich technologii, dobraniu wzorców architektonicznych i w razie potrzeby odpowiedniego systemu zarządzania danymi.

## 2.1. Wymagania funkcjonalne i niefunkcjonalne

Do wymagań funkcjonalnych, czyli takich które opisują funkcjonalności które realizować ma system. Poniżej przedstawiono postawione wymagania funkcjonalne:

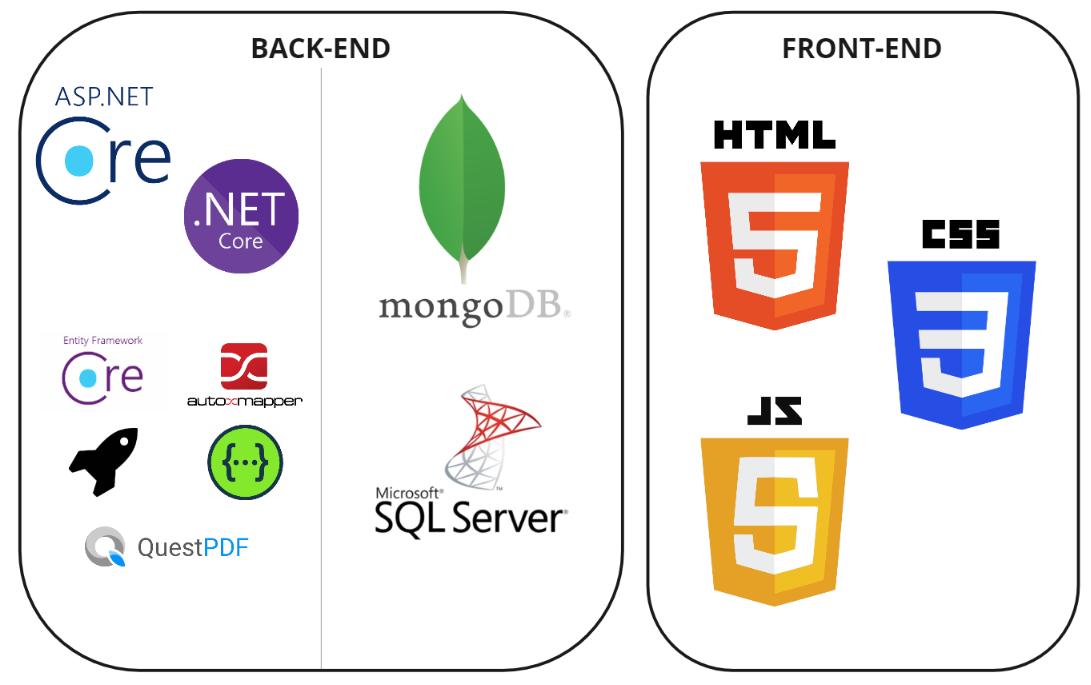
* Import danych z plików CSV, XSLX, generowanych przez analizator. Import powinien obejmować automatyczne rozpoznawanie pliku, typu danych, separatorów, oraz wykrywać i oznaczać brakujące, błędnie sformatowane dane.
* Automatyczne rozpoznawanie parametrów energii elektrycznej. Detekcja i kategoryzacja poszczególnych parametrów powinna opierać się o nagłówki importowanych plików.
* Tworzenie szablonów na podstawie których wygenerowane zostaną raporty. Szablony powinny być zapisywane i przechowywane w bazie danych. Każdy szablon powinien zawierać zestaw predefiniowanych wykresów przebiegów poszczególnych parametrów na zadanym okresie.
* Sprawdzanie zgodności danych z zadanego okresu z normami przedstawionymi w rozporządzeniu Ministra Gospodarki z dnia 4 maja 2007 roku [TODO przypis]. W przypadku niespełnienia takich wymagań, system powinien podać przybliżone informacje o niezgodności.
* Generowanie raportów na podstawie zdefiniowanych szablonów, dla zadanego okresu czasu oraz wybranego punktu pomiarowego. Raport powinien zostać wygenerowany w formacie PDF. System powinien udostępniać możliwość pobrania oraz wysyłkę wygenerowanego raportu przez system e-mail.

Wymagania niefunkcjonalne definiują działanie systemu, są to ograniczenia i standardy które system musi spełniać. Poniżej przedstawiono postawione wymagania niefunkcjonalne:

* System, ma opierać się na technologii webowej, nie powinien wymagać od użytkownika dodatkowych instalacji
* System powinien zapewniać efektywne przetwarzanie dużych ilości danych, tj. brak limitów na wielkość pliku
* System powinien być łatwy do użycia dla użytkowników, z intuicyjnym interfejsem użytkownika i jasnymi instrukcjami
* System powinien zapewniać kompatybilność dla różnych wariacji importowanych danych, powinien obsługiwać różne systemy
* Zbierane dane powinny być zapisywane w sposób ciągły, tj. zbiór danych powinien być unifikowany i łączony niezależnie od ilości i formatu importowanych plików

## 2.2. Wybrane technologie

Dobór odpowiednich narzędzi i technologii powinien być podyktowany przez wszystkim możliwościami w spełnieniu postawionych wymagań. W następnym kroku powinny zapewniać skalowalność, bezpieczeństwo oraz umożliwiać integrację z jak największa ilością systemów. Wybór konkretnych rozwiązań ma kluczowe znaczenie dla dalszego kształtu prac oraz wpływa na ostateczną użyteczność i przydatność systemu. Dlatego warto stawiać na technologię, nowe, z długim wsparciem technicznym od firm z ustabilizowaną pozycją rynkową. Jedną z takich technologii jest technologia *.NET* wraz z językiem *C#* firmy Microsoft i to właśnie na tych technologiach została oparta duża część systemu. Wybrane technologie zostały ukazane na Rys. 1



Rysunek 1. Stack technologiczny budowanej aplikacji.

### 2.3.1. ASP.NET Core

Kluczowym wyborem technologii dla opisywanego systemu, było wybranie głównej platformy. Zgodnie z celami pracy i wymaganiami stawianymi systemowi, wybór padł na *ASP.NET Core*. Jest to wieloplatformowa platforma typu open source . Umożliwia ona tworzenie wielu różnych aplikacji webowych. Jest to technologia bieżąco rozwijana przez Microsoft, co gwarantuje ciągły rozwój, oraz wsparcie techniczne. Dla tej technologii stworzono dedykowane środowisko programistyczne Visual Studio, które pozwala znacząco skrócić czas pracy przez wiele wbudowanych prototypów, łatwe testowanie, oraz narzędzia wspomagające budowę aplikacji w technologiach *.NET*. Technologia *ASP.NET Core* posiada wbudowane wsparcie dla różnych systemów zarządzania bazami danych, co pozwala na szybkie i elastyczne zarządzanie danymi. Technologia ta pozwala tworzyć aplikację wykorzystując takie języki programowania jak *C#, F#* lub *Visual Basic .NET*. Na potrzeby budowy systemu wybrano język *C#*, ze względu na jego wszechstronność, obiektowość oraz największe wsparcie ze stronu firmy Microsoft. Framework MVC sprawdza się przy tworzeniu aplikacji internetowych i interfejsów API, ponieważ umożliwia wykorzystanie wzorca MVC (Model View Controller), technologii Razor Pages, oraz łatwe integracje z systemami baz danych. Ponadto do stworzenia aplikacji wykorzystano kilka gotowych bibliotek, dostępnych w systemie *NuGet*, który jest otwartoźródłowym, domyślnym systemem zarządzania pakietami dla platfromy *.NET.* Wśród nich znalazły następujące bilbioteki i frameworki:

* Automapper – biblioteka służąca do automatycznego, mapowania (kopiowania) wartości z jednego modelu do drugiego. Działanie takiego frameworku opiera się na refleksji, dzięki czemu pola z jednego modelu zostaną automatycznie dopasowane opierając się na podobieństwie nazw i typów. Szczególnie przydatna przy wykorzystaniu konceptu *Obiektu transferu danych* (*DTO, ang Data transfer object*).
* Json.Net – biblioteka wspomagająca operację z wykorzystaniem standardu formatu *JSON* takie jak serializacja, umożliwienie operacji *LINQ* na formatach *JSON.* W projekcie wykorzystana głównie do wspomagania operacji powiązanych z MongoDB, jako, że baza ta przechowywuje dane w formacie BSON (Binary JSON) który jest ściśle powiązany z JSON
* Entity Framework Core – lekka bibilioteka, służaca do mapowania obiektowo relacyjnego (*ORM, ang Object-relational mapping).* Umożliwia deweloperom na pracę z danymi za pomocą obiektów, wyspecyfikowany dla problemów domenowych. Pozwala na znaczne pominięcie podstawowych szczegółów bazy danych.
* QuestPDF – bilbioteka, pozwalająca na zwinne tworzenie dokumentów w formacie PDF, wykorzystująca podejście „code-only”, czyli cała logika i struktura dokumentu jest stworzona na bazie kodu źródłowego. Pozwala to programiście na pełną kontrolę nad dokumentem nie wychodząc poza, kod źródłowy aplikacji. Biblioteka ta pozawala na dodawanie wielu bazowych elementów takich jak teksty, obrazy, tabele, tła, granice. Razem z biblioteką udsotępnione zostało narzędzie które pozwala w czasie rzeczywistym obserwować tworzony pdf, wystarczy zapis kodu źródłowego, a w narzędziu zobaczymy plik pdf z naniesionymi zmianami.
* ScottPlot – poboczna biblioteka służąca do wygenerowanie wykresów na podstawie danych uzyskanych z bazy danych , oraz umieszczeniu ich w pliku PDF przy pomocy *QuestPDF*.
* Swasbuckle – pakiet bibliotek, ułatwiający generowanie dokumentacji interfejsu API opartej na standardzie OpenAPI. Pomaga testować oraz usuwać błędy, Generuje interfejs który pozwala w łatwy sposób wprowadzić dane wejściowe i przetestować daną komende, pozwala łatwo sprawdzić walidacje danych, czy poprawność zwracanych błędów bez pisania specjalnego testowego klienta który będzie konsumował tak wystawione API.

### 2.3.2. Razor

Razor to silnik generujący widoki wprowadzony w wersji trzeciej ASP.NET MVC zastępując bardziej złożoną i mniej przystępną technologię WebForms. Technologia ta działa po stronie serwera i pozwala na dynamiczne budowanie widoków skoncentrowanych na stronach. Umożliwia on używanie fragmentów kodu po stronie serwera do osadzenia w znacznikach HTML, co pozwala na dynamiczną generację treści dla przeglądarki. Ze strony architektury, technologia Razor pozwala na implementację wzorca MVC, promując podział zadań. W praktyce silnik ten pozwala na mieszanie składni języka C# oraz znaczników HTML. Dla zobrazowania przedstawiono przykładowy fragment kodu z silnika Razor: Rys. 2

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Rysunek 2. Fragment składni silnika Razor.

### 2.3.3. JavaScript

JavaScript to język programowania, który jest standardem wśród technologii do tworzenia interaktywnych stron internetowych. W tym projekcie JavaScript, desygnowany jest jako narzędzie wspomagające do manipulacji elementami strony. Ponieważ JavaScript działą po stronie klienta, pozwala na natychmiastowe reakcję działania użytkownika, bez przeładowywania strony. Tym samym strona internetowa staje się bardziej responsywna a interfejsy użytkownika będą bardziej dynamiczne i interaktywne. Użycie JavaScript, w połączeniu z technologią ASP.NET oraz silnikiem generującym Razor, zapewniają kompleksowe i efektywne narzędzie do tworzenia prostych jak i bardzo rozbudowanych stron internetowych.

### 2.3.4. SQL Server, MongoDB

Podczas wyboru odpowiednich technologii do zarządzania danymi w systemie, kluczową rolę odgrywa zarówno natura danych, jak i specyfika projektu. W ramach realizowanej pracy zdecydowano się na wykorzystanie dwóch różnych systemów zarządzania bazami danych – MS SQL oraz MongoDB.

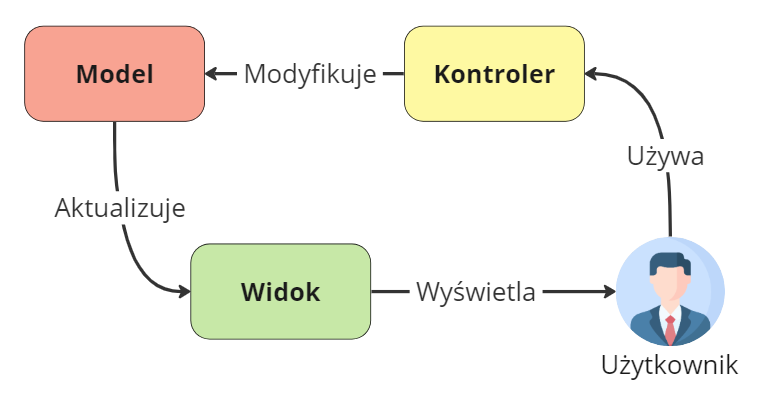
Jako relacyjną bazę danych wybrano MS SQL, jest to naturalny wybór dla aplikacji opartych na platformach .NET. System ten tak jak technologia .NET jest opracowywany przez firmę Microsoft, czego następstwem jest ogromna łatwość integracji między tymi platformami. Również Środowisko Visual Studio jest dobrze przystosowane do obsługi tego systemu baz danych i oferuję wiele wbudowanych narzędzi wspomagających taką integrację.

Na nierelacyjną bazę danych wybrano natomiast MongoDB. Jest to system zarządzania bazami danych typu NoSQL, oferuje możliwość przechowywania danych w formacie dokumentów (BSON). Szczególnie ważnym aspektem wyboru MongoDB jest wsparcie dla kolekcji typu *Timeseries*, która jest szczególnie przydatna w przechowywaniu danych z szeregów czasowych – a takie dane będą importowane przez analizatory jakości energii elektrycznej.

## 2.3. Architektura systemu

### 2.3.1. MVC

MVC czyli Model-View-Controller to wzorzec architektonicznym który jest często wykorzystywany w projektowaniu aplikacji internetowych. Pozwala on na osiągniecie jednej z kluczowych zasad dotyczących architektury system czyli separacji problemów. Zasada ta opisuje, iż oprogramowanie musi być podzielone na podstawie wykonywanej pracy. Każda z takich sekcji powinna być odpowiedzialna za konkretną, niezależną funkcję. Ten koncept jest kluczowym elementem wzorca jakim jest Model-View-Controller. Separacja jest realizowana przez podział aplikacji na komponenty: model, widok, kontroler. Każdy z nich odpowiadający za konkretną sferę działania. Relacja ta przedstawiona na Rys. 3



Rysunek 3. Model wzorca MVC.

Model jest reprezentacją pewnego stanu aplikacji oraz logiki biznesowej która powinna być przez niego wykonywana. Logika biznesowa wraz z logiką utrzymującą stan aplikacji powinna być enkapsulowana w modelu.

Widok jest odpowiedzialny za prezentację treści przez interfejs użytkownika. Logika znajdująca się w widokach powinna być związana jedynie z prezentacją treści. Widoki powinny prezentować użytkownikowi dane z modelu przekazanego przez kontroler. Ten element jest również odpowiedzialny za to z czym użytkownik wchodzi w interakcję, czyli którą metodę i z jakimi danymi przekazać do kontrolera.

Kontroler odpowiada za komunikację między modelem i widokiem. Jest to element który reaguje na dane wejściowe i interakcje przekazane z widoku. Wykonuje logikę biznesową, wywołuje modyfikacje modelu, dobiera widok który ma być wyświetlony użytkownikowi.

### 2.3.2. Wzorzec Repozytorium/Serwis

Wzorzec architektoniczny MVC w swoim podejściu jest dość ogólny, jego zasady opisują separację elementów prezentacji danych, manipulowania danymi oraz komunikacji między tymi elementami. Jednak taki ogólny podział często nie jest wystarczający, ponieważ zarządzanie danymi to często rozbudowany i skomplikowany kod, dlatego łączenie wzorców MVC, oraz Repozytorium/Serwis jest powszechną znaną praktyką w tworzeniu oprogramowania. Połączenie tych wzorców pomoże zapewnić organizację kodu, oraz pełną separację obowiązków. We wzorcu Repozytorium/Serwis możemy oddzielić warstwę prezentacji danych oraz komunikacji z modelem (odpowiednio widoków i kontrolerów) od dostępu do danych i logiki biznesowej aplikacji.

Warstwa dostępu do danych (DAL – data access layer) składa się z repozytoriów oraz powiązanych z nimi modeli domenowych. Warstwa ta ma za zadanie zapewnić bezpośredni dostęp do danych komunikując się z bazą danych. Pojedyncze repozytorium powinno zapewnić dostęp do pojedynczej encji danych. Biorąc pod uwagę wymagania stawiane systemowi przedstawione w rozdziale 2.1, można wyróżnić, że aplikacja będzie składała się z repozytorium próbek danych z analizatora, ale również takich jak repozytorium szablonów, punktów pomiarowych. Takie repozytorium składa się najczęściej z prostych operacji typu CRUD (Create, Read, Update, Delete)

Warstwa biznesowa składająca się z serwisów służy do operacji bardziej złożonych niż proste operację CRUD. W warstwie tej powinna znaleźć się główna logika i reguły związane z biznesowym procesem tworzonej aplikacji. Często zdarza się że jeden serwis korzysta z kilku repozytoriów, złą praktyką również nie jest wykorzystanie w jednym serwisie innego serwisu, jednak należy mieć na uwadze, iż pojedynczy serwis powinien odpowiadać za operację powiązane z jednym modułem, najczęściej powiązany z pojedynczym widokiem. Ponownie analizując wymagania stawiane systemowi przedstawione w rozdziale 2.1, można łatwo wyróżnić, że jednym z serwisów będzie moduł odpowiadający za szablony. Sam zapis szablonów, czyli operacje dodania, modyfikacji będą znajdować się w repozytorium, natomiast logika związana z walidacją, wyborem odpowiednich danych pomiarowych w szablonie, zakodowaniem ich do odpowiedniego formatu zajmować musi się serwis. Takie podejście na spełnienie jednej z fundamentalnych zasad dobrych praktyk programowania obiektowego - *Single Responsibility Principle* – zasada pojedynczej odpowiedzialności.

Podział odpowiedzialności w architekturze z zachowaniem wzorca Repozytorium/Serwis dobrze obrazuje schemat przedstawiony w Rys. 4

Obraz zawierający tekst, diagram, zrzut ekranu, Czcionka

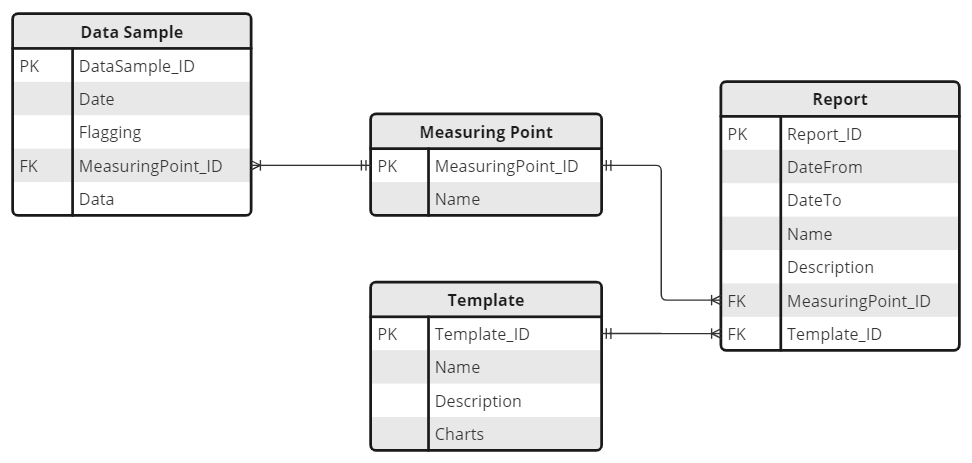
Opis wygenerowany automatycznie

Rysunek 4. Model wzorcu Repozytorium/Serwis.

## 2.4. Schemat bazy danych

Kluczowym aspektem realizowanego projektu jest analiza i porównanie różnych podejść baz danych. W pracy zostaną zbadane trzy modele: relacyjny z wykorzystaniem MS SQL, nierelacyjny oparty o MongoDB, oraz podejście hybrydowe wykorzystujące zalety obu tych systemów. We wszystkich wymienionych podejściach, modele bazy danych będą się nieznacznie różnić, jednak wszystkie będą składały się z następujących encji:

* **DataSample:** Jest to główna encja, która będzie zajmować dominującą część bazy danych. To w tej encji znajdują się wszystkie próbki danych pomiarowych, oraz pomocnicze informację jak data i czas pomiaru, punkt pomiarowy, itp.
* **Template:** Zawiera informacje o szablonach takie jak nazwa,opis, czy informację o wykresach jakie mają być wygenerowane z użyciem tego szablonu.
* **MeasuringPoint:** Reprezentują poszczególne punkty pomiarowe, z której gromadzone są dane.
* **Report:** Zawiera informację na temat wygenerowanych raportów, oraz sam raport w postaci danych binarnych.

Schemat tych encji oraz relacji między nimi przedstawia diagram EDR ukazany na Rys. 5

Rysunek 5. Schemat modelu relacyjnego bazy danych.

Dla modelu bazy nierelacyjnej, encje będą wyglądać niemalże identycznie, jednak będą to osobne kolekcje bez relacji. Natomiast dla modelu hybrydowego [TODO Dopisać]

# 3. Implementacja proponowanego rozwiązania

W niniejszym rozdziale, zostaną przedstawione kluczowe decyzje, które kształtowały proces implementacji, metodykę pracy, a także wyzwania i problemy napotkane podczas budowy aplikacji. Szczegółowo omówione zostaną zastosowania wskazanych technologii z poprzedniego rozdziału, oraz sposób, w jaki zaprojektowane narzędzia wpływają na efektywność zarządzania danymi z pomiarów z pomiarów parametrów jakości energii elektrycznej. Poszczególny podrozdział, może być traktowany jako opis poszczególnego kontrolera, któremu w dalszym etapie będzie odpowiadać osobna sekcja w interfejsie użytkownika. Dzięki zastosowaniu wzorca *Dependency injection*, budowanie takich kontrolerów jest traktowane jak dołączanie odpowiednich modułów (serwisów i repozytoriów), co zostało zaprezentowanie w opisie poszczególnych kontrolerów w podrozdziałach 3.1. 3.2. 3.3. 3.4.

## 3.1. Import danych

Pierwszym kluczowym etapem w procesie budowania aplikacji było stworzenie sekcji odpowiedzialnej za import danych z pliku. Dla tej funkcjonalności postawiono następujące wymagania:

* Stworzenie modelu opisującego parametr pomiaru który w jednoznaczny sposób będzie go identyfikował, zarówno w bazie danych jak również w pozostałych modułach aplikacji.
* Automatyczne rozpoznawanie parametrów, na podstawie nagłówków znajdujących się w importowanych plikach.
* Zapewnienie efektywności i wydajności tego procesu. Dla systemu nie powinno być ograniczeń wielkościowych importowanego pliku, dlatego dla dużej ilości importowanych danych należy zastosować mechanizmy optymalizujące ta czynność.

Analizując nagłówki z przykładowych dokumentów wyeksportowanych z przykładowych analizatorów, można zauważyć, że występują wiele kombinacji parametrów. Dla samego napięcia wyznaczono osiemnaście różnych parametrów, ponieważ może to być napięcie fazowe, międzyfazowe, wartości mogą być średnimi, minimami, maksimami itp. Podobnie jest dla prądu, mocy czynnej lub biernej. Finalnie możliwe jest uzyskanie setek różnych kombinacji parametrów, tworzenie dla każdego z nich osobnego typu wyliczeniowego, lub mapowania bazując na jakimś wspólnym identyfikatorze wydaje się trudnym i mało efektywnym rozwiązaniem. Jednak zauważono, iż dla każdego z tych parametrów można wyznaczyć pomniejsze kategorie które w jednoznaczny sposób definiują dany parametr. W tworzonym systemie przyjęto trzy takie kategorie:

* Typ – definiuje główną kategorię pomiaru taką jak: napięcie, natężenie, moc, częstotliwość, ale również informację o danej próbce danych takie jak data i czas pomiaru.
* Rodzaj pomiaru – definiuje jak został dokonany pomiar konkretnego typu. W tej kategorii znajdują się informacje które są bardziej specyficzne dla każdego typu, jednak można odnaleźć również części wspólne. Takie informacje to przykładowo, czy pomiar dotyczy wielkości fazowej, międzyfazowej, całkowitej itp.
* Rodzaj wartości – ta kategoria definiuje rodzaj wartości z danej próbki tj. czy dany parametr zawiera wielkości średnie, maksymalne, minimalne itp.

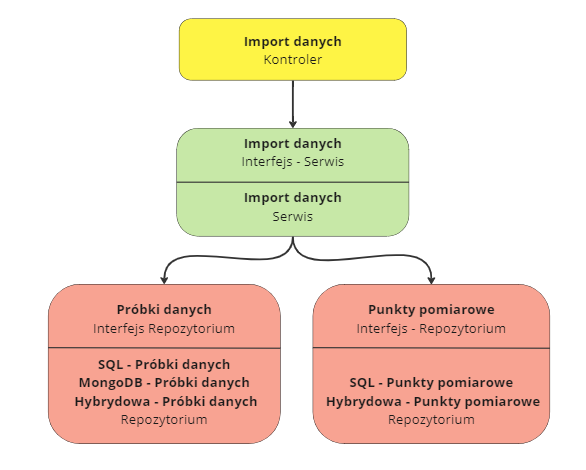
Wykorzystując tak przyjęte kategorię, można jednoznacznie opisać każdy z parametrów, wybierając nie z setek różnych możliwości a ustalając wartości dla tych trzech kategorii, z kilku możliwych opcji. Wiele z parametrów nie potrzebuje opisywać rodzaju pomiaru oraz wartości, ponieważ typ zawiera również wiele wartości unikalnych jak czas pomiaru. Dla lepszego zrozumienia tej koncepcji przedstawiono przykład wraz z opisem na Rys. 6



Rysunek 6. Podział parametru na kategorie.

Do rozpoznania i przyporządkowania parametrów do odpowiednich kategorii, wykorzystano wyrażenia regularne (Regex). Wyrażenia regularne to sekwencje znaków tworzące wzorzec, który można wykorzystać do dopasowania, wyszukiwania czy podziału tekstu. Jest to potężne narzędzie w przetwarzaniu tekstów, które pozwala na wykonanie skomplikowanych operacji na ciągach znaków w sposób skuteczny oraz efektywny czasowo. W tworzonej aplikacji, stworzono trzy różne pule wyrażeń regularnych, co umożliwia precyzyjne dopasowanie do odpowiedniej wartości w każdej z przedstawionych wyżej kategorii.

Dla zapewnienia efektywności i wydajności procesu importu, podzielona, tę cześć na dwa etapy. W pierwszym etapie po wybraniu importowanego pliku na serwerze zostanie zapisany cały plik, aplikacja odczyta tylko nagłówki i kilka rekordów próbek danych, gdy system poradzi sobie już z tą czynnością użytkownikowi zostanie wyświetlony w jaki sposób aplikacja odczytała nagłówki, zostanie udostępniona również możliwość edycji nagłówków. Drugi etap następuje po zaakceptowaniu przez użytkownika odczytanych nagłówków. W jego trakcie zostaną przekopiowane wartości z pliku zapisanego na serwerze do systemu bazy danych z odpowiadającym mu nagłówkiem. Czynność ta będzie odbywać się w tle, pozwalając użytkownikowi przejść do następnych widoków interfejsu użytkownika i rozpoczęcie przygotowywania szablonu lub raportu. Innym rozwiązaniem mającym zapewnić wydajność wskazanego procesu jest zastosowanie kopiowania wartości z pliku do bazy danych w paczkach. Schemat zawierający z jakich modułów składa się opisany kontroler przedstawiono na Rys. 7.



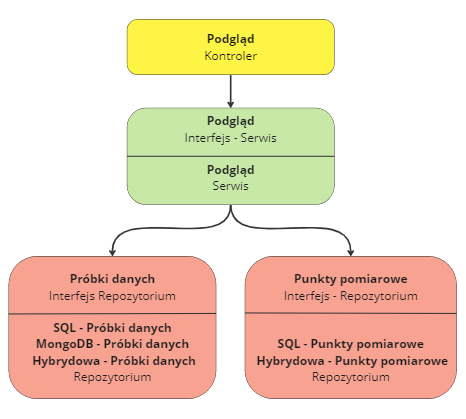
Rysunek 7. Schemat kontrolera importu danych.

## 3.2. Dostęp do danych

W procesie implementacji kluczową rolę odgrywa sprawny dostęp do danych. W celu zapewnienia elastycznego i optymalnego dostępu dla różnych podejść modeli bazodanowych zastosowano wzorzec projektowy wstrzykiwania zależności (*ang. Dependency injection, DI*). Polega on na dostarczaniu obiektowi jego zależności z zewnątrz, zamiast tworzenia ich wewnątrz obiektu. Takie podejście zwiększa możliwości testowalności i rozbudowy kodu przez odseparowanie logiki tworzenia obiektu od logiki biznesowej, oraz zapewnia łatwe podmieniania konkretnej implementacji bez modyfikowania głównego kodu.

Jednym z wykorzystanych narzędzi w procesie budowania logiki dostępu do danych jest Entity Framework Core. Jest to typowy przedstawiciel narzędzi typu ORM (*ang. Object-Relational Mapping*), który umożliwia zarządzanie danymi relacyjnymi bez bezpośredniego pisania zapytań SQL. Dzięki niemu, operacje na danych stają się znacznie prostsze i czytelniejsze z perspektywy programisty, zachowując jednocześnie wysoką wydajność. Dla bazy nierelacyjnej, wykorzystano podobne narzędzie o nazwie MongoDB Driver. Jest to również narzędzie typu ORM stworzone do integracji z bazą danych MongoDB. Zapewnia on podobną funkcjonalność jak Entity Framework, jednak ze względu na architekturę nierelacyjnej bazy danych, operację te są jeszcze łatwiejsze i szybsze.

Kluczowym aspektem operacji na zbiorach danych w C# jest wykorzystanie technologii LINQ (*Language integrated Query).* Pozwala ona na pisanie zapytań bezpośrednio w języku C#. Dzięki kompatybilności zarówno z Entity Framework jak i MongoDB Driver, możliwe jest tworzenie zapytań do różnych źródeł baz danych tak, jakby były one kolekcjami w języku C#. Zapewnia to większą kontrolę i czytelność dostępu do bazy danych, ponieważ całość kodu z tym związana znajduje się w jednym miejscu, w znanej składni języka C#. Ponadto kolejnymi zaletami tej technologii jest pełne wsparcie asynchroniczności, co przy budowaniu interaktywnych interfejsów jest szczególnie ważne, ze względu na brak wstrzymywania aplikacji podczas wykonywania operacji. Jednak najważniejszą zaletą tej technologii jest leniwa ewaluacja, wg dokumentacji „odroczona egzekucja” [TODO odnośnik do dokumentacji]. Oznacza to, że operacja zostanie wywołania i wykonana dopiero wtedy, gdy będzie to potrzebne. Leniwa ewaluacja obiektu oznacza, że jego tworzenie jest odroczone do momentu, gdy jest po raz pierwszy używane. Leniwa ewaluacja jest głównie używana do poprawy wydajności, unikania zbędnych obliczeń i zmniejszenia wymagań pamięci programu. [TODO odnośnik do dokumentacji]. Takie podejście jest możliwe dzięki słowu kluczowemu języka C# *yield* które wskazuje że metoda lub operator w którym się pojawia, jest iteratorem, co powoduje że zwracany jest każdy element pojedynczo, oszczędzając pamięć, bez strat na wydajności. Leniwa ewaluacja wraz z możliwościami łączenia zapytań, wpływa dobrze na optymalizowanie zapytań przez silnik baz danych. Takie zapytania często są łączone i optymalizowane Ponadto technologia ta, posiada ogromne wsparcie i jest mocno rozbudowana o czym może świadczyć około stu wbudowanych instrukcji takich jak: *select, where, any, join, intersect* i wiele innych. Schemat zawierający z jakich modułów składa się opisany kontroler przedstawiono na Rys. 8.



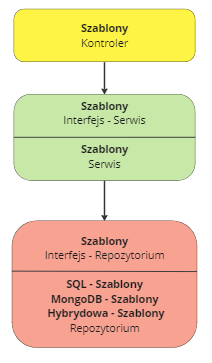
Rysunek 8. Schemat kontrolera podglądu danych.

## 3.3. Szablony

Na potrzeby obsługi szablonów, stworzono indywidualny kontroler. Umożliwia on tworzenie edycję i zarządzanie szablonami raportów. Jest to ważny moduł, dzięki któremu możliwe jest tworzenie spersonalizowanych raportów przedstawiających wybrane dane w postaci wykresów. Każdy z szablonów składa się z dwóch sekcji:

* Części stałej: Jest to część szablonu która nie jest możliwa do edycji z poziomu użytkownika aplikacji. Składa się ona z informacji o szablonie, tj. danych wprowadzonych przy generowaniu raportu takich jak zakres dat, czy nazwa punktu pomiarowego, notatki czym jest i jak został wygenerowany raport. Następnym fragmentem jest opis norm wskazanych w *Rozporządzeniu Ministra Gospodarki,* oraz tabela z wynikami weryfikacji wskazanych danych z normami. Do części stałej dołączono również podstawowe przebiegi takie jak wartości średnie, maksymalne, minimalne napięcia.
* Części edytowalnej: W tej sekcji użytkownik może zdefiniować dodatkowe przebiegi, które powinny znaleźć się w raporcie. Szablon pozwala na dodanie tytułu przebiegu i wskazanie parametrów które mają się na nim znaleźć.

Dzięki zastosowaniu szablonów, generowanie raportów staje się bardziej elastyczne i dostosowane do różnych potrzeb użytkowników. Kontroler obsługujący szablony został zaprojektowany w taki sposób, aby zapewnić niezbędną funkcjonalność, bez potrzeby integracji z innymi modułami opisanymi w tym rozdziale. Wykorzystano jedyne narzędzie ułatwiające wskazanie danych które mają znaleźć się na przebiegach. Opis jak identyfikowane są badane wielkości został wskazany w rozdziale 3.1. na Rys. 6. Schemat zawierający z jakich modułów składa się ten kontroler przedstawiono na Rys. 9.



Rysunek 9. Schemat kontrolera szablonów.

## 3.4. Raporty

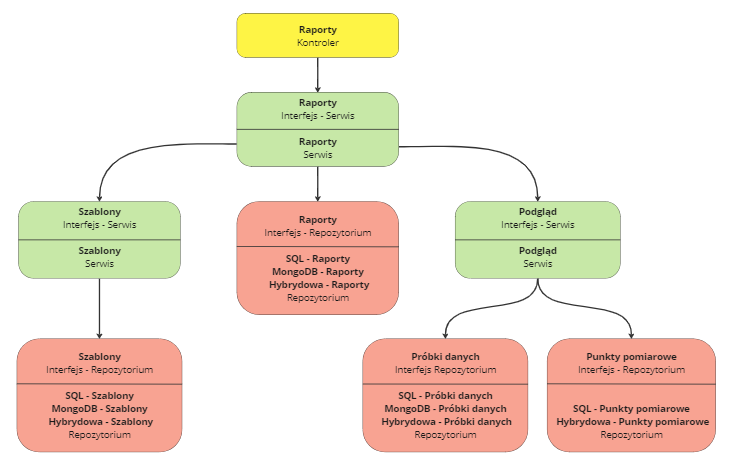
Osobny moduł zostaw stworzony dla kluczowej funkcjonalności tworzonej aplikacji webowej jaką jest tworzenie i obsługa raportów. Jest to kompleksowa część systemu, której zadaniem jest:

* Wyciąganie z bazy danych, próbek danych które są potrzebne przy badaniu zgodności z normą, jak również do przedstawiania na przebiegach wskazanych szablonach.
* Przeliczenie i sprawdzenie odpowiednich parametrów, pod kątem zgodności z normami.
* Generowanie przebiegów zdefiniowanych przez użytkownika w szablonie.
* Zarządzanie raportami, czyli operacje takie jak podgląd wygenerowanego raportu w przeglądarce, pobranie go na dysk, oraz przechowywanie w bazie danych.
* Obsłużenie serwera SMTP, umożliwiając przy tym wysyłkę wygenerowanego raportu bezpośrednio na pocztę wskazaną przez użytkownika

Zdecydowano się na przechowywanie raportów w bazie danych, ponieważ z założenia takich raportów nie będzie wielu, ponadto po pobraniu lub wysłaniu mailem, przechowywanie ich w bazie nie jest konieczne. Dlatego można założyć, iż takie rozwiązanie nie będzie spowalniać pracy bazy danych. Takie pliki przechowywane są w bazie w postaci plików binarnych, zapewniając oryginalny format i strukturę pliku, bez ryzyka utraty jakości czy zniekształceń powodowanych konwersją lub kompresją. Eliminuje to również potrzebę konwersji do formatów tekstowych, obniżając przy tym zajętość pamięci. Ponadto dane binarne umożliwiają bezpośredni zapis i odczyt bez dodatkowego przetwarzania, dzięki temu takie operacje są znacznie szybsze.

Do wysyłki maili, wykorzystano serwer SMTP, co jest uniwersalnym rozwiązaniem, zapewniającym prostą, szybką i niezawodną wysyłkę wiadomości e-mail. W pracy wykorzystano serwer Gmail, który w wersji darmowej umożliwia pełną funkcjonalność o ile nie jest ona wykorzystywana w celach komercyjnych.

W tym kontrolerze natomiast niezbędne okazało się dołączenie, większości modułów opisanych w tym rozdziale, ponieważ korzysta i jest bezpośrednio zależny od takich encji jak próbki danych oraz szablony. Schemat połączeń serwisów i repozytoriów przedstawiono na rysunku Schemat zawierający z jakich modułów składa się ten kontroler przedstawiono na Rys. 10.



Rysunek 10. Schemat kontrolera Raportów.

# 4. Interfejs użytkownika

W dobie ogromnego rozwoju rozwiązań webowych, a co za tym idzie dostępnej różnorodności, użytkownik został przyzwyczajony do nowoczesnych, estetycznych interfejsów użytkownika. To właśnie interfejs decyduje o pierwszym wrażeniu i ogólnym odbiorze aplikacji przez użytkownika. Dlatego ważne jest, aby interfejs zapewniał przyjazny dla oka design, ale również był prosty w obsłudze, oraz pozwalał na dynamiczną komunikację z użytkownikiem. Aby spełnić te wymagania, nie wystarczy dostarczenie interfejsu opierającego się o domyślnie stylizowane pola tekstowe, czy przyciski. Dążenie do stworzenia interfejsu, który będzie funkcjonalny i interaktywny, należy dostarczyć, nowocześnie wystylizowane obiekty, zapewnić animacje i responsywność wszelkiego rodzaju pól, przycisków, list. W proponowanym rozwiązaniu, poza możliwościami które oferują widoki z silnika Razor, wykorzystano również najczęściej stosowane rozwiązanie składające się z trzech elementów: HTML, CSS, oraz JavaScript.

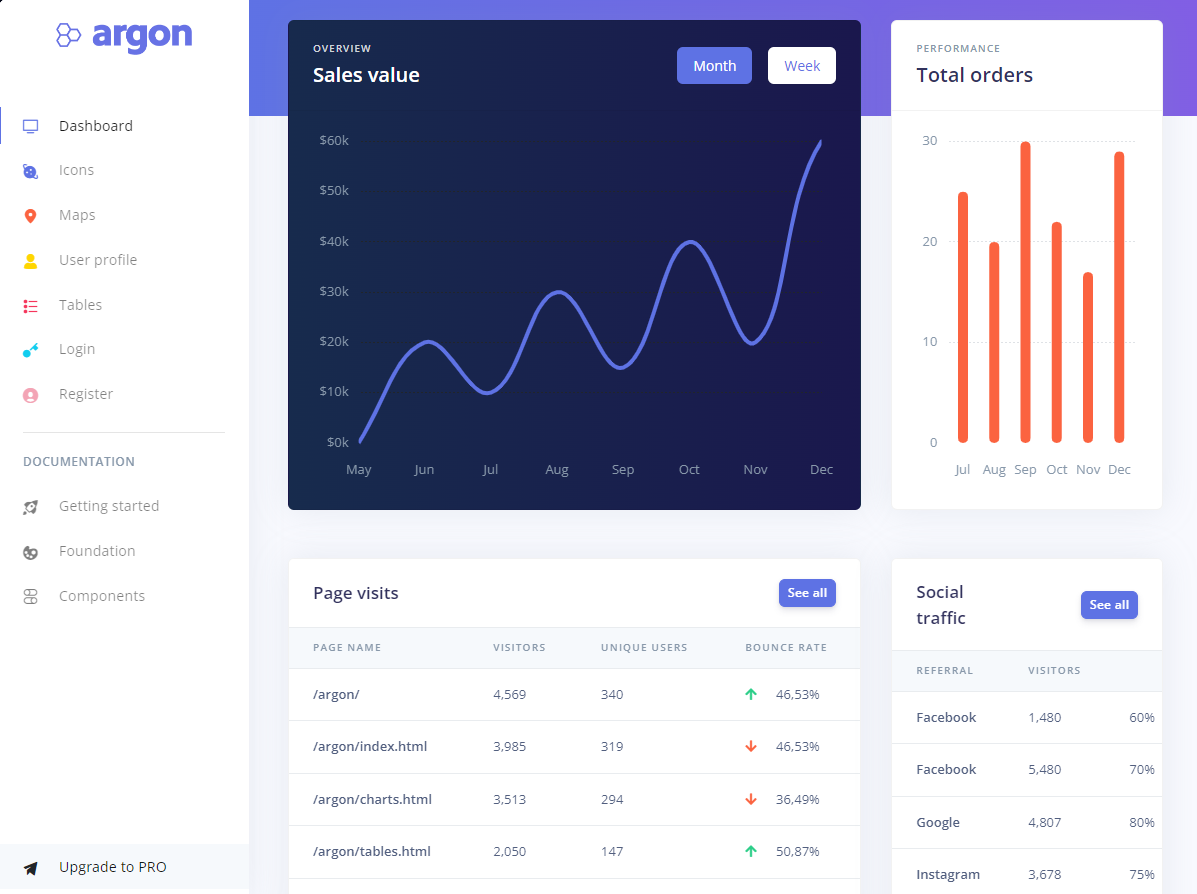
HTML jest językiem opisowym, odpowiadającym za główną strukturę wyświetlanych treści. Definiuje on takie elementy jak nagłówki, paragrafy, elementy nawigacji. CSS jest to język stylów, który opisuje prezentację dokumentów HTML, Pozwala on kontrolować wygląd, poprzez ustawianie kolorów, czcionek, odstępów czy kształtów elementów treści opisanych w HTML. Zastosowanie obu tych języków pozwala w łatwy sposób oddzielić strukturę od zarządzania stylami, co znacznie upraszcza konstruowanie części wizualnej aplikacji internetowych. Uzupełnieniem dla tych dwóch języków jest JavaScript, jest to dynamiczny język programowania którego skrypty znajdują się po stronie klienta umożliwiając natychmiastową reakcję, bez komunikacji z serwerem. Dzięki zastosowaniu JavaScript możliwe jest dynamiczne manipulowanie strukturą dokumentu HTML, zmienianie treści, reagowanie na zdarzenia użytkownika, komunikacja z serwerem.

W tworzonej aplikacji wykorzystano bibliotekę JavaScript o nazwie JQuery. Jest to łatwo rozszerzalna, biblioteka typu open-source upraszczająca podstawowe wykorzystanie języka JavaScript. Najważniejszymi zaletami JQuery jest uproszczenie manipulacji DOM (document object model), czyli znajdowania zmieniania i ukrywania elementów dokumentu HTML. Wiele najczęściej wykorzystywanych zdarzeń jak kliknięcia czy przeładowania, zostały skrócone do postaci krótkich jednolinijkowych wyrażeń. Kolejną ogromną zaletą jest uproszczenie korzystania z AJAX. Czyli zbioru technik umożliwiających interakcje użytkownika z serwerem, bez konieczności przeładowywania strony HTML. Pozwala to na dynamiczne aktualizowanie treści na stronie, co prowadzi do bardziej responsywnego i interaktywnego doświadczenia użytkownika.

## 4.1. Argon Dashboard

Stworzenie nowoczesnego i estetycznie atrakcyjnego wyglądu interfejsu użytkownika stanowi spore wyzwanie, które wymaga nie tylko technicznego doświadczenia, ale również wyczucia estetyki. Celem uproszczenia tego procesu i zagwarantowania spójności wyglądu w projekcie zdecydowano się wykorzystać szablon interfejsu *Argon Dashboard*. Jest to gotowy zestaw stylów CSS, komponentów HTML. Oraz skryptów JavaScript, który został zaprojektowany, aby wspierać twórców w szybkim tworzeniu eleganckich i funkcjonalnych interfejsów użytkownika. Szablon ten został udostępniony na zasadach licencji MIT, co za tym idzie można go wykorzystywać zarówno w projektach prywatnych i komercyjnych. Szablon interfejsu *Argon Dashboard* wykorzystuje również popularny framework CSS Bootstrap. Udostępnia on zestaw narzędzi, stylów które pozwala na szybkie i efektywne projektowanie interfejsów użytkownika. Jego komponenty opierają się na wielu dodatkowych bibliotekach JavaScript, jednak na potrzeby budowanej aplikacji niezbędne okazało się wykorzystanie jedynie bibliotek: [TODO Jakiś odnośnik]

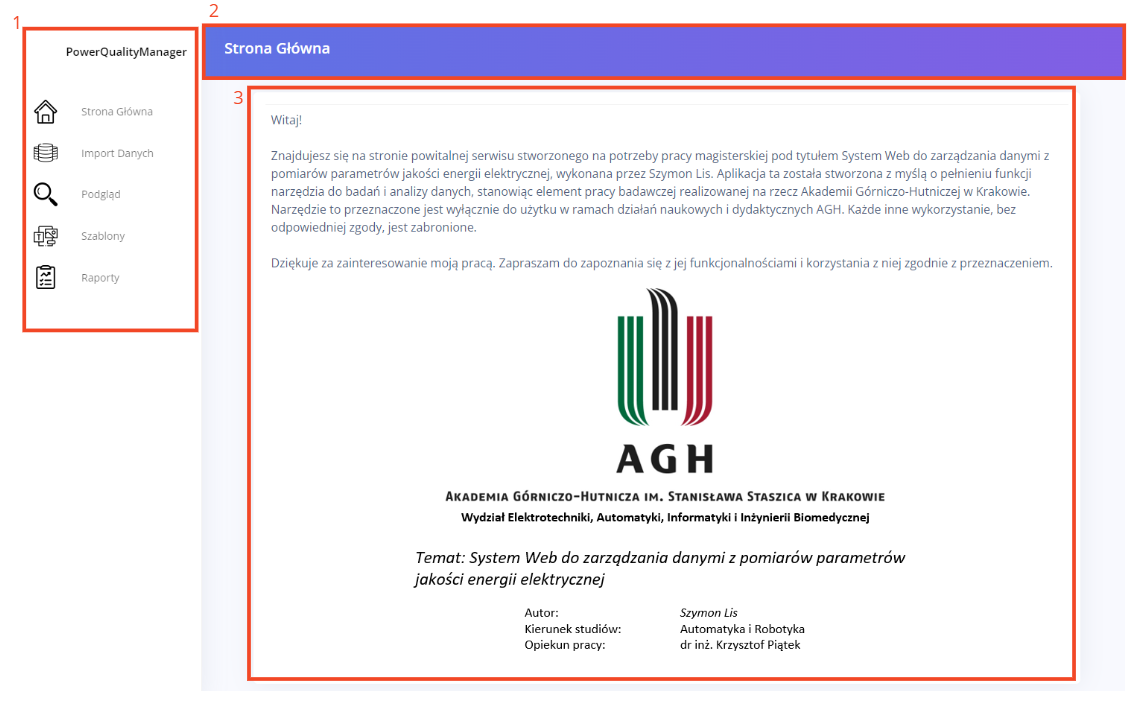
* Flatpickr – prosty, interaktywny selektor dat
* Datatables – biblioteka wspomagająca budowanie rozbudowanych tabel z danymi



Rysunek 11. Przykładowy interfejs zbudowany z wykorzystaniem Argon Dashboard

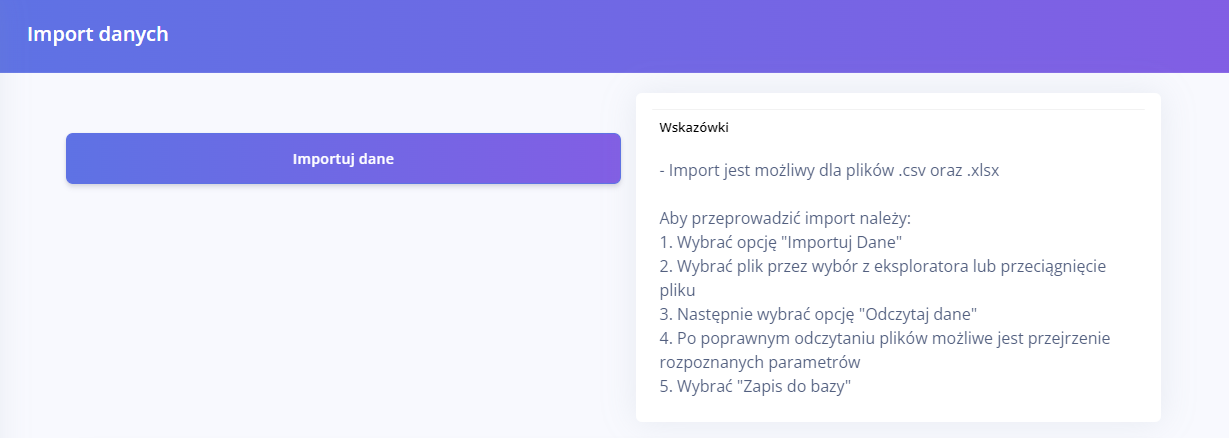
## 4.2. Opis funkcjonalności

W tym podrozdziale przedstawiono elementy i funkcje dostępne w interfejsie użytkownika aplikacji. Ukazanie tych aspektów, pozwoli poznać możliwości oferowane przez system. Opisane również zostaną rozwiązania które zapewniają spełnienie wymagań postawionych w podrozdziale 2.1.



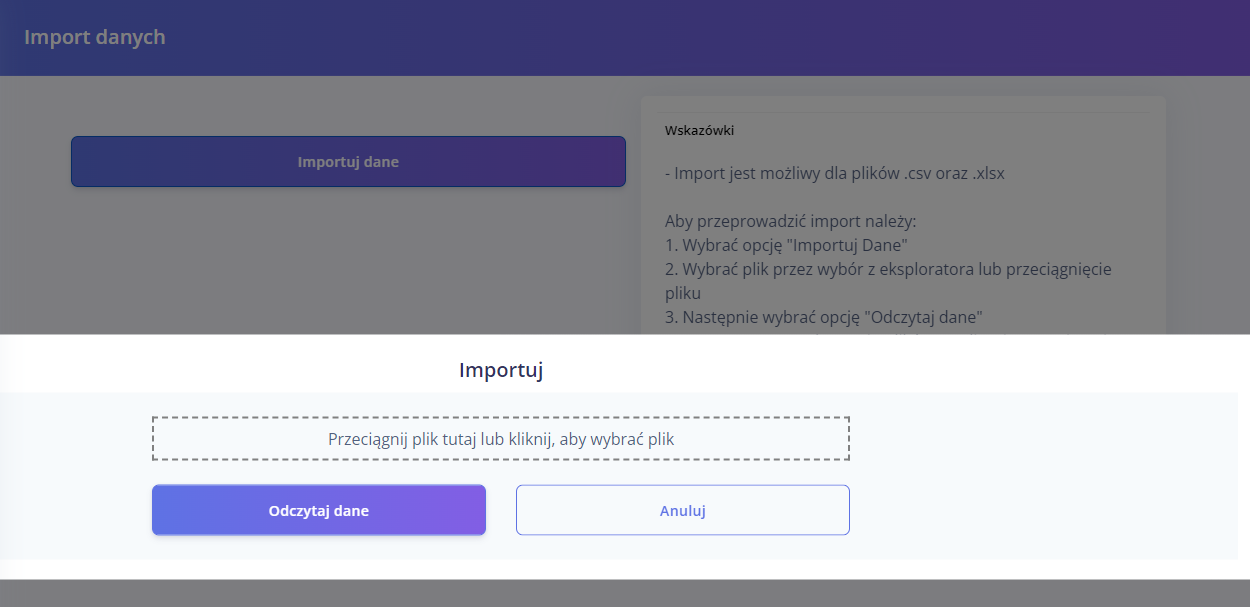
Rysunek 12. Interfejs użytkownik – Strona główna

Na Rys. 12. Ukazane zostały główne elementy interfejsu. Każda z podstron aplikacji *PowerQualityManager* składa się dokładnie z tych trzech elementów, jednak zmienia się treść w poszczególnych elementach. Elementem nr 1 oznaczono boczny pasek nawigacyjny, pasek ten przy mniejszych oknach wyświetlania automatycznie zwija się do boku nie zasłaniając treści głównej. Na pasku tym znajdują się przekierowania do poszczególnych funkcji aplikacji. Każda z funkcji reprezentuje osobny kontroler, dokładniej opisany w podrozdziałach 3.1. 3.2. 3.3. 3.4. Ponadto dołączona została strona główna zawierająca statyczne informacje o celu powstania i przeznaczeniu aplikacji, jej dokładna treść została ukazana na Rysunku 12. Element nr 2 to pasek górny, zawierający jedynie nagłówek z nazwą podstrony na której obecnie znajduje się użytkownik. Element nr 3. Zawiera treść główna dla poszczególnej podstrony. Aby skupić się na konkretnych elementach interfejsu, w dalszej części pracy fragmenty interfejsu zostaną ukazane bez bocznego paska nawigacji (element nr 1), ponieważ jest on identyczny w całej aplikacji.



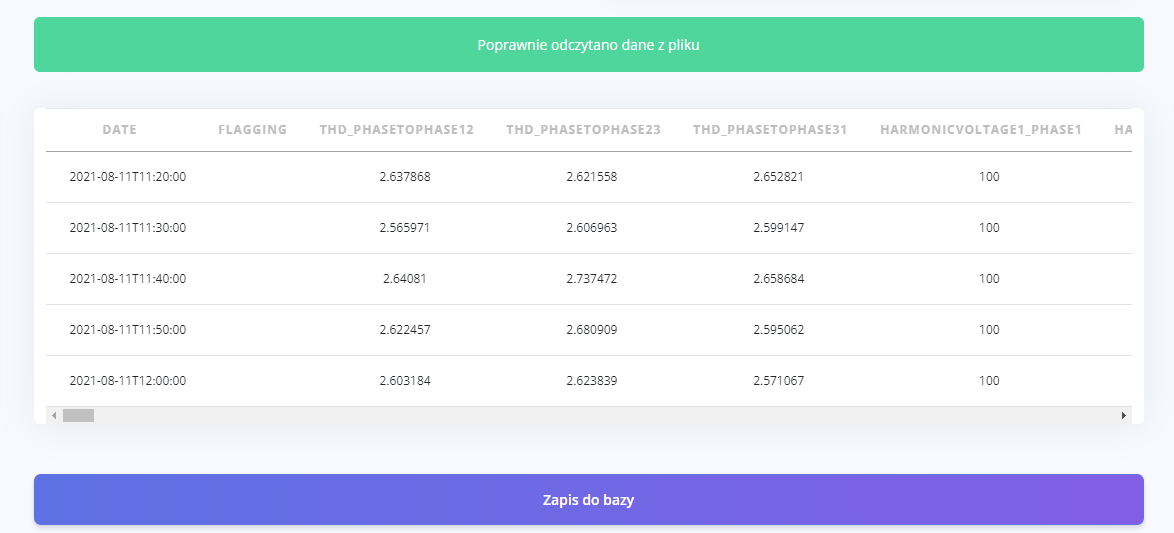
Rysunek 13. Interfejs użytkownika - import danych.

Podstawowy ekran importu danych został przedstawiony na Rys. 13. Na każdym z głównych ekranów tak jak i na tym ekranie została umieszczona karta z krótką instrukcją postępowania, informacjami ogólnymi dla danej funkcjonalności.



Rysunek 14. Interfejs użytkownika - okno wyboru importowanego pliku

Do wyboru pliku udostępniono zarówno klasyczną metodę wyboru pliku przez eksplorator plików, który jest uruchamiany po kliknięciu we wskazane pole, lub popularną metodę typu drag and drop, umożliwiając „przeciąganie” pliku z urządzenia do wskazanego pola. Okno wyboru wskazanego pliku przedstawiono na Rys. 14.



Rysunek 15. Interfejs użytkownika - odczytane nagłówki

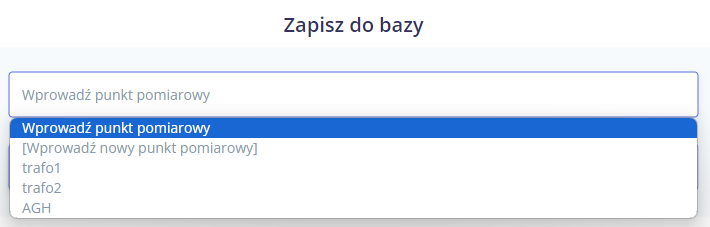
Po wybraniu i zaimportowaniu pliku rozpoczyna się odczytywanie nagłówków zgodnie z tym co opisano w podrozdziale 3.1. Po zakończeniu odczytywania wyświetlony zostanie komunikat z informacją o pozytywnym, lub negatywnym odczytaniu nagłówków, oraz tabela zawierające odczytane nagłówki oraz kilka wierszy przykładowych danych, co pozwoli użytkownikowi na weryfikację poprawności odczytanych nagłówków. Przykładowy rezultat przedstawiono na Rys. 15.

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

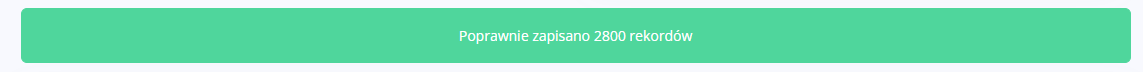
Rysunek 16. Interfejs użytkownika - edytor nagłówków

Dzięki wprowadzeniu opisu parametrów złożonego z trzech oddzielnych kategorii, przedstawionego na Rys. 6, możliwe jest optymalne zarządzanie kategoriami opierające się o wybranie wartości z trzech niedużych list, takie rozwiązanie jest zarówno szybsze jak i bardziej intuicyjne niż wybieranie spośród listy kilkuset różnych kombinacji, lub ręcznym wpisywaniu nazw nagłówków.



Rysunek 17. Interfejs użytkownika - zapis do bazy

Proces importu danych kończy się wybraniem punktu pomiarowego z listy punktów pomiarowych znajdujących się w bazie, lub dodaniem zupełnie nowego punktu pomiarowego. Przykładowa lista z wyborem punktu pomiarowego została ukazana na Rys. 16. Po potwierdzeniu, rozpoczyna się proces importu, który ze względu na sporą ilość danych jest uruchamiany jako zadanie w tle, nie blokując innych operacji na interfejsie użytkownika. Po zakończeniu operacji importu wyświetlony zostanie komunikat niezależnie od podstrony na której znajduje się użytkownik o sukcesie bądź błędzie importu – Rys. 17.



Rysunek 18. Interfejs użytkownika – komunikat

Następna zakładka oferuje oferuje podgląd danych zapisanych w bazie filtrując po punkcie pomiarowym oraz dacie początku i końca. Dla pola w formularzu dotyczącego punktu pomiarowego zastosowano, rozwijaną listę z wyborem, natomiast dla wyboru dat, zastosowano interaktywny kalendarz z biblioteki flatpickr zawartej w szablonie interfejsu ArgonDashboard. Dodatkowo na kalendarz nałożone są ograniczenia, które pozwalają na wybranie dat które znajdują się w bazie danych dla wskazanego punktu pomiarowego. Tutaj wykorzystano dynamiczne zapytania na podstawie wybranego punktu pomiarowego z wykorzystaniem opisywanej technologii AJAX oraz odpowiednio przygotowanej logice po stronie backendu. Przykładowe dane wraz z opcjami podstrony podglądu danych zostały ukazane na Rys. 19.

Obraz zawierający tekst, zrzut ekranu, numer, oprogramowanie

Opis wygenerowany automatycznie

Rysunek 19. Interfejs użytkownika – podgląd

Następna z sekcji obejmuje narzędzia i funkcję, które pozwalają na zarządzanie szablonami w systemie. Użytkownik ma możliwość dodania, edycji oraz usuwania szablonów. Warto podkreślić że istnieje jeden specyficzny szablon bazowy, który służy jako podstawa dla wszystkich pozostałych. Jest to element stały – nieedytowalny oraz nieusuwalny. Widok podstrony obejmującej obsługę szablonów przedstawiono na Rys. 20.

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie**

Rysunek 20. Interfejs użytkownika – szablony

Do dodawania oraz edycji stworzono jeden formularz różniący się jedynie nagłówkiem między tymi dwoma sekcjami, składa się on z unikalnej nazwy szablonu według której będzie identyfikowany w bazie danych, dodatkowego opisu, który może być wskazówka dla innych użytkowników zawierająca informację o celu i przeznaczeniu szablonu. Ostatnia część formularza szablonu, obejmuje tabelę z nazwą wyświetlaną dla poszczególnych wykresów, oraz informacją jakie parametry mają się znajdować się na takim szablonie. Tutaj przyjęto następujące uproszczenia względem użytkownika, domyślnie dla wyboru poszczególnych parametrów nie jest wskazywana wartość dla poszczególnych kategorii. Przykładowo po wybraniu kategorii pomiaru jako fazowe, na przebiegach zostaną umieszczone wartości ze wszystkich dostępnych faz, oczywiście wraz z legendą. W tym przypadku tak samo jak w przypadku modyfikatora nagłówków podczas importu zastosowano osobne okno, pozwalające na wygenerowanie nagłówka. Narzędzie to przedstawiono na Rys. 16. W tym przypadku jest ono jedynie dokładniej przefiltrowane, aby możliwe wskazanie było jedynie parametrów, a nie wartości technicznych jaka flaga, czy znacznik czasu. Widok formularza dodawania/edycji szablonów został przedstawiony na Rys. 21.

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Opis wygenerowany automatycznie**

Rysunek 21. Interfejs użytkownika - formularz dodawania/edycji szablonu

Ostatnia z sekcji interfejsu użytkownika obejmuje generowania i zarządzanie raportami. Struktura tej podstrony jest niemal identyczna jak podstrony, obejmuje jednak szerszą funkcjonalność, tutaj również występuje lista z informacjami o raporcie, w której po wskazaniu raportu pojawiają się opcję takie jak podgląd raportu w przeglądarce, pobranie na dysk, usunięcie lub wysyłka wiadomość e-mail z raportem. Widok z tej części interfejsu, wraz z przykładowymi danymi przedstawiono na Rys. 22.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

Rysunek 22. Interfejs użytkownika - Raport

Sekcja raportów obejmuje dodatkowo dwa formularze, pierwszy z nich obejmuje wygenerowanie nowego raportu. Składa się on z pól takich jak nazwa, nazwa wykorzystywanego szablonu, data rozpoczęcia oraz zakończenia oraz wskazanie punktu pomiarowego. Logika obejmująca ułatwiania wprowadzania danych została przedstawiona we wcześniejszych elementach interfejsu użytkownika. Wzór takiego formularza ukazano na Rys. 23. Kolejny natomiast jest to formularz wysyłki wiadomości e-mail. Zawiera on pola tekstowe obejmujące adresata, tytuł oraz treści wiadomości. Załącznik jest automatycznie dodawany w zależności od wskazanego raportu (Rys. 22). Przykład formularza obejmującego wysyłkę wiadomości e-mail został ukazany na Rys. 24.

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Rysunek 23. Interfejs użytkownika - formularz generowania nowych raportów

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Rysunek 24. Interfejs użytkownika - formularz wysyłki wiadomości e-mail

## 4.3. Raport

Raport jest dokumentem w formie pliku PDF, generowany jest programistycznie, oparty na określonej logice, co jest spowodowane potrzebą zamieszczania dodatkowych wykresów dodawanych w szablonach. Elementy statyczne, takie jak teksty czy stałe opisy przechowywane są w plikach .resx. Są to pliki zasobów, szeroko wykorzystywane i dobrze zintegrowane z technologią .NET. Opierają się o format XM, oraz definiują wartości w parach typu nazwa/wartość. Mogą być tworzone oraz modyfikowane zarówno z poziomu kodu oraz edytora plików. [TODO dokumentacja Microsoftu] Takie podejście umożliwia centralne zarządzenie tekstem, lub adaptacje aplikacji do różnych języków bez konieczności modyfikacji kodu aplikacji. Generowany raport składa się z czterech sekcji:

* Strony tytułowej z informacjami o wygenerowanym raporcie
* Definicji i opisu norm
* Wynikami spełnienia norm dla wskazanych danych
* Przebiegami parametrów wskazanych przez użytkownika w szablonie

Strona tytułowa oraz definicja norm to statyczne teksty, wycinki z tych części raportu przedstawiono na Rys. 25, Rys. 26

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Rysunek 25. Raport - strona tytułowa

Obraz zawierający tekst, zrzut ekranu, Czcionka, Równolegle

Opis wygenerowany automatycznie

Rysunek 26. Raport - opis norm

Definicje tych norm pochodzą z [TODO przypis], opisują one parametry jakościowe energii elektrycznej dla sieci funkcjonującej bez zakłóceń. Przedstawione wyżej wymogi są stawiane grupom przyłączeniowym III – VI.

Wyniki zgodności z normą, zostały natomiast przedstawione w formie tabelarycznej. Gdzie kolumnami jest odpowiednio nr normy, wynik, oraz powód braku zgodności, wiersze natomiast to kolejne normy, rozbite ewentualnie na poszczególne fazy w przypadku danych z więcej niż jednej fazy. Przykładowe wyniki wraz ze wzorem tabeli przedstawiono na Rys. 27.

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

Rysunek 27. Raport - wyniki zgodności z normą

Poszczególne parametry na przebiegach prezentowane są pojedynczo, łączone są jedynie przebiegi dotyczące tej samej wartości dla różnych faz, przykładowy przebieg przedstawiony został na Rys 28. Jednostki dla poszczególnych parametrów nie są zapisywane w bazie danych, oraz nie zawsze są zawarte w poszczególnych nagłówkach, dla dostarczonych danych podczas importu następuje konwersja dla jednostek odstających od podstawowych – tyczy się to głównie przedrostków. Dlatego przebiegi prezentowane są zawsze w jednostce podstawowej dla danego parametru.

Obraz zawierający tekst, linia, Wykres, zrzut ekranu

Opis wygenerowany automatycznie

Rysunek 28. Raport - przykładowy przebieg

# 5. Porównanie modeli bazy danych

[TODO]

## 5.1. Założenia teoretyczne

## 5.2. Badane podejścia

## 5.3. Porównanie wyników

# 6. Podsumowanie

## 6.1. Wnioski

## 6.2. Możliwości rozwoju

# Bibliografia

1. „Rozporządzenie Ministra Gospodarki z dnia 4 maja 2007 r. w sprawie szczegółowych warunków funkcjonowania systemu elektroenergetycznego,” (Dz.U. 2007 nr 93 poz. 623).
2. Freeman Adam, Pro ASP.NET Core MVC: sixth edition, Apress Berkeley, ISBN-13 : 978-1-4842-0389-9
3. Troelsen Andrew, Japikse Philip, Pro C# 7 eighth edition, Apress Berkeley, ISBN-13 : 978-1-4842-3017-6
4. Lock Andrew, ASP .NET Core in action, Manning Publications, ISBN 9781617294617
5. Microsoft, „Overview of ASP.NET Core” [Online]. Dostęp: <https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core>
6. Microsoft, „Architectural principles” [Online]. Dostęp: <https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/architectural-principles>
7. Microsoft, „Overview of ASP.NET Core MVC” [Online]. Dostęp: <https://learn.microsoft.com/en-us/aspnet/core/mvc>
8. Microsoft, „Introduction to Razor Pages in ASP.NET Core” [Online]. Dostęp: <https://learn.microsoft.com/en-us/aspnet/core/razor-pages>
9. MongoDb, „Welcome to the MongoDB Documentation” [Online], Dostęp: <https://www.mongodb.com/docs>
10. Microsoft, „What's new in SQL Server 2022 ” [Online], Dostęp: <https://learn.microsoft.com/en-us/sql/sql-server/what-s-new-in-sql-server-2022>
11. Creative Tim, „Argon Dashboard Bootstrap” [Online], Dostęp: <https://www.creative-tim.com/learning-lab/bootstrap/overview/argon-dashboard>
12. Microsoft, „Create resource files for .NET apps” [Online], Dostęp: <https://learn.microsoft.com/en-us/dotnet/core/extensions/create-resource-files>
13. Firlit Andrzej, Błajszczak Grzegorz, Narzędzia do oceny i analizy jakości energii elektrycznej, Miesięcznik **Energetyka**, 2020, nr 9 (795)
14. Martin Robert, Czysty Kod. Podręcznik dobrego programisty, tłum. Paweł Gonera, Gliwice: Helion, ISBN: 978-83-8322-344-5
15. Martin Robert, Czysta architektura. Struktura i design oprogramowania. Przewodnik dla profesjonalistów,tłum.Wojciech Moch, Gliwice: Helion, ISBN: 978-83-283-9109-3