

**WYDZIAŁ PODSTAWOWYCH PROBLEMÓW TECHNIKI
POLITECHNIKI WROCŁAWSKIEJ**

ZARZĄDZANIE WYDATKAMI W GOSPODARSTWIE DOMOWYM

ARTUR PTASZEK

Praca inżynierska napisana
pod kierunkiem
dr Wojciecha Macyny

WROCŁAW 2010

Spis treści

1	Zagadnienia	3
1.1	SPA - Single Page Application	3
1.2	MVC	3
1.3	Logowanie tokenowe	3
2	Technologie	4
2.1	Serwer	4
2.2	Klient	6
3	Baza danych	6
3.1	Diagramy przedstawiające zależności między tabelami	6
3.2	Opis tabel	6
4	Implementacja systemu	9
4.1	Omówienie kodów źródłowych - Serwer	9
4.2	Omówienie kodów źródłowych - Klient	14
5	Instalacja i wdrożenie	19

Wstęp

Praca swoim zakresem obejmuje stworzenie aplikacji do zarządzania wydatkami w gospodarstwie i wyświetlanie wielu statystyk z nimi związanymi. Ma ona na celu ułatwienie w takim zarządzaniu poprzez wyświetlanie licznych wykresów. Użytkownikiem docelowym jest każdy zarabiający/studujący chcący organizować swoje wydatki. Celem pracy zaprojektowanie i oprogramowanie aplikacji o następujących założeniach funkcjonalnych:

- Dostosowanie interfejsu do szerokiego wachlarzu urządzeń (telefony komórkowe, tablety, komputery)
- SPA (Single Page Application) czyli aplikacja działająca bez odświeżenia strony
- Prosty i intuicyjny interfejs
- Logowanie i rejestracja użytkowników
- Wprowadzanie i edycja przychodów i wydatków
- Tagowanie przychodów i wydatków
- Filtrowanie wydatków (po tagach i po czasie)
- Możliwość sprawdzenia czy można sobie pozwolić na jakiś większy wydatek w przyszłych miesiącach (Symulacja)
- Możliwość tworzenia wyzwań na tagi czyt. ustawienie sobie jakiegoś limitu na konkretną jednostkę czasu
- Możliwość tworzenia wykresów z podsumowaniami wydatków dla tagów
- Stworzenie predefiniowanych wykresów
 - Wykres kołowy ukazujący wykorzystanie tagowania w wydatkach
 - 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

lecz aplikacja będzie w formie strony internetowej, przez co będzie dostępna na wszystkie platformy. Także idea Responsive Web Design będzie zwiększała zasięg platformowy. Aplikacja mBanku i ING jest bardzo mało intuicyjna przez co można trafić do klientów tych aplikacji. Praca składa się z n rozdziałów ...

1 Zagadnienia

W pracy zostały wykorzystane liczne technologie, wzorce i zagadnienia, które poniżej zostaną omówione.

1.1 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 jest rosnące wrażenia użytkownika przy korzystaniu z aplikacji (User Experience), a także zminimalizowanie transferu danych między przeglądarką a serwerem, przez co czas odpowiedzi serwera jest zdecydowanie szybszy i użytkownik szybciej zobaczy efekt. Koncepcja wiele czerpie z najnowszych technologii jakimi są HTML5, CSS3, JavaScript, AJAX.

1.2 MVC

Złożony wzorzec architektoniczny służący do organizowania struktury aplikacji posiadających interfejsy graficzne. Wykorzystuje on 3 proste wzorce jakimi są Strategia, Obserwator, Kompozyt. MVC zakłada podzielenie aplikacji na poniżej wymienione części składowe:

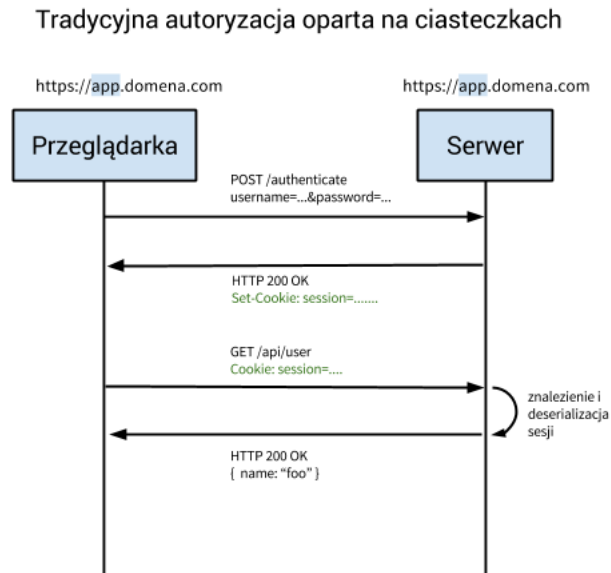
- Model - dane wymagane do utworzenia Widoku
- Widok - opisuje jak przedstawić część modelu użytkownikowi
- Kontroler - agreguje dane z wielu modeli i przygotowuje je aby przekazać do widoku

1.3 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 1: Diagram przepływu dla logowania tradycyjnego

2 Technologie

Aplikacja została podzielona na 2 części logiczne, którymi są:

- Serwer
- Klient

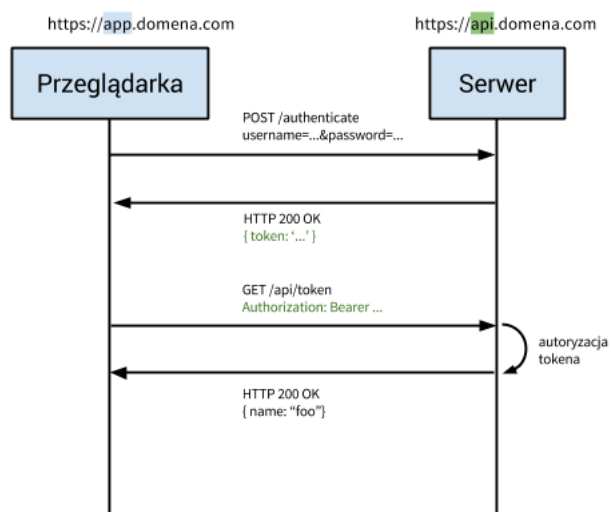
Każda z nich używa innych technologii, ponieważ działają w różnych środowiskach np. klient pracuje jedynie w przeglądarce.

2.1 Serwer

Została napisana 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

Nowoczesna autoryzacja oparta na tokenach



Rysunek 2: Diagram przepływu dla logowania tokenowego

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 te 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.

2.2 Klient

Klient został w większości 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.

Interfejsu 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 generowane przy pomocy HighCharts.js, który sprawia, że wykresy są proste w generowaniu.

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

3 Baza danych

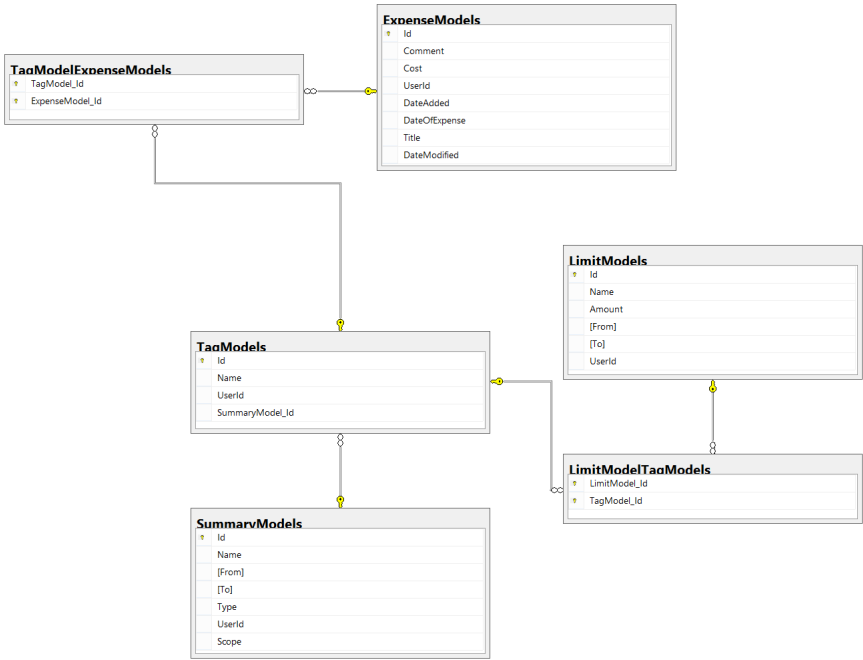
Jak wyżej zostało napisane baza danych to Microsoft SQL Server 2012 Express LocalDB Poniżej projekt bazy danych.

3.1 Diagramy przedstawiające zależności między tabelami

3.2 Opis tabel

ExpenseModels Przechowuje wydatki i przychody

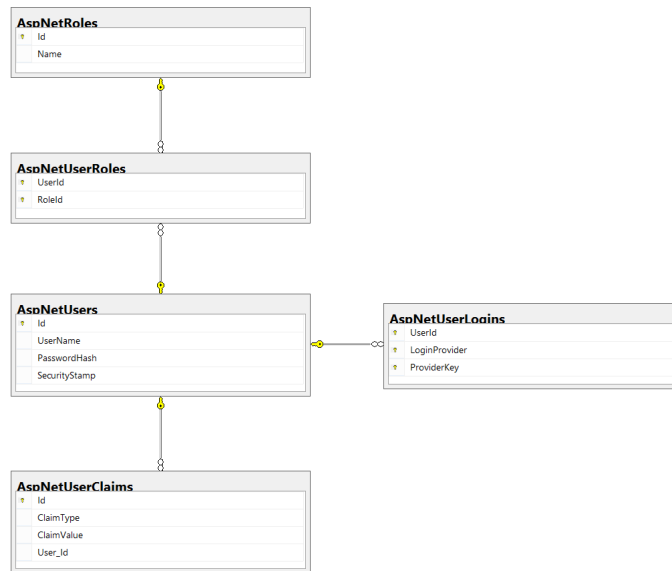
- Id int IDENTITY (Identyfikator)
- Comment nvarchar(max) (Komentarz)
- Cost decimal(18,2) (Wartość wydatku/przychodu)
- UserId nvarchar(max) (Identyfikator użytkownika)
- DateAdded datetime (Data dodania wydatku/przychodu)
- DateOfExpense datetime (Data wydatku/przychodu wprowadzona przez użytkownika)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



Rysunek 3: Tabele aplikacji

MigrationHistory			
Id	MigrationId		
	ContextKey		
	Model		
	ProductVersion		

Rysunek 4: Tabela migracyjna



Rysunek 5: Tabele autoryzacyjne

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 tagami

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

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)

TagModelExpenseModels Tabela łącząca tabele wydatków/przychodów z tagami

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

TagModels Tabela przechowująca tagi

- Id int IDENTITY (Identyfikator)
- Name nvarchar(max) (Nazwa tagu)
- UserId nvarchar(max) (Identyfikator użytkownika)
- SummaryModelId int (Identyfikator podsumowania)

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

4 Implementacja systemu

4.1 Omówienie kodów źródłowych - 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 to napisana przeze mnie, 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 na 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 lecz dla Web Service'u RESTowego.

```

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. Zamiarem było także włączenie opcji logowania przez Facebook i Google+ lecz czas na to nie pozwolił.

Katalog `Controllers` zawiera klasy controllerów są tam przedewszystkim kontrolery Web Api 2, lecz jest tam jeden, który posiada tylko jedną 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 66zniej
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<HttpResponseMessage> 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 ==
                expense.Id && i.UserId == userId);
129
130         var tagExist = expenseDB.Tags.Select(i => i.Id);
131         var tagIds = expense.Tags.Select(i => i.Id);
132
133         expenseDB.Tags.AddRange(await _db.Tags.Where(i => tagIds.Contains(i.Id) && !
                tagExist.Contains(i.Id)).ToListAsync());
134         expenseDB.Tags.RemoveAll(i => !tagIds.Contains(i.Id));
135         await _db.SaveChangesAsync();
136
137         return Request.CreateResponse(HttpStatusCode.Created);
138     }
139
140     return Request.CreateResponse(HttpStatusCode.BadRequest);
141 }
142
143 public async Task<HttpResponseMessage> Delete(int id)
144 {
145     var userId = User.Identity.GetUserId();
146     var expense = await _db.Expenses.Where(i => i.UserId == userId && i.Id == id).
        FirstOrDefaultAsync();
147     if (expense == null)
148     {
149         return Request.CreateResponse(HttpStatusCode.NotFound);
150     }
151     _db.Expenses.Remove(expense);
152     await _db.SaveChangesAsync();
153
154     return Request.CreateResponse(HttpStatusCode.OK);
155 }
156
157 protected override void Dispose(bool disposing)
158 {
159     if (disposing)
160     {
161         _db.Dispose();
162     }
163     base.Dispose(disposing);
164 }
165 }
166 }

```

Listing 4: Kontroller ExpensesController

Klasa kontrolera dziedziczy po klasie abstrakcyjnej ApiController, by mogłabyć rozpoznana przez mechanizm refleksji jako kontroller 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` mówi bibliotece `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`). Linijka 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` robi `LEFT OUTER JOIN` z tabelą `TagsModels`, zaś metoda `OrderByDescending` dodaje sortowanie po `DateOfExpense` (`ORDER BY DateOfExpense DESC`), `Skip` opuszcza liczbę wpisów podanych w parametrze, a `Take` bierze 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, nad którym się znajduje polem `PRIMARY KEY`. Pole `Tags` jest to tak naprawdę odwołanie do tabeli `TagsModels`. `Entity Framework` sam stwierdza jakiego typu relacjami mają być połączone tabele 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.

4.2 Omówienie kodów źródłowych - Klient

Cały kod aplikacji został zorganizowany według rekomendowanej przez `Angulara` struktury katalogów [4].

Przy asynchronicznych akcjach jest wykorzystywana okrojona biblioteka `Q.js` [5] zaimplementowana w `Angularze`. Implementuje ona ustandaryzowany system obietnic `Promises/A+` [3]

```
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     jshintsrc: '.jshintrc',
81     reporter: require('jshint-stylish')
82   },
83   all: [
84     'Gruntfile.js',
85     '<%=_yeoman.appSrc_%>/**/*.js'
86   ],
87   test: {
88     options: {
89       jshintsrc: 'test/.jshintrc'
90     },
91     src: ['<%=_yeoman.testSpec_%>/**/*.js']
92   }
93 },
94 karma: {
95   unit: {
96     configFile: 'karma.conf.js',
97     singleRun: true
98   }
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 [8] 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 [6]. Jest on wydzielony 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 [2]

```

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

Kolejnym jest przedstawiony na Listingu 9 moduł 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 jedynie on w trakcie ładowania widoków, czy mogą być pokazane użytkownikowi.

```

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
19         return service;
20       }
21     });

```

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

Do wymiany danych używany jest mechanizm \$resource [1]. Przykład implementacji przedstawiony został na Listingu 11. Udostępnia on łatwy interfejs do komunikacji z serwerem.

```

1  'use_strict';
2  /**
3   * resources.summaries Module
4   *
5   * ExpenseModel
6   */
7  angular.module('resources.summaries', [])
8    .factory('Summaries', function($resource) {
9      return $resource('/api/summaries/:id',
10        {id: '@id'},
11        {
12          'update': { method: 'PUT' }
13        });
14    });

```

Listing 11: Summary Resource

5 Instalacja i wdrożenie

Aby wdrożyć aplikację musimy potrzebujemy następujących składników:

- Systemu Windows

Literatura

- [1] [https://docs.angularjs.org/api/ngResource/service/\\$resource](https://docs.angularjs.org/api/ngResource/service/$resource).
- [2] List of http status codes. http://en.wikipedia.org/wiki/List_of_HTTP_status_codes.
- [3] Promises/a+. <http://promises-aplus.github.io/promises-spec/>.
- [4] Google. Best practice recommendations for angular app structure. <https://docs.google.com/document/d/1XXMvRe08-Awi1EZXAXS4PzDzdNvV6pGcuaF4Q9821Es/pub>.
- [5] Kris Kowal. Q.js. <https://github.com/krisowal/q>.
- [6] Pawel Kozlowski Peter Bacon Darwin. Mastering Web Application Development with AngularJS. PackPub, 2013.
- [7] Alberto Pose. Cookies vs tokens. getting auth right with angular.js. <https://auth0.com/blog/2014/01/07/angularjs-authentication-with-cookies-vs-token/>.
- [8] Aaron Smith. Building minification-safe angular.js applications. <http://thegreenpizza.github.io/2013/05/25/building-minification-safe-angular.js-applications/>, May 2013.