

RESTful API в вашем .NET приложении: Как, зачем и почему?

Юлия Цисык, CUSTIS

MSK DOT NET MeetUp #2

Зачем...

- Нужно думать об API?

Почему...

- Именно REST и чем он хорош?

Как...

- Делать АПИ хорошо?
- Версионировать?
- “Открыть” данные с помощью OData?
- Проектировать доменную модель?

Что такое API?

Application Program Interface

- Набор правил и механизмов

Почему хороший API это важно?

- Легко использовать и поддерживать
- API - это UI для разработчиков
- Увеличивает популярность сервиса

Какие виды API бывают?

- Web service APIs
 - XML-RPC and JSON-RPC
 - SOAP
 - REST
- WebSockets APIs
- Library-based APIs
 - Java Script
- Class-based APIs
 - C# API, Java
- OS function and routines
 - Access to file system
 - Access to user interface
- Object remoting APIs
 - CORBA
 - .Net remoting
- Hardware APIs
 - Video acceleration (OpenCL...)
 - Hard disk drives
 - PCI bus
 - ...

Какие виды API нас интересуют?

Web service APIs

XML-RPC and JSON-RPC

SOAP – Simple ☺ Object Access Protocol

REST

Какие виды API нас интересуют?

Web service APIs

~~XML RPC and JSON RPC~~

~~SOAP — Simple ☺ Object Access Protocol~~

REST

Что такое REST?

Representative **S**tate **T**ransfer

НЕ протокол. НЕ стандарт. Это архитектурный стиль

Принципы REST?

- Клиент-серверная архитектура
- Любые данные являются ресурсом
- Любой ресурс имеет ID
- Ресурсы связаны между собой
- Используются стандартные методы HTTP
- Сервер не хранит состояние

Чем REST хорош?

- Он простой!
- Переиспользует существующие стандарты
- REST базируется на HTTP => доступны все плюшки
 - Кэширование
 - Масштабирование
 - Минимум накладных расходов
 - Стандартные коды ошибок
- Очень хорошая распространённость (даже IoT)

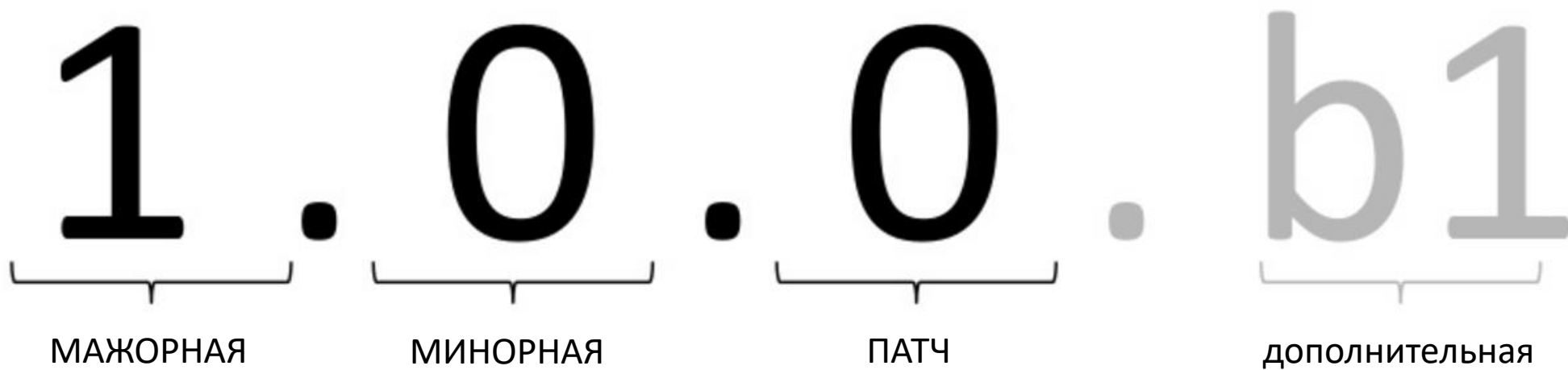
Best-practices (независимые от технологий)

- SSL везде
- Методы POST, PUT должны возвращать данные
- Фильтрация, сортировка и постраничный вывод
- Поддержка MediaType
- Pretty print & gzip
- Стандартное кэширование ETag & Last-Modified
- Стандартные коды ошибок
- Документация и версионирование

Версионирование

- Если вы однажды опубликовали контракт, то вы обязаны его соблюдать
- Breaking changes можно делать только при изменении мажорной версии

Semantic Versioning



Подходы к версионированию

Type	Sample	Complexity
URL	<code>{host}/api/v2/...</code>	Minimum
Custom Header	<code>api-version:2</code>	Average
Custom Accept Header	<code>Accept:application/vnd.trainmodel.v2+json</code>	Maximum

Библиотека Climax.Web.Http

```
[VersionedRoute("v2/values", Version = 2)]
```

```
config.ConfigureVersioning(  
    versioningHeaderName: "version", versioningMediaTypes: null);
```

```
config.ConfigureVersioning(  
    versioningHeaderName: null,  
    versioningMediaTypes: new [] { "application/vnd.model" });
```

Что такое RESTful API?

Это такой сервис, который
удовлетворяет принципам REST



Выбираем технологию

WCF Services

- webHttpBinding only
- Поддерживаются только HTTP GET & POST
- + Разные форматы XML, JSON, ATOM

Web Api

- + Очень простой
- + Open source
- + Все возможности HTTP
- + Все возможности MVC
- + Легкий
- + Также поддерживает кучу форматов

Выбираем хостинг для WebApi

- ASP.NET MVC
- OWIN
 - IIS
 - Self-hosted
- Azure

OWIN

Open Web Interface for .Net

Идея OWIN

- Это спецификация (не библиотека и не платформа)
- Устраняет сильную связанность веб-приложения с реализацией сервера



Katana - реализация OWIN от Microsoft

```
[assembly: OwinStartup(typeof (Startup))]  
namespace RestApiDemo  
{  
    public class Startup  
    {  
        public void Configuration(IAppBuilder app)  
        {  
            var config = new HttpConfiguration();  
            config.MapHttpAttributeRoutes();  
            app.UseWebApi(config);  
        }  
    }  
}
```

Проектируем интерфейс

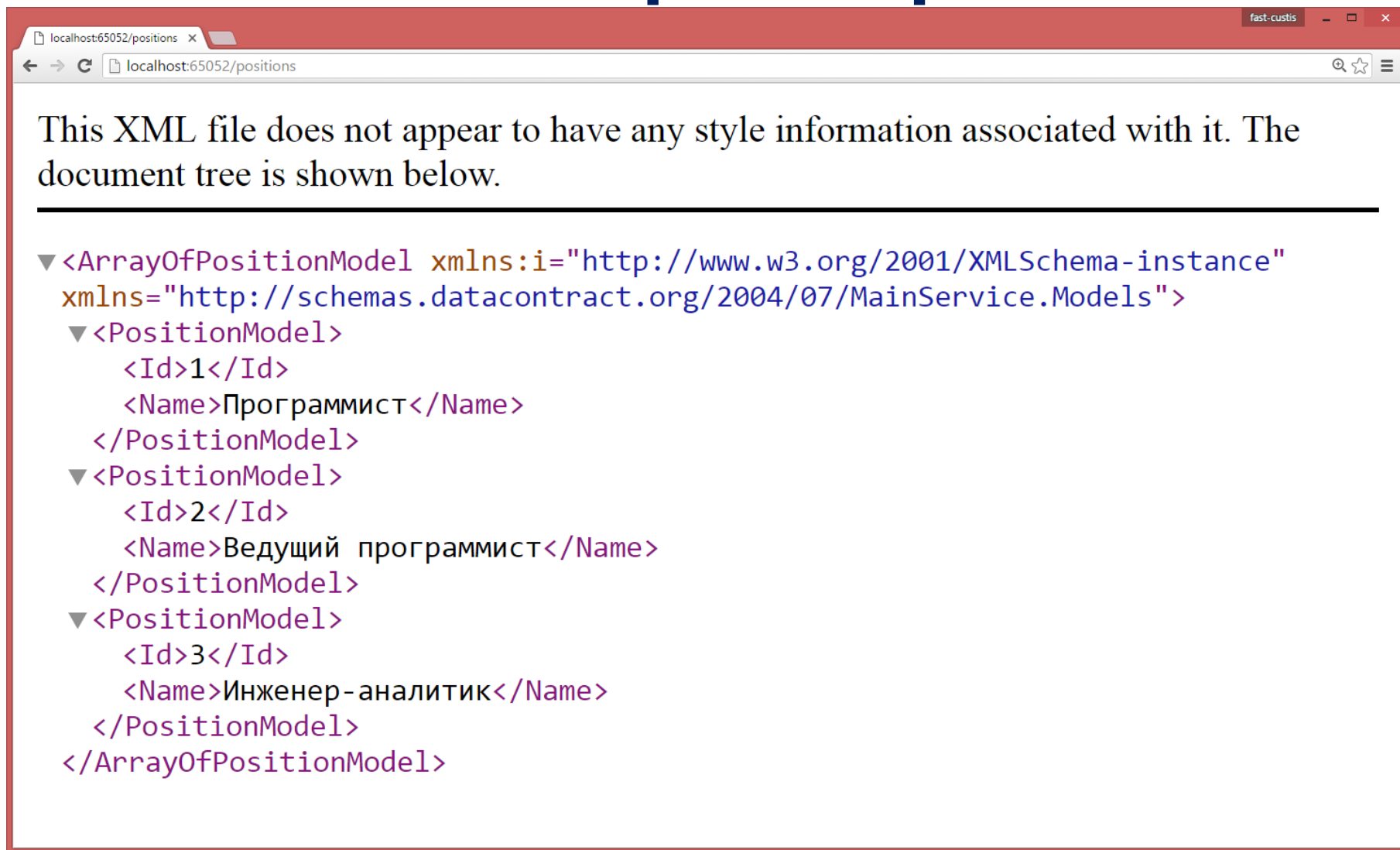
- Все ресурсы в REST существительные (множественное число)
- Корневые сущности API
 - GET /positions - Все вакансии
 - GET /positions/1 - Информация по вакансии с ID = 1
 - GET /competitors - Все соискатели
- Зависимые сущности
 - GET /positions/7/interviews – собеседования, назначенные на вакансию с id = 7

Простейший контроллер

```
[RoutePrefix("positions")]
public class PositionsController : ApiController
{
    [HttpGet]
    [Route]
    public IEnumerable<PositionModel> GetAll()
    {
        return testData;
    }

    PositionModel[] testData = /*initialization here*/
}
```

Простейший контроллер



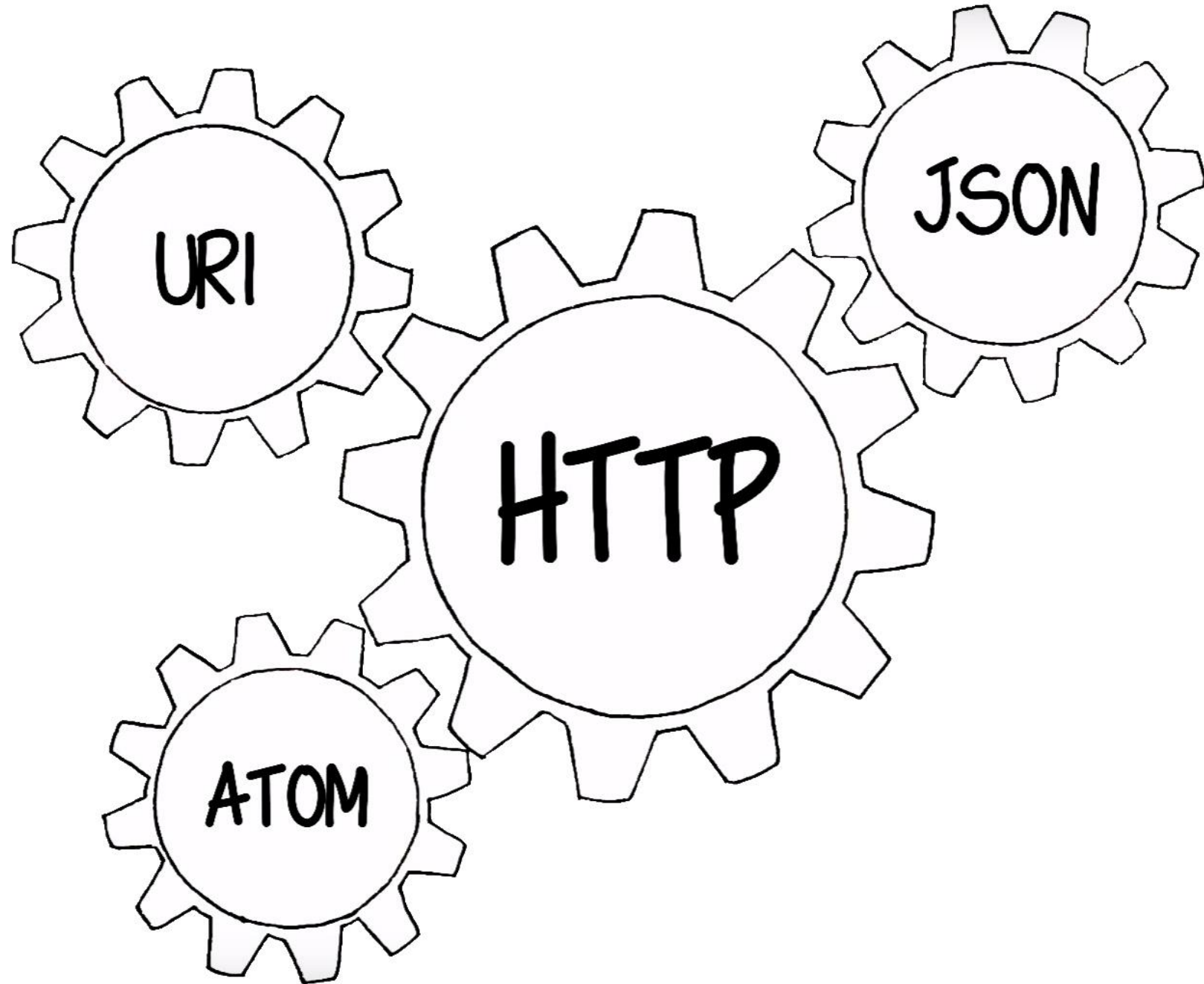
Зависимый контроллер

```
[RoutePrefix("positions/{positionId}/interviews")]
public class InterviewsForPositionController : InterviewsController
{
    [HttpGet]
    [Route]
    [EnableQuery]
    public IQueryable<InterviewModel> GetAll(int vacancyId)
    {
        return GetAllInterviews().Where(x => x.VacancyId == vacancyId);
    }
}
```

Базовый CRUD

- **POST – создать новую сущность**
 - POST /positions – JSON описание сущности целиком. Действие добавляет новую сущность в коллекцию
 - Возвращает созданную сущность
- **PUT – изменить сущность**
 - PUT /positions/12 – Изменить сущность с ID = 12.
 - Возвращает измененную сущность
- **DELETE**
 - DELETE /positions/12 – Удалить сущность с ID = 12.

OData



OData

```
[RoutePrefix("positions")]
public class PositionsController : ODataController
{
    [HttpGet]
    [Route]
    [EnableQuery]
    public IQueryable<PositionModel> Get()
    {
        return testData.AsQueryable();
    }

    PositionModel[] testData = /*initialization here*/
}
```

OData



A screenshot of a web browser window with a red title bar. The browser has three tabs: 'The entity 'PositionModel'', 'localhost:65052/positions', and 'localhost:65052'. The address bar shows 'localhost:65052'. The main content area displays a JSON response representing OData metadata. The JSON structure is as follows:

```
{
  "@odata.context": "http://localhost:65052/$metadata", "value": [
    {
      "name": "positions", "kind": "EntitySet", "url": "positions"
    }
  ]
}
```

`http://.../api/Orders?$filter=Total gt 1000&$orderby=ShippedAt desc`

`http://.../api/Orders(451)/Lines(4)/Product/Name`

`http://.../api/Orders?$top=10&$skip=10`

`http://.../api/Orders?$expand=Lines($select=Product,Quantity)&$format=json`



Параметры запросов

Query Option	Sample
\$filter	Positions?\$filter=Name eq 'Программист' Positions?\$filter=contains(Name, 'инженер')
\$select	Positions?\$select=Name, Id
\$orderby	Positions?\$orderby=Name desc
\$top	Competitors?\$top=10
\$skip	Interviews?\$skip=15&\$top=15

EnableQuery Атрибут

- AllowedArithmeticOperators
- AllowedFunctions
- AllowedLogicalOperators
- AllowedOrderByProperties
- AllowedQueryOptions
- EnableConstantParameterization
- EnsureStableOrdering
- HandleNullPropagation
- MaxAnyAllExpressionDepth
- MaxExpansionDepth
- MaxNodeCount
- MaxOrderByNodeCount
- MaxSkip
- MaxTop
- PageSize

Операторы OData

Оператор	Описание	Пример
Логические операторы		
eq	Равно	/Customers?filter = City eq 'Moscow'
ne	Не равно	/Customers?filter = City ne 'Moscow'
gt	Больше	/Product?\$filter = Price gt 20
gteq	Больше или равно	/Orders?\$filter = Freight gteq 800
lt	Меньше	/Orders?\$filter = Freight lt 1
lteq	Меньше или равно	/Product?\$filter = Price lteq 20
and	И	/Product?\$filter = Price lteq 20 and Price gt 10
or	Или	/Product?\$filter = Price lteq 20 or Price gt 10
not	Не	/Orders?\$filter = not endswith(ShipPostalCode, '100')
Арифметические операторы		
add	Сложение	/Product?filter = Price add 5 gt 10
sub	Вычитание	/Product?filter = Price sub 5 gt 10
mul	Умножение	/Orders?\$filter = Freight mul 800 gt 2000
div	Деление	/Orders?\$filter = Freight div 10 eq 4
mod	Деление по модулю	/Orders?\$filter = Freight mod 10 eq 0
Операторы группировки		
()	Группировка по приоритетам	/Product?filter = (Price sub 5) gt 10

Функции OData

Строковые функции

<code>bool contains(string p0, string p1)</code>
<code>bool endswith(string p0, string p1)</code>
<code>bool startswith(string p0, string p1)</code>
<code>int length(string p0)</code>
<code>int indexof(string p0)</code>
<code>string insert(string p0, int position, string p1)</code>
<code>string remove(string p0, int position)</code>
<code>string remove(string p0, int position, int lenght)</code>
<code>string replace(string p0, string find, string replace)</code>
<code>string substring(string p0, int position)</code>
<code>string substring(string p0, int position, int length)</code>
<code>string tolower(string p0)</code>
<code>string toupper(string p0)</code>
<code>string trim(string p0)</code>
<code>string concat(string p0, string p1)</code>

Функции типов

<code>bool isof(type p0)</code>
<code>bool isof(expression p0, type p1)</code>
<code><p0> cast(type p0)</code>
<code><p1> cast(expression p0, type p1)</code>

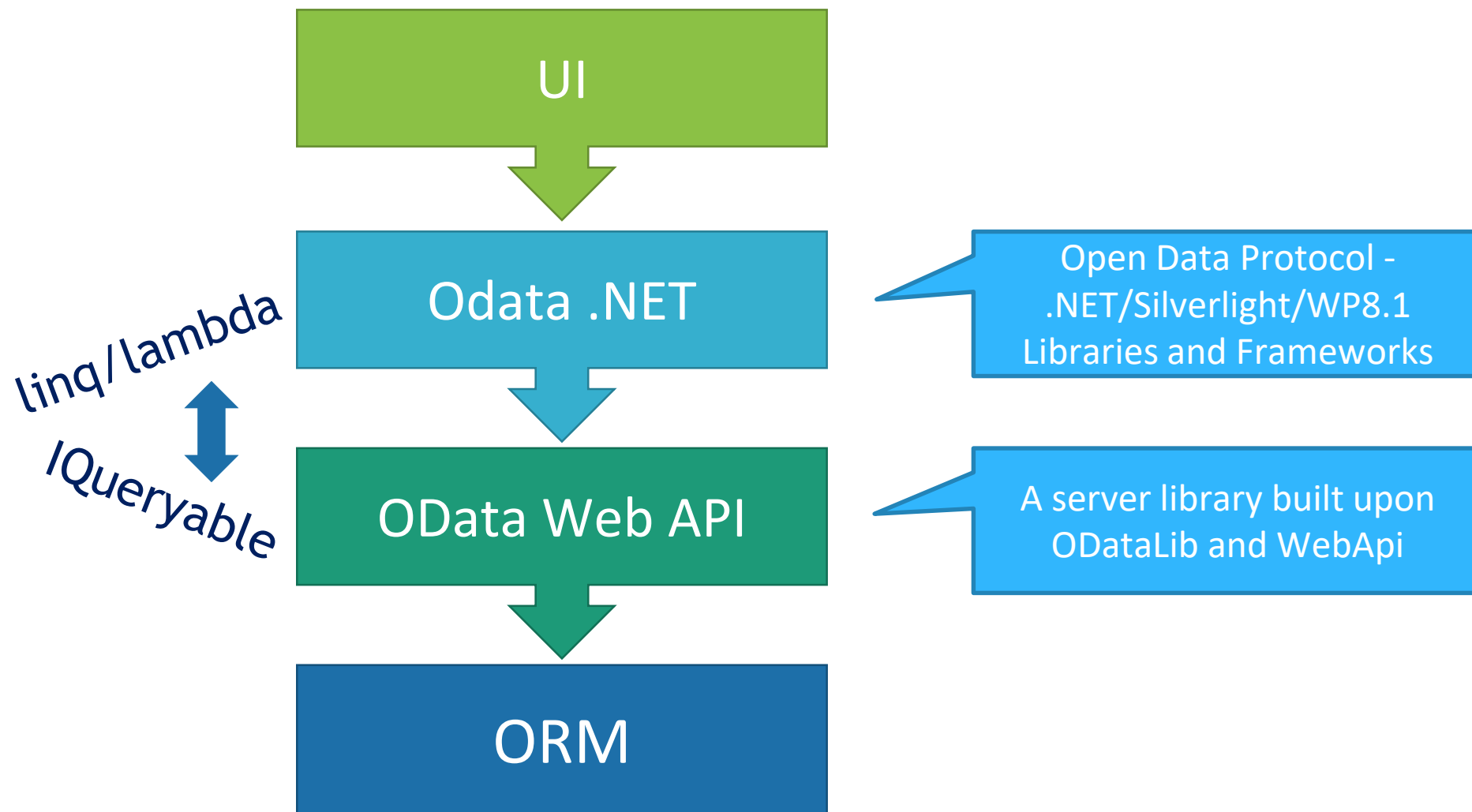
Функции работы с датой

<code>int day(DateTime p0)</code>
<code>int hour(DateTime p0)</code>
<code>int minute(DateTime p0)</code>
<code>int month(DateTime p0)</code>
<code>int second(DateTime p0)</code>
<code>int year(DateTime p0)</code>

Математические функции

<code>double round(double p0)</code>
<code>decimal round(decimal p0)</code>
<code>double floor(double p0)</code>
<code>decimal floor(decimal p0)</code>
<code>double ceiling(double p0)</code>
<code>decimal ceiling(decimal p0)</code>

Клиент



Клиент

Справочник планов вы...

Планы вывоза коллекц...

Обновить

Импорт из RT

Изменить

Изменить все

История

Сформировать график

Удалить график

Перейти

Экспорт в Excel

Фильтры

Торговая сеть:

Большая торговая сеть

План вывоза:

Вывоз SS15

Источник:

Код

Название

Склад-получатель:

Код

Название

Регион:

Наименование

Коды товаров:

Цветомодель:

Код

Категория товара:

Код

Подкатегория товара:

Код

Товарная группа:

Код

Дата начала вывоза:

Дата конца вывоза:

Логистическое назначение:

Показать только красные строки:

Статус:

Новая

Включена в график

Удалена

Источник		Склад-получатель		Товар		Дата начала вывоза	Объем	Дата конца вывоза	Количество		План вывоза	Приоритет
Код	Название	Код	Название	Код	Название	Дата начала вывоза	Объем	Дата конца вывоза	Количество вывоза	Количество	План вывоза	Приоритет
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	0380103	038000013 полуботинки черный	08.12.2015	4,8000	12.12.2015	50	0	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	0819031-4	08190312 Перчатки	06.12.2015	0,2200	15.12.2015	110	3	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	1368010115	1368-010 11,5 Ботинки черный	02.12.2015	0,4030	03.12.2015	31	1	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	215225010	2152-250 10 Полуботинки светло-коричневый	05.12.2015	5,48	05.12.2015	548	1	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	21702537	BM2170-253 7 Полуботинки мужские Howell Men's медный p.7	04.12.2015	1,2960	07.12.2015	144	2	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	242170108	24217- 8 мужские кроссовки белый/красный	25.11.2015	3,2880	12.12.2015	411	0	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	262700107	26270-7 кроссовки женские белый/черный	01.12.2015	1,5030	11.12.2015	167	0	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	2628201075	26282- 7,5 кроссовки белый/т.синий p.7,5	20.11.2015	3,2550	26.11.2015	465	0	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	303480-06	303480-06 туфли	04.12.2015	3,3840	04.12.2015	564	3	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	3126692-1	312669-142 Обувь спортивная (кроссовки)	09.11.2015	6,51	23.11.2015	651	1	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	4186P11-6	4186-P110 6 Футболка с коротким рукавом нежнорозовый	12.11.2015	0	10.12.2015	82	2	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	4207101105	42071105 кроссовки черный	25.11.2015	9,0720	03.12.2015	648	3	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	460710107	460717 полуботинки черный	11.11.2015	3,6890	12.11.2015	527	3	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	4644601075	4644675 сапоги женские песочный	29.11.2015	6,7060	14.12.2015	479	3	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	6969262	69692666 Шорты	07.12.2015	1,7790	10.12.2015	593	2	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	893374-7	893374 Обувь спортивная (кроссовки)	20.11.2015	5,3440	22.11.2015	668	1	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	6066952	6066952 Футболка темно-синий	12.11.2015	0,3990	02.12.2015	399	3	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	80496	80496 Брюки мужские темно-серый p.S	09.12.2015	0,7740	14.12.2015	258	3	Вывоз SS15	100
> 11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	9139848	9139848 Куртка графит/оранжевый/серый p.48	14.11.2015	4,0920	24.11.2015	341	0	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	646640	646640 обувь для футбола Goal синий/серебристый	14.11.2015	4,2840	12.12.2015	476	3	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	701645	701-64 5 Перчатки детские темно-синий	06.11.2015	0,3180	07.12.2015	159	3	Вывоз SS15	100
11	Дискаунтер ТЦ Щелково	1	Склад Московского региона	01051S	01051Куртка голубой							

Метаданные



The screenshot shows a web browser window with the address bar at `localhost:65052/$metadata`. The page content displays an XML document tree for an OData metadata file. The XML structure is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<edmx:Edmx xmlns:edmx="http://docs.oasis-open.org/odata/ns/edmx" Version="4.0">
  <edmx:DataServices>
    <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="MainService.Models">
      <EntityType Name="PositionModel">
        <Key>
          <PropertyRef Name="Id"/>
        </Key>
        <Property Name="Id" Type="Edm.Int64" Nullable="false"/>
        <Property Name="Name" Type="Edm.String" Nullable="false"/>
      </EntityType>
    </Schema>
    <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="Demos">
      <EntityContainer Name="DefaultContainer">
        <EntitySet Name="positions" EntityType="MainService.Models.PositionModel"/>
      </EntityContainer>
    </Schema>
  </edmx:DataServices>
</edmx:Edmx>
```

T4 шаблон

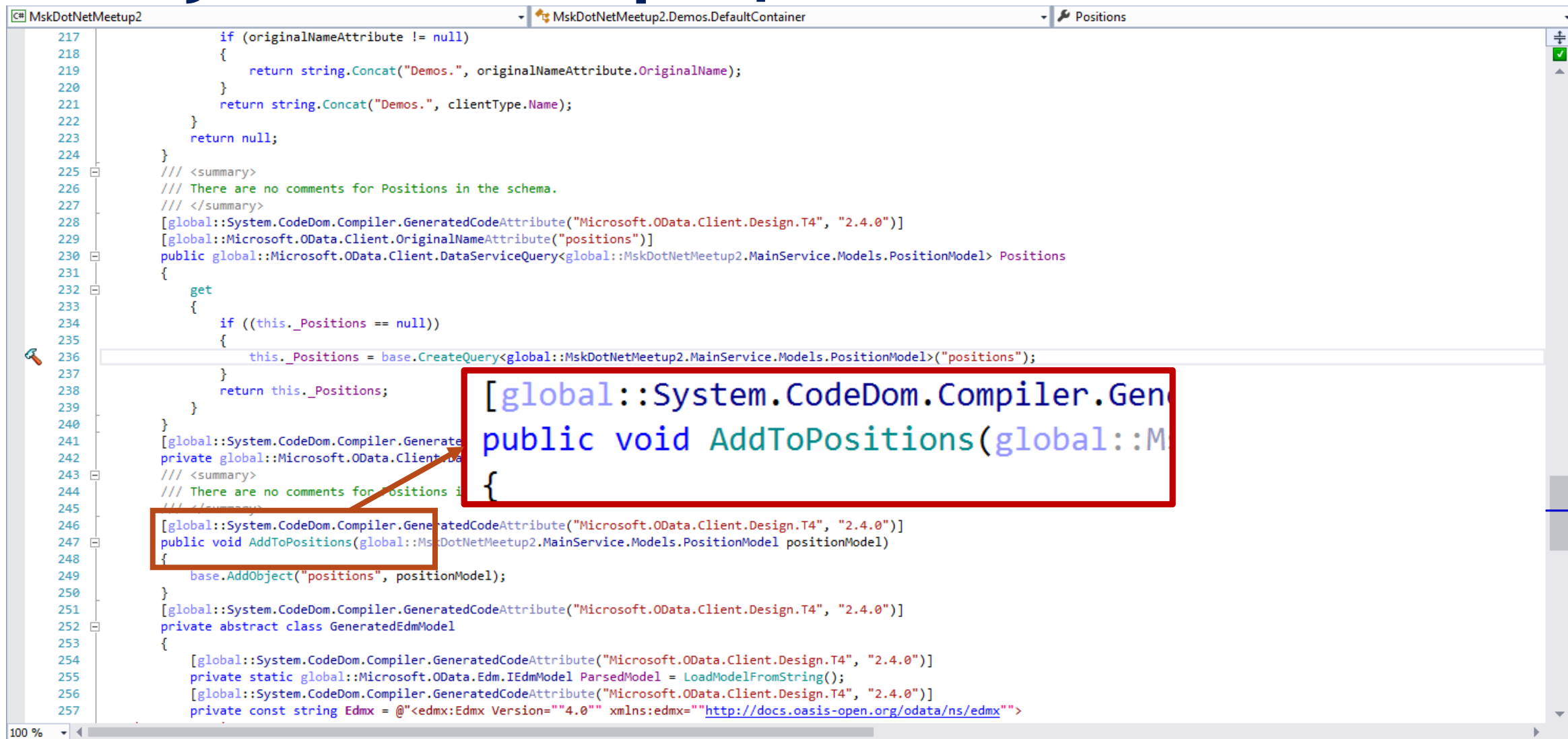
```
ServiceReference.cs  ServiceReference.tt  WebApiConfig.cs  PositionModel.cs  PositionsController.cs  MainWindow.xaml  MainWindow.xaml.cs
1  <#@ include file="ServiceReference.ttinclude" #>
2  <#+
3  public static class Configuration
4  {
5      // The URI of the metadata document. The
6      // eg : "http://services.odata.org/V4/OData/OData.svc/", "File:///C:/Odata.edmx", or @"C:\Odata.edmx"
7      // ### Notice ### If the OData service requires authentication for accessing the metadata document, the value of
8      // MetadataDocumentUri has to be set to a local file path, or the client code generation process will fail.
9      public const string MetadataDocumentUri = "http://localhost:65052/$metadata";
10
11     // The use of DataServiceCollection enables entity and property tracking. The value must be set to true or false.
12     public const bool UseDataServiceCollection = true;
13
14     // The namespace of the client code generated. It replaces the original namespace in the metadata document,
15     // unless the model has several namespaces.
16     public const string NamespacePrefix = "MskDotNetMeetup2";
17
18     // The target language of the generated client code. The value must be set to "CSharp" or "VB".
19     public const string TargetLanguage = "CSharp";
20
21     // This flag indicates whether to ignore unexpected elements and attributes in the metadata document and generate
22     public const bool EnableIgnoreUnexpectedElementsAndAttributes = true;
23
24     // This flag indicates whether to ignore unexpected elements and attributes in the metadata document and generate
25     // the client code if any. The value must be set to true or false.
26     public const bool IgnoreUnexpectedElementsAndAttributes = true;
27 }
```

`"http://localhost:65052/$metadata";`

`"http://localhost:65052/$metadata";`

`NamespacePrefix = "MskDotNetMeetup2";`

Результат генерации



```
217         if (originalNameAttribute != null)
218         {
219             return string.Concat("Demos.", originalNameAttribute.OriginalName);
220         }
221         return string.Concat("Demos.", clientType.Name);
222     }
223     return null;
224 }
225 /// <summary>
226 /// There are no comments for Positions in the schema.
227 /// </summary>
228 [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.OData.Client.Design.T4", "2.4.0")]
229 [global::Microsoft.OData.Client.OriginalNameAttribute("positions")]
230 public global::Microsoft.OData.Client.DataServiceQuery<global::MskDotNetMeetup2.MainService.Models.PositionModel> Positions
231 {
232     get
233     {
234         if ((this._Positions == null))
235         {
236             this._Positions = base.CreateQuery<global::MskDotNetMeetup2.MainService.Models.PositionModel>("positions");
237         }
238         return this._Positions;
239     }
240 }
241 [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.OData.Client.Design.T4", "2.4.0")]
242 private global::Microsoft.OData.Client.DataServiceQuery<global::MskDotNetMeetup2.MainService.Models.PositionModel> _Positions;
243 /// <summary>
244 /// There are no comments for Positions in the schema.
245 /// </summary>
246 [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.OData.Client.Design.T4", "2.4.0")]
247 public void AddToPositions(global::MskDotNetMeetup2.MainService.Models.PositionModel positionModel)
248 {
249     base.AddObject("positions", positionModel);
250 }
251 [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.OData.Client.Design.T4", "2.4.0")]
252 private abstract class GeneratedEdmModel
253 {
254     [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.OData.Client.Design.T4", "2.4.0")]
255     private static global::Microsoft.OData.Edm.IEdmModel ParsedModel = LoadModelFromString();
256     [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.OData.Client.Design.T4", "2.4.0")]
257     private const string Edmx = @"<edmx:Edmx Version=""4.0"" xmlns:edmx=""http://docs.oasis-open.org/odata/ns/edmx"">
```

Результат генерации

```
34     /// </summary>
35     public PositionModelSingle(global::Microsoft.OData.Client.IEdmEntitySet entitySet, global::Microsoft.OData.Client.IEdmEntityKey key)
36         : base(query) {}
37
38 }
39
40 /// <summary>
41 /// There are no comments for PositionModel in the schema.
42 /// </summary>
43 /// <KeyProperties>
44 /// Id
45 /// </KeyProperties>
46 [global::Microsoft.OData.Client.Key("Id")]
47 [global::Microsoft.OData.Client.OriginalNameAttribute("PositionModel")]
48 public partial class PositionModel : global::Microsoft.OData.Client.BaseEntityType, global::System.ComponentModel.INotifyPropertyChanged
49 {
50     /// <summary>
51     /// Create a new PositionModel object.
52     /// </summary>
53     /// <param name="ID">Initial value of Id.</param>
54     /// <param name="name">Initial value of Name.</param>
55     [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.OData.Client.Design.T4", "2.4.0")]
56     public static PositionModel CreatePositionModel(long ID, string name)
57     {
58         PositionModel positionModel = new PositionModel();
59         positionModel.Id = ID;
60         positionModel.Name = name;
61         return positionModel;
62     }
63     /// <summary>
64     /// There are no comments for Property Id in the schema.
65     /// </summary>
66     [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.OData.Client.Design.T4", "2.4.0")]
67     [global::Microsoft.OData.Client.OriginalNameAttribute("Id")]
68     public long Id
69     {
70         get
71         {
72             return this._Id;
73         }
74         set
75         {
76             this._Id = value;
77             this.OnPropertyChanged("Id");
78         }
79     }
80 }
```


Клиент

```
DataContext.MdmDeliveryTypes.Where(x => x.IsShippingInRm).OrderBy(x => x.Name)
```

```
DataContext.TradingNetworkSettings.Where(o => o.Id == listDto.Key))  
    .Expand(o => o.TradingNetwork)  
    .Expand(o => o.Creator)  
    .Expand(o => o.Modifier);
```

```
DataContext.PlannedScheduleItems  
    .Where(x => (x.ReturnPlanItem.ReturnPlan.Id == filterData.ReturnPlanId) &&  
        (filterData.RegionId == null || x.ReturnPlanItemNormalizationKey.SourceShop.Region.Id == filterData.RegionId) &&  
        (filterData.SourceShopId == null || x.ReturnPlanItemNormalizationKey.SourceShop.Id == filterData.SourceShopId) &&  
        (filterData.RecipientWarehouseId == null  
            || x.ReturnPlanItemNormalizationKey.RecieverStorage.Id == filterData.RecipientWarehouseId) &&  
        (filterData.NextWarehouseId == null || x.NextWarehouse.Id == filterData.NextWarehouseId) &&  
        (filterData.DeliveryTypeId == null || x.DeliveryType.Id == filterData.DeliveryTypeId) &&  
        (filterData.ShopClusterId == null || x.ShopCluster.Id == filterData.ShopClusterId) &&  
        (filterData.DateFrom == null || x.ReturnDate >= filterData.DateFrom.Value) &&  
        (filterData.DateTo == null || x.ReturnDate < filterData.DateTo.Value))
```

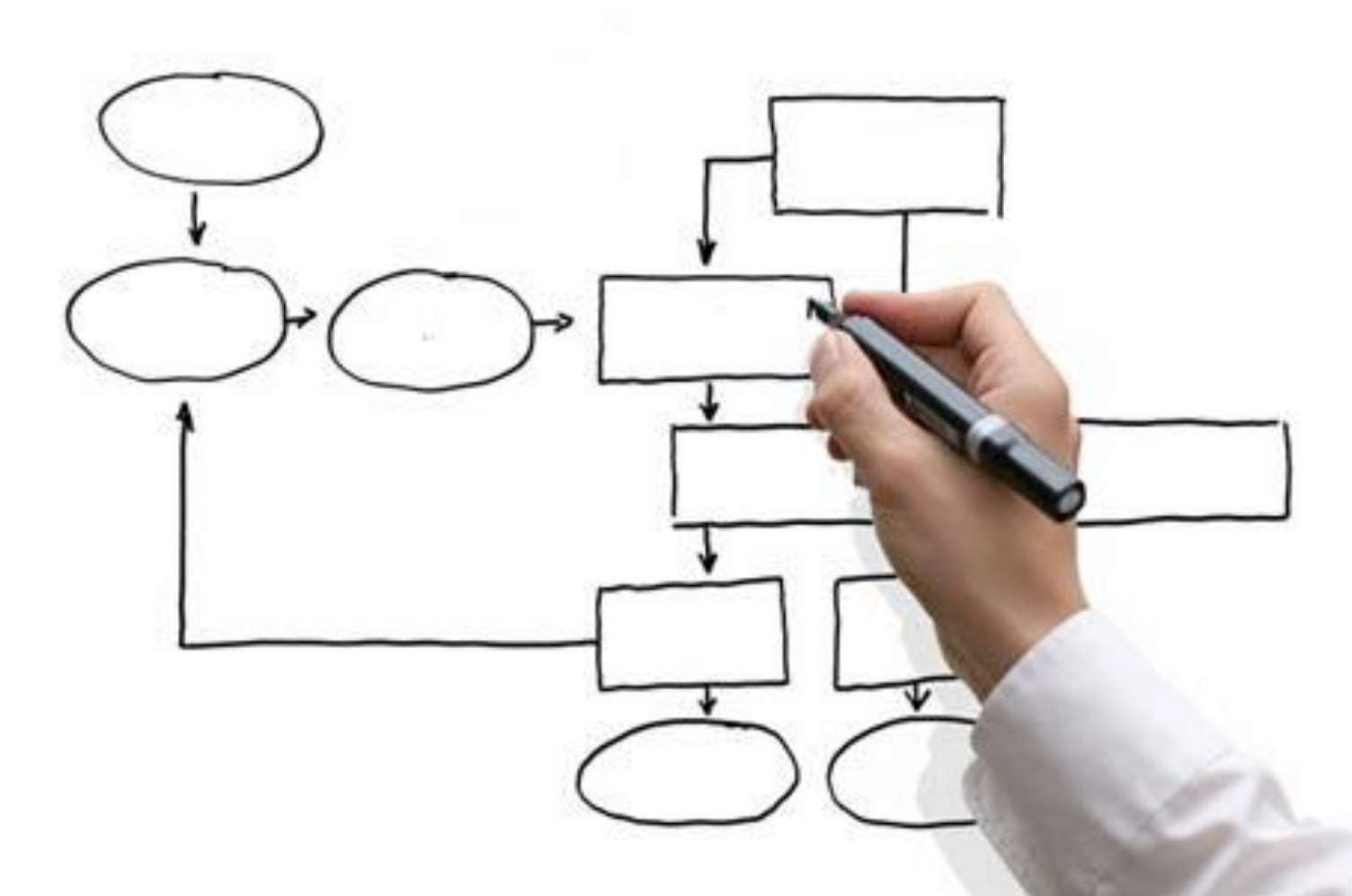
Клиент

```
GET /Service/ReturnPlanItemDtos?$filter=  
  (BeginDate lt ReturnPlan/BeginDate or EndDate gt ReturnPlan/EndDate)  
  and TradingNetwork/Id eq 19  
  and ReturnPlanItemNormalizationKey/SourceShop/Code eq 301  
  and Product/Subcategory/Id eq 66  
  and BeginDate ge 2015-11-02T00:00:00+03:00  
  and EndDate lt 2015-11-28T00:00:00+03:00  
  and EditStatus eq ReturnPlanItemEditStatus'Rms'  
  and (Status eq PlanItemStatus'New' or PlanItemStatus'IncludedInSchedule')
```

Клиент Telerik ASP.NET

```
<telerik:RadMenu RenderMode="Lightweight" runat="server" ID="RadMenu1">
  <WebServiceSettings Path="http://services.odata.org/OData/OData.svc">
    <ODataSettings ResponseType="JSONP">
      <Entities>
        <telerik:ODataEntityType Name="Category"
                                DataValueField="ID" DataTextField="Name" />
      </Entities>
      <EntityContainer>
        <telerik:ODataEntitySet EntityType="Category" Name="Categories" />
      </EntityContainer>
    </ODataSettings>
  </WebServiceSettings>
</telerik:RadMenu>
```

Проектируем API

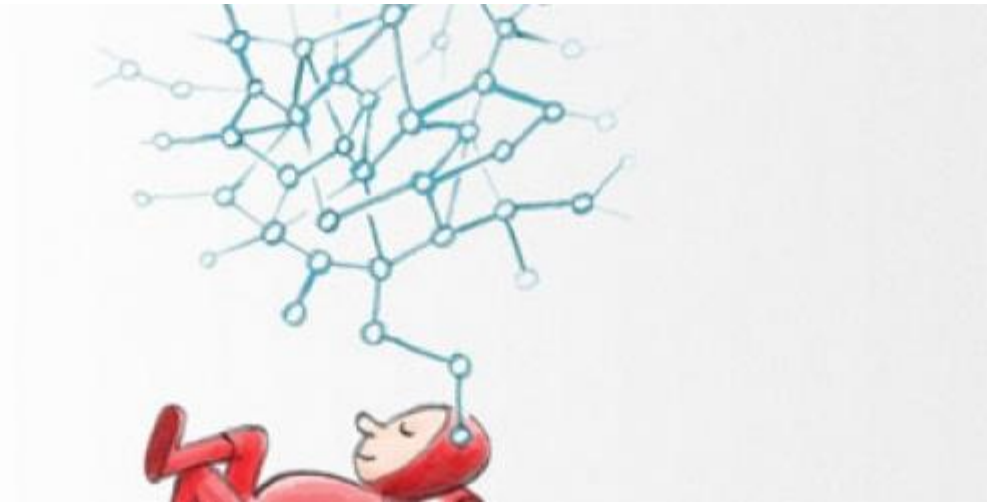


Проектируем API

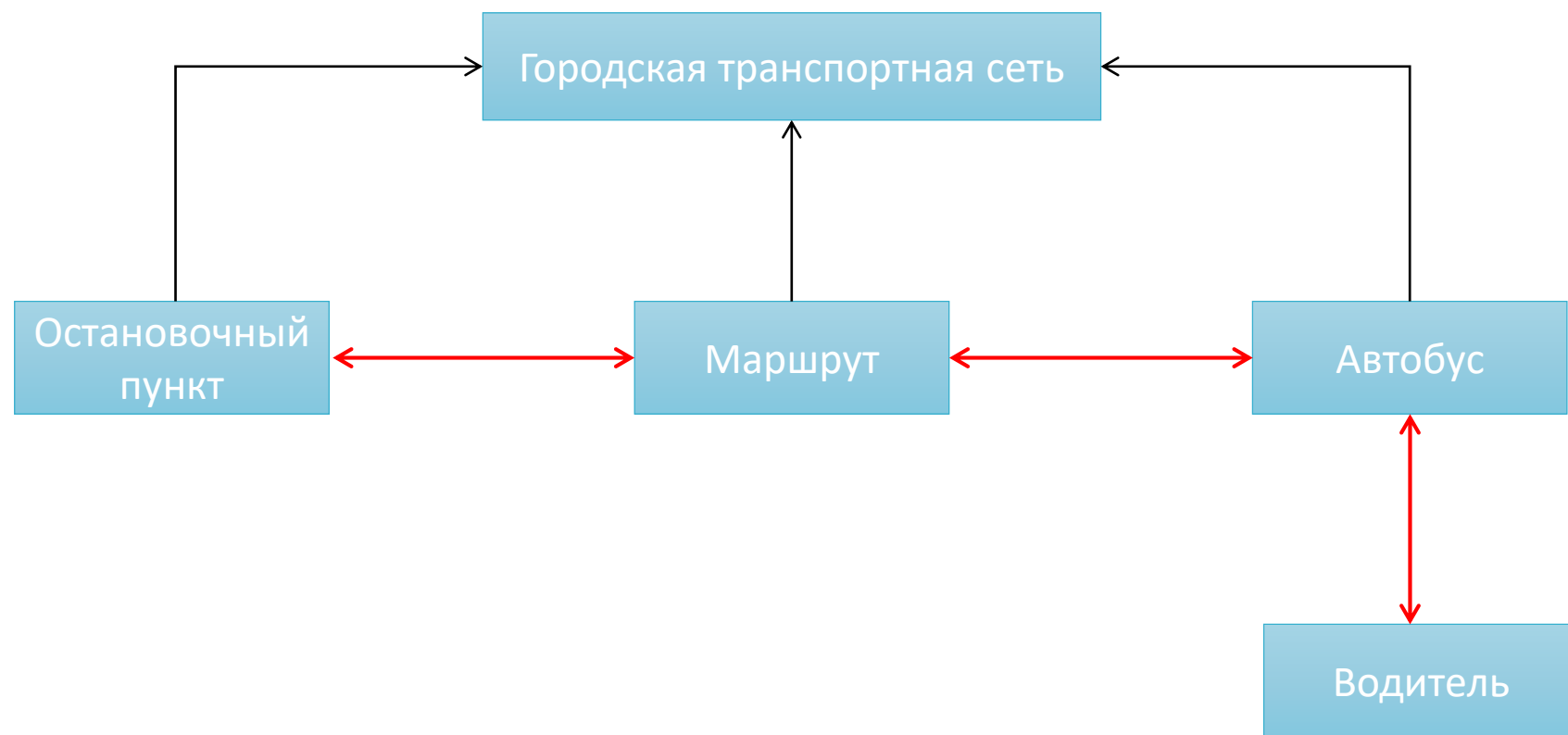
REST \neq
CRUD via HTTP

Понятия DDD

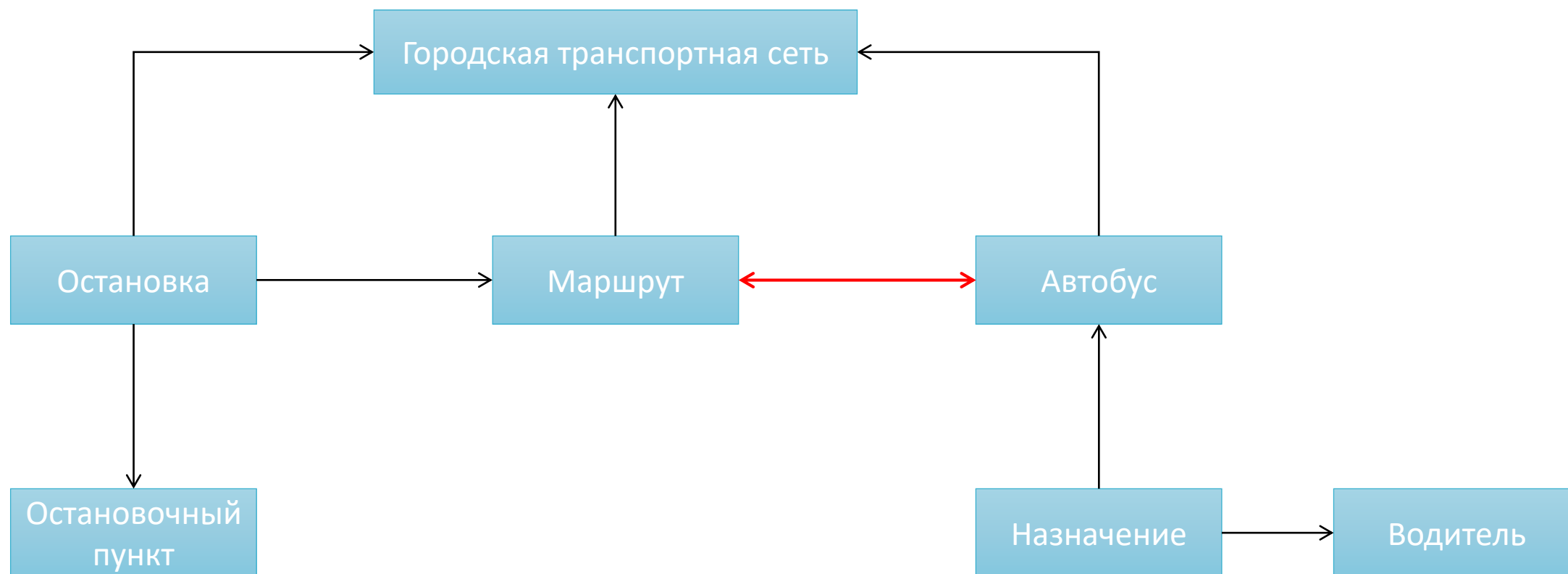
- Bounded Context
- Aggregates
- Domain Entities
- Value Objects



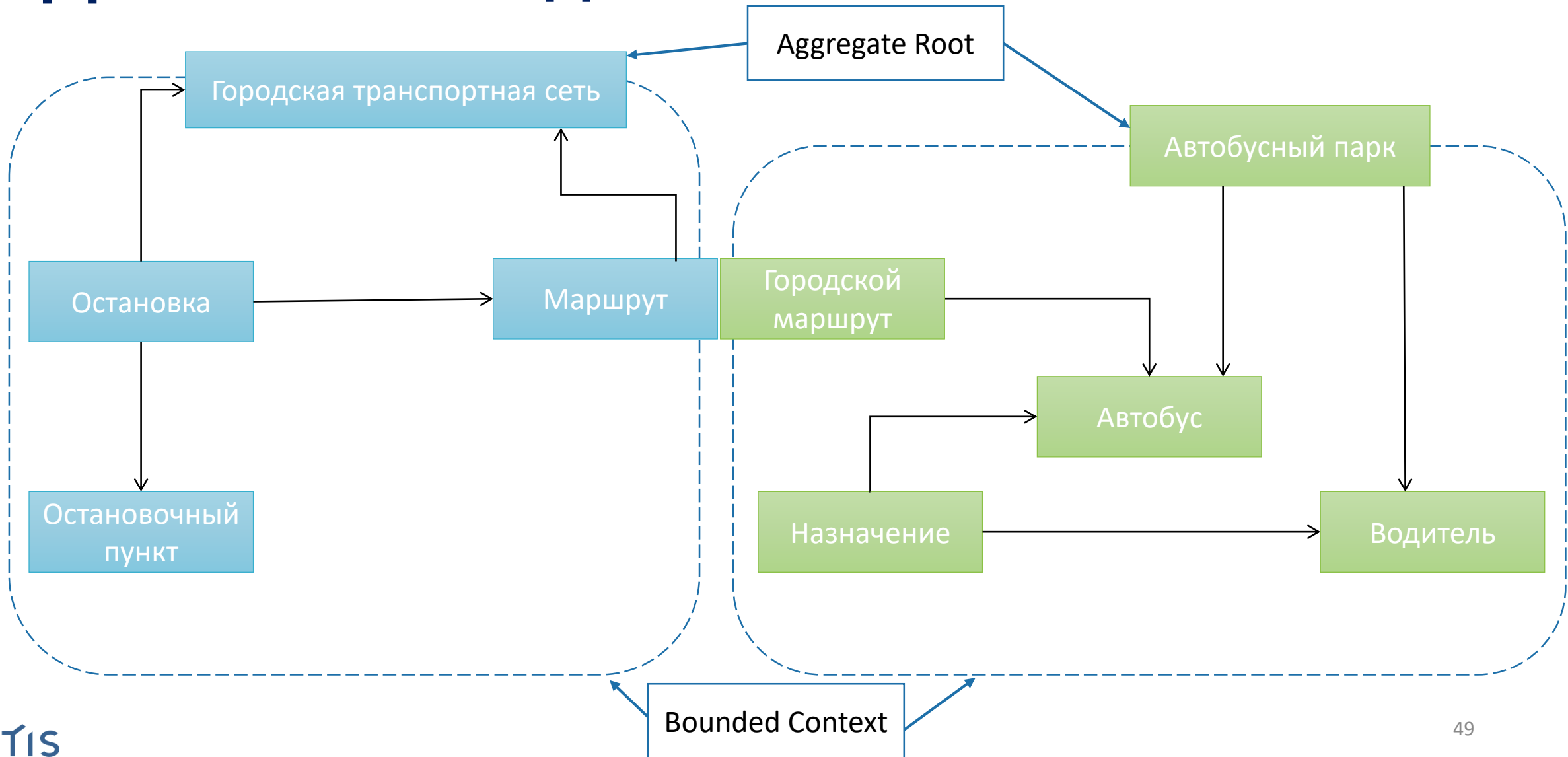
Доменная модель



Доменная модель



Доменная модель



Зачем...

...нужно думать об API?

Чтобы увеличить популярность сервиса

Почему...

...именно REST и чем он хорош?

Это современно и удобно

Как?

Проектируем по DDD

Основываясь на REST

Реализуем с OData

Хостим с помощью OWIN

Что читать?

<https://github.com/OData>

<http://www.odata.org/>

<http://dontpanic.42.nl/2012/04/rest-and-ddd-incompatible.html>

<http://www.dataart.ru/blog/2016/02/podhody-k-proektirovaniyu-restful-api/>

<https://msdn.microsoft.com/en-us/magazine/dn451439.aspx>

<http://martinfowler.com/bliki/CQRS.html>

<http://www.telerik.com/odata>

<http://semver.org/>

<https://github.com/climax-media/climax-web-http>

CUSTIS



Юлия Цисык, CustIS
yulia@tsisyk.com
