POLITECHNIKA GDAŃSKA

WYDZIAŁ ELEKTRONIKI, TELEKOMUNIKACJI I INFORMATYKI

Imię i nazwisko studenta: Mikołaj Jaskulski
Nr albumu: 131521
Studia drugiego stopnia
Forma studiów: stacjonarne
Kierunek studiów: Informatyka
Specjalność/profil: Aplikacje rozproszone i systemy internetowe

PRACA DYPLOMOWA MAGISTERSKA

Tytuł pracy w języku polskim: Graficzny kreator stron www służących do zdalnego monitoringu urządzeń

Tytuł pracy w języku angielskim: Graphic web pages creator for remote devices monitoring

Potwierdzenie przyjęcia pracy	
Opiekun pracy podpis	Kierownik Katedry/Zakładu podpis
dr inż. Łukasz Kuszner	

Data oddania pracy do dziekanatu:

POLITECHNIKA GDAŃSKA
WYDZIAŁ ELEKTRONIKI, TELEKOMUNIKACJI I INFORMATYKI

OŚWIADCZENIE

Imię i nazwisko: Mikołaj Jaskulski
Data i miejsce urodzenia: 21.12.1991, Gdynia
Nr albumu: 131521
Wydział: Wydział Elektroniki, Telekomunikacji i Informatyki
Kierunek: informatyka
Poziom studiów: II stopnia
Forma studiów: stacjonarne

Ja, niżej podpisany(a), wyrażam zgodę/nie wyrażam zgody* na korzystanie z mojej pracy dyplomowej zatytułowanej: Graficzny kreator stron www służących do zdalnego monitoringu urządzeń
do celów naukowych lub dydaktycznych.[1]

Gdańsk, dnia ....
podpis studenta

Świadomy(a) odpowiedzialności karnej z tytułu naruszenia przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. z 2006 r., nr 90, poz. 631) i konsekwencji dyscyplinarnych określonych w ustawie Prawo o szkolnictwie wyższym (Dz. U. z 2012 r., poz. 572 z późn. zm.),[2] a także odpowiedzialności cywilno-prawnej oświadczam, że przedkładana praca dyplomowa została opracowana przeze mnie samodzielnie.

Niniejsza(y) praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadaniem tytułu zawodowego.

Wszystkie informacje umieszczone w ww. pracy dyplomowej, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami zgodnie z art. 34 ustawy o prawie autorskim i prawach pokrewnych.

Potwierdzam zgodność niniejszej wersji pracy dyplomowej z załączoną wersją elektroniczną.

Gdańsk, dnia ....
podpis studenta

Upoważniam Politechnikę Gdańską do umieszczenia ww. pracy dyplomowej w wersji elektronicznej w otwartym, cyfrowym repozytorium instytucjonalnym Politechniki Gdańskiej oraz poddawania jej procesom weryfikacji i ochrony przed przywłaszczaniem jej autorstwa.

Gdańsk, dnia ....
podpis studenta

*) niepotrzebne skreślić

---

[1] Zarządzenie Rektora Politechniki Gdańskiej nr 34/2009 z 9 listopada 2009 r., załącznik nr 8 do instrukcji archiwalnej PG.

[2] Ustawa z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym:
Art. 214 ustęp 4. W razie podejrzenia popełnienia przez studenta czynu podlegającego na przypisaniu sobie autorstwa istotnego fragmentu lub innych elementów cudzego utworu rektor niezwłocznie poleca przeprowadzenie postępowania wyjaśniającego.
Art. 214 ustęp 6. Jeżeli w wyniku postępowania wyjaśniającego zebrany materiał potwierdza popełnienie czynu, o którym mowa w ust. 4, rektor wstrzymuje postępowanie o nadanie tytułu zawodowego do czasu wydania orzeczenia przez komisję dyscyplinarną oraz składa zawiadomienie o popełnieniu przestępstwa.

STRESZCZENIE

The **Streszczenie** should correspond to the **Abstract** and includes the same key elements. It should be prepared according to faculty rules. The **Streszczenie** is a chapter written only by Polish authors who decide to write the diploma thesis in English and it is not applicable to foreign students.

Słowa kluczowe: [keywords] Zażółć gęślą jaźń.

Dziedzina nauki i techniki, zgodnie z wymogami OECD: <dziedzina>, <technika>,. . . [field of science and technology in accordance with OECD requirements: <field>, <technology>,. . .]

SPIS TREŚCI

LIST OF IMPORTANT SYMBOLS AND ABBREVIATIONS

i, j, l, m	—	indeksy
MVC	—	wzorzec projektowy Model View Controller

1. OPIS ZASTOSOWANEGO ROZWIąZANIA

Rozdział ten zawiera opis zastosowanego przeze mnie rozwiązania służącego do zdalnego nadzorowania pracy urządzeń. Rozwiązanie to zostało zaimplementowane jako aplikacja internetowa składająca się z dwóch modułów. Decyzja ta wynika z faktu, iż aplikacja jest przeznaczona dla dwóch typów użytkowników. Pierwszym z nich jest projektant widoków – osoba odpowiedzialna za ich tworzenie, która posiada podstawową wiedzę na temat urządzeń. W żargonie branży kolejowej termin "widok"można zastąpić słowem "wizualizacja". Drugim typem użytkownika jest osoba upoważniona do oglądania widoków.

Zadaniem pierwszego modułu aplikacji jest projektowanie widoków - stron złożonych z elementów graficznych, które pozwalają na wizualizację stanu urządzenia. Drugi służy do wyświetlania tych widoków wraz z nawiązaniem połączenia z urządzeniami i dostarczeniem danych w odpowiednie miejsca wizualizacji. Kolejne podrozdziały zawierają szczegółowy opis funkcjonalności modułów.

1.1. Elements Designer - Moduł projektowania widoków

Moduł Designer pozwala na projektowanie widoków urządzeń. Jego interfejs składa się z dwóch głównych elementów – menu bocznego oraz sceny, która prezentuje obecny stan projektu. Dzięki temu rozwiązaniu projektant widzi efekt swojej pracy natychmiast po dokonaniu zmian. Projektowanie widoku polega na wybieraniu tzw. komponentów z menu, umieszczaniu ich w odpowiednim miejscu na scenie i konfigurowaniu wedle potrzeb.

1.1.1. Tworzenie projektu

Po wybraniu modułu z odpowiedniej sekcji menu aplikacji użytkownikowi ukazuje się strona wyboru projektu. Każdy projekt widoczny jest jako kafelek z nazwą i opisem projektu. Użytkownik może z tego miejsca przejśc do edycji istniejącego projektu lub stworzyć nowy. W takiej sytuacji należy uzupełnić nazwę, opis oraz grupę urządzeń, dla której projektowany będzie widok. Po zatwierdzeniu tych danych użytkownikowi ukazuje się Scena.

1.1.2. Scena

Scena jest obszarem roboczym, który gromadzi wszystkie komponenty. Po zapisaniu projektu, możliwe będzie wyświetlenie jej przez moduł Viewer wraz ze wszystkimi komponentami naniesionymi na nią w trakcie projektowania. Jedynym atrybutem sceny, który można konfigurować jest kolor jej tła.

1.1.3. Komponenty

Komponenty to podstawowe elementy składowe projektu, które pełnią różne funkcje. W praktyce jest to każdy element dodany przez użytkownika do Sceny w trakcie projektowania. W aplikacji Elements istnieją trzy rodzaje komponentów, które można rozróżnić na podstawie ich funkcji:

- Kontener – służy do grupowania innych komponentów. Można w nim umieścić dowolny komponent, ale sam również może być umieszczony w innym kontenerze. Aby umieścić element w kontenerze należy chwycić go myszą i „upuścić” nad wybranym komponentem typu kontener. Czynność tę można odwrócić wybierając opcję „Przenieś wyżej” z menu kontekstowego komponentu.
- TextBox – jego funkcja to przechowywanie tekstu, który można edytować. Dzięki temu możliwe jest tworzenie opisów innych elementów projektu. Tekst wpisywany jest pod-

czas tworzenia komponentu. TextBox można edytować w każdej chwili wybierając odpowiednią opcję z jego menu kontekstowego.

- RefreshedVariable – element ten powiązany jest z konkretną zmienną urządzenia, którą należy wybrać podczas tworzenia komponentu. Jego główną funkcją jest regularne odświeżanie wartości zmiennej urządzenia w przypadku jej zmiany. Dodatkowo można skonfigurować zmianę tekstu i koloru tła komponentu w chwili gdy wartość zmiennej wyniesie lub przekroczy zdefiniowaną wartość. Pozwala to na zwrócenie uwagi użytkownika na zmienne w wyjątkowych sytuacjach. Można też ustawić komponentowi wartość domyślną. Dzięki temu komponent nie musi wcale zawierać liczby, a jedynie tekst informujący o stanie w jakim znajduje się urządzenie. Kolejną opcją jest nałożenie maski bitowej na zmienną. Przed wyświetleniem jej wartości element przemnoży ją przez liczbę ustawioną wcześniej jako maska. Jest to przydatne w przypadku gdy jedną zmienną należy interpretować jako więcej niż jedną liczbę. Ostatnią możliwością jest ustawienie wartości mnożnika. Komponent po prostu przemnoży wartość zmiennej przez mnożnik przed jej wyświetlaniem. W przypadku wybrania przez użytkownika opcji maski bitowej i mnożnika jednocześnie, mnożenie wykonywane jest jako ostatnia operacja.

Wszystkie komponenty posiadają dodatkowo kilka wspólnych właściwości, które można konfigurować wedle upodobań. Są nimi kolor tła, kolor obramowania, współrzędne oraz rozmiar. Wszystkie wyżej wspomniane właściwości można zmienić w dowolnej chwili posługując się menu kontekstowym dostępnym pod prawym przyciskiem myszy. Dla wygody użytkownika rozmieszczenie komponentu określa się przeciągając go po scenie z miejsca na miejsce. Rozmiar natomiast reguluje się chwytając za wybrany narożnik komponentu i rozciągając go.

W momencie gdy użytkownik uzna iż projekt jest już gotowy musi go zapisać, aby zmiany nie zostały utracone. Służy do tego guzik z ikoną dyskietki znajdujący się w menu bocznym. Do edycji można wrócić w każdej chwili wybierając projekt ponownie z listy projektów.

1.2. Elements Viewer – moduł wyświetlania widoków

Moduł Viewer służy do prezentowania widoków użytkownikom końcowym – głównie kolejarzom. Viewer jest w stanie wczytać projekt widoku i wyświetlić go użytkownikowi utrzymując przy tym połączenie z urządzeniami i wyświetlając ich zmienne w skonfigurowany przez projektanta sposób.

1.2.1. Lista urządzeń

Pierwszą stroną, która ukazuje się użytkownikowi po uruchomieniu modułu jest lista wszystkich urządzeń podłączonych do aplikacji Elements. Konfiguruje się ją edytując odpowiedni plik xml w folderze aplikacji. Każde urządzenie na liście reprezentowane jest przez pojedynczy kafelek, który wyświetla jego nazwę. Po kliknięciu na guzik z ikoną lupy użytkownik zostaje przeniesiony do listy projektów dostępnych dla urządzenia.

1.2.2. Lista projektów

Interfejs listy projektów prezentuje się analogicznie do listy urządzeń. Każdy projekt reprezentowany jest przez kafelek z jego nazwą i opisem. Aplikacja przyporządkowuje widoki do urządzeń na podstawie przynależności urządzenia do grupy. Oznacza to, że wiele urządzeń może zostać przeniesionych do tego samego widoku, który wyświetli w nim zmienne wybranego urządzenia.

1.2.3. Wizualizacje pracy urządzeń

Głównym elementem widocznym na stronie projektu jest Scena. Wygląda ona prawie identycznie jak w module Designer. Różnica polega na tym, że menu projektowania jest ukryte. Użytkownik nie może też przesuwać komponentów ani wywołać menu kontekstowego. Widok jest przeznaczony jedynie do oglądania.

Po wybraniu projeku użytkownikowi ukazuje się układ komponentów wraz z nadanymi im atrybutami przez projektana. Aplikacja nawiązuje połączenie z serwerem wymiany danych z urządzeniami w celu uzyskania wartości zmiennych dla komponentów typu RefreshedVariable. Połączenie z serwerem utrzymywane jest przez cały czas, gdy użytkownik znajduje się na stronie projektu. Dzieje się tak, ponieważ aplikacja regularnie kontroluje, czy wartości zmiennych nie uległy zmianie i w takim przypadku aktualizuje je w komponentach. Aktualizacja zmiennej polega na wyświetleniu jej w komponencie RefreshedVariable, który jej z nią związany. Przed wyświetleniem zmiennej aplikacja wykonuje dodatkowe czynności, uwzględniając konfigurację danego komponentu przez projektanta (zmiana tekstu, zmiana koloru, maska bitowa, mnożnik).

2. IMPLEMENTACJA

Aplikacja Elements została zaprojektowana w taki sposób, by z łatwością można było ją dopasować do innych składowych systemu ogrzewania rozjazdów kolejowych. W związku z tym, aplikacja przygotowana jest do instalacji na serwerze xsp4 zintegrowanym z apache, przy użyciu maszyny wirtualnej Mono. Wyżej wymienione technologie są kompatybilne z systememami z rodziny Unix, na których działa reszta aplikacji systemu.

2.1. Wykorzystane biblioteki

2.1.1. Razor

Razor jest silnikiem generowania widoków autorstwa Microsoft, który w swojej składni jest znacznie prostszy niż jego poprzednik – aspx. Dzięki temu tworzenie stron, które wymagają użycia logiki staje się jeszcze łatwiejsze. Uproszczenie składni polega głównie na zastąpieniu znaczników aspx jednym znakiem - „@”. Służy on do zadeklarowania w widoku intencji użycia zmiennych dostarczanych do niego przez kontroler. Wydawać by się mogło, że nie jest to duża różnica, lecz w przypadku skomplikowanych konstrukcjach warunkowych łatwo dostrzec przewagę silnika Razor nad aspx. Dla przykładu następującą składnię aspx:

```
<%if (Model.Length == 0)
{%>
    <p>Brak</p>
 <%}
        else
 {%>
    <p><%=Model.Item%></p>
<%} %>
```

można zastąpić w taki sposób:

```
@if (Model.Length == 0)
{
        <p>Brak</p>
}
 else
{
        <p>@Model.Item</p>
}
```

W widoku można umieścić sekcję zawierającą logikę. W tym celu należy umieścić linie kodu w konstrukcji @{...}. Warto jednak nie zapominać o podstawowym założeniu MVC – widok powinien wyświetlać dane, a nie służyć do ich generowania. [1] Razor zawiera również prosty mechanizm tworzenia szablonów stron. Dzięki temu ilość kodu w plikach widoków znacznie się zmniejsza, gdyż powtarzalne części strony takie jak menu, czy stopka można wyeksportować do odrębnych plików.

2.1.2. NHibernate

Nhibernate jest biblioteką, która służy do „rzutowania” tabel baz danych na obiekty (Object Related Mapping) [2]. Dzięki niej możliwe jest posługiwanie się prostymi obiektami, które swoją

strukturą odpowiadają odpowiednim tabelom. Dużą zaletą NHibernate w stosunku do innych bibliotek spełniających tę funkcję jest możliwosć konfiguracji w kodzie. Skonfigurować w kodzie można zarówno sposób wiązania obiektów z tableami, jak i samo połączenie z bazą. Najprostszą metodą konfiguracji jest wskazanie, w którym pakiecie znajdują się obiekty, które powinny zostać powiązane z tabelami za pomocą klasy Configuration. Należy jedynie pamiętać, by nazwy obiektów oraz ich pól były identyczne jak odpowiednie nazwy tabel oraz ich kolumn. Dzięki klasie Configuration uzyskujemy dostęp do obiektu typu SessionFactory, kóry pozwala na otwieranie i zamykanie sesji z bazą danych.

Kolejną zaletą NHibernate jest możliwość wykorzystania technologii LINQ do wykonywania zapytań. Dzięki temu kod C# zawierający zapytanie jest bardzo podobny do odpowiadającemu mu SQL i naturalnie opisuje intencje programisty. Na przykład zapytanie SQL o postaci:

```
SELECT
    Name
FROM
    Device
WHERE
    Id = 1;
```

można zapisać w następujący sposób:

```
string name = session.QueryOver<Device>()
    .Where(d => d.Id == 1)
    .Select(d => d.Name)
    .SingleOrDefault<string>();
```

Używanie biblioteki opiera się głównie na wykorzystaniu obiektu SessionFactory. Służy on do otwierania połączenia z bazą danych, wykonywania operacji oraz zamykania połączenia. Wszystkimi wymienionymi czynnościami zajmuje się biblioteka, pozwalając programiście skupić się na logice biznesowej programu.

2.1.3. jQuery

jQuery to biblioteka, która pozwala na osiąganie atrakcyjnych efektów wizualnych na stronie WWW niewielkim nakładem pracy programisty. Jak wiadomo z dokumentacji [3], całość wykonana jest w oparciu o technologię JavaScript. Biblioteka jest niezależna od przeglądarki więc programista nie musi dostosowywać kodu do wielu różnych przeglądarek WWW. Korzystanie z biblioteki polega głównie na korzystaniu z obiektu jQuery lub znaku "$", który jest aliasem do tego obiektu. jQuery umożliwia wybieranie tablicy węzłów DOM strony internetowej, a następnie wykonywanie na nich różnych działań za pomocą języka JavaScript. Wybieranie węzłów wykonuje się wykorzystując selektory języka CSS3. Przykładowe działania to:

- dodawanie, usuwanie węzłów
- odczytywanie i modyfikowanie atrybutów i zawartości węzłów
- modyfikowanie stylu węzłów
- animacje
- rozbudowana obsługa zdarzeń takich jak kliknięcie lub przesunięcie kursora nad element

Wszystko to można osiągnąc za pomocą kilku linii kodu, na przykład jeśli programista chciałby zmienić kolor tekstu wszystkich węzłów o klasie żed", wystarczy posłużyć się następującym kodem:

```
$('.red').css('color': 'red')
```

Kolejną istotną funkcją jQuery z punktu widzenia mojej aplikacji jest możliwość wykonywania zapytań synchronicznych jak i asynchronicznych AJAX. Znacznie przyspiesza to tworzenie aplikacji, gdyż różne przeglądarki wymagają obsługi wyżej wymienionych zapytań na różne sposoby.

W internecie dostępnych jest wiele wtyczek do biblioteki, które rozszerzają funkcjonalność jQuery. Dzięki nim można na przykład w szybki sposób tworzyć interaktywne tabele z danymi (DataTables), przybornik do wyboru koloru (Evol), czy efektywny pokaz slajdów (Fotorama).

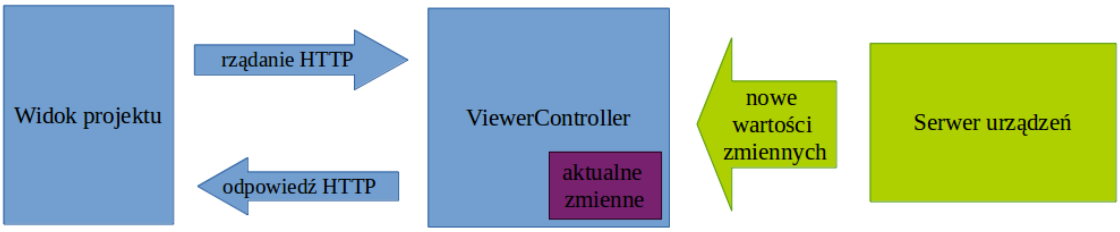
2.2. Architektura aplikacji

Zastosowane przeze mnie rozwiązane zostało zaimplementowane jako aplikacja internetowa. Moduł Viewer ma za zadanie wyświetlanie stanu urządzeń w przeglądarce internetowej, jest to zatem jedyny słuszny wybór. Drugi z modułów służy do projektowania tych widoków. Aby maksymalnie ułatwić użytkownikowi tę czynność, moduł Designer został zrealizowany w ten sam sposób. Dzięki temu projektant widoków ma natychmiastową możliwość oceny rezultatów swojej interakcji z aplikacją. Aplikacja została nazwana Ęlements". Nawiązuje to do małych elementów, z których budowane są widoki.

Architektura projektu Elements składa się z trzech warstw zgodnie z ideą wzorca projektowego MVC. Jest to wymuszone przez framework ASP.NET MVC4.

2.2.1. Warstwa prezentacji - Widoki

Warstwą prezentacji (View) jest interfejs graficzny modułów Designer oraz Viewer. Został on wykonany w dwóch technologiach. Strony internetowe budowane są za pomocą silnika tworzenia widoków Razor. Widoki te zawierają również logikę napisaną w języku TypeScript, który kompiluje się do kodu Javascript. Dzięki temu połączeniu użytkownik jest w stanie poruszać się po interfejsie aplikacji w płynny sposób. Nie ma konieczności przeładowywania formularza na stronach z każdą jego akcją dzięki zastosowaniu zapytań asynchronicznych AJAX. Właśnie na nich oparty w większości jest moduł Viewer. Moduł regularnie wysyła zapytania asynchroniczne do serwera aplikacji w celu pozyskania aktualnych zmiennych. W ten sposób osiągniety został efekt monitorowania urządzeń w czasie rzeczywistym.



Rys. 2.1. Mechanizm odświerzania wartości zmiennych urządzeń

Komponenty zostały zaimplementowane za pomocą klas Typescript. Klasy reprezentujące komponent dziedziczą po klasie Component oraz implementują któryś z interfejsów - Container, TextBox, RefreshedVariable, w zależności od ich przeznaczenia. Dzięki temu aplikacja może za pomocą jednego mechanizmu wyświetlić każdy komponent, zachowując przy tym jego specyficzne zachowanie. Scena jest wyjątkowym komponentem typu kontener. W module Designer pozwala to na przechowywanie komponentów, organizując je w strukturę drzewa. W module Viewer Scena regularnie odświeża zmienne w posiadanych komponentach typu RefreshedVariable. Dzieje się to w następujący sposób:

- Scena iteruje po drzewie komponentów które zawiera i gromadzi listę zmiennych związanych z elementami typu RefreshedVariable
- wysyłane jest zapytanie asynchroniczne z listą zmiennych w celu pozyskania ich
- po odebraniu odpowiedzi z serwera Scena iteruje po swoich komponentach i powiadamia je o zdarzeniu przybycia zmiennej
- Każdy komponent reaguje na przyjście nowej zmiennej w sposób wcześniej skonfigurowany przez projektanta widoków. Wykorzystany został tutaj wzorzec projektowy Observer-Observable.

Komponenty są zdolne do reagowania na zdarzenie przyjścia nowej zmiennej dzięki posiadanych przez nie list tzw. działań (rodzina klas dziedzicząca po klasie Behavior). W momencie przyjścia odpowiedzi z serwera zawierającej zmienne, Scena przekazuje ich wartości do odpowiednich komponentów. Komponenty te uruchamiają metody posiadanych działań, które sprawdzają czy wartość zmiennej spełnia warunek wykonania akcji np. czy wartość zmiennej jest większa od nadanej przez projektanta wartości. Jeśli tak jest, działanie modyfikuje odpowiednie pola komponentu zmieniając np. jego kolor lub tekst. Taki model klas nazywany jest wzorcem projektowym Dekorator. W Elements zostały zaimplementowane następujące zachowania:

- ColorChangeBehavior - zmienia kolor komponentu
- TextChangeBehavior - zmienia tekst komponentu
- ValueChangeBehavior - zmienia wartość numeryczną komponentu - każdy komponent domyślnie posiada to zachowanie

Do komponentu można dodać wiele zachowań jednocześnie. W takim przypadku kolejność zmian atrybutów komponentów jest zależna od kolejności dodania zachowania.

2.2.2. Warstwa logiki biznesowej - Kontrolery

Do warstwy logiki biznesowej w Elements należy część aplikacji wykonana w języku C#, która wykonywana jest przez serwer aplikacyjny. Logika warstwy zawarta jest w tzw. kontrolerach - klasach, których zadaniem jest odbiór rządania HTTP od warsty prezentacji i odesłania im odpowiedzi. Odpowiedzią może być odpowiednio spreparowany widok lub pojedyncza wartość. Jako że aplikacja Elements jest oparta w dużej mierze na zapytaniach asynchronicznych, zadaniem każdego kontrolera jest dostarczenie odpowiedniego widoku oraz obsługa jego zapytań AJAX.

Głównym zadaniem kontrolera modułu Designer (klasa DesignerControler) jest dostarczanie widoków do tworzenia, usuwania i edytowania projektów. Zapis projektu odbywa się porzez serializację klasy JSONProject dostarczanej z widoku, która zawiera drzewo komponentów, nazwę oraz opis projektu. Kontroler jest w stanie zapisać tę kalsę do pliku jak i do bazy danych. Przesłanie projektu do widoku wykonane jest analogicznie, poprzez pobranie danych o projekcie z bazy i przesłanie ich do widoku w postaci obiektu klasy JSONProject.

Kontroler modułu Viewer (ViewerController) odpowiada za dostarczenie widoków prezentujących listę projektów oraz wyświetlających ich zawartość. Projekty przesyłane są do widoku w taki sam sposób jak opisano powyżej. Głównym zadaniem kontrolera jest dostarczanie zmiennych do widoku projektu. Utrzymuje on stałe połączenie z serwerem urządzeń. Dzieki protokołowi wymiany danych P5, serwer urządzeń powiadamia kontroler o nadejściu nowych zmiennych oraz umożliwia natychmiastowe pobranie ich. Wartości zmiennych przechowywane są w słownikowej strukturze danych, która dodatkowo zawiera informacje o tym, czy określona zmienna zmieniła się od ostatniej aktualizacji widoku. Aby maksymalnie zwiększyć szybkość odświerzania wszystkich zmiennych widoku, kontroler przesyła do widoku jedynie te zmienne, które zmieniły swoje wartości od ostatniej aktualizacji.

Pozostałe kontrolery (klasy DeviceSchemesControler, MainControler) zajmują się tworzeniem i edycją grup urządzeń zapisywanych do bazy danych oraz nawigacją po aplikacji.

2.2.3. Warstwa obsługi danych - Modele

Aplikacja Elements zawiera dwa rodzaje modeli:

- Pasywne - wszystkie klasy po stronie serwera zaimplementowane w tehcnologii C#. Służą głównie do przesyłu danych z widoku do kontrolera oraz do komunikacji z bazą danych. Potrzebne jedynie do reprezentacji fragmentów realizowanego projektu. Nie zmieniają same swojego stanu, gdyż nie posiadają one żadnej logiki.
- Aktywne - zaimplementowane w technologii Typescript. Służą do reprezentacji graficznych elementów interfejsu użytkownika np. scena, komponenty lub urządzenia. Są w stanie zmieniać swój stan, odświeżając przy tym interfejs.

Do przechowywania danych została użyta baza PostgreSQL. Wynika to z faktu, iż jest ona już wykorzystywana w istniejącym systemie ogrzewania i oświetlania rozjazdów. Dodatkowymi atutami bazy PostgreSQL są: [4]

- konkurencyjna wydajność
- bogata w typy danych
- dojrzałość projektu, duża społeczność

3. PODSUMOWANIE

Every diploma thesis must include a chapter entitled **Summary**. It should appear before the **Bibliography** and include a review of the main points of the thesis and/or obtained results. The chapter should also state what should be realized if research into the subject of the thesis is continued.

BIBLIOGRAFIA

[1] Helm R. Vlissides J. Gamma E., Johnson R. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[2] NHibernate Community. *NHibernate documentation*. 1 wrzesień 2015.

[3] jQuery Foundation. *jQuery API Documentation*. 1 wrzesień 2015.

[4] Leo S. Hsu Regina O. Obe. *PostgreSQL: Up and Running*. O'Reilly Media, 2012.

SPIS RYSUNKÓW

SPIS TABLIC