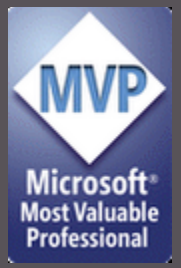
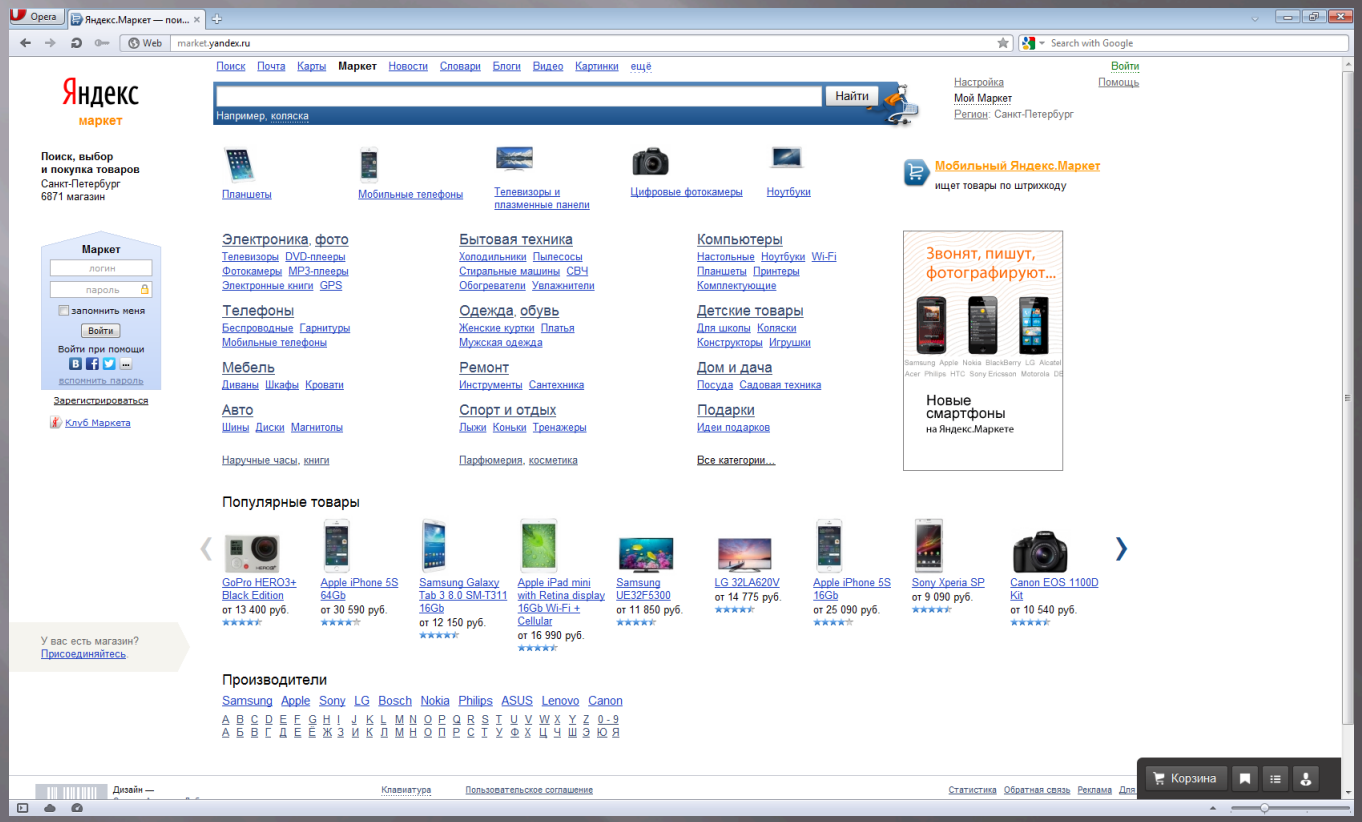
Интеграция ASP.NET MVC приложений для онлайн торговли с Яндекс Маркетом

Алексей Садомов, к.ф.-м.н., MVP
http://sadomovalex.blogspot.com/

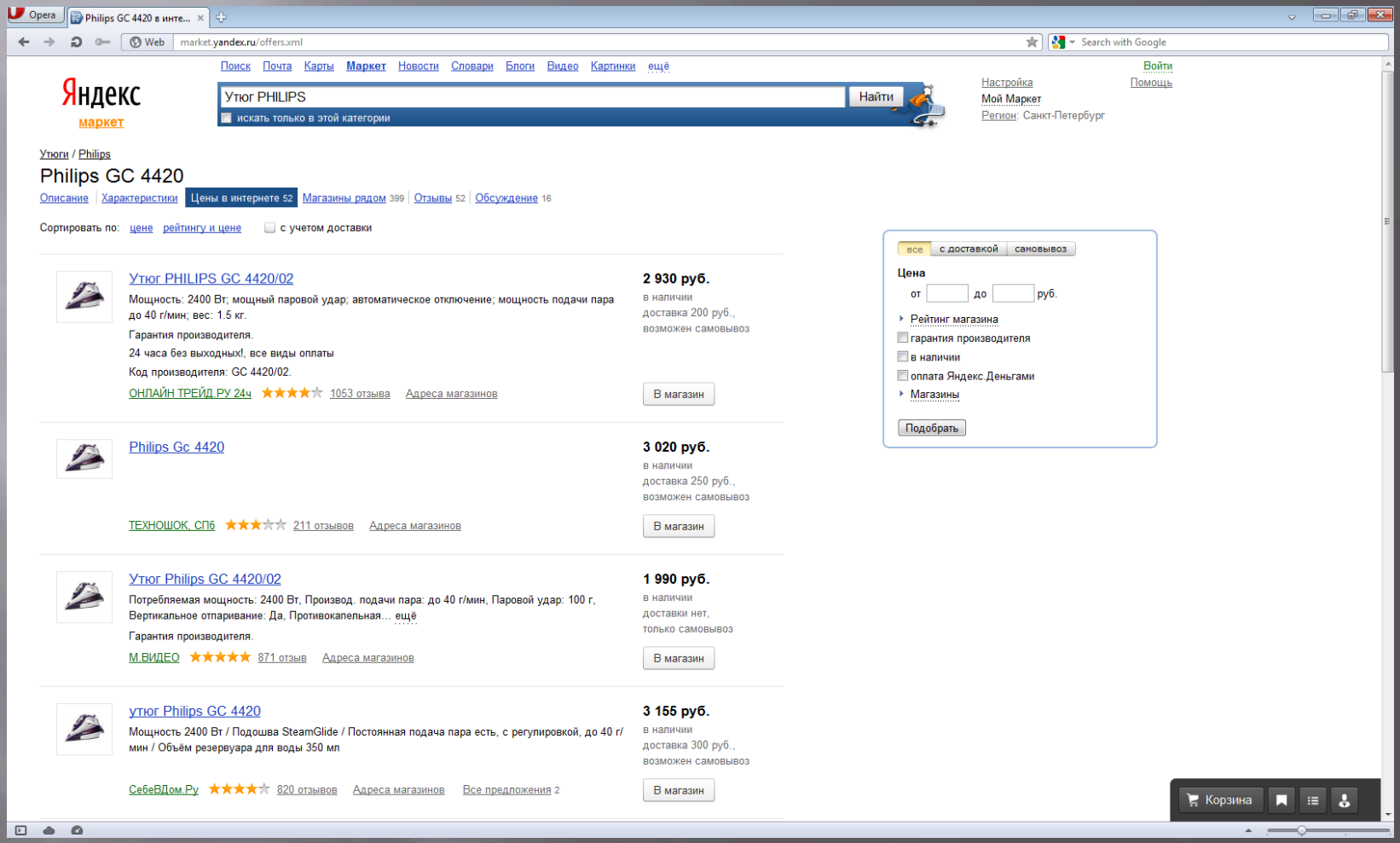


Что такое Яндекс Маркет?

- http://market.yandex.ru
- ~13 тыс магазинов
- ~17 млн посетителей в месяц

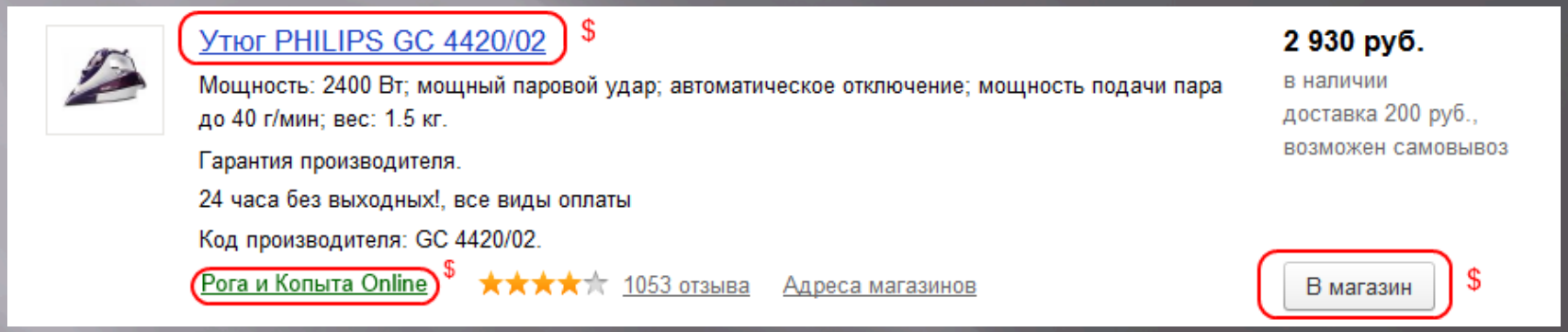


Что такое Яндекс Маркет?

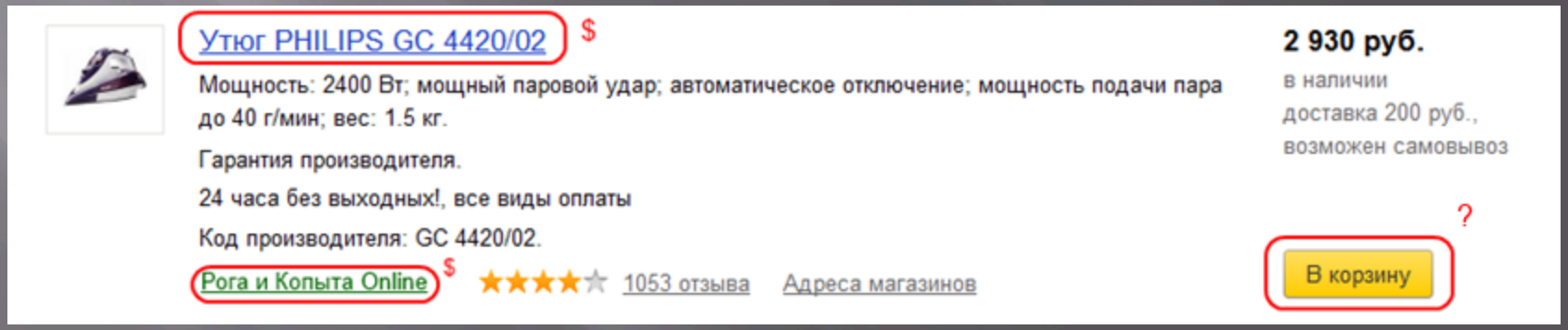


Модели размещения

- Cost per click (CPC)

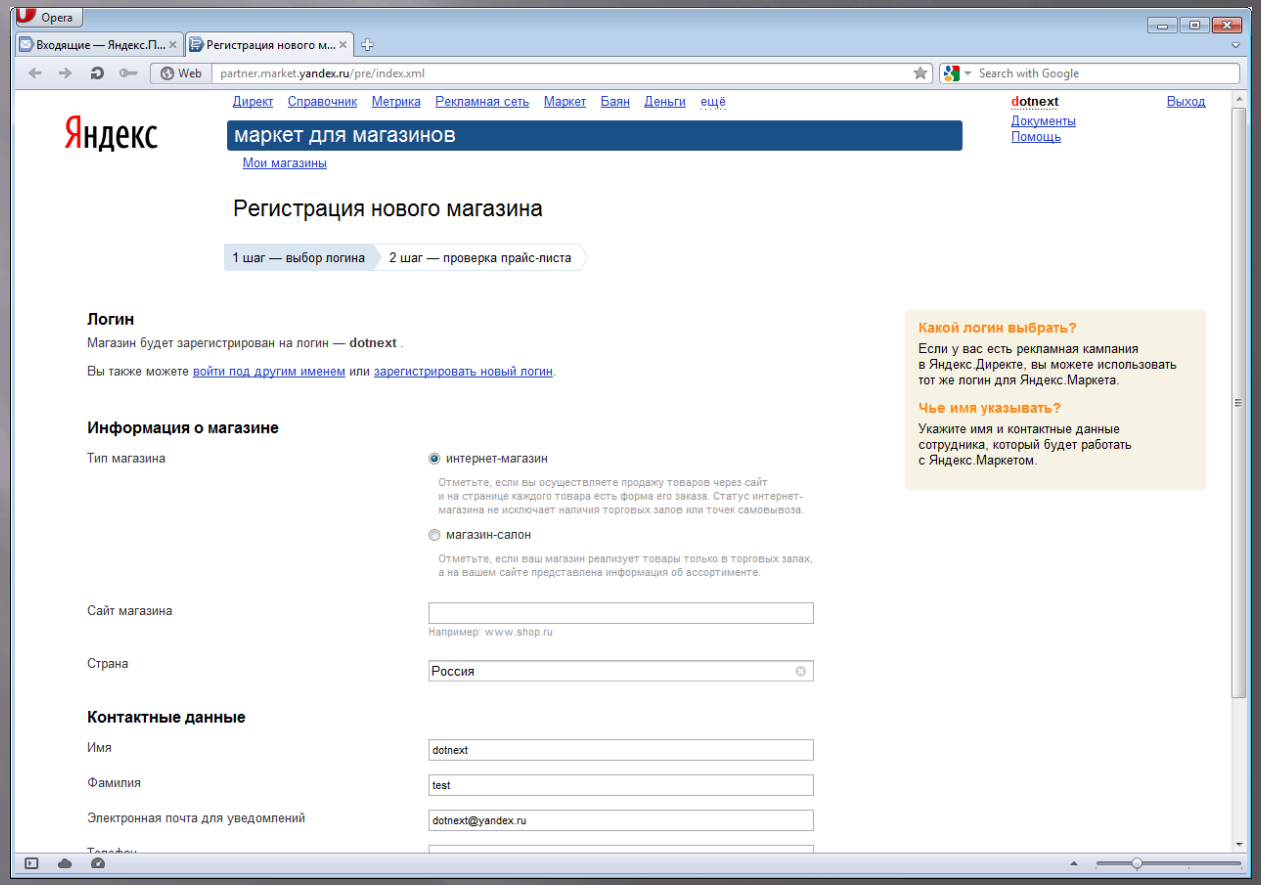


- Cost per action (CPA)



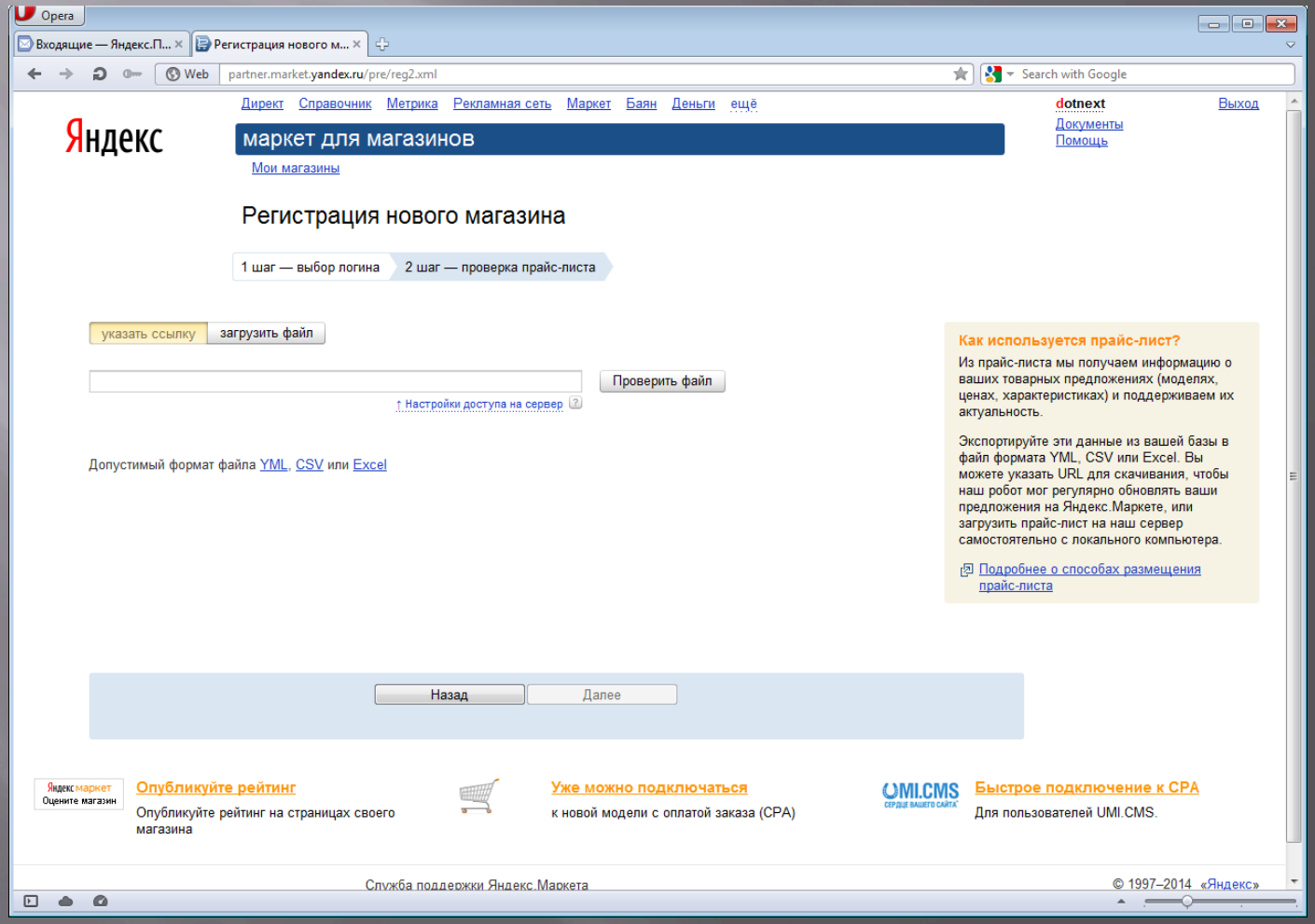
Как товары попадают на Маркет?

- Шаг 1: регистрация магазина в партнерском интерфейсе маркета http://partner.market.yandex.ru/



Как товары попадают на Маркет?

- Шаг 2: подготовка и проверка прайс-листа



Выгрузка прайс-листа

- 3 доступных формата:
  - YML (Yandex Market Language)
  - CSV
  - Excel
- Форматы CSV и Excel имеют ряд ограничений по сравнению с YML
- YML - стандарт, разработанный Яндексом для принятия и размещения информации в базе данных Маркета. Основан на стандарте XML

Yandex Market Language

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE yml_catalog SYSTEM "shops.dtd">
- <yml_catalog date="2014-03-12 17:40">
  - <shop>
      <name>example.com</name>
      <company>example.com</company>
      <url>http://example.com</url>
    - <currencies>
        <currency plus="0" rate="1" id="RUR"/>
      </currencies>
    - <categories>
        <category id="53">Категория 1</category>
        <category id="54" parentId="53">Подкатегория 2</category>
        <category id="55" parentId="53">Подкатегория 1</category>
        <category id="56" parentId="53">Подкатегория 3</category>
        <category id="57">Категория 2</category>
        <category id="58">Категория 3</category>
      </categories>
    - <offers>
      - <offer type="vendor.model" id="487" available="true">
          <url>http://example.com/product/487</url>
          <price>1100</price>
          <currencyId>RUR</currencyId>
          <categoryId>53</categoryId>
          <picture> http://example.com/productimage/487</picture>
          <pickup>true</pickup>
          <delivery>true</delivery>
          <local_delivery_cost>200</local_delivery_cost>
          <vendor>Тест</vendor>
          <model>Блендер</model>
          <country_of_origin>Россия</country_of_origin>
        </offer>
      + <offer type="vendor.model" id="480" available="true">
      + <offer type="vendor.model" id="483" available="true">
      + <offer type="vendor.model" id="477" available="true">
      + <offer type="vendor.model" id="481" available="true">
      + <offer type="vendor.model" id="485" available="true">
      + <offer type="vendor.model" id="488" available="true">
      + <offer type="vendor.model" id="479" available="true">
      + <offer type="vendor.model" id="482" available="true">
      + <offer type="vendor.model" id="478" available="true">
      + <offer type="vendor.model" id="484" available="true">
      + <offer type="vendor.model" id="486" available="true">
      </offers>
  </shop>
</yml_catalog>
```

Заголовок

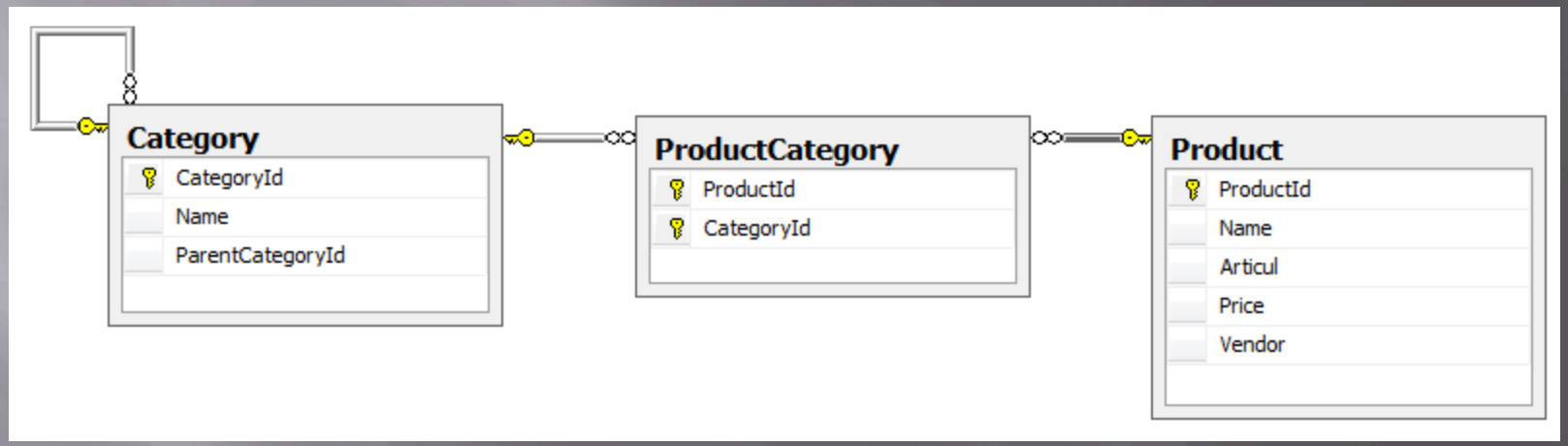
Общая информация о магазине

Валюта

Дерево категорий

Предложения

Товары в БД простого магазина



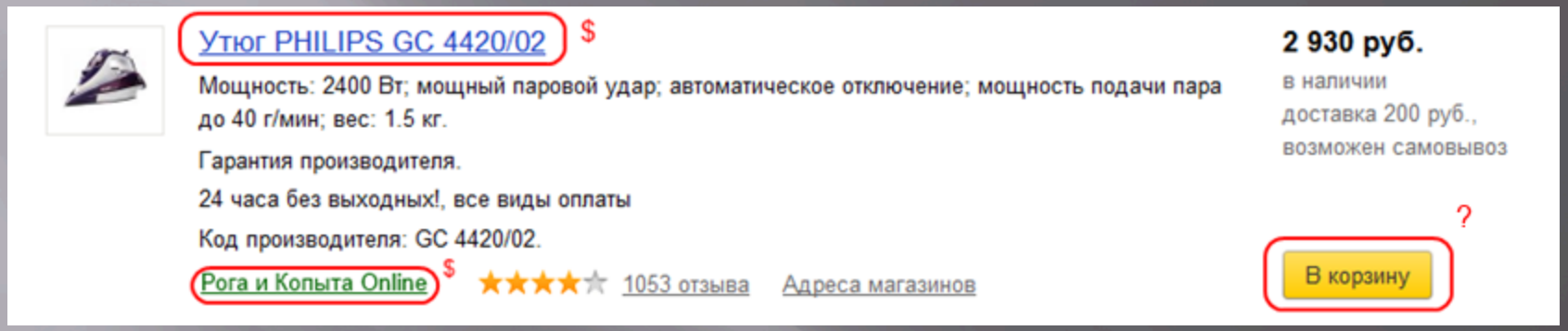
NVelocity шаблон для выгрузки в YML

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE yml_catalog SYSTEM "shops.dtd">
<yml_catalog date="$yml.Created">
  <shop>
    <name>$yml.ShopName</name>
    <company>$yml.Company</company>
    <url>$yml.Url</url>
    <currencies>
      <currency id="$yml.Currency" rate="1" plus="0"/>
    </currencies>
    <categories>
      #if($yml.Categories)
        #foreach($c in $yml.Categories)
        #each
          #if($c.ParentId)
            <category id="$c.Id" parentId="$c.ParentId">$c.Name</category>
          #else
            <category id="$c.Id">$c.Name</category>
          #end
        #end
      #end
    </categories>
    <offers>
      #if($yml.Offers)
        #foreach($o in $yml.Offers)
        #each
          <offer id="$o.Id" type="vendor.model" available="$o.Available">
            <url>$o.Url</url>
            <price>$o.Price</price>
            <currencyId>$yml.Currency</currencyId>
            <categoryId>$o.CategoryId</categoryId>
            <picture>$o.Picture</picture>
            <pickup>true</pickup>
            <delivery>true</delivery>
            <local_delivery_cost>$o.DeliveryCost</local_delivery_cost>
            <vendor>$o.Vendor</vendor>
            #if($o.Articul && $o.Articul != "")
            <vendorCode>$o.Articul</vendorCode>
            #end
            <model>$o.Name</model>
            #if($o.CountryOfProduction && $o.CountryOfProduction != "")
            <country_of_origin>$o.CountryOfProduction</country_of_origin>
            #end
          </offer>
        #end
      #end
    </offers>
  </shop>
</yml_catalog>
```

Генерация YML с помощью NVelocity

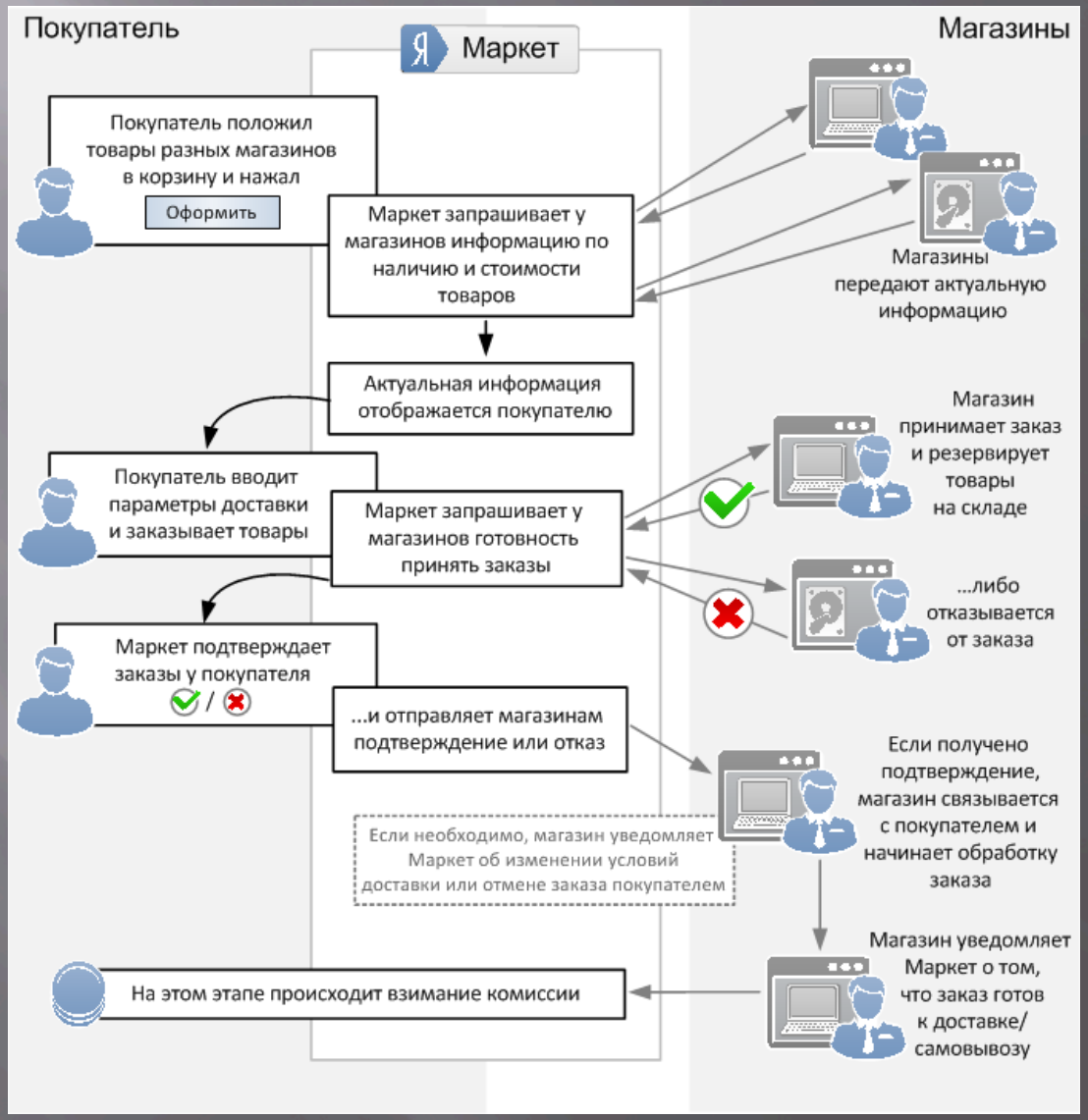
```
private string export(string template)
{
    string name = "Рога и Копыта Онлайн";
    var categories = this.getCategories();
    var offers = this.getOffers();
    var p = new
    {
        Created = DateTime.Now.ToString("yyyy-MM-dd HH:mm"),
        ShopName = name,
        Company = name,
        Url = "http://example.com",
        Categories = categories,
        Offers = offers,
        Currency = "RUB"
    };

    var context = new VelocityContext();
    context.Put("yml", p);

    var velocity = new VelocityEngine();
    var props = new ExtendedProperties();
    velocity.Init(props);
    using (var sw = new StringWriter())
    {
        velocity.Evaluate(context, sw, "", template);
        sw.Flush();
        return sw.ToString();
    }
}
```

CPA: Покупка на Маркете



- можно не только выбрать товары, но и сразу оформить заказ
- единый интерфейс оформления заказа и общая корзина для товаров из разных магазинов
- комиссия 1% с подтвержденных заказов

CPA: Покупка на Маркете



Методы API покупки

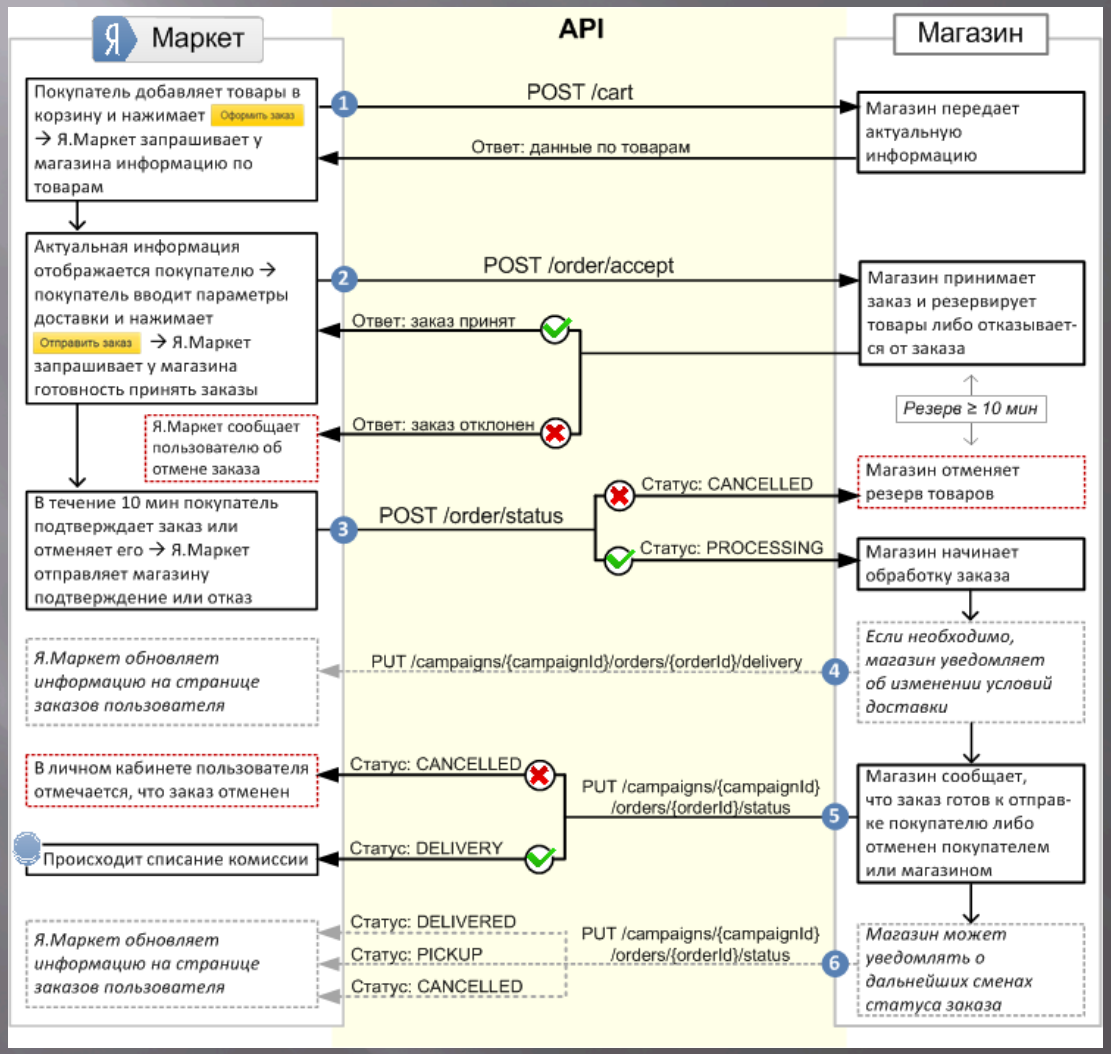
- Методы, с помощью которых Маркет выполняет запросы к магазину:

Метод	Краткое описание
POST /cart	Запрашивает у магазина информацию о товарах в корзине.
POST /order/accept	Передает заказ магазину и запрашивает подтверждение принятия заказа магазином.
POST /order/status	Уведомляет магазин о смене статуса заказа.

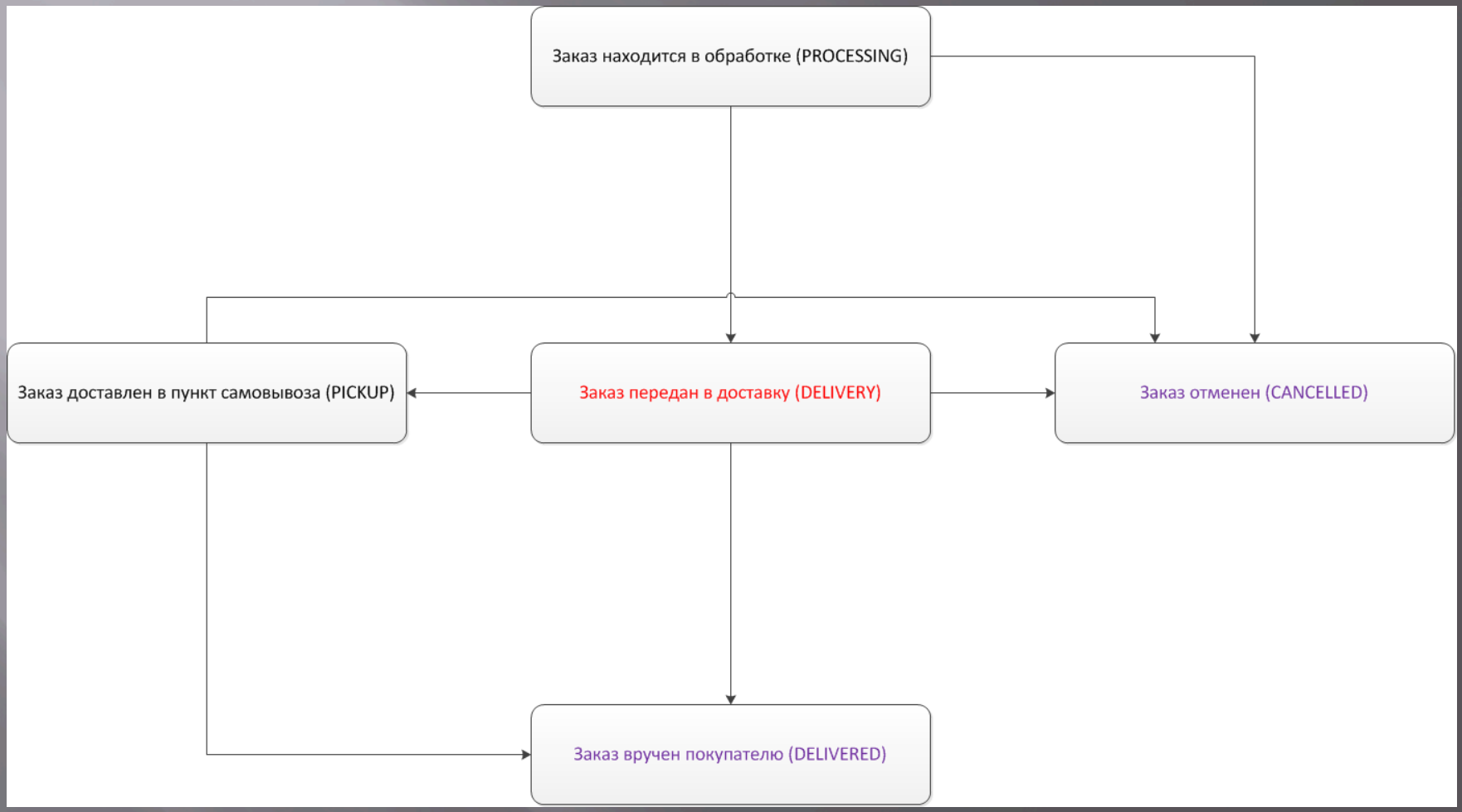
- Методы, с помощью которых магазин выполняет запросы к Яндекс.Маркету:

Метод	Краткое описание
PUT /campaigns/{campaignId}/orders/{orderId}/status	Изменяет статус заказа.
PUT /campaigns/{campaignId}/orders/{orderId}/delivery	Изменяет условия доставки заказа.
GET /campaigns/{campaignId}/orders/{orderId}	Запрашивает информацию о конкретном заказе, оформленном на магазин в интерфейсе Яндекс.Маркета.
GET /campaigns/{campaignId}/orders	Запрашивает информацию о заказах, оформленных на магазин в интерфейсе Яндекс.Маркета.

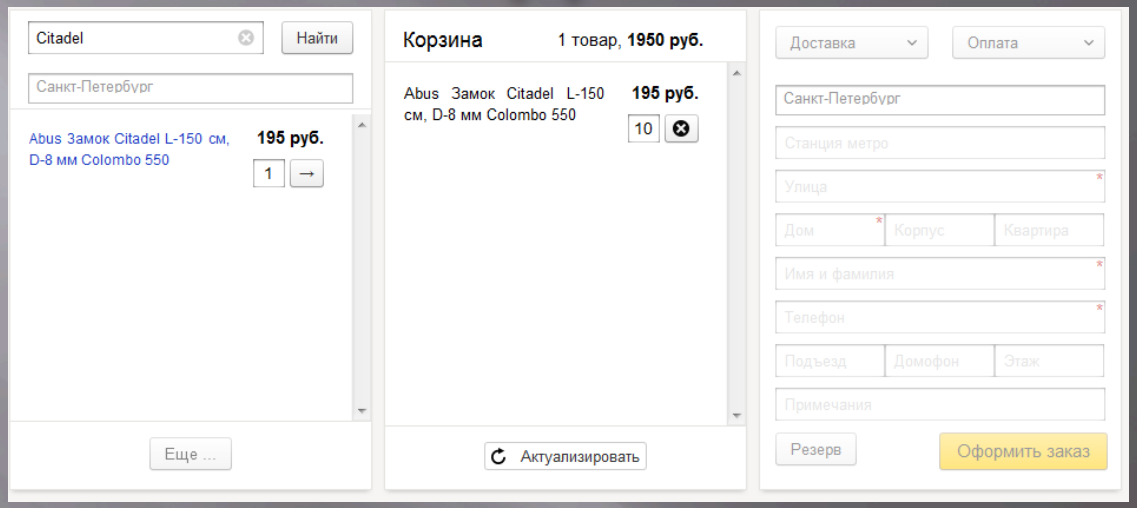
Порядок взаимодействия через API



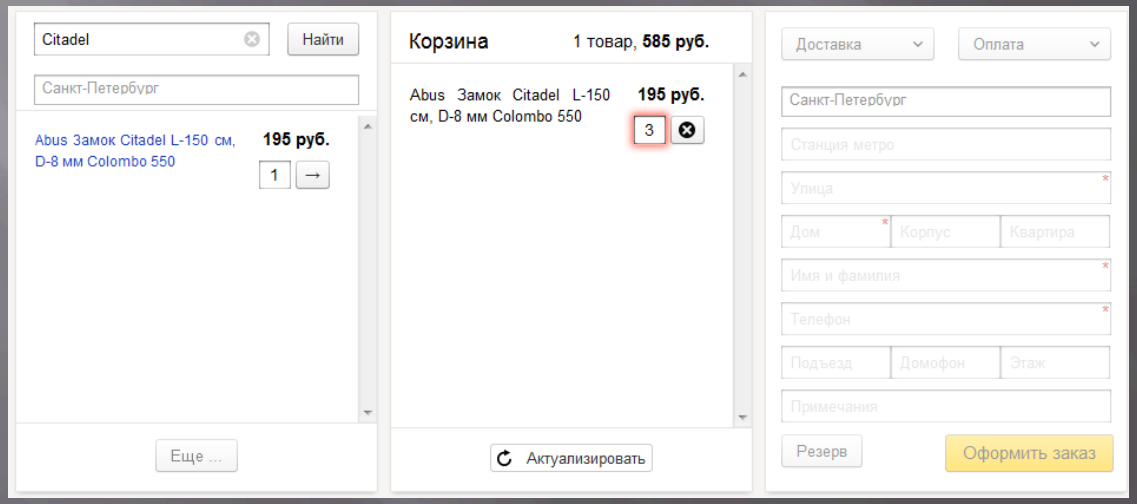
CPA: workflow заказа



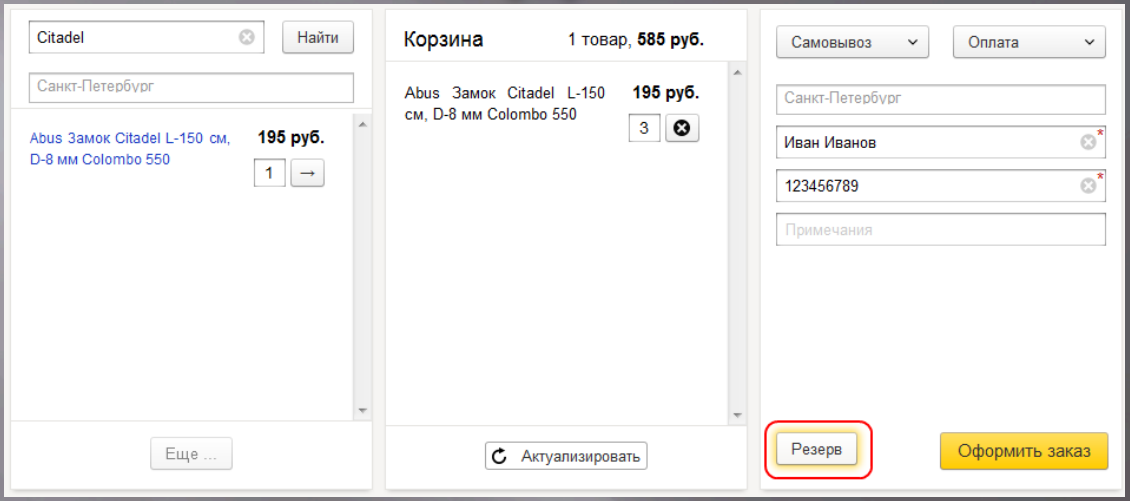
Отладка API



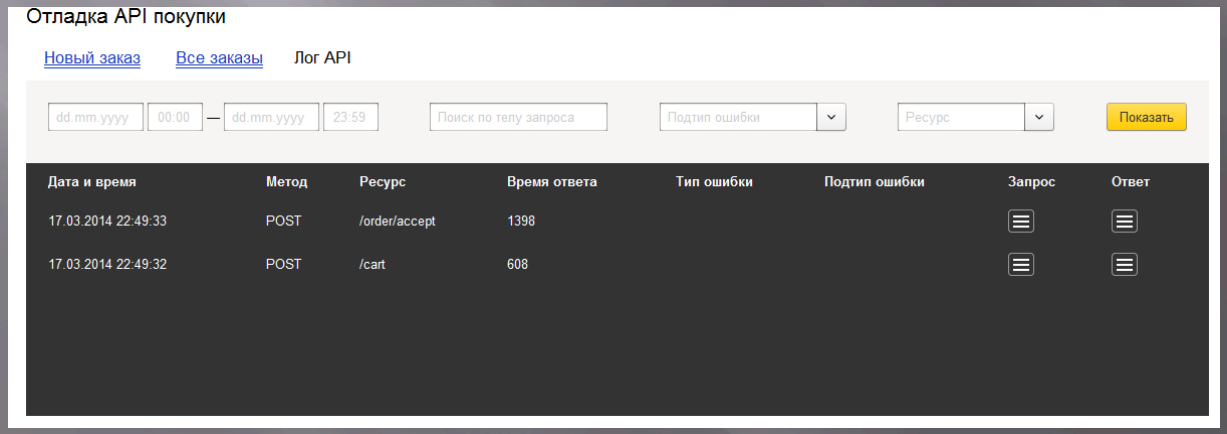
post / cart

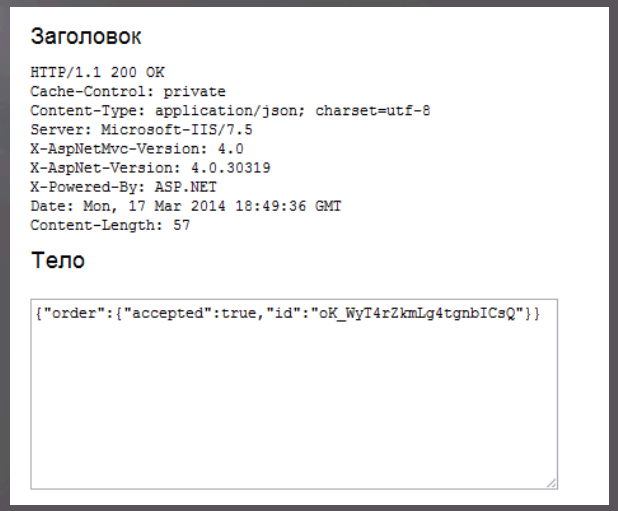


Отладка API

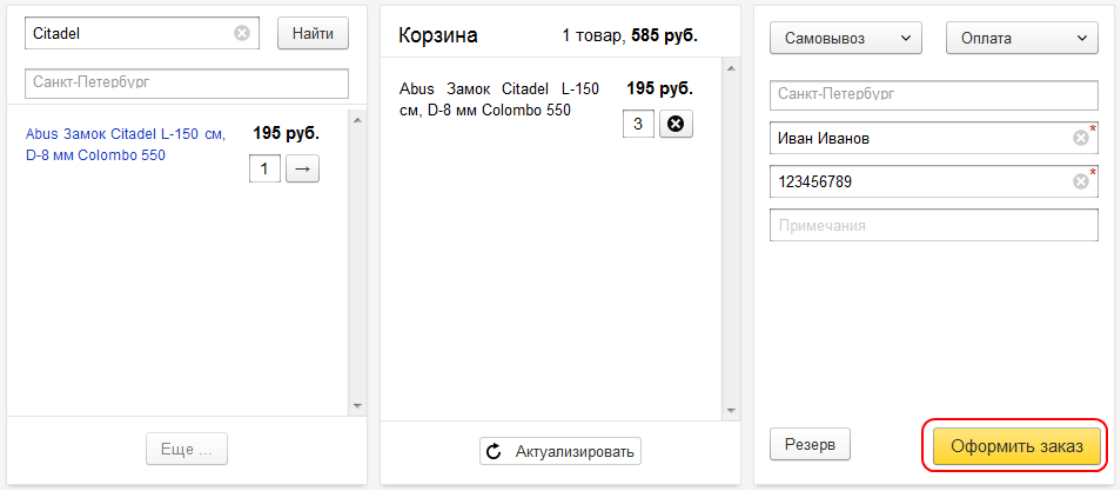


post /order/accept

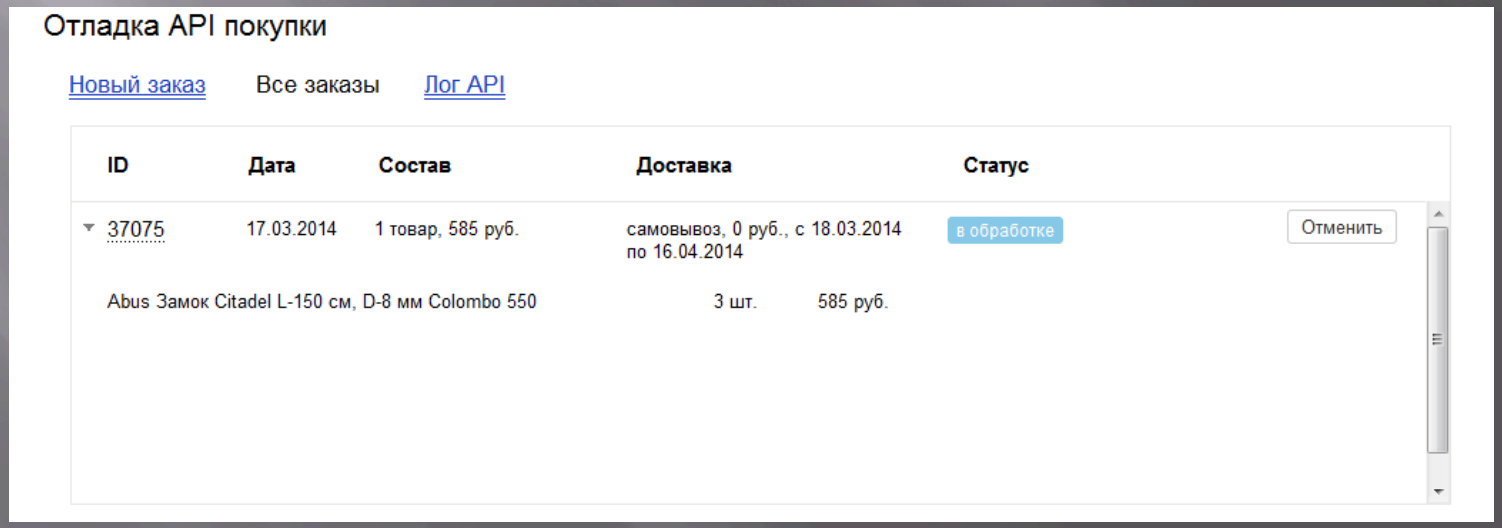




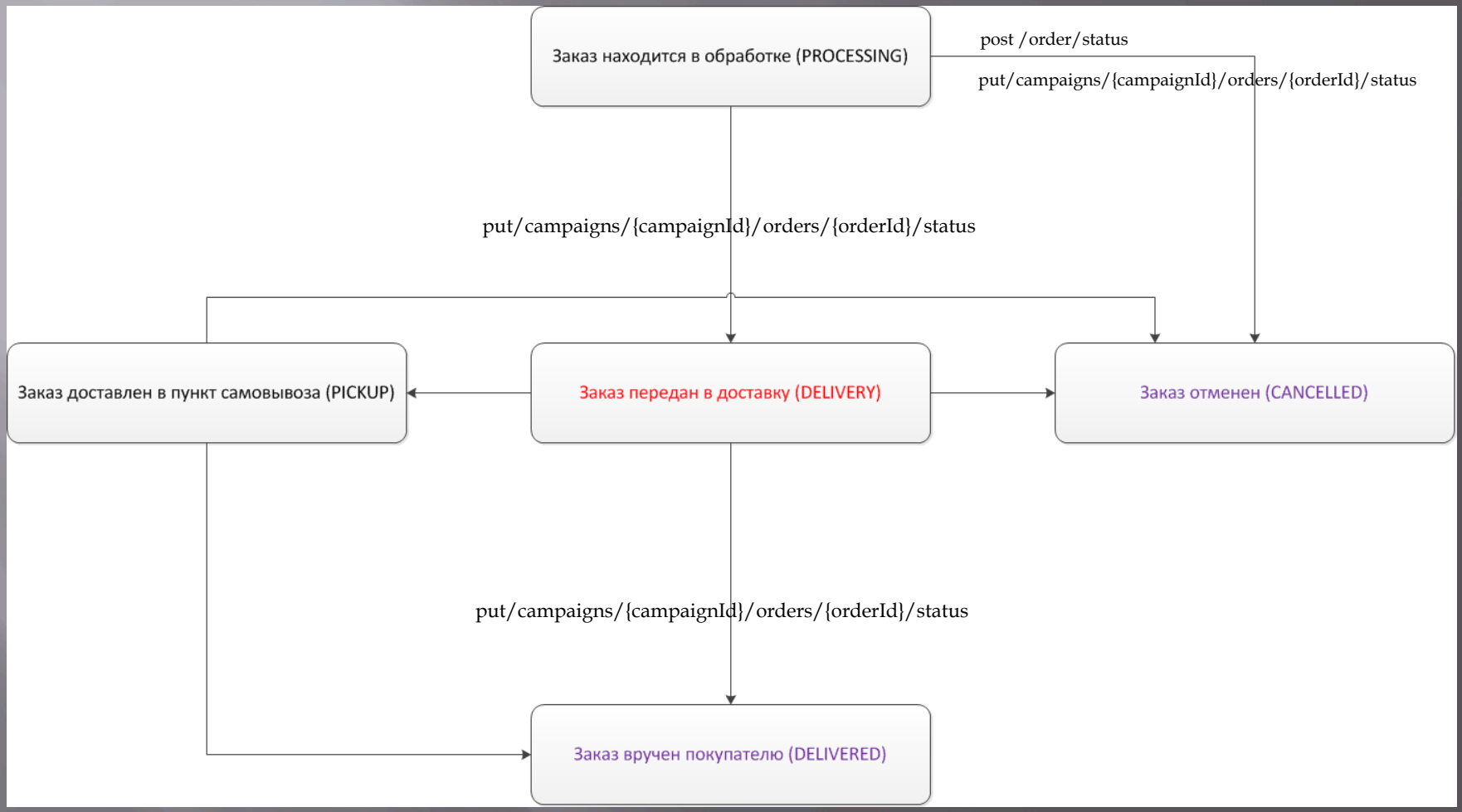
Отладка API



post /order/status



Отладка API


post /order/status
Заказ находится в обработке (PROCESSING)
post /order/status
put/campaigns/{campaignId}/orders/{orderId}/status
put/campaigns/{campaignId}/orders/{orderId}/status
Заказ доставлен в пункт самовывоза (PICKUP)
Заказ передан в доставку (DELIVERY)
Заказ отменен (CANCELLED)
put/campaigns/{campaignId}/orders/{orderId}/status
Заказ вручен покупателю (DELIVERED)

Подготовка параметров для MVC контроллеров

- 2 формата данных:
  - Xml
  - JSON

```xml
<cart currency="{string}">
    <items>
        <item feed-id="{number}" offer-id="{string}" feed-category-id="{string}" offer-name="{string}" count="{number}"/>
        ...
    </items>
    <delivery>
      <region id="{number}" name="{string}" type="{string}">
          <parent id="{number}" name="{string}" type="{string}">
              ...
          </parent>
      </region>
      <address country="{string}" postcode="{string}" city="{string}" subway="{string}" street="{string}" house="{string}"
block="{string}" floor="{string}"/>
    </delivery>
</cart>
```

```json
{
  "cart":
  {
    "currency":"{string}",
    "items":
    [
      {
        "feedId":{number},
        "offerId":"{string}",
        "offerName":"{string}",
        "count":{number},
        "feedCategoryId":"{string}"
      },
      ...
    ],
    "delivery":
    {
      "region":
      {
        "id":{number},
        "name":"{string}",
        "type":"{string}",
        "parent":
        {
          "id":{number},
          "name":"{string}",
          "type":"{string}",
          "parent":
          {
            ...
          }
        }
      },
      "address":
      {
        "country":"{string}",
        "postcode":"{string}",
        "city":"{string}",
        "subway":"{string}",
        "street":"{string}",
        "house":"{string}",
        "block":"{string}",
        "floor":"{string}"
      }
    }
  }
}
```

Подготовка параметров для MVC контроллеров

- Запрос

```
{
  "cart":
  {
    "currency":"{string}",
    "items":
    [
      {
        "feedId":{number},
        "offerId":"{string}",
        "offerName":"{string}",
        "count":{number},
        "feedCategoryId":"{string}"
      },
      ...
    ],
    "delivery":
    {
      "region":
      {
        "id":{number},
        "name":"{string}",
        "type":"{string}",
        "parent":
        {
          "id":{number},
          "name":"{string}",
          "type":"{string}",
          "parent":
          {
            ...
          }
        }
      },
      "address":
      {
        "country":"{string}",
        "postcode":"{string}",
        "city":"{string}",
        "subway":"{string}",
        "street":"{string}",
        "house":"{string}",
        "block":"{string}",
        "floor":"{string}"
      }
    }
  }
}
```

POCO

```
public class YMCartRequestWrapper
{
    public YMCartRequest cart { get; set; }
}

public class YMCartRequest
{
    public string currency { get; set; }
    public YMCartRequestItem[] items { get; set; }
    public YMCartRequestDelivery delivery { get; set; }
}

public class YMCartRequestItem
{
    public int feedId { get; set; }
    public int offerId { get; set; }
    public string offerName { get; set; }
    public int count { get; set; }
    public int feedCategoryId { get; set; }
}

public class YMCartRequestDelivery
{
    public YMCartRequestRegion region { get; set; }
    public YMCartRequestAddress address { get; set; }
}

public class YMCartRequestRegion
{
    public string id { get; set; }
    public string name { get; set; }
    public string type { get; set; }
    public YMCartRequestRegion parent { get; set; }
}

public class YMCartRequestAddress
{
    public string country { get; set; }
    public string postcode { get; set; }
    public string city { get; set; }
    public string subway { get; set; }
    public string street { get; set; }
    public string house { get; set; }
    public string block { get; set; }
    public string floor { get; set; }
}
```

Подготовка параметров для MVC контроллеров

- Ответ:

```
{
  "cart":
  {
    "items":
    [
      {
        "feedId":{number},
        "offerId":"{string}",
        "price":{number},
        "count":{number},
        "delivery":{boolean}
      },
      ...
    ],
    "deliveryOptions":
    [
      {
        "id":"{string}",
        "type":"{string}",
        "serviceName":"{string}",
        "price":{number},
        "dates":
        {
          "fromDate":"{string}",
          "toDate":"{string}"
        },
        "outlets":
        [
          {
            "id":{number}
          },
           ...
        ]
      },
      ...
    ],
    "paymentMethods":
    [
      "{string}",
        ...
    ]
  }
}
```

POCO

```
public class YMCartResponseWrapper
{
    public YMCartResponse cart { get; set; }
}

public class YMCartResponse
{
    public YMCartResponseItem[] items { get; set; }
    public YMCartResponseDeliveryOption[] deliveryOptions { get; set; }
    public string[] paymentMethods { get; set; }
}

public class YMCartResponseItem
{
    public int feedId { get; set; }
    public string offerId { get; set; }
    public decimal price { get; set; }
    public int count { get; set; }
    public bool delivery { get; set; }

    public YMCartResponseItem() {}
}

public class YMCartResponseDeliveryOption
{
    public string id { get; set; }
    public string type { get; set; }
    public string serviceName { get; set; }
    public decimal price { get; set; }
    public YMCartResponseDeliveryOptionDates dates { get; set; }
    public YMCartResponseDeliveryOptionOutlet[] outlets { get; set; }

    public YMCartResponseDeliveryOption() {}
}

public class YMCartResponseDeliveryOptionDates
{
    public string fromDate { get; set; }
    public string toDate { get; set; }
}

public class YMCartResponseDeliveryOptionOutlet
{
    public int id { get; set; }
}
```

Создаем контроллер

```
public class YMController : Controller
{
    ...

    [HttpPost]
    [LogYMReqRes]
    [YMAuth]
    public ActionResult Cart(YMCartRequestWrapper request)
    {
        try
        {
            List<string> errors = null;
            if (!this.ymValidator.IsCartRequestValid(request, out errors))
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest, error);
            }

            var response = this.ymService.CreateCartResponse(request);
            if (response == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.InternalServerError);
            }

            var result = new JsonResult();
            result.Data = response;
            return result;
        }
        catch (Exception x)
        {
            // log
            return new HttpStatusCodeResult(HttpStatusCode.InternalServerError);
        }
    }

}
```

ActionFilter для отладки

```
[AttributeUsage(AttributeTargets.Method)]
public class LogYMReqResAttribute : FilterAttribute, IActionFilter
{
    public ILogger Logger { get; set; }

    public void OnActionExecuting(ActionExecutingContext filterContext)
    {
        try
        {
            var req = Utils.Serialize(filterContext.ActionParameters["request"]);
            string headers = Utils.SerializeHttpHeaders(filterContext.HttpContext.Request.Headers);
            string url = filterContext.HttpContext.Request.RawUrl;
            this.Logger.Info("New request is received ({0}):\n{1}\n{2}", url, req, headers);
        }
        catch
        {
        }
    }

    public void OnActionExecuted(ActionExecutedContext filterContext)
    {
        try
        {
            var res = Utils.Serialize(filterContext.Result);
            string headers = Utils.SerializeHttpHeaders(filterContext.HttpContext.Response.Headers);
            this.Logger.Info("Response sent:\n{0}\n{1}", res, headers);
        }
        catch
        {
        }
    }
}
```

Настройки авторизации для вызовов из Маркета в магазин

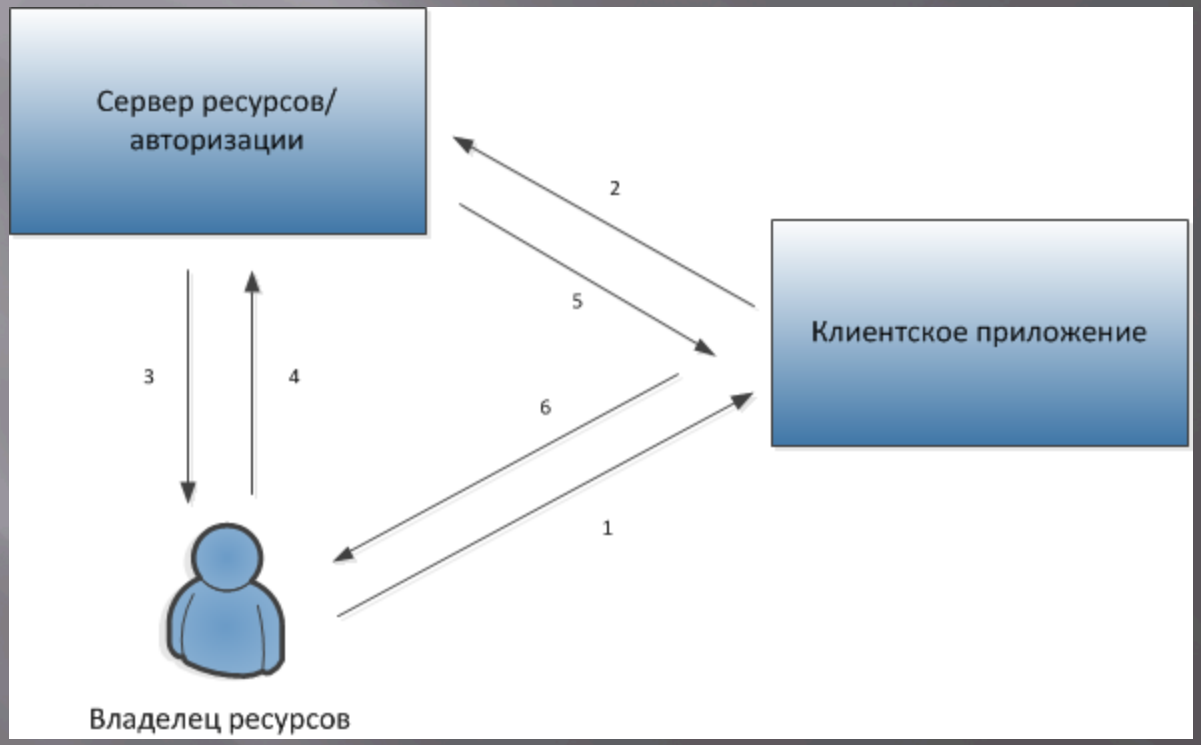
- Устанавливаются в партнерском интерфейсе:
  - URL API
    - только https
  - SHA1 fingerprint:
    - из ssl сертификата
  - Авторизационный токен:
    - Авторизационный токен передается при каждом вызове API и используется для авторизации. Реализация API должна проверять передачу корректного токена для каждого вызова. При попытке передачи некорректного токена или отсутствии его как такового, реализация должна возвращать ошибку HTTP 403
  - Тип авторизации
    - Header
    - URL

AuthorizationFilter для проверки токена авторизации

```
[AttributeUsage(AttributeTargets.Method)]
public class YMAuthAttribute : FilterAttribute, IAuthorizationFilter
{
    public ISettingsProvider SettingsProvider { get; set; }

    public void OnAuthorization(AuthorizationContext filterContext)
    {
        string authTokenFromRequest = filterContext.HttpContext.Request.Headers[HttpRequestHeader.Authorization.ToString()];
        if (string.IsNullOrEmpty(authTokenFromRequest))
        {
            filterContext.Result = new HttpStatusCodeResult(HttpStatusCode.Forbidden, "auth token not specified");
            return;
        }

        string authTokenFromSite = ...;
        if (string.IsNullOrEmpty(authTokenFromSite))
        {
            filterContext.Result = new HttpStatusCodeResult(HttpStatusCode.Forbidden, "auth token not set");
            return;
        }

        if (string.Compare(authTokenFromRequest, authTokenFromSite, true) != 0)
        {
            filterContext.Result = new HttpStatusCodeResult(HttpStatusCode.Forbidden, "auth token is not valid");
            return;
        }
    }
}
```

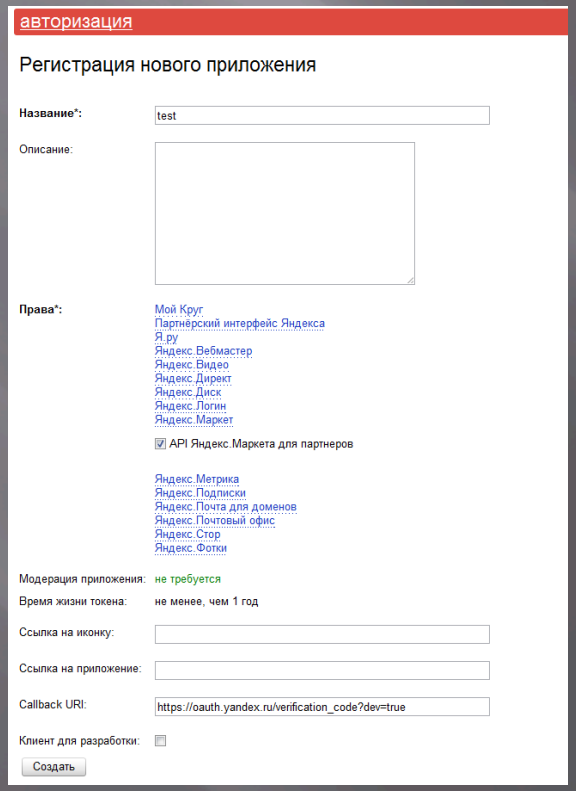
Авторизация вызовов из магазина в Маркет

- Классическая модель вызовов OAuth 2.0:



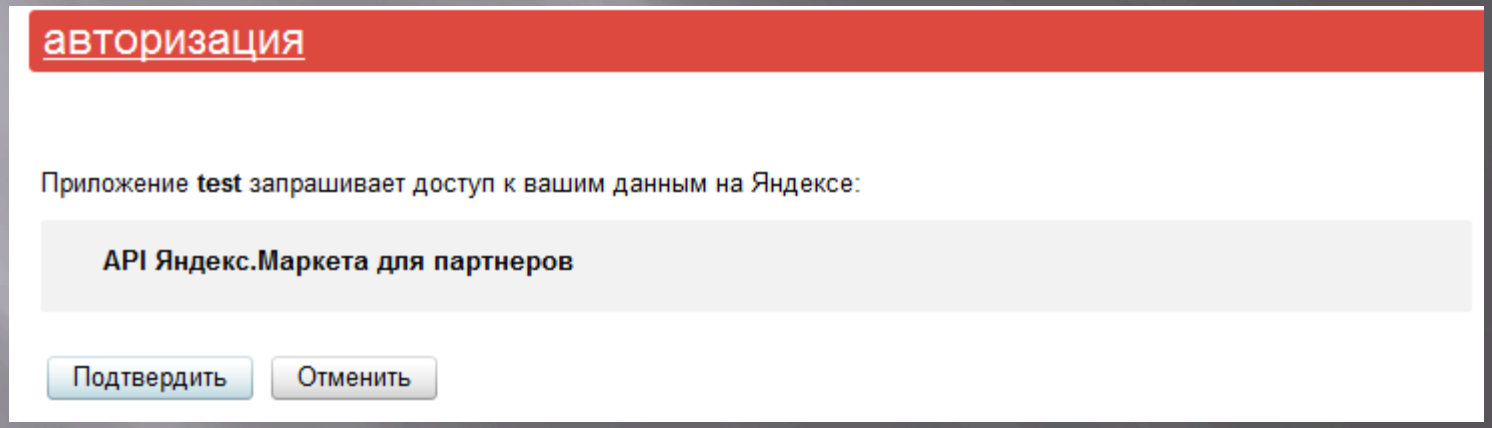
Настройки авторизации для вызовов из магазина в Маркет

- Получение токена:
  - Регистрируем приложение на http://oauth.yandex.ru
  - В Callback URI устанавливаем: https://oauth.yandex.ru/verification_code?dev=true



Настройки авторизации для вызовов из магазина в Маркет

- Получение токена:
  - https://oauth.yandex.ru/authorize?response_type=token&client_id=<идентификатор приложения>



  - Копируем токен доступа из URL-а

Настройки авторизации для вызовов из магазина в Маркет

- Использование токена:

```
var request = (HttpWebRequest)WebRequest.Create(url);
request.Method = "PUT";
request.ContentType = "application/json";
request.ContentLength = json.Length;

string oauth_token = ...;
string oauth_client_id = ...;
string oauth_login = ...;
request.Headers[HttpRequestHeader.Authorization] = string.Format("OAuth oauth_token={0}, oauth_client_id={1}, oauth_login={2}",
    oauth_token,
    oauth_client_id,
    oauth_login);

var data = Encoding.UTF8.GetBytes(json);
var dataStream = request.GetRequestStream();
dataStream.Write(data, 0, data.Length);
dataStream.Close();

var response = (HttpWebResponse)request.GetResponse();
string result = string.Empty;
using (var sr = new StreamReader(response.GetResponseStream(), Encoding.UTF8))
{
    result = sr.ReadToEnd();
}
```

Ссылки и материалы

- http://habrahabr.ru/company/yandex/blog/ - блог Яндекса на хабре
- http://partner.market.yandex.ru/ - партнерский интерфейс Маркета
- http://partner.market.yandex.ru/legal/tt/ - описание формата YML
- http://help.yandex.ru/partnermarket/purchase/about.xml - общее описание программы Покупка на Маркете
- http://api.yandex.ru/market/partner/doc/dg/reference/purchase-methods.xml - описание API Покупки на Маркете
- https://oauth.yandex.ru – регистрация приложений для использования API служб Яндекса