

**WYDZIAŁ PODSTAWOWYCH PROBLEMÓW TECHNIKI
POLITECHNIKI WROCŁAWSKIEJ**

ZARZĄDZANIE WYDATKAMI W GOSPODARSTWIE DOMOWYM

ARTUR PTASZEK

Praca inżynierska napisana
pod kierunkiem
dr Wojciecha Macyny

WROCŁAW 2014

Spis treści

1	Cel i zakres pracy	2
2	Projekt systemu	4
2.1	Diagram przypadków użycia	4
2.2	Projekt bazy danych	4
2.3	Diagramy przedstawiające zależności między tabelami	4
2.4	Opis tabel	6
3	Implementacja systemu	8
3.1	Opis technologii	8
3.1.1	Zagadnienia	8
3.1.2	Technologie	9
3.2	Omówienie kodów źródłowych	10
3.2.1	Serwer	10
3.2.2	Klient	15
3.3	Prezentacja systemu	25
4	Instalacja i wdrożenie	33

Wstęp

Praca swoim zakresem obejmuje stworzenie aplikacji do zarządzania wydatkami w gospodarstwie domowym oraz do prezentowania wielu statystyk z nimi związanych. Dzięki zastosowaniu licznych wykresów projekt ma na celu ułatwienie zarządzania przychodami oraz wydatkami. Użytkownikiem docelowym jest każdy zarabiający/studiujący chcący organizować swoje wydatki. Projekt ten wyróżnia od innych to, że ma bardzo intuicyjny wygląd, co ułatwia korzystanie z tego narzędzia. Jest dużo ciekawszy w użyciu i zgodny z konwencją Responsive Web Design.

Praca złożona jest z 5 rozdziałów. Rozdział pierwszy przedstawia zakres pracy oraz cel projektu. Zawiera również opis podstawowych funkcjonalności aplikacji, które zostały zrealizowane w pracy dyplomowej. Dodatkowo zaprezentowane są przykładowe, podobnie działające pod względem funkcjonalności.

W kolejnym rozdziale znajduje się projekt systemu. Przedstawione zostały grupy użytkowników oraz założenia. Umieszczone są diagram przypadków użycia, a także opisany schemat bazy danych oraz omówiono konkretne tabele w niej.

Trzeci rozdział jest poświęcony implementacji systemu. Zaprezentowane zostały użyte technologie, a także poszczególne fragmenty kodów źródłowych z omówieniem ich. Co więcej umieszczone są zrzuty ekranu, pokazujące odrębne widoki i funkcjonalności.

Czwarty rozdział zawiera szczegółową instrukcję instalacji wytworzonego oprogramowania.

W ostatnim znajduje się podsumowanie z całej zrealizowanej pracy oraz wnioski dotyczące możliwości rozszerzenia aplikacji na potrzeby większego projektu.

1 Cel i zakres pracy

Celem pracy dyplomowej było stworzenie aplikacji do zarządzania wydatkami w gospodarstwie oraz wyświetlanie statystyk z nimi związanymi. Jest ona dostosowana do poprawnego działania na wszelkich urządzeniach takich jak: komputery, telefony komórkowe czy tablety. Zadaniem było zaprojektowanie oraz oprogramowanie następujących funkcjonalności:

- Dostosowanie interfejsu do szerokiego wachlarzu urządzeń (telefony komórkowe, tablety, komputery)
- Prosty i intuicyjny interfejs
- Logowanie i rejestracja użytkowników
- Wprowadzanie i edycja przychodów oraz wydatków
- Wydatki i przychody można oznaczać etykietami tj. elektronika lub wypłata, które służą do rozpoznawania typu wydatku
- Filtrowanie wydatków (po etykietach i po czasie)
- Możliwość sprawdzenia czy można sobie pozwolić na jakiś większy wydatek w przyszłych miesiącach (Symulacja)
- Możliwość tworzenia wyzwań na etykiety czyt. ustawienie sobie jakiegoś limitu na konkretną jednostkę czasu

- Możliwość tworzenia wykresów z podsumowaniami wydatków dla etykiet
- Wyświetlanie predefiniowanych wykresów
 - Wykres kołowy ukazujący udział procentowy wydatków oznaczonych konkretnymi etykietami
 - Wykres liniowy ukazujący stosunek przychodów do wydatków w ostatnim roku
 - Wykres liniowy ukazujący bilans przychodów do wydatków w ostatnim roku
 - Wykres liniowy ukazujący stosunek przychodów do wydatków w ostatnim miesiącu

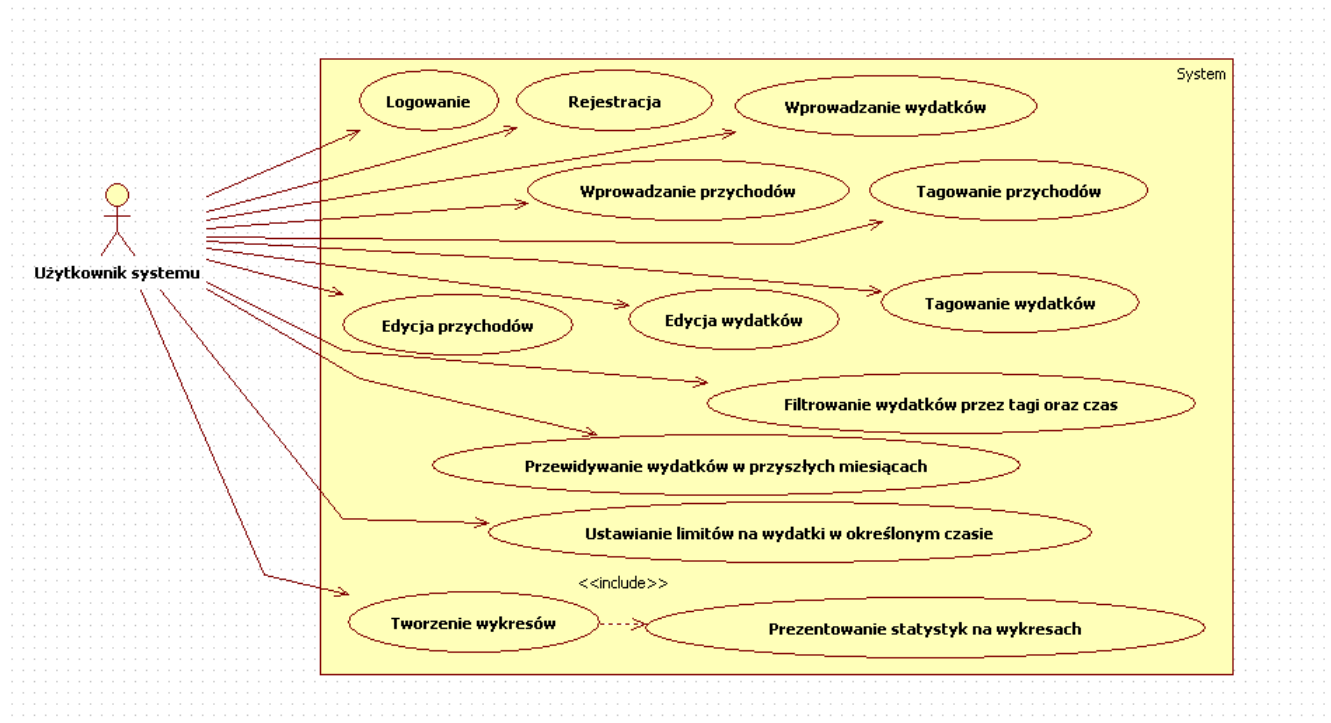
Istnieje wiele aplikacji o podobnej funkcjonalności, które zazwyczaj są połączone z kontem bankowym:

- mBank - posiada interfejs do zarządzania wydatkami
- ING Bank - posiada interfejs do zarządzania wydatkami
- MoneyWiz - Personal Finance
- Money (with sync)
- Bills
- iFinance

Zrealizowana aplikacja jest w formie strony internetowej, przez co jest dostępna na wszystkie platformy. Również idea Responsive Web Design zwiększa zasięg platformowy. Natomiast aplikacje mBanku oraz ING są bardzo mało intuicyjne w użyciu, przez co zrealizowana aplikacja może łatwo trafić do klientów tych programów.

2 Projekt systemu

2.1 Diagram przypadków użycia



Rysunek 1: Diagram przypadków użycia

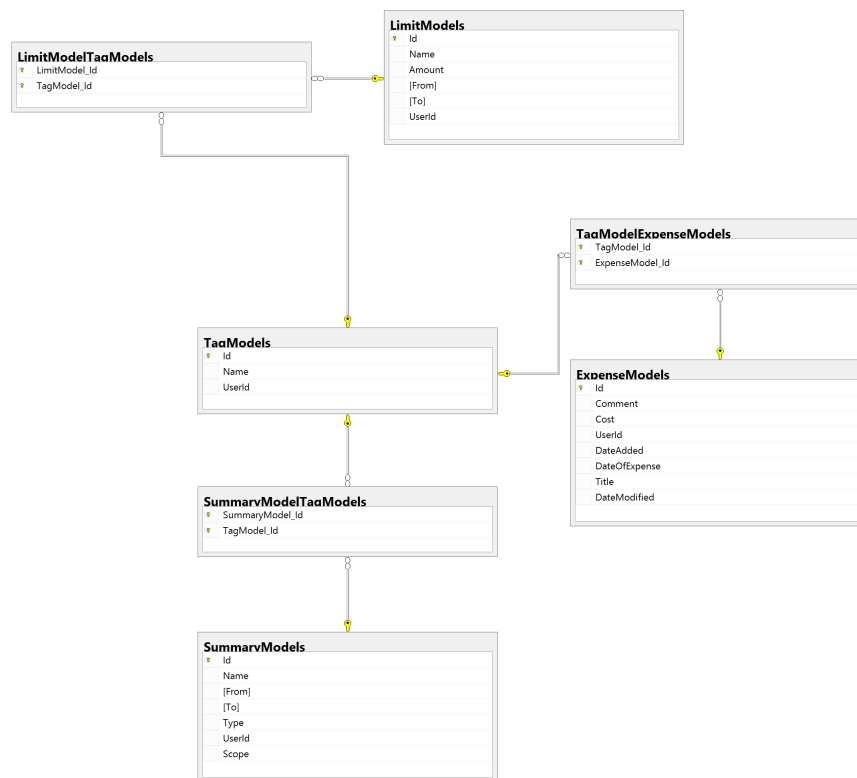
2.2 Projekt bazy danych

Jak wyżej zostało napisane baza danych to Microsoft SQL Server 2012 Express LocalDB Poniżej projekt bazy danych.

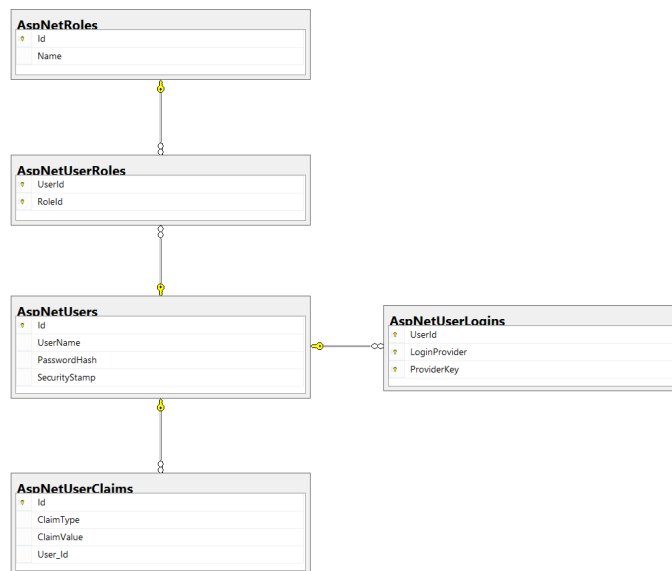
2.3 Diagramy przedstawiające zależności między tabelami

MigrationHistory	
MigrationId	
ContextKey	
Model	
ProductVersion	

Rysunek 2: Tabela migracyjna



Rysunek 3: Tabele aplikacji



Rysunek 4: Tabele autoryzacyjne

2.4 Opis tabel

ExpenseModels Przechowuje wydatki i przychody

- Id int IDENTITY (Identyfikator)
- Comment nvarchar(max) (Komentarz)
- Cost decimal(18,2) (Wartość wydatku/przychodu)
- UserId nvarchar(max) (Identyfikator użytkownika)
- DateAdded datetime (Data dodania wydatku/przychodu)
- DateOfExpense datetime (Data wydatku/przychodu wprowadzona przez użytkownika)
- Title nvarchar(max) (Nazwa wydatku)
- DateModified datetime (Data modyfikacji wydatku/przychodu)

LimitModels Przechowuje dane wyzwań

- Id int IDENTITY (Identyfikator)
- Name nvarchar(max) (Nazwa wyzwania)
- Amount decimal(18,2) (Wartość wyzwania)

- From datetime (Data, od której zaczyna się wyzwanie)
- To datetime (Data, do której trwa wyzwanie)
- UserId nvarchar(max) (Identyfikator użytkownika)

LimitModelTagModels Tabela łącząca tabele wyzwań z etykietami

- LimitModelId int (Id wyzwania)
- TagModelId int (Id etykiety)

SummaryModels Przechowuje dane podsumowań

- Id int IDENTITY (Identyfikator)
- Name nvarchar(max) (Nazwa podsumowania)
- From datetime (Data, od której zaczyna się wyzwanie)
- To datetime (Data, do której trwa wyzwanie)
- Type int (Typ podsumowania)
- Scope int (Zasięg podsumowania np. roczne, miesięczne itd.)
- UserId nvarchar(max) (Identyfikator użytkownika)

SummaryModelTagModels Tabela łącząca tabele podsumowań z etykietami

- SummaryModelId int (Id podsumowania)
- TagModelId int (Id tagu)

TagModelExpenseModels Tabela łącząca tabele wydatków/przychodów z etykietami

- ExpenseModelId int (Id wydatku/przychodu)
- TagModelId int (Id tagu)

TagModels Tabela przechowująca etykiety

- Id int IDENTITY (Identyfikator)
- Name nvarchar(max) (Nazwa etykiety)
- UserId nvarchar(max) (Identyfikator użytkownika)

Pozostałe tabele są wykorzystywane przez standardowy mechanizm autentykacji ASP.NET MVC.

3 Implementacja systemu

W poniższym rozdziale przedstawione zostały liczne technologie, wzorce oraz zagadnienia. Dodatkowo zawiera on przykładowe kody źródłowe programu, które prezentują różne funkcjonalności. W celu zaprezentowania widoków aplikacji załączone zostały zrzuty ekranów poszczególnych zakładek wraz z opisem.

3.1 Opis technologii

3.1.1 Zagadnienia

SPA - Single Page Application Jest to koncepcja, która mówi, że strona powinna być załadowana raz, a każde przejście na kolejną stronę ma być wykonywane asynchronicznie. Wszystkie zmiany strony są pokazywane przez modyfikację drzewa DOM w dokumencie HTML. Uzasadnieniem tego typu stron są rosnące doświadczenia użytkownika przy korzystaniu z aplikacji (User Experience), a także zminimalizowanie transferu danych między przeglądarką a serwerem, przez co czas odpowiedzi serwera jest zdecydowanie szybszy, a użytkownik szybciej zobaczy efekt. Koncepcja wiele czerpie z najnowszych technologii jakimi są HTML5, CSS3, JavaScript, AJAX.

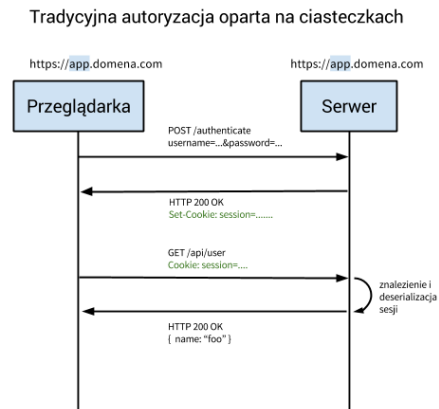
MVC Złożony wzorec architektoniczny służący do organizowania struktury aplikacji posiadających interfejsy graficzne. Wykorzystuje on 3 proste wzorce jakimi są Strategia, Obserwator, Kompozycja.

MVC zakłada podzielenie aplikacji na poniżej wymienione części składowe:

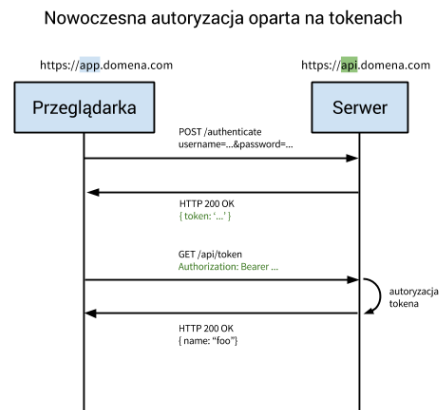
- Model - dane wymagane do utworzenia Widoku
- Widok - opisuje jak przedstawić część modelu użytkownikowi
- Kontroler - agreguje dane z wielu modeli i przygotowuje je aby przekazać do widoku

Logowanie tokenowe Logowanie to działa na podobnej zasadzie jak logowanie przy pomocy ciasteczek, lecz znosi pewne ograniczenia jakie wcześniej wspomniano i najpopularniejszy w dzisiejszych czasach system identyfikacji sesji posiadał. Użytkownik, który chce się zalogować wpisuje swój login i hasło, na których podstawie jest generowany token (identyfikator sesji), który z kolei podajemy przy każdym wymagającym autentykacji żądaniu do serwera (poprzez dodanie w nagłówku żądania HTTP pola `Authentication: Bearer <token>`). Token jest wydawany jedynie na określony czas i służy do autentykacji tylko dla jednego użytkownika. Jedną z zalet tego logowania jest to, że możemy w aplikacjach SPA zalogować się bez odświeżenia strony i dodatkowo aplikacje klienckie możemy “hostować” na innej domenie. Nowoczesne strony wykorzystują już tego typu logowanie, idealnym przykładem jest protokół OAuth 2, który wykorzystują liczne strony:

- Facebook
- Google+
- Twitter



Rysunek 5: Diagram przepływu dla logowania tradycyjnego



Rysunek 6: Diagram przepływu dla logowania tokenowego

3.1.2 Technologie

Aplikacja została podzielona na 2 części logiczne, którymi są:

- Serwer
- Klient

Każda z nich używa innych technologii, ponieważ działają w różnych środowiskach np. klient pracuje jedynie w przeglądarce.

Serwer Został napisany w języku C# wykorzystując ASP.NET MVC 5 i ASP.NET Web Api 2. Statyczne pliki są udostępniane przy pomocy MVC, a reszta serwera została udostępniona w formie Web Service'u.

Silnik bazy danych, który został wykorzystany jest to Microsoft SQL Server 2012 w wersji Express LocalDB. Cała baza danych została zaprojektowana i wygenerowana przy pomocy Entity Framework 6, który także wspomaga wykonywanie zapytań do bazy danych. Zapytania w tej bibliotece nie wykonuje się bezpośrednio przy pomocy języka strukturalnego SQL, lecz pisząc kod i operując na encjach, a zapytania są generowane w locie. Wielką zaletą tego typu rozwiązania jest możliwość przeniesienia aplikacji na inny silnik bazy danych bez zmiany jakiegokolwiek fragmentu kodu. Jedyną rzeczą, którą trzeba będzie wykonać to zmienić parametry połączenia. Jest to typowe rozwiązanie mapowania obiektowo-relacyjnego w skrócie ORM.

Klient Klient został zaimplementowany przy pomocy biblioteki AngularJS. Jest to narzędzie napisane w JavaScript, które umożliwia tworzenie aplikacji za pomocą wzorca projektowego MVC, a także ułatwia tworzenie według koncepcji Single Page Application.

Interfejs graficzny wykorzystuje bibliotekę Bootstrap, która nie tylko ładnie i schludnie wygląda, lecz przy zastosowaniu pewnych reguł zapewnia nam kompatybilność z innymi urządzeniami tj. telefony komórkowe, tablety, czytniki itd. Tego typu zgodność jest nazywana Responsive Web Design i jest coraz częściej stosowana w Internecie.

Wykresy są tworzone przy pomocy HighCharts.js, dzięki czemu w łatwy sposób się je generuje.

Aplikacja została napisana modułowo i każdy takich modułów da się przetestować testami jednostkowymi, a wszystko dzięki narzuconemu schematowi przez AngularJS.

3.2 Omówienie kodów źródłowych

3.2.1 Serwer

```

1 namespace AngularPlanner
2 {
3     public class RouteConfig
4     {
5         public static void RegisterRoutes(RouteCollection routes)
6         {
7             routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
8
9             routes.MapHtml5Routes(new List<string>()
10             {
11                 "login",
12                 "login/{url}",
13                 "register",
14                 "expenses/add",
15                 "expenses",
16                 "expenses/{id}",
17                 "expenses/tag/{tag}",
18                 "expenses/tag/{tag}/{id}",
19                 "expenses/date/{date}",
20                 "expenses/date/{date}/{id}",
21                 "expenses/edit",
22                 "account",
23                 "statistics",
24                 "limits",
25                 "summaries",
26                 "simulations",
27                 "users"
28             }, new { controller = "Home", action = "Index" });
29 
```

```

30         routes.MapRoute(
31             name: "Default",
32             url: "{controller}/{action}/{id}",
33             defaults: new { controller = "Home", action = "Index", id = UrlParameter.
                Optional }
34         );
35     }
36 }
37 }

```

Listing 1: Mapowanie ścieżek routingu

Powyższy kod służy do nauczania serwera IIS interpretowania ścieżek adresu wpisywanych przez użytkownika. Pierwsza linijka metody `RegisterRoutes` jest wymagana przez rozszerzenie `Elmah`, które służy do monitorowania aplikacji. Mówi ona, że ten typ adresów ma być przez naszą aplikację ignorowany. Adresy, które spełniają ten warunek to np. `http://domena.pl/elmah.axd` lub `http://domena.pl/elmah.axd/1234`. Kolejna użyta metoda jest napisana na potrzeby aplikacji metoda, która linkuje każdy z adresów pasujących do wyrażen z pierwszego argumentu metody `MapHtml5Routes` na controller podany w drugim parametrze. Zabieg ten jest wymagany przez router `Angular`'a, który w aplikacji działa w `HTML5Mode` i daje dostęp do adresów URL bez poprzedzania każdej ścieżki znakiem `#`. Kod metody przedstawia Listing 2. Podobnym plikiem jest `AngularPlanner/AngularPlanner/App_Start/WebApiConfig.cs`, który ustawia ścieżki routingu dla `Web Service`'u `REST`owego.

```

1 namespace AngularPlanner.Extensions
2 {
3     public static class RouteCollectionExtension
4     {
5         public static void MapHtml5Routes(this RouteCollection route, List<string> routes,
            object defaults)
6         {
7             routes.ForEach(i => route.MapRoute(i, i, defaults));
8         }
9     }
10 }

```

Listing 2: Metoda `MapHtml5Routes`

W pliku `AngularPlanner/AngularPlanner/App_Start/BundleConfig.cs` są tworzone paczki plików `JavaScript`, które na środowisku produkcyjnym są zmniejszone i zoptymalizowane. Zaletą tego rozwiązania są mniejsze transfery danych i szybszy kod po stronie przeglądarki.

Plik `AngularPlanner/AngularPlanner/App_Start/Startup.Auth.cs` zawiera konfigurację mechanizmu autentykacji tokenowej dla naszej aplikacji. Ponadto istnieje możliwość dodania opcji logowania przez `Facebook` i `Google+`.

Katalog `Controllers` zawiera klasy controllerów. Przedewszystkim są tam kontrolery `Web Api` 2, lecz jest tam jeden, który posiada tylko akcję, która z kolei zwraca użytkownikowi widok podany na Listing 3.

```

1 <!doctype html>
2 <!--[if lt IE 7]> <html class="no-js_lt-ie9_lt-ie8_lt-ie7"> <![endif]-->
3 <!--[if IE 7]> <html class="no-js_lt-ie9_lt-ie8"> <![endif]-->
4 <!--[if IE 8]> <html class="no-js_lt-ie9"> <![endif]-->
5 <!--[if gt IE 8]><!-->
6 <html class="no-js">
7 <!--<![endif]-->
8 <head>
9     <meta charset="utf-8">
10    <meta http-equiv="X-UA-Compatible" content="IE=edge">
11    <title>SmartPlanner</title>
12    <meta name="description" content="">

```

```

13     <meta name="viewport" content="width=device-width">
14     @Styles.Render("~/Content/css")
15 </head>
16 <body ng-app="app">
17     <div class="modal_fade" id="connectionChecker" tabindex="-1" role="dialog" aria-
18         labelledby="myModalLabel" aria-hidden="true">
19         <div class="modal-dialog">
20             <div class="modal-content">
21                 <div class="modal-body">
22                     Problem z łączeniem. ęProsz 6csprbowa 63pniej
23                 </div>
24             </div>
25         </div>
26 <div ng-view>
27     @RenderBody()
28 </div>
29 <!--[if lt IE 9]>
30     @Scripts.Render("~/bundles/compatibility")
31 <![endif]-->
32     @Scripts.Render("~/bundles/vendor")
33
34     @Scripts.Render("~/bundles/app")
35 </body>
36 </html>

```

Listing 3: Widok zwracany przez akcję Index kontrolera Home

Pozostałe kontrollery są do siebie podobne, a kod wybranego z nich można zobaczyć poniżej.

```

1 namespace AngularPlanner.Controllers
2 {
3     [Authorize]
4     [ElmahHandleErrorApi]
5     public class ExpensesController : ApiController
6     {
7         private readonly AngularPlannerContext _db;
8
9         public ExpensesController()
10         {
11             _db = new AngularPlannerContext();
12         }
13
14         [HttpGet]
15         [ActionName("Get")]
16         public async Task<List<ExpenseModel>> GetListByDate(string date, int page = 1)
17         {
18             var userId = User.Identity.GetUserId();
19
20             var timespan = TimeSpanHelper.GetTimeSpan(date);
21
22             try
23             {
24                 return await _db.Expenses
25                     .Where(i => i.UserId == userId && i.DateOfExpense >= timespan.Lower && i.
26                         DateOfExpense <= timespan.Higher)
27                     .Include("Tags")
28                     .AsNoTracking()
29                     .OrderByDescending(i => i.DateOfExpense)
30                     .Skip((page - 1) * 20)
31                     .Take(20)
32                     .ToListAsync();
33             }
34             catch (Exception e)
35             {
36                 Elmah.ErrorSignal.FromCurrentContext().Raise(e);
37             }
38
39             return new List<ExpenseModel>();
40         }
41     }
42 }

```

```

39     }
40
41     [HttpGet]
42     [ActionName("Get")]
43     public async Task<List<ExpenseModel>> GetListByTag(string tag, int page = 1)
44     {
45         var userId = User.Identity.GetUserId();
46
47         try
48         {
49             var query = _db.Expenses.AsQueryable();
50
51             query = tag == "notag"
52                 ? query.Where(i => i.UserId == userId && !i.Tags.Any())
53                 : query.Where(i => i.UserId == userId && i.Tags.Any(j => j.Name == tag));
54
55             return await query.Include("Tags")
56                             .AsNoTracking()
57                             .OrderByDescending(i => i.DateOfExpense)
58                             .Skip((page - 1) * 20)
59                             .Take(20)
60                             .ToListAsync();
61         }
62         catch (Exception e)
63         {
64             Elmah.ErrorSignal.FromCurrentContext().Raise(e);
65         }
66
67         return new List<ExpenseModel>();
68     }
69
70     [HttpGet]
71     [ActionName("Get")]
72     public async Task<List<ExpenseModel>> GetList(int page = 1)
73     {
74         var userId = User.Identity.GetUserId();
75
76         try
77         {
78             return await _db.Expenses
79                             .Where(i => i.UserId == userId)
80                             .Include("Tags")
81                             .AsNoTracking()
82                             .OrderByDescending(i => i.DateOfExpense)
83                             .Skip((page - 1) * 20)
84                             .Take(20)
85                             .ToListAsync();
86         }
87         catch (Exception e)
88         {
89             Elmah.ErrorSignal.FromCurrentContext().Raise(e);
90         }
91
92         return new List<ExpenseModel>();
93     }
94
95     [HttpGet]
96     [ActionName("Get")]
97     public async Task<ExpenseModel> GetSingle(int id)
98     {
99         var userId = User.Identity.GetUserId();
100         return await _db.Expenses.Where(i => i.UserId == userId && i.Id == id).
101                                 FirstOrDefaultAsync();
102     }
103
104     public async Task<HttpResponseMessage> Post([FromBody]ExpenseModel expense)
105     {
106         if (ModelState.IsValid)
107         {

```

```

107         var tagIds = expense.Tags.Select(i => i.Id);
108
109         expense.Tags = await _db.Tags.Where(i => tagIds.Contains(i.Id)).ToListAsync();
110         expense.DateAdded = DateTime.Now;
111         expense.UserId = User.Identity.GetUserId();
112         _db.Expenses.Add(expense);
113
114         await _db.SaveChangesAsync();
115
116         return Request.CreateResponse(HttpStatusCode.Created);
117     }
118
119     return Request.CreateResponse(HttpStatusCode.BadRequest);
120 }
121
122 public async Task<HttpStatusCode> Put([FromBody]ExpenseModel expense)
123 {
124     if (ModelState.IsValid)
125     {
126         var userId = User.Identity.GetUserId();
127         var expenseDB =
128             await _db.Expenses.Include(i => i.Tags).FirstOrDefaultAsync(i => i.Id ==
129                 expense.Id && i.UserId == userId);
130
131         var tagExist = expenseDB.Tags.Select(i => i.Id);
132         var tagIds = expense.Tags.Select(i => i.Id);
133
134         expenseDB.Tags.AddRange(await _db.Tags.Where(i => tagIds.Contains(i.Id) && !
135             tagExist.Contains(i.Id)).ToListAsync());
136         expenseDB.Tags.RemoveAll(i => !tagIds.Contains(i.Id));
137         await _db.SaveChangesAsync();
138
139         return Request.CreateResponse(HttpStatusCode.Created);
140     }
141
142     return Request.CreateResponse(HttpStatusCode.BadRequest);
143 }
144
145 public async Task<HttpStatusCode> Delete(int id)
146 {
147     var userId = User.Identity.GetUserId();
148     var expense = await _db.Expenses.Where(i => i.UserId == userId && i.Id == id).
149         FirstOrDefaultAsync();
150     if (expense == null)
151     {
152         return Request.CreateResponse(HttpStatusCode.NotFound);
153     }
154     _db.Expenses.Remove(expense);
155     await _db.SaveChangesAsync();
156
157     return Request.CreateResponse(HttpStatusCode.OK);
158 }
159
160 protected override void Dispose(bool disposing)
161 {
162     if (disposing)
163     {
164         _db.Dispose();
165     }
166     base.Dispose(disposing);
167 }

```

Listing 4: Kontroller ExpensesController

Klasa kontrolera dziedziczy po klasie abstrakcyjnej ApiController, by mogłabyć rozpoznana przez mechanizm refleksji jako kontroler Web Api 2 i dodatkowo uzyskać dostęp do kilku przydat-

nych metod charakterystycznych dla tego typu kontrolerów. Trochę wyżej są 2 atrybuty. `Authorize` oznacza, że dostęp do kontrolera mają jedynie zalogowani użytkownicy, a atrybut `ElmahHandleErrorApi` informuje bibliotekę `Elmah`, że ma przechwytywać wyjątki z tego kontrolera. W konstruktorze mamy tworzenie instancji połączenia z bazą danych przy pomocy `Entity Framework`. W klasie mamy zaimplementowane wszystkie akcje CRUD, czyli `Create` (metoda `Post`), `Read` (metody `GetListByDate`, `GetListByTag`, `GetList`, `GetSingle`), `Update` (metoda `Put`) i `Delete` (metoda `Delete`). Linia nr 100 przedstawia przykładowe zapytanie do bazy. Odpowiada to zapytaniu w SQLu

```
SELECT TOP 1 * FROM dbo.ExpenseModels WHERE UserId = <UserId> AND Id = <Id>
```

Kolejnym dobrym przykładem zapytania są linie od 78 do 85. Metoda `Include` wykonuje `LEFT OUTER JOIN` z tabelą `TagsModels`, znowu metoda `OrderByDescending` dodaje sortowanie po `DateOfExpense` (`ORDER BY DateOfExpense DESC`), `Skip` opuszcza liczbę wpisów podanych w parametrze, a `Take` pobiera 20 rekordów. Ważnym elementem tego zapytania jest także `ToListAsync`. Do póki nie użyjemy tej metody zapytanie nie zostanie wykonane, a jedynie przygotowane. Po wykonaniu zapytania (zmaterializowaniu) dane są typu `List`, a wcześniej były `IQueryable`.

Pliki w katalogu `Models` z wyłączeniem `AngularPlannerContext` są to pliki reprezentujące bazę danych. Każda klasa reprezentuje rzeczywistą tabelę w bazie. Dla przykładu została przedstawiony model `Expense` w Listingu 5

```
1 namespace AngularPlanner.Models
2 {
3     public class ExpenseModel
4     {
5         [Key]
6         public int Id { get; set; }
7         public List<TagModel> Tags { get; set; }
8         public string Title { get; set; }
9         public string Comment { get; set; }
10        public decimal Cost { get; set; }
11        public string UserId { get; set; }
12        public Nullable<DateTime> DateAdded { get; set; }
13        public Nullable<DateTime> DateModified { get; set; }
14        public Nullable<DateTime> DateOfExpense { get; set; }
15    }
16 }
```

Listing 5: Model `ExpenseModel`

Atrybut `Key` oznacza pole, które jest typu `PRIMARY KEY`. Pole `Tags` jest to tak naprawdę odwołanie do tabeli `TagsModels`. `Entity Framework` sam stwierdza jakiego typu relacjami mają być połączone table w bazie danych (jeden-do-wielu, wiele-do-wielu itd.). Pola `Nullable<DateTime>` oznaczają, że to nie są wymagane i mogą mieć wartość `NULL` w bazie.

3.2.2 Klient

Cały kod aplikacji został zorganizowany według rekomendowanej przez Angulara struktury katalogów [9].

Przy asynchronicznych akcjach jest wykorzystywana okrojona biblioteka `Q.js` [10] zaimplementowana w Angularze. Implementuje ona ustandaryzowany system obietnic `Promises/A+` [8]

```
1 'use_strict';
2
3 module.exports = function (grunt) {
4
5     // Load grunt tasks automatically
6     require('load-grunt-tasks')(grunt);
7 }
```



```

8 // Time how long tasks take. Can help when optimizing build times
9 require('time-grunt')(grunt);
10
11 // Define the configuration for all the tasks
12 grunt.initConfig({
13   // Project settings
14   yeoman: {
15     // configurable paths
16     appSrc: 'AngularPlanner/App_src',
17     appDest: 'AngularPlanner/App',
18     testSpec: 'AngularPlanner.Tests/App/Spec',
19     css: 'AngularPlanner/Content',
20     images: 'AngularPlanner/Images',
21     views: 'AngularPlanner/Views'
22   },
23   // Watches files for changes and runs tasks based on the changed files
24   watch: {
25     js: {
26       files: ['<%=yeoman.appSrc%>/**/*.js'],
27       tasks: ['newer:ngmin:app', 'newer:jshint:all'],
28       options: {
29         livereload: true
30       }
31     },
32     views: {
33       files: ['<%=yeoman.appSrc%>/**/*.html'],
34       tasks: ['newer:copy:views'],
35       options: {
36         livereload: true
37       }
38     },
39     jsTest: {
40       files: ['<%=yeoman.testSpec%>/**/*.js'],
41       tasks: ['newer:jshint:test', 'karma']
42     },
43     gruntfile: {
44       files: ['Gruntfile.js']
45     },
46     livereload: {
47       options: {
48         livereload: 35729
49       },
50       files: [
51         '<%=yeoman.views%>/**/*.cshtml',
52         '<%=yeoman.appSrc%>/**/*.html',
53         '<%=yeoman.css%>/**/*.css',
54         '<%=yeoman.images%>/**/*.{png,jpg,jpeg,gif,webp,svg}'
55       ]
56     }
57   },
58   ngmin: {
59     app: {
60       cwd: '<%=yeoman.appSrc%>',
61       expand: true,
62       src: ['/**/*.js'],
63       dest: '<%=yeoman.appDest%>'
64     }
65   },
66   copy: {
67     views: {
68       cwd: '<%=yeoman.appSrc%>',
69       expand: true,
70       src: ['/**/*.html'],
71       dest: '<%=yeoman.appDest%>'
72     }
73   },
74 },
75
76

```

```

77 // Make sure code styles are up to par and there are no obvious mistakes
78 jshint: {
79   options: {
80     jshintrc: '.jshintrc',
81     reporter: require('jshint-stylish')
82   },
83   all: [
84     'Gruntfile.js',
85     '<%=_yeoman.appSrc_%>/**/*.js'
86   ],
87   test: {
88     options: {
89       jshintrc: 'test/.jshintrc'
90     },
91     src: ['<%=_yeoman.testSpec_%>/**/*.js']
92   }
93 },
94
95 karma: {
96   unit: {
97     configFile: 'karma.conf.js',
98     singleRun: true
99   }
100 }
101 });
102
103
104 grunt.registerTask('dev', function () {
105   grunt.task.run([
106     'watch'
107   ]);
108 });
109
110 grunt.registerTask('test', [
111   'karma'
112 ]);
113
114 grunt.registerTask('default', function() {
115   grunt.task.run([
116     'watch'
117   ]);
118 });
119 });

```

Listing 6: Plik konfiguracyjny systemu budującego Grunt

Kod przedstawiony na powyższym listingu jest plikiem konfiguracyjnym dla systemu do automatyzacji pewnych zadań Grunt. Można go porównać do Makefile, ponieważ jest to narzędzie tego typu, lecz istnieje do niego wiele rozszerzeń, które ułatwiają pracę. Powyżej definiujemy 3 zadania: default, test, dev. Każdy z nich można uruchomić za pomocą polecenia `grunt <nazwa_zadania>`. Dev i default nasłuchują katalogi z plikami *.js, *.css, *.html, a następnie robią wstępną kompilację, aby kod napisany był możliwy do pomniejszenia [12] i sprawdzany pod względem poprawności składni. Także informowana jest przeglądarka o zmianach, których następstwem jest automatyczne odświeżenie okna przeglądarki. Jest to czynność, która znacznie przyspiesza proces wytwarzania oprogramowania.

Plik z Listingu 7 jest plikiem startowym aplikacji. Definiuje się tutaj jakie moduły wchodzi w skład aplikacji i konfiguruje środowisko do dalszej pracy. W tym przypadku został włączony `html5Mode` i domyślne przekierowanie na stronę `domena.pl/statistics`. W pliku z Listingu 3 jest podłączenie aplikacji do drzewa DOM za pomocą `<body ng-app="app">`.

```

1 'use_strict';
2
3 /**

```

```

4  * app Module
5  *
6  * init app module
7  */
8  angular.module('app', [
9      'ngSanitize',
10     'ngResource',
11     'ngCookies',
12     'ngRoute',
13     'http',
14     'login',
15     'register',
16     'index',
17     'expenses',
18     'navbar',
19     'nav',
20     'simulations',
21     'statistics',
22     'limits',
23     'progressbar',
24     'connectionChecker',
25     'summaries',
26     'simulations'
27 ]);
28
29 angular.module('app')
30     .config(function($routeProvider, $locationProvider) {
31         $locationProvider.html5Mode(true);
32
33         $routeProvider
34             .otherwise({redirectTo: '/statistics'});
35     });

```

Listing 7: Plik startowy aplikacji

Mechanizm autentykacji po stronie klienta został napisany na podstawie książki [11]. Jest on podzielony na trzy moduły. Pierwszy z nich przedstawiony na Listingu 8 pośredniczy przy żądaniach i odpowiedziach dodając Token w nagłówkach żądania, jeśli jest on zapisany w `localStorage` przeglądarki. Jeśli Token wygaśnie lub jest nieprawidłowy, serwer poinformuje klienta statusem kodu 401 – Unauthorized [7]

```

1  'use_strict';
2  /**
3   * auth.interceptor Module
4   *
5   * Module for handling authentication errors
6   */
7  angular.module('auth.interceptor', []).factory('authInterceptor', function($window, $location){
8      return {
9          'request': function(config) {
10             config.headers = config.headers || {};
11             if($window.localStorage.token) {
12                 config.headers.Authorization = 'Bearer_' + $window.localStorage.token;
13             }
14             return config;
15         },
16         'response': function(config) {
17             if(config.status === 401) {
18                 $location.path('/login');
19             }
20             return config;
21         }
22     };
23 })
24 .config(function($httpProvider) {
25     $httpProvider.interceptors.push('authInterceptor');

```

```
26 });
```

Listing 8: Moduł pośredniczący przy żądaniach i odpowiedziach HTTP

Kolejny przedstawiony na Listingu 9 moduł to udostępniający najważniejsze metody do autentykacji i sprawdzenia zalogowania użytkownika.

```
1  'use_strict';
2
3  /**
4   * auth.service Module
5   *
6   * Service for logging etc
7   */
8  angular.module('auth.service', [])
9    .factory('auth', function($http, $q, $window){
10     var service = {
11       register: function(username, password, password2) {
12         if(!username || !password || !password2) {
13           throw new Error('[App]_Security:_You_must_specify_username_and_password');
14         }
15       },
16       return $http
17         .post('/api/account/register', {
18           userName: username,
19           password: password,
20           confirmPassword: password2
21         });
22     },
23     logout: function() {
24       delete $window.localStorage.token;
25     },
26     login: function(username, password) {
27       var defer = $q.defer();
28
29       if(!username || !password) {
30         throw new Error('[App]_Security:_You_must_specify_username_and_password');
31       }
32
33       $http.postUrlEncoded('/token', {
34         'grant_type': 'password',
35         username: username,
36         password: password
37       })
38         .success(function(token) {
39           $window.localStorage.token = token.access_token;
40           defer.resolve();
41         })
42         .error(function(data) {
43           defer.reject(data);
44         });
45
46       return defer.promise;
47     },
48     isAuthenticated: function() {
49       var defer = $q.defer();
50
51       $http
52         .get('/api/account/userInfo')
53         .success(function(userInfo) {
54           defer.resolve(userInfo);
55         })
56         .error(function(err, status) {
57           console.log('[App]_Security:_(' + status + '):_ ' + err);
58           defer.reject(err);
59         });
60
61       return defer.promise;
62     }
63   }
```

```

63     };
64
65     return service;
66 });

```

Listing 9: Moduł wystawiający interfejs do autentykacji

Udostępnia on metody login, logout, register, isAuthenticated. W przypadku login serwer jest odpytywany czy użytkownik z takimi danymi logowania istnieje w bazie danych, jeżeli tak jest, to w odpowiedzi dostajemy Token autoryzacyjny, który zostaje zapisany w pamięci przeglądarki.

Ostatnim z modułów odpowiedzialnych za logowanie jest przedstawiony poniżej. Sprawdza czy widoki mogą być pokazane użytkownikowi. Adresy końcowe dla autentykacji zostały zaczerpnięte z tutoriala [1].

```

1  'use_strict';
2  /**
3   * auth.checker Module
4   *
5   * promises for $routeProvider
6   */
7  angular.module('auth.checker', ['auth.service'])
8    .provider('authChecker', {
9      require: function(authChecker) {
10         return authChecker.require();
11      },
12      $get: function(auth) {
13         var service = {
14           require: function() {
15             return auth.isAuthenticated();
16           }
17         };
18         return service;
19       }
20     });
21

```

Listing 10: Moduł sprawdzający czy dany widok jest dostępny dla anonimowego użytkownika

Do wymiany danych używany jest mechanizm \$resource [2]. Przykład implementacji przedstawiony został na Listingu 11. Udostępnia on łatwy interfejs do komunikacji z serwerem.

```

1  'use_strict';
2  /**
3   * resources.tags Module
4   *
5   * tags resource
6   */
7  angular.module('resources.tags', [])
8    .factory('Tags', function($resource){
9      return $resource('/api/tags/:id', {id: '@id'},
10        {
11          'update': { method: 'PUT' }
12        });
13    });

```

Listing 11: Tags Resource

Zastosowanie w/w mechanizmu jest podane w Listingu 12. Poprzez Tags.query(callback) odpytujemy serwer o listę wszystkich etykiet na serwerze. Poprzez utworzenie nowej instancji tworzymy nową etykietę, a wywołując na nim metodę \$.save(callback) zapisujemy go na serwerze.

```

1  'use_strict';
2  /**

```

```

3  * tagsPicker Module
4  *
5  * Pick tags
6  */
7  angular.module('tagsPicker', ['resources'])
8    .controller('TagsPickerCtrl', function($scope, Tags, $rootScope){
9      $scope.checked = [];
10     $scope.tagsList = [];
11
12     function updateChecked() {
13       if(!$scope.tags) {
14         $scope.tags = [];
15       }
16       $scope.checked = [];
17       $scope.tagsList.forEach(function(tag) {
18         var founded = false;
19         $scope.tags.forEach(function(selectedTag) {
20           if(selectedTag.id === tag.id) {
21             founded = true;
22           }
23         });
24
25         $scope.checked.push(founded);
26       });
27     }
28
29     $scope.delete = function($index) {
30       $scope.tagsList[$index].$delete();
31       $rootScope.$broadcast('tagsPicker:splice', $index);
32     };
33
34     $scope.toggle = function($index) {
35       var selectedTag = $scope.tagsList[$index],
36         indexOf = -1;
37
38       $scope.tags.forEach(function(tag, index) {
39         if(tag.id === selectedTag.id) {
40           indexOf = index;
41         }
42       });
43
44       if(indexOf !== -1) {
45         $scope.tags.splice(indexOf, 1);
46       } else {
47         $scope.tags.push(selectedTag);
48       }
49     };
50
51     $scope.$on('tagsPicker:push', function(e, tag) {
52       $scope.tagsList.push(tag);
53     });
54
55     $scope.$on('tagsPicker:check', function(e, tag) {
56       $scope.tags.push(tag);
57       e.stopPropagation();
58     });
59
60     $scope.$on('tagsPicker:splice', function(e, $index) {
61       $scope.tagsList.splice($index, 1);
62     });
63
64     $scope.$watchCollection('tags', updateChecked);
65
66     (function() {
67       Tags.query(function(tags) {
68         $scope.tagsList = tags;
69         updateChecked();
70       });
71     })();

```

```

72  })
73  .controller('TagsPickerAddCtrl', function(Tags, $scope, $rootScope){
74    $scope.show = false;
75    $scope.clear = function() {
76      $scope.show = false;
77      delete $scope.tag;
78      $scope.form.$setPristine(true);
79    };
80
81    $scope.add = function() {
82      var tag = new Tags($scope.tag);
83
84      $scope.show = false;
85      delete $scope.tag;
86      $scope.form.$setPristine(true);
87
88      tag.$save(function(tag) {
89        $rootScope.$broadcast('tagsPicker:push', tag);
90        $scope.$emit('tagsPicker:check', tag);
91      });
92    };
93  })
94  .directive('tagsPicker', function() {
95    return {
96      scope: {
97        tags: '='
98      },
99      controller: 'TagsPickerCtrl',
100      restrict: 'EA',
101      templateUrl: '/App/components/tagsPicker/tagsPicker.html',
102      replace: true
103    };
104  });

```

Listing 12: Dyrektywa <tags-picker />

Warto także wspomnieć, że Listing 12 tworzy nowy element HTMLa, który jest nazwany <tags-picker />. Definicja takiego elementu jest zawarta w liniach od 94 do 104. Parametr scope nakazuje utworzyć obiekt podgląd, który jest obiektem dla tego elementu. Danymi źródłowymi są atrybuty dostarczone do elementu. Wartość '=' nakazuje zachować 2-way data binding [3] między widokiem, a kontrolerem dla tej właściwości. controller wskazuje, który kontroler ma być wykorzystany. Jest on zaimplementowany wyżej. Parametr restrict determinuje jakiego typu ma być dyrektywa [4]. Więcej o konfiguracjach można przeczytać [5].

Metody i właściwości są udostępnianie widokowi przy pomocy \$scope. W kontrolerze jest kilka przykładów: \$scope.delete = function() {}, \$scope.tags = []. Także dla właściwości w \$scope udostępniony został mechanizm obserwowania zmian w tej właściwości. Jest to klasyczne zastosowanie wzorca Obserwator. Istnieje też system wiadomości, w którym możemy nasłuchiwać na konkretną wiadomość (\$scope.\$on('tagsPicker:push')) i je wysyłać w górę drzewa podglądów (\$scope.\$emit('tagsPicker:push')) lub w dół (\$scope.\$emit('tagsPicker:push')).

```

1  <div class="row">
2    <button type="button" ng-click="toggle($index)" class="btn btn-default btn-xs pull-left tag"
      ng-class="{ 'btn-primary': _checked[$index] }" ng-repeat="tag in tagsList">{{tag.name}}
3    <span class="glyphicon glyphicon-remove" ng-click="delete($index)"></span>
4  </button>
5  <form class="form-xs pull-left" role="form" name="form" ng-submit="add()" ng-controller="
      TagsPickerAddCtrl">
6    <div class="form-group has-feedback input-xs" ng-class="{ 'has-success': _form.tagName.$valid,
      _has-error': _form.tagName.$dirty && _form.tagName.$invalid }">
7      <label for="tagName" class="control-label sr-only">Nazwa</label>
8      <input ng-show="show" ng-model="tag.name" type="text" name="tagName" id="tagName" class="
        form-control" value="" required placeholder="Nazwa" />

```

```

9      <span ng-show="form.tagName.$dirty_&&_form.tagName.$invalid" class="glyphicon glyphicon-
      remove_form-control-feedback"></span>
10     <span ng-show="form.tagName.$valid" class="glyphicon glyphicon-ok_form-control-feedback">
      </span>
11   </div>
12   <button type="submit" class="submit-hidden"></button>
13   <button ng-show="show_&&!readonly" ng-click="clear()" type="button" class="btn btn-default
      btn-xs"><span class="glyphicon glyphicon-remove"></span></button>
14   <button ng-show="!show_&&!readonly" ng-click="show=_true" type="button" class="btn btn-
      success btn-xs"><span class="glyphicon glyphicon-plus-sign"></span></button>
15 </form>
16 </div>

```

Listing 13: Widok dla dyrektywy <tags-picker />

W widoku podpięte są metody na akcje. Dla przykładu można podać `ng-click="delete($index)"`. Kliknięcie w element wywołuje tą metodę z kontrolera. Atrybut `ng-repeat="tag in tagsList"` nakazuje powielić element, do którego jest podpięty tyle razy ile jest elementów w tablicy `tagsList`.

```

1  'use_strict';
2  /**
3   * Expenses Module
4   *
5   * Module for expenses
6   */
7  angular.module('expenses', ['auth', 'resources', 'tagsPicker'])
8    .config(function($routeProvider, authCheckerProvider) {
9      $routeProvider
10        .when('/expenses/edit', {
11          templateUrl: '/App/expenses/expenses-edit.html',
12          controller: 'ExpensesEditCtrl',
13          resolve: {
14            currentUser: authCheckerProvider.require
15          }
16        })
17        .when('/expenses', {
18          templateUrl: '/App/expenses/expenses-list.html',
19          controller: 'ExpensesListCtrl',
20          resolve: {
21            currentUser: authCheckerProvider.require,
22            expenses: function(Expenses) {
23              return Expenses.query().$promise;
24            }
25          }
26        })
27        .when('/expenses/:page', {
28          templateUrl: '/App/expenses/expenses-list.html',
29          controller: 'ExpensesListCtrl',
30          resolve: {
31            currentUser: authCheckerProvider.require,
32            expenses: function(Expenses, $route) {
33              return Expenses.query({page: $route.current.params.page}).$promise;
34            }
35          }
36        })
37        .when('/expenses/tag/:tag', {
38          templateUrl: '/App/expenses/expenses-list.html',
39          controller: 'ExpensesListCtrl',
40          resolve: {
41            currentUser: authCheckerProvider.require,
42            expenses: function(Expenses, $route) {
43              return Expenses.query({
44                tag: $route.current.params.tag,
45                page: $route.current.params.page
46              }).$promise;
47            }
48          }
49        })

```



```

50     .when('/expenses/tag/:tag/:page', {
51       templateUrl: '/App/expenses/expenses-list.html',
52       controller: 'ExpensesListCtrl',
53       resolve: {
54         currentUser: authCheckerProvider.require,
55         expenses: function(Expenses, $route) {
56           return Expenses.query({
57             tag: $route.current.params.tag,
58             page: $route.current.params.page
59           }).$promise;
60         }
61       }
62     })
63     .when('/expenses/date/:date', {
64       templateUrl: '/App/expenses/expenses-list.html',
65       controller: 'ExpensesListCtrl',
66       resolve: {
67         currentUser: authCheckerProvider.require,
68         expenses: function(Expenses, $route) {
69           return Expenses.query({
70             date: $route.current.params.date,
71             page: $route.current.params.page
72           }).$promise;
73         }
74       }
75     })
76     .when('/expenses/date/:date/:page', {
77       templateUrl: '/App/expenses/expenses-list.html',
78       controller: 'ExpensesListCtrl',
79       resolve: {
80         currentUser: authCheckerProvider.require,
81         expenses: function(Expenses, $route) {
82           return Expenses.query({
83             date: $route.current.params.date,
84             page: $route.current.params.page
85           }).$promise;
86         }
87       }
88     });
89   })
90   .controller('ExpensesListCtrl', function($scope, expenses, Expenses, $route, $location) {
91     if(!$route.current.params.page) {
92       $route.current.params.page = 1;
93     }
94
95     $scope.expenses = expenses;
96
97     $scope.page = $route.current.params.page;
98
99     $scope.$on('expenses-invalidate', function(e) {
100       e.stopPropagation();
101       $location.path('/expenses');
102     });
103
104     $scope.$on('expenses:editor:close', function() {
105       $scope.editor = false;
106     });
107
108     $scope.edit = function($index) {
109       $scope.editor = true;
110       $scope.editorIndex = $index;
111       $scope.$broadcast('expenses:editor:open', $scope.expenses[$index]);
112     };
113
114     $scope.delete = function($index) {
115       var tmp = $scope.expenses[$index];
116       $scope.expenses.splice($index, 1);
117       tmp.$delete();
118     };

```

```

119
120     $scope.next = function() {
121         $location.path('/expenses/' + (parseInt($route.current.params.page) + 1));
122     };
123
124     $scope.prev = function() {
125         $location.path('/expenses/' + (parseInt($route.current.params.page) - 1));
126     };
127 })
128 .controller('ExpensesListEditCtrl', function($scope){
129     $scope.$on('expenses:editor:open', function(e, expense) {
130         $scope.expense = expense;
131     });
132
133     $scope.save = function() {
134         $scope.expense.$update();
135     };
136
137     $scope.close = function() {
138         $scope.$emit('expenses:editor:close');
139     };
140 })
141 .controller('ExpenseAddCtrl', function($scope, Expenses){
142     $scope.more = false;
143     $scope.tagsList = [{id:1}, {id:2}];
144
145     $scope.add = function() {
146         var expense = new Expenses({
147             dateOfExpense: $scope.expense.date,
148             title: $scope.expense.title,
149             cost: $scope.expense.cost,
150             comment: $scope.expense.comment,
151             tags: $scope.expense.tags
152         });
153
154         delete $scope.expense;
155
156         $scope.form.$setPristine(true);
157
158         expense.$save(function() {
159             $scope.$emit('expenses-invalidate');
160         });
161     };
162 });

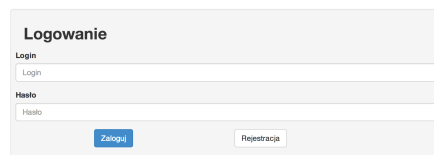
```

Listing 14: Przykładowy kod podstrony

Na samej górze mamy definicje adresów, które prowadzą do podstrony. Każdy z nich ma podany widok i kontroler, przez które będzie obsługiwany adres. W właściwości są zależności, które są wymagane do przejścia na podstronę np. `currentUser: authCheckerProvider.require` jest odpowiedzialny za autoryzację. Jeżeli, któraś z tych zależności nie zostanie spełniona to przejście nie zostanie wykonane, a sam `$routeProvider` wyśle komunikat `$routeChangeError` [6]. Poniżej mamy definicję samego kontrolera.

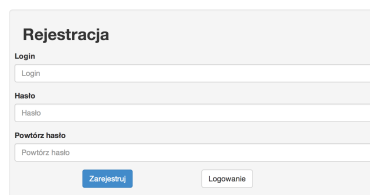
3.3 Prezentacja systemu

Poniżej został przedstawiony ekran logowania. Jest to także pierwszy ekran widziany przez użytkownika. Po zalogowaniu ma dostęp do swoich wykresów i wydatków. Jeżeli użytkownik nie ma konta może nacisnąć przycisk Rejestracja i zostanie przeniesiony do widoku pokazanego na rysunku nr 8.



Rysunek 7: Ekran logowania

Na rysunku nr 8 jest ekran rejestracji, którego pomocy możemy założyć nowe konto. Przyciskiem Logowanie wracamy do ekranu logowania.

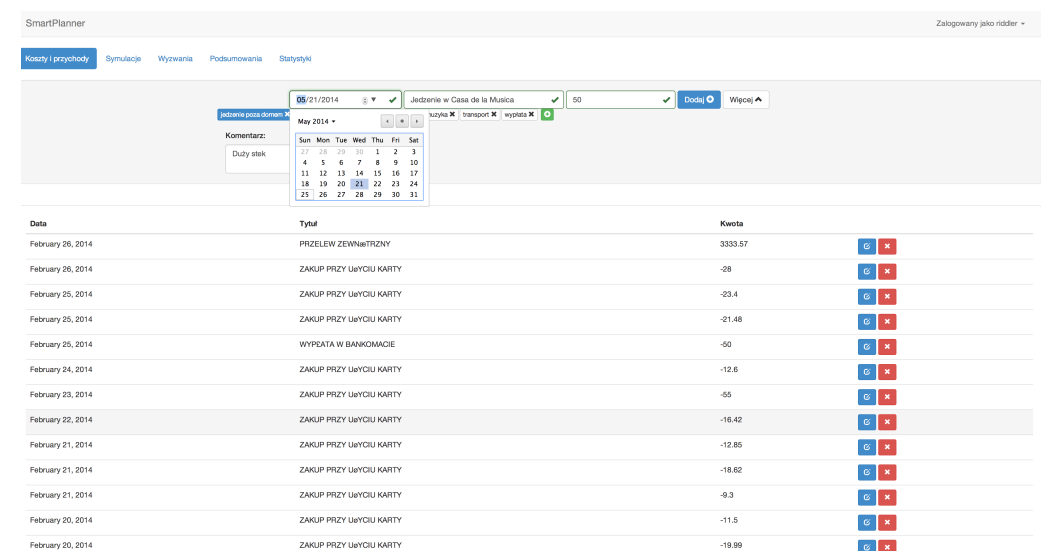


Rysunek 8: Ekran rejestracji

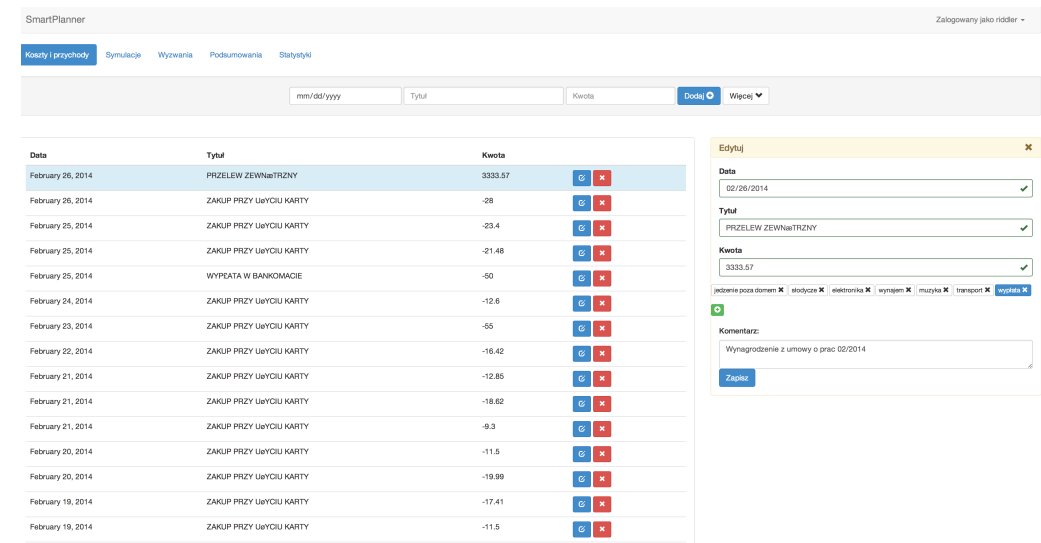
Na liście wszystkich wydatków i przychodów możemy dodawać nowe wydatki i przychody, a także usuwać i edytować istniejące. Aby dodać nowy wydatek trzeba skorzystać z formularza na górze i wprowadzić wszystkie wymagane dane. Dodatkowe dane można uzupełnić klikając na przycisk *Więcej*. Ujawniają się wtedy etykiety do wyboru i komentarz do uzupełnienia.

Edycja elementu następuje po naciśnięciu niebieskiego przycisku z ikoną długopisa, po naciśnięciu którego pojawia się panel do edycji przedstawiony na rysunku nr 10. Znajdują się tam wszystkie opcje dostępne przy zakładaniu nowego elementu. Aby zapisać wydatek trzeba potwierdzić zmiany przez naciśnięcie *Zapisz*.

Obok przycisku edycji znajduje się czerwony przycisk do usuwania.

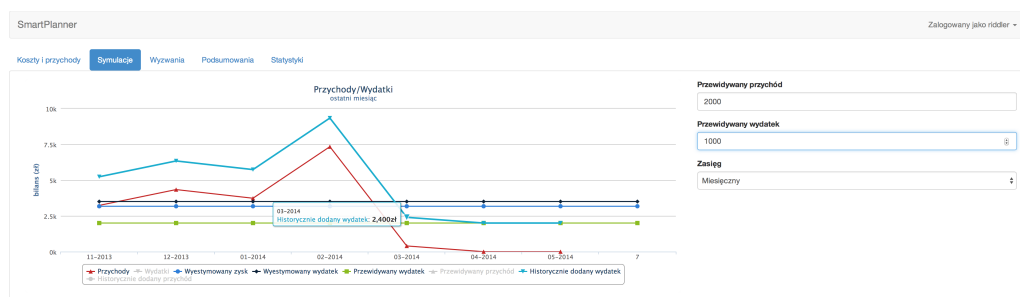


Rysunek 9: Lista wydatków i dodawanie



Rysunek 10: Edycja wydatków

Poniżej jest ekran do planowania wydatków. W panelu po prawej wprowadzamy planowany przychód i wydatek na przyszłe miesiące/dni/lata i na żywo jest przedstawiana zdolność do realizacji wyżej wymienionego planu. Na wykresie jest przedstawione kilka serii, lecz każdą można wyłączyć przez naciśnięcie etykiety serii w legendzie.

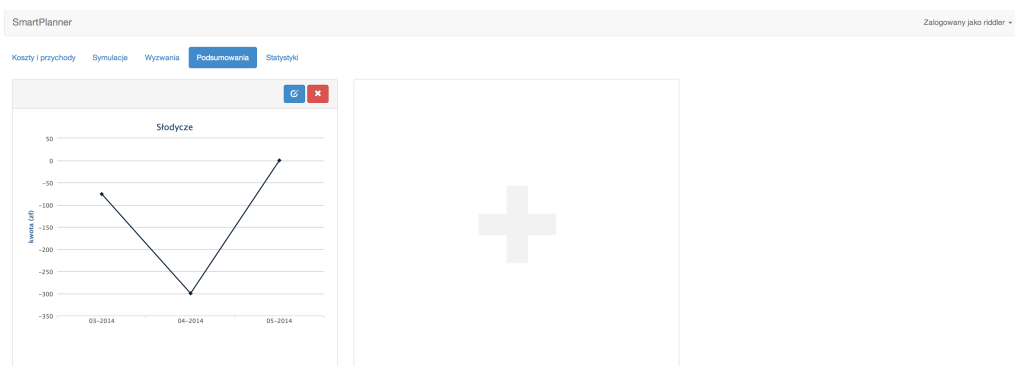


Rysunek 11: Planowanie wydatków

Podsumowania służą do tworzenia raportów. Można przykładowo stworzyć raport wydatków za ostatni miesiąc i przedstawić w formie wykresu. Przykład przedstawiony jest poniżej.

Dodawanie wydatku przedstawione na rysunku nr 13 pozwala utworzyć podsumowanie. Aby przejść do tego widoku trzeba nacisnąć kafelek z ikoną plusa. Na tym widoku kiedy wybierzemy etykiety to aplikacja zawęzi wyszukiwanie wydatków do tych właśnie etykiet.

Edycja z rysunku 14 działa analogicznie jak dodawanie.



Rysunek 12: Lista podsumowań

SmartPlanner Zalogowany jako rddler

Koszty i przychody Symulacje Wyzwania **Podsumowania** Statystyki

Nowe podsumowanie

Nazwa: Skodycze za kawieci ✓

Od: 04/01/2014 ✓

Do: 04/30/2014 ✓

Do: Miejsce

Tagi: jedzenie poza domem ✕ Skodycze ✕ elektronika ✕ wynajem ✕ muzyka ✕ transport ✕ wyjazd ✕

Dodaj

Rysunek 13: Dodawanie podsumowań

SmartPlanner Zalogowany jako rddler

Koszty i przychody Symulacje Wyzwania **Podsumowania** Statystyki

Edytuj podsumowanie

Nazwa: Skodycze ✓

Od: 01/01/2014 ✓

Do: 05/27/2014 ✓

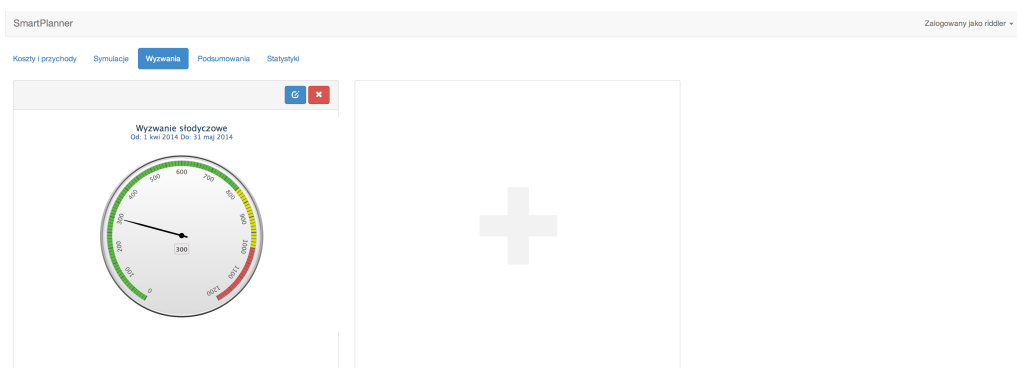
Do: Miejsce

Tagi: jedzenie poza domem ✕ Skodycze ✕ elektronika ✕ wynajem ✕ muzyka ✕ transport ✕ wyjazd ✕

Zapisz

Rysunek 14: Edycja podsumowania

Wyzwania służą do wyznaczania sobie celów na najbliższe miesiące. Pokazują aktualny stan wydanych pieniędzy na wydatki oznaczone wybranymi etykietami. Poniżej zrzuty z listy, edycji, dodawania, powodzenia w wykonaniu i niepowodzeniu wyzwania.



Rysunek 15: Lista wyzwań

Rysunek 16: Dodawanie wyzwania

SmartPlanner Zalogowany jako riddler

Koszty i przychody Symulacja **Wyzwania** Podsumowanie Statystyki

Edycja wyzwania

Nazwa
Wyzwanie słodczowe ✓

Od
04/01/2014 ✓

Do
05/31/2014 ✓

Tagi
jedzenie poza domem ✕ slodkosc ✕ elektronika ✕ wynajem ✕ muzyka ✕ transport ✕ wypalac ✕

Kwota wyzwania
1000 ✓

Zapisz

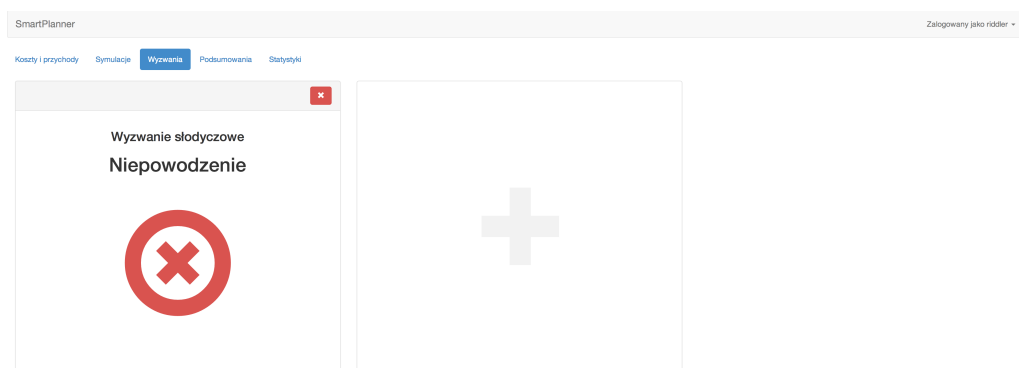
Rysunek 17: Edycja wyzwania

SmartPlanner Zalogowany jako riddler

Koszty i przychody Symulacja **Wyzwania** Podsumowanie Statystyki

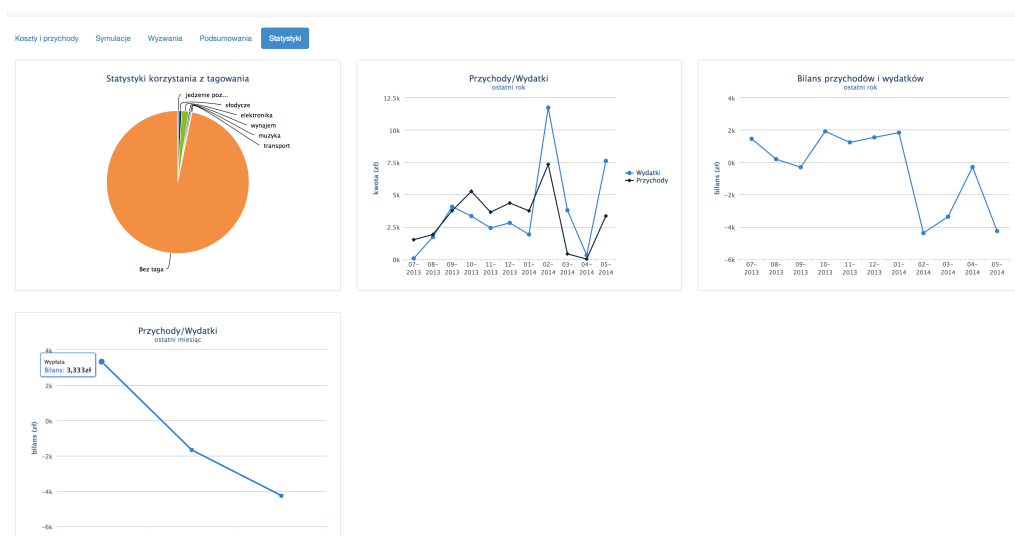
Wyzwanie słodczowe
Sukces

Rysunek 18: Sukces w wyzwaniu



Rysunek 19: Niepowodzenie w wyzwaniu

Na ekranie pokazane są statystyki predefiniowane przez twórcę aplikacji i nie można ich zmienić. Po naciśnięciu jakiegoś elementu na wykresie zostajemy przeniesieni na widok z rysunku nr 9 z wyfiltrowanymi wydatkami.



Rysunek 20: Statystyki

4 Instalacja i wdrożenie

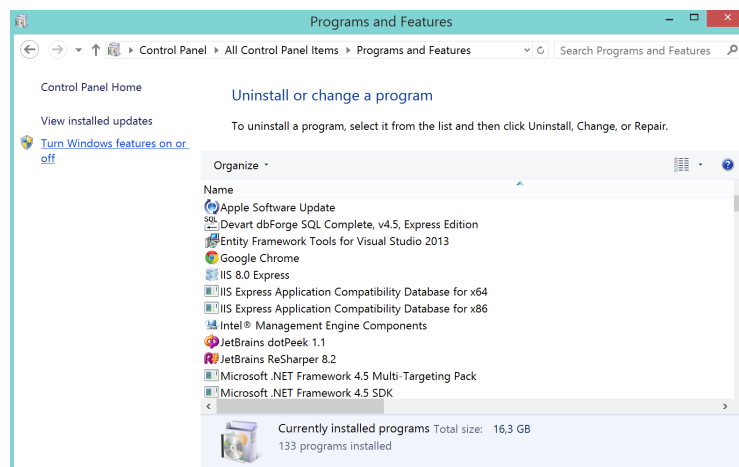
Aby wdrożyć aplikację potrzebujemy następujących składników:

- Microsoft Windows 2012 Server/7/8 lub wyższe
- .NET Framework 4.5
- Microsoft SQL Server 2012 lub wyższe
- Microsoft IIS Server 7

Przedstawiona poniżej instalacja została wykonana na systemie Windows 8.1

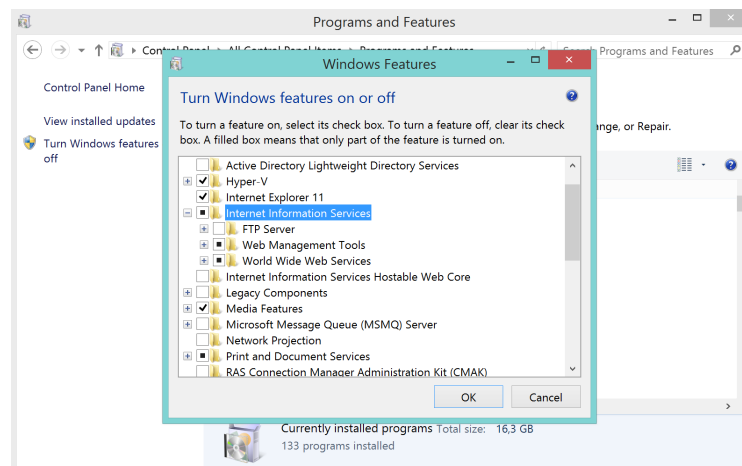
Pierwszym krokiem jest instalacja serwera IIS. Aby to zrobić trzeba przejść do Control Panel, a następnie do Programs and features pokazane na obrazku nr 21.

Tutaj należy kliknąć Turn Windows features on or off.



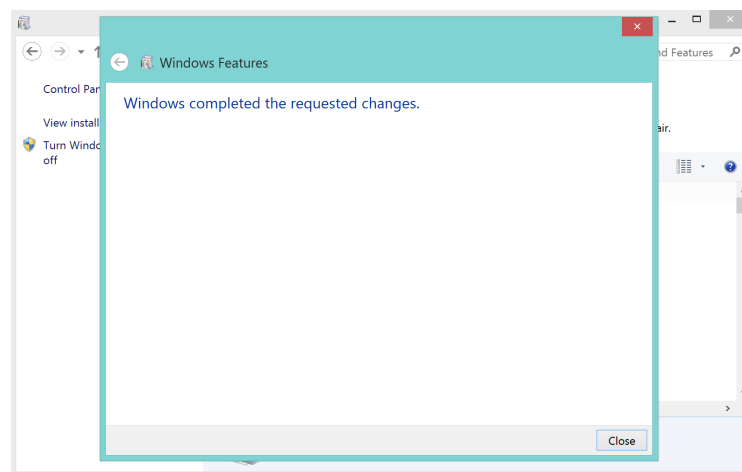
Rysunek 21: Programs and features

Zaznaczyć Internet Information Services, a następnie kliknąć ok, po naciśnięciu którego nastąpi instalacja IIS.



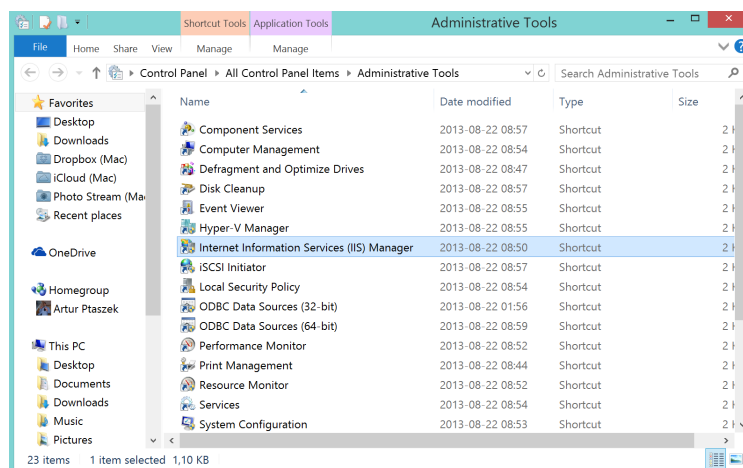
Rysunek 22: Turn Windows features on or off

Ten ekran prezentuje powodzenie instalacji IIS.



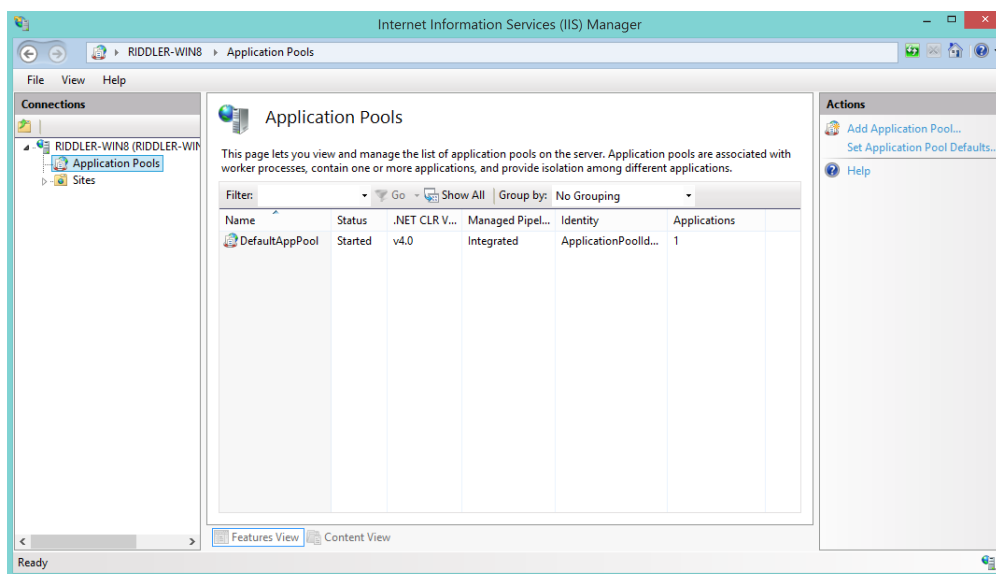
Rysunek 23: Powodzenie instalacji IIS

Kolejnym krokiem jest skonfigurowanie IIS pod aplikację. Aby to wykonać trzeba przejść do **Control Panel > Administrative Tools > Internet Information Services (IIS) Manager**



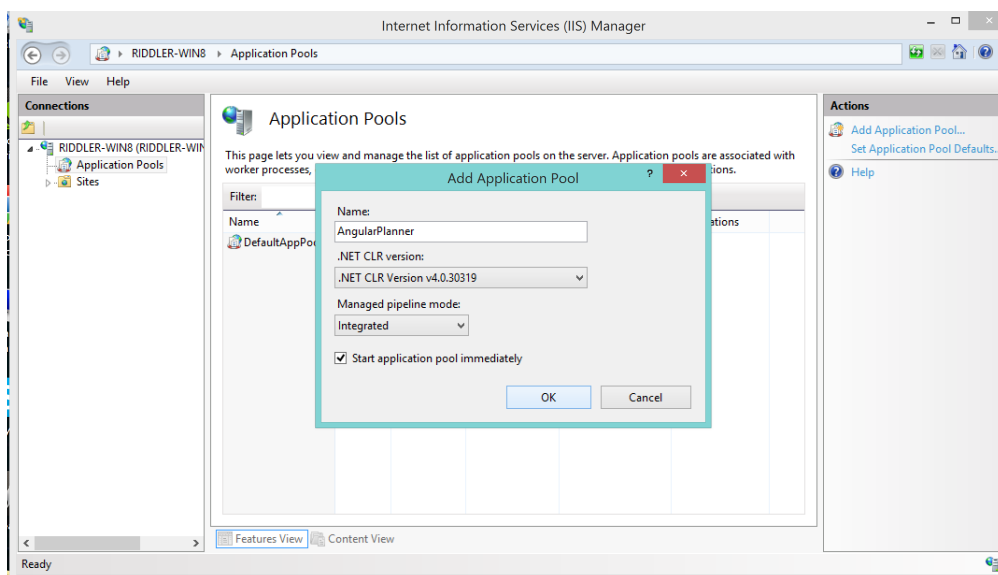
Rysunek 24: Internet Information Services (IIS) Manager

W lewej kolumnie wybieramy Application Pools i w prawej Add Application Pool...



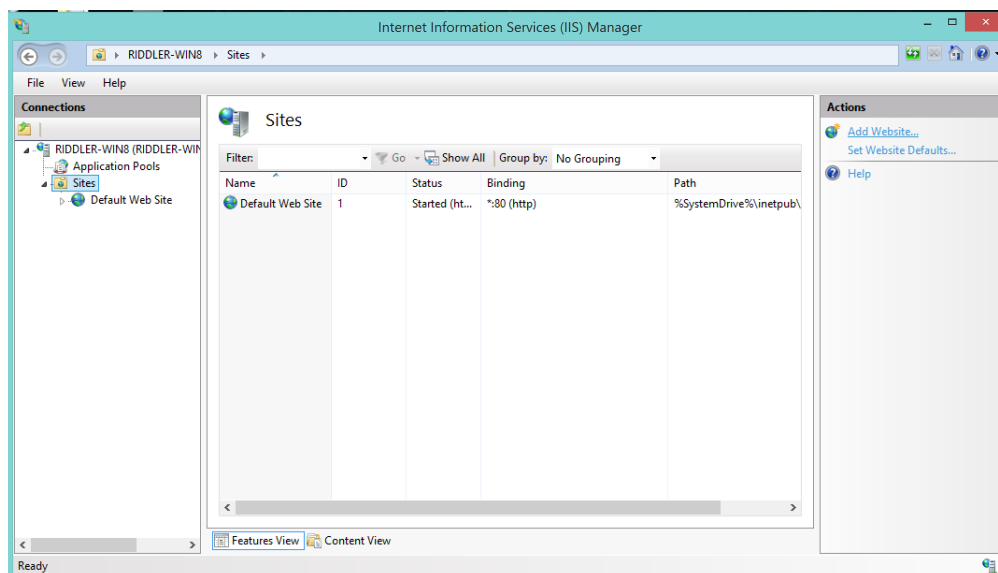
Rysunek 25: Internet Information Services (IIS) Manager

Należy wypełnić dane tak jak w okienku poniżej i kliknąć OK.

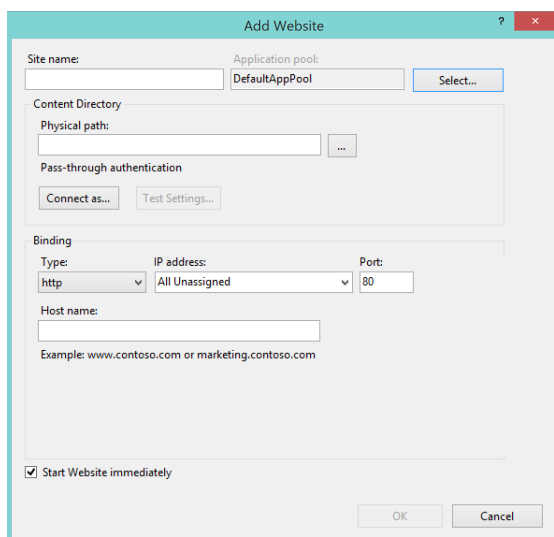


Rysunek 26: Dodawanie nowej puli

Z lewej kolumny należy wybrać Sites i w prawej kliknąć Add Website.

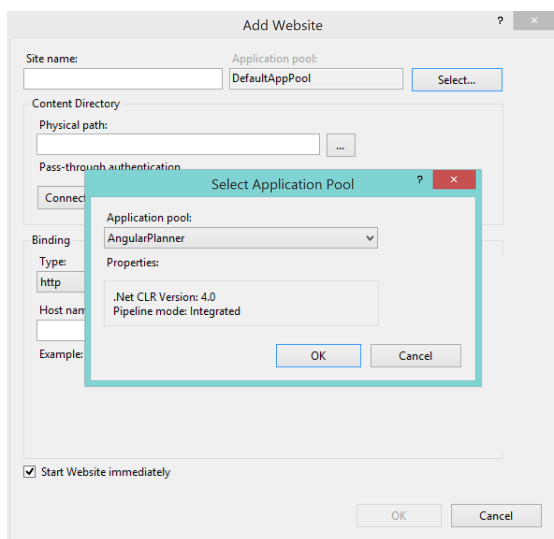


Rysunek 27: Internet Information Services (IIS) Manager



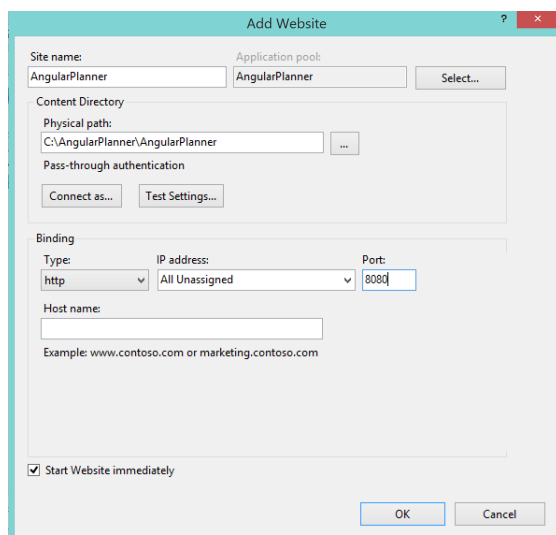
Rysunek 28: Dodawanie strony

Należy kliknąć przycisk `Select...` i wybrać pulę `AngularPlanner`.



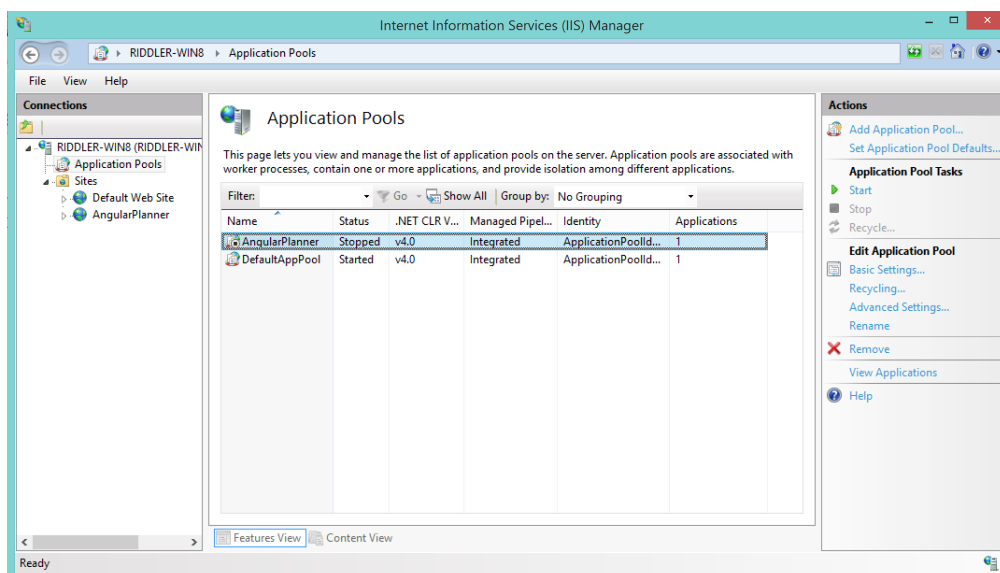
Rysunek 29: Wybór puli

Pozostałe pola wypełnić jak poniżej. Jedynie można zmienić ścieżkę do aplikacji. Zatwierdzić wszystko przyciskiem `OK`.



Rysunek 30: Uzupełnianie danych

Przejsć do Application Pools wybrać pulę AngularPlanner i w prawej kolumnie kliknąć Start.



Rysunek 31: Uruchamianie puli

Podsumowanie

Podczas prac implementacyjnych większość założonych funkcjonalności zostało wykonanych. Nie udało się zrobić zakładki ze statystykami wszystkich użytkowników systemu. Oczywiście byłyby one bardzo ogólne, aby dbać o ochronę danych klientów. Także można było rozwinąć funkcję przewidywania wydatków o precyzyjniejsze i bardziej skomplikowane algorytmy. Jeden z najważniejszych celów został osiągnięty, a było nim uproszczenie wszelkich operacji, aby użytkownik bez instrukcji użytkowania wiedział do czego służą funkcje i jak je obsłużyć.

Najwięcej problemów było z biblioteką AngularJS i stworzeniem mechanizmu logowania w tej technologii. Biblioteka ta oferuje bardzo dużo funkcjonalności w prosty sposób, lecz dopiero po dłuższym użytkowaniu. Dla początkujących użytkowników jest ciężka w rozumieniu. Logowanie także nie jest prostą sprawą, ponieważ mało jest materiałów w Internecie i literaturze o logowaniu tokenowym.

Literatura

- [1] <http://www.asp.net/web-api/overview/security/individual-accounts-in-web-api>.
- [2] [https://docs.angularjs.org/api/ngResource/service/\\$resource](https://docs.angularjs.org/api/ngResource/service/$resource).
- [3] <https://docs.angularjs.org/guide/databinding>.
- [4] [https://docs.angularjs.org/api/ng/service/\\$compile#-restrict-](https://docs.angularjs.org/api/ng/service/$compile#-restrict-).
- [5] [https://docs.angularjs.org/api/ng/service/\\$compile](https://docs.angularjs.org/api/ng/service/$compile).
- [6] [https://docs.angularjs.org/api/ngRoute/service/\\$route#\\$routeChangeError](https://docs.angularjs.org/api/ngRoute/service/$route#$routeChangeError).
- [7] List of http status codes. http://en.wikipedia.org/wiki/List_of_HTTP_status_codes.
- [8] Promises/a+. <http://promises-aplus.github.io/promises-spec/>.
- [9] Google. Best practice recommendations for angular app structure. <https://docs.google.com/document/d/1XXMvRe08-Aw1EZAXS4PzDzdNvV6pGcuaF4Q9821Es/pub>.
- [10] Kris Kowal. Q.js. <https://github.com/kriskowal/q>.
- [11] Pawel Kozlowski Peter Bacon Darwin. Mastering Web Application Development with AngularJS. PacktPub, 2013.
- [12] Aaron Smith. Building minification-safe angular.js applications. <http://thegreenpizza.github.io/2013/05/25/building-minification-safe-angular.js-applications/>, May 2013.