

# How to create 15 complex e-commerce projects in 8 months and save your mental health

Chapter 1. Sites development

# Who I am

My name is Dmitry, I'm a fullstack developer

- With MODX since 2013 (Evo/Revo)
- I have created over 100 websites with MODX in last 10 years
- I have experience with MODX, Wordpress, Joomla, Bitrix, Pure PHP, JS, jQuery, Vue & React
- I was a team-leader in the biggest Russian IT corporation "LANIT" with my own team in "diHouse" – one of the biggest distributors in Russia as a subcompany of "Lanit"
- Now I'm a React developer in "Orion Inc."

# Disclaimer

- This is quite a simple story, and it might be useless for real pro developers. But as I know a lot of developers still avoid some rules and principles and today I'm gonna show you how you can manage your development process to make it simpler and faster.
- This story isn't about PHP at all. This story is between managing, business and technology.

# What it is about

- diHouse is a distribution company with a lot of partners around the world. Our biggest clients were such companies as Apple (we were the main Apple distributor in Russia before the war), Nokia, Nvidia, Ecovacs Robotics, XIAOMI, SONY, and a lot more.
- Several years ago our company decided to present itself on retail market besides the distribution
- Our first clients in that direction were Nokia, Nvidia, Ecovacs Robotics and some Russian companies, like Z-Store (personal brand of the popular Russian rapper BASTA)

# What we had to develop

- Product catalog, synced with company ERP to check stocks and prices (inner API)
- Public API for our customers to get sale statistics
- A powerful basket with promo-codes and other discount activities
- Different types of shops – public, only basket, only for company employees
- Loyalty program
- CRM integration, different delivery systems, card payment

# And we did it

- I can't show these projects, because they were closed when the war began
- But I have some pictures , taken from WebArchive
- You also can check it by following this link
- <https://web.archive.org/web/20220402041747/https://mobileshop.nokia.ru/>
- And it was made with MODX



Смартфоны и планшеты



Телефоны



Аудио



Аксессуары



Комплекты



Рассрочка



Гарантия на телефоны



 Защищен от попадания влаги и песка

 Прочный экран из закаленного стекла Corning® Gorilla® Glass Victus™

 Три года обновлений ОС

## Nokia XR20

Ваш новый смартфон с вами надолго

Компактная беспроводная колонка Nokia SP101 Black в ПОДАРОК

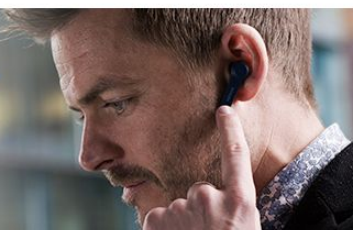
Оформить >

### Nokia BH-805

Стильные наушники с качественным шумоподавлением

5 990 Р

Оформить >



### Nokia 6310

Возвращение легенды

Оформить >

Надежные телефоны Nokia

Новые аудио аксессуары

Подробнее >



# The winter is coming

- In 2022, from February till April, all our retail customers left
- Our company made a new contracts with a lot of Chinese brands in 2021 and wanted to create new sites on our platform
- Our CTO decided to switch from MODX to Bitrix for these new projects (because his last job was about “1C” – the developer of Bitrix)
- But our retail department was so in love in our MODX, and I decided to keep the idea and move it into Bitrix



# A new beginning

- We should create 15 new e-commerce projects from February to December 2022.
- All previous functions should be saved
- A new API should have created, because the old one had some performance issues
- And all of it with only 2 developers.
- Great!

# The main conditions

- We should have launched at least 1 shop per month, 2 – is optimal
- This project should be easy to develop and maintain by new developers or outsourcing
- Easy from-site-to-site switching process. A developer should know everything about this site even it his first time on this particular site. This is how we can develop it fast
- Our timings: 1 week – frontend development, 1-2 weeks CMS integration, 1 day – launch
- We should keep our project as much simple as we can. Only KISS, only hardcore.

# What we did and how we did it

- We have delegated frontend development process to outsourcing company with few rules:
  - Just HTML + JS. No jQuery, no Vue, no React. A lot of developers still can't use react-like frameworks and we don't have time to teach them. Also, Vue and React requires a specific API to work, but MODX and Bitrix don't have this specific functions – only old-fashion way. Also, I don't want to spend my time to configure SSR, because we have to have a good SEO, but it's too hard with React or Vue.
  - Simple frontend. Minimum animations, circles, 3D, canvas, etc.
  - Every site should have the same HTML classes, the same structure (as much as possible), and the same file structure
  - BEM methodology
  - Gulp as a bundler. Webpack is too complicated and it's hard to support.
  - Regular code-review from our side
  - GIT (yeah, it's still a big problem for a lot of companies)

# Backend rules

- Files elements only. Fenom for MODX, TWIG for Bitrix.
- GIT (we have our own Gitlab server). I will show you our CI/CD in next slides
- The same file structure from site to site. Easy to support, easy to learn, easy to develop – you don't have to think how you should call every new chunk, snippet or plugin.
- We should try to use only simple packages and modules from marketplaces. PDOTools for MODX, D7 for Bitrix. MiniShop2 for MODX, Bitrix has his own integrated webshop module already.

# After we did a first project with these rules, all we had left to do was:

- Create new context (and named folder inside core/elements/)
- Copy new frontend and put Fenom/TWIG placeholders and snippets
- Some little different improvements from site to site (different menu, different filters, etc)
- Fill the site with some content
- Run
  
- Yeah, that's it. I'm not kidding.

# Just imagine that

- It's hard to believe, but when we have optimized our approach, we launched our last 5 sites just spending for 3 days on each.
- Let's take a look a little bit closer

# The trick

- As I said before, we have decided to follow some rules:
  - The same structure of each particular site
  - The same frontend layout (HEADER – HERO – BREADCRUMBS – CONTENT - FOOTER)
  - The same BEM-methodology and the same class names
  
- Now let's check our sites



# КОМПАКТНЫЙ СКЛАДНОЙ ЭЛЕКТРОВЕЛОСИПЕД ADO A16

ADO A16 – универсальный байк с облегченной конструкцией обеспечивает высокую мобильность и скорость передвижения.

- Запас хода на электротяге – до 35 км, в смешанном режиме – до 70 км.
- Диаметр колес 16 дюймов.
- 7-ступенчатая трансмиссия Shimano: быстрое переключение передач, полный контроль над движениями.
- Полная зарядка аккумулятора занимает всего 5-6 часов.
- Вес – 20 кг, складная алюминиевая рама, яркая LED-фара, удобная приборная панель.



**62 990 ₪**

Добавить  
к сравнению

Нет в наличии



[Главная](#) | 
 [Каталог товаров](#) | 
 Комплект из заварочного чайника и двух чашек Kiss Kiss Fish Boogie Woogie Tearpot (зеленый)

Артикул: TEAP08-U

# Комплект из заварочного чайника и двух чашек Kiss Kiss Fish Boogie Woogie Tearpot (зеленый)

- Лаконичный и эстетичный дизайн.
- Удобная конструкция с заварочным ситечком.
- Объемный и легкий – 800 мл.
- Создан из боросиликатного стекла высокой прочности.
- Две чашки в комплекте.

**Цвет:**



**Количество:**



## Botslab Outdoor Dual-lens Camera W302

Два объектива для панорамного обзора.

- Разрешение: 4MP 2.5K.
- Панорамный обзор на 170° благодаря двум объективам.
- Поддержка двух диапазонов Wi-Fi: 2,4 и 5 ГГц.
- Двусторонняя голосовая связь.
- Защита от атмосферных воздействий класса IP66.
- Инфракрасный и полноцветный режим ночного видения.
- Голосовое управление через Алису.

**Количество:**



~~10 766 Р~~

7 690 Р

В КОРЗИНУ



[Главная](#) | [Каталог товаров](#) | [Электромельница HuoHou Electric Pepper & Salt Grinder \(Белая\)](#)

Артикул: HU0142

## Электромельница HuoHou Electric Pepper & Salt Grinder (Белая)

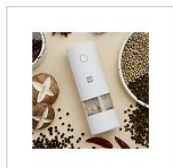
- Работает от 6 батареек AAA.

Цвет:



1 390 ₺

Нет в наличии





# Smartmi Evaporative Humidifier 2

Увлажняет и освежает воздух в доме, задерживает рост и размножение бактерий.

- Удобное управление и отображение параметров на дисплее.
- Здоровое увлажнение: подавляет рост бактерий, очищает воздух от пыли, аллергенов и шерсти животных.
- Равномерное увлажнение воздуха в доме без образования конденсата.
- Работает бесшумно, тратит минимум энергии.
- Качество и скорость увлажнения — 260 мл/ч.
- Он оснащен резервуаром для воды емкостью 4 литра, который обеспечивает до 15 часов работы в максимальном режиме.

Количество:



~~15 587 ₽~~

11 990 ₽

В корзину





ГЛАВНАЯ → КАТАЛОГ ТОВАРОВ → РОБОТЫ-ПЫЛЕСОСЫ → РОБОТ-ПЫЛЕСОС ECOVACS DEEBOT OZMO T8 AIVI



## Робот-пылесос ECOVACS DEEBOT OZMO T8 AIVI

Артикул DBX11-11

Инновационный робот-пылесос с искусственным интеллектом.

DEEBOT OZMO T8 AIVI с передовой технологией искусственного интеллекта поднимает уборку на новый уровень. Это идеальный помощник, который одновременно моет и пылесосит, строит подробные карты помещений и прокладывает эффективные маршруты с обходом препятствий. Пропуск или повторение участков исключены. Благодаря технологии TrueMapping и лазерному датчику dToF робот-пылесос не запутается в проводах, не повредит мебель и не утащит за собой мелкие предметы. DEEBOT OZMO T8 AIVI самостоятельно определяет типы поверхностей, корректирует режим уборки и может автоматически выгружать мусор.

**59990 Р**[Купить](#)[Добавить к сравнению](#)[ФУНКЦИИ](#)[ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ](#)[КОМПЛЕКТАЦИЯ](#)

### Видеотрансляция в реальном времени

Потоковое видео по запросу — это новая революционная функция. Она позволяет

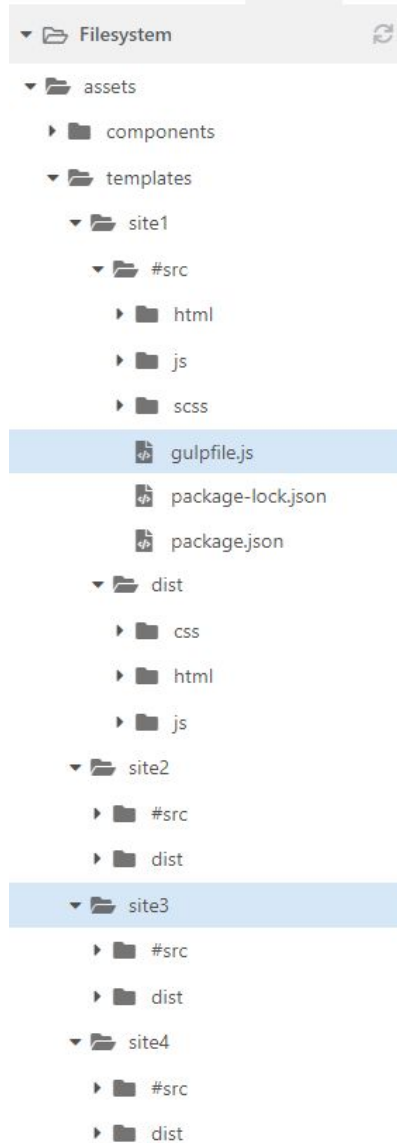
# The trick

- We put some rules for our designers – just keep it the same structure as much as you can. You can change fonts, sizes, images, icons, paddings, margins, colors, but please – stay in the structure.
- Yeah, it looks very identical, but if only you have visited all of them. In the real world it's impossible. No one will remember some familiar site back in time.
- ALL SITES ARE FAMILIAR. ESPECIALLY E-COMMERCE.

# The trick

- And because of that our project became simple and powerful in the same time. We can develop & support it extremely fast and this is what our company expects from us.
- Let's take a look at the code

# Each frontend files stores in templates/context\_key



Название файла: gulpfile.js

Путь: assets/templates/site1/#src/gulpfile.js

Размер файла: 10771 B

Последнее обращение: 2024-04-17 21:12

Изменён: 2024-04-17 21:12

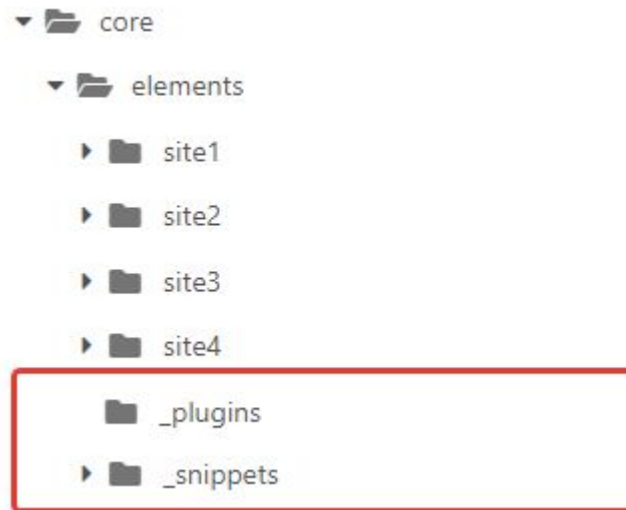
```
export default gulp.series(  
  clean,  
  gulp.parallel(twigBuild, scss, copy, script, sprite),  
  gulp.parallel(serve)  
)  
  
/*  
 * Declare helpers functions  
 */  
  
const getFolders = (base, folders) => {  
  return folders.map((item) => {  
    return path.join(base, item);  
  });  
}  
  
const getFoldersSrc = (base, folders) => gulp.src(  
  folders.map((item) => path.join(base, item)), {  
    base: base,  
    allowEmpty: true  
  })  
})
```







# Each backend things stores in core/elements/context\_key



Common plugins and snippets are stored in core/elements/



Also we have a special component called Dihouse with some special methods, utils and plugins code

-  model
  -  pluginshandler
  -  processors/dihouse
  -  utils
-

# This is how our plugins work

I saw this trick on MODX.pro someday

- events
- pluginhandler.class.php
- dihousemultipluginhandler.class.php
  - cartpluginhandler.class.php
  - initpluginhandler.class.php
  - productpluginhandler.class.php
  - userspluginhandler.class.php

```
/**
 * Class dihousekInitPluginHandler
 */
class dihouseInitPluginHandler extends dihouseMultiPluginHandler
{

    public function OnMODXInit()
    {
        $this->modx->loadClass('msOrder');
        $this->modx->map['msOrder']['fields']['seller_id'] = 0;
        $this->modx->map['msOrder']['fieldMeta']['seller_id'] = array(
            'dbtype' => 'int',
            'precision' => 10,
            'attributes' => 'unsigned',
            'phptype' => 'integer',
            'null' => true,
            'default' => 0,
        );
        return true;
    }
}
```

# The MAIN trick

- If we have the same structure, the same code, the same principles – we don't have to think about the code. Just COPY/PASTE!
- And this is how we finished it.
- Let's take a look on the Template

```

<div class="product-full-info__preview">
  <div class="product-preview">
    <div class="container product-preview__body">
      //FENOM EXAMPLE
      <div class="product-preview__col-1">
        <div class="product-preview__breadcrumb">
          <ul class="breadcrumb">
            <li class="breadcrumb__item"> <a class="breadcrumb__link" href="/">MAIN</a></li>
            <li class="breadcrumb__item"> <a class="breadcrumb__link" href="/catalog/">Products</a></li>
            <li class="breadcrumb__item">{<name}</li>
          </ul>
        </div>

        <div class="product-preview__slider">
          <div class="product-preview__slider-thumbs swiper" data-slider="product-preview-slider-thumbs">
            <div class="swiper-wrapper">
              {<foreach result['PROPERTIES']['DETAIL_GALLERY']['THUMB'] as gallery_thumb_img>
                <div class="swiper-slide product-preview__slider-thumbs-item">
                  }">
                </div>
              </foreach>
            </div>
            <div class="product-preview__slider-thumbs-btn" data-action="slider-next"><i class="icon-chevron-down"></i></div>
          </div>
          <div class="product-preview__slider-images swiper" data-slider="product-preview-slider-images" data-zoom-container>
            //TWIG EXAMPLE
            {<% if result['PROPERTIES']['PROMO_STATUS']['VALUE'] %>
            <div class="product-preview__slider-images-label">
              {<% for status_key, status in result['PROPERTIES']['PROMO_STATUS']['VALUE'] %>
                <div class="product-label product-label--{{ result['PROPERTIES']['PROMO_STATUS']['VALUE_XML_ID'][status_key] }}">{{ status }}</div>
              <% endfor %>
            </div>
            {<% endif %>
            <div data-zoom-zone></div>
            <div class="swiper-wrapper">
              {<% if result['DISPLAY_PROPERTIES']['DETAIL_GALLERY']['FILE_VALUE']['SRC'] %>
                {<% set only_one_img = result['DISPLAY_PROPERTIES']['DETAIL_GALLERY']['FILE_VALUE']['SRC'] %>
                <div class="swiper-slide product-preview__slider-images-item">
                  }" alt="{<{{ result['NAME'] }}>}">
                </div>
              <% else %>
                {<% for gallery_img in result['DISPLAY_PROPERTIES']['DETAIL_GALLERY']['FILE_VALUE'] %>
                  <div class="swiper-slide product-preview__slider-images-item">
                    }" alt="{<{{ result['NAME'] }}>}">
                  </div>
                <% endfor %>
              <% endif %>
            </div>
            <div class="product-preview__slider-images-dots">
              <div class="slider-dots" data-slider-dots></div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

```

Just put another HTML and change the placeholders!

# What about the rest of the code?

- Nothing special, actually. Just PDOTools with Fenom, some custom snippets & plugins and MiniShop2.

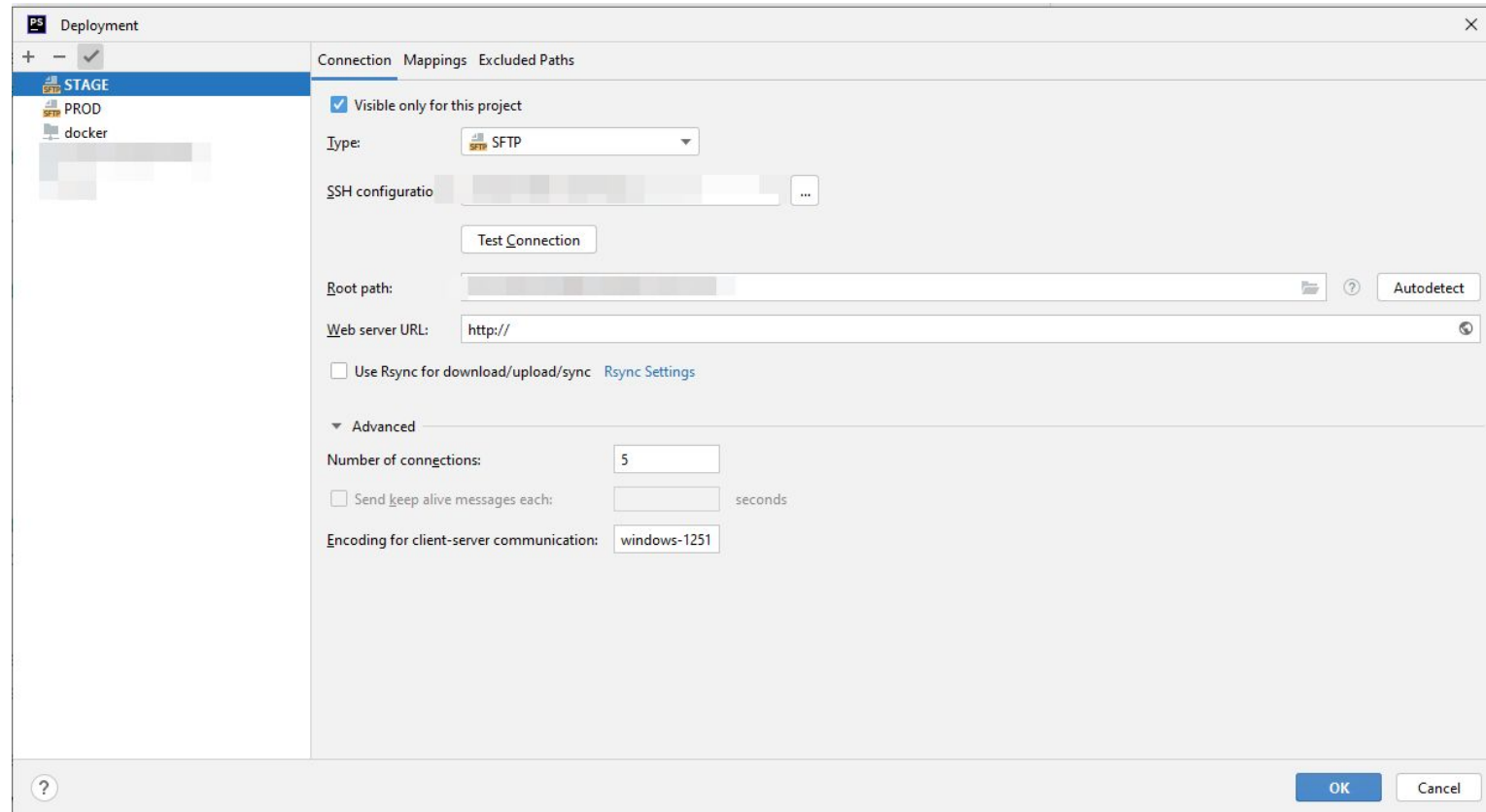
# CI/CD

- We had our Gitlab server, but no one in the whole company knew how to set up the CI/CD
  - I tried, but no one gave me ROOT to the server.
  - No root – no SSL.
  - NO CI/CD.
- 
- You can start laughing, it's really funny, I know.



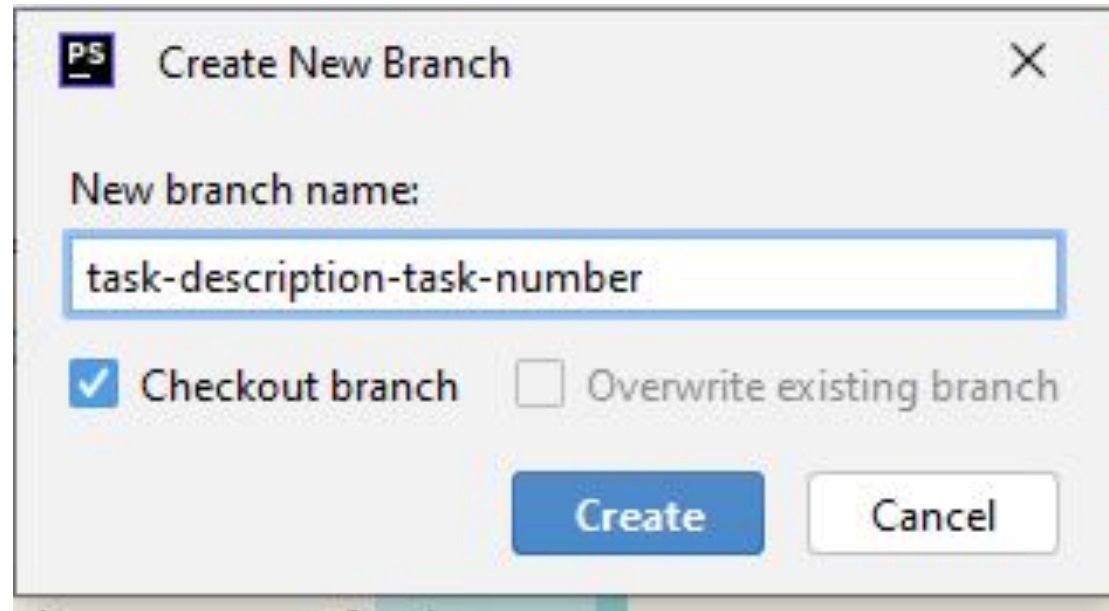
# God save JetBrains

- You can create a lot FTP/SFTP/Local connections in PhpStorm
- We had DEV, STAGE and PROD. PROD is only for me, the others for the second developer



# God save JetBrains

- Every new task – is a new branch. Master is protected.



# God save JetBrains

Push branch on GitLab

Create a PR

Merge

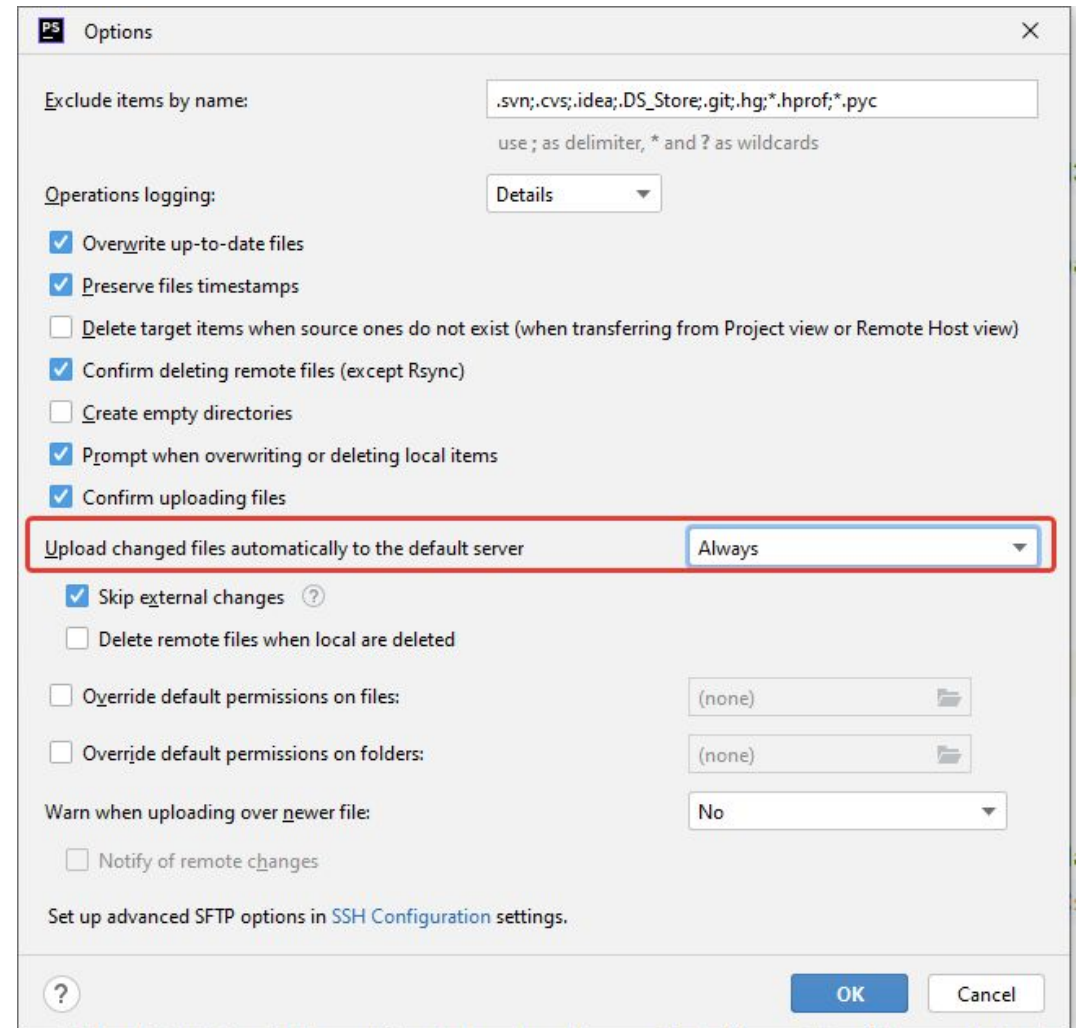
Switch back to master in IDE

Set PROD as default Deployment server (2 slides back)

Pull

All changes will be uploaded on the PROD

Done! You are great!



# Conclusion

- Being a programmer is not only about THE CODE.
- Good planning and architecture can save your mental health and probably the job
- Limitations are good. Don't be shy to put anyone into frames if you know why you are doing this
- MODX is still awesome. Can you do it with some other CMS?

# Chapter 2. API development

# The task

- As I said before, we have to develop new API to transfer the data
  - Product items, Categories, Prices, Stocks, Users, Pictures, Product specifications, Orders and Orders statuses
- Company stock is about 10 000 items. The system should process it fast.
- Our sites are only for clients. No actions with products in /Manager, never.

# Limitations

- No existing solutions. We should have created it from scratch
- No existing API. We have to create it.
- Our ERP-system does not support the REST.
- No PUT/GET/DELETE. Only POST.
- Our CTO wanted to use KAFKA so bad

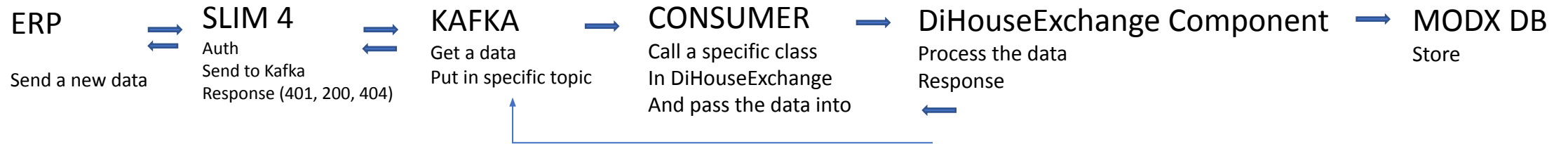
# The stack

- Slim 4
- Kafka
- MySQL
- Our PHP component DiHouseExchange to process the receiving data
- MODX / Bitrix
- PHP Unit



# The scheme

## When ERP starting the exchange



## When MODX starting the exchange



## MySQL

Keep all requests for logs  
For 1 week, then delete it  
(really useful thing)

# Slim 4

When ERP is sending a data (new products, new stocks, new prices)

- Receive the data
- Check auth
- Put data into Kafka
- Send response (401, 200, 404)

When our site is sending a data (new order, new user, new order status)

- Get the data from Kafka
- Create a request
- Send the data to ERP
- Receive the response

# Kafka

UI for Apache Kafka v0.4.0 (521ba0c)

Dashboard

local +

Brokers

Topics

Consumers

## All Topics

+ Add a Topic

Search by Topic Name   Show Internal Topics

Topic Name	Total Partitions	Out of sync replicas	Replication Factor	Number of messages	Size
<input type="checkbox"/> INCOMING	1	0	1	2	46KB
<input type="checkbox"/> RETRY	1	0	1	0	0Bytes
<input checked="" type="checkbox"/> IN__consumer_offsets	50	0	1	21	1KB
<input type="checkbox"/> answers	1	0	1	0	82KB
<input type="checkbox"/> client	1	0	1	1	1KB
<input type="checkbox"/> redkin	1	0	1	0	0Bytes
<input type="checkbox"/> responseFromTC	1	0	1	0	0Bytes
<input type="checkbox"/> test	1	0	1	0	0Bytes
<input type="checkbox"/> test_topic	1	0	1	0	0Bytes

Previous Next

- INCOMING topic for all incoming requests. Then it will be transferred to the specific topic
- Each route in API = single topic in Kafka
- Every topic in Kafka = single consumer script in PHP
- For errors we had RETRY topic with `retry_count` param.
- If `retry_count > 5` – send a warning to developer
- Answers topic for responses

# DiHouseExchange component

- Main class with base methods (request/response/process)
- Each entity has a specific parent class to process the data in CRUD style
  - Products
    - Create Product
    - Update Product
    - Delete Product
    - Get Product
  - Order
    - Create Order
    - Update Order
    - Get Order
  - etc

## The main class

```
1 <?php
2
3 namespace Api {
4
5     use Api\Log;
6     use Api>Main;
7     use Api\Config;
8     use Api\Iblocks;
9     use Api\KafkaProducer;
10
11     class ApiHandler
12     {
13         public Config $config;
14         public Iblocks $iblock;
15         public ResponseHandler $responseHandler;
16         public FieldsMap $mapper;
17         public array $last_query;
18         public string $last_error;
19         public KafkaProducer $producer;
20         public Log $logger;
21         public ?string $last_api_status;
22
23
24         public function __construct(Config $cfg)
25         {
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53         /**
54          * process all input data, search method & execute, return status
55          * @param array $data
56          * @return bool
57          */
58         public function process($data): bool
59         {
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101         /**
102          * call response handler & send result message to Kafka
103          * @param $data - data processing result
104          * @param $has_errors
105          * @return bool
106          */
107         public function response($data, $has_errors): bool
108         {
109
110
111
112
113
114
115
116
117         protected function success(array $data = null): array
118         {
119
120
121
122
123
124
125         protected function error(): array
126         {
127
128
129
130
131
132
133         protected function setResult(bool $result, array $data = null): array
134         {
135             return $result ? $this->success($data) : $this->error();
136         }
137
138     }
139 }
```



# A Consumer

To make it stable all consumers are running from  
Supervisord – daemon service to keep consumers alive

```
1 #!/usr/bin/php
2 <?php
3 define("NO_KEEP_STATISTIC", true);
4 define("NOT_CHECK_PERMISSIONS", true);
5 define("BX_CAT_CRON", true);
6 define('NO_AGENT_CHECK', true);
7
8 require($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/main/include/prolog_before.php");
9 \Bitrix\Main\Loader::includeModule('api');
10
11 $config = new \Api\Config([]);
12
13 $broker = $config->options['KAFKA_CONNECTION'];
14 $topic_name = $config->options['VENDORS_TOPIC'];
15
16 if ($broker == '' && $topic_name == ''){
17     die('P&Pp C&P&P°P°P°P&C< PIP°CBP°PjPpC,CBC< PIPsPrPeP»C&C+PpP&P&C!!'.PHP_EOL);
18 }
19
20 $conf = new RdKafka\Conf();
21
22 $conf->set('group.id', 'group_id');
23
24 $rk = new RdKafka\Consumer($conf);
25 $rk->addBrokers($broker);
26
27 $topicConf = new RdKafka\TopicConf();
28 $topicConf->set('auto.commit.interval.ms', 100);
29
30 $topicConf->set('auto.offset.reset', 'smallest');
31
32 $topic = $rk->newTopic($topic_name, $topicConf);
33 $topic->consumeStart(0, RD_KAFKA_OFFSET_STORED);
34
35 while (true) {
36     $message = $topic->consume(0, 120 * 10000);
37     switch ($message->err) {
38         case RD_KAFKA_RESP_ERR_NO_ERROR:
39             if (isset($message->payload) && $message->payload != '') {
40                 $data = json_decode($message->payload, 1);
41                 $site_id = $config->getSiteByShopId($data['shopId']);
42                 $config->set('IBLOCK_ID', $config->options[$site_id . '__VENDORS_IBLOCK_ID']);
43                 $vendors = Api\Main::$container->make('Api\Vendors', [
44                     'cfg' => $config
45                 ]);
46                 $vendors->process($data);
47                 $topic->offsetStore(0, $message->offset);
48                 unset($vendors);
49             }
50             break;
51         case RD_KAFKA_RESP_ERR_PARTITION_EOF:
52             echo "No more messages; will wait for more\n";
53             break;
54         case RD_KAFKA_RESP_ERR_TIMED_OUT:
55             echo "Timed out\n";
56             break;
57         default:
58             throw new \Exception($message->errstr(), $message->err);
59             break;
60     }
61 }
```

# The API

- We took OpenAPI and made it simpler for our purposes
- Our API was stored in POSTMAN to collaborate between PHP-developers and our ERP-developers
- BaseAuth (1C still cannot use JWT in 2024)
- Async



Sorry for Russian labels, but this is all I have found

In this particular example we are transferring the order from our site to our ERP.

Please, pay your attention – we can update created order  
And also we can split the payment – total amount,  
total discounts, how many Money was spent  
with Loyalty Program, and what was the delivery price.

No one should go to the Manager to check the data.  
All data is already sent to ERP and CRM

```
2  "collection": "orders",
3  "method": "update",
4  "version": 1,
5  "apiKey": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9",
6  "shopId": "adoebike",
7  "requestId": "285b01cd-61e5-45cd-960e-879f72979416",
8  "info": {
9    "user": {
10     "id": "1",
11     "fio": "Администратор Сайта",
12     "phone": "7937777777",
13     "email": "dev-it@di-house.ru"
14   },
15   "delivery": {
16     "type": "Мексиканским курьером",
17     "address": "Пэ шарман долителейн сидней",
18     "price": 500
19   },
20   "payment": [
21     {
22       "type": "Наличный расчет",
23       "price": 155000
24     },
25     {
26       "type": "Бонусный счет",
27       "price": 480
28     }
29   ],
30   "order": {
31     "number": "ADO-145",
32     "guid": "bx_6319bbe441722",
33     "status": "N",
34     "amount": 155480,
35     "create_date": "08.09.2022 12:54:44"
36   },
37   "products": [
38     {
39       "quid": "4b142e1f-8651-41ab-b30e-de7e30fcad55",
40       "quantity": 1,
41       "price": 73990,
42       "discount": 1000,
43       "warehouse": "4b142e1f-8651-41ab-b30e-de7e30fsklad2"
44     },
45     {
46       "quid": "4b142e1f-8651-41ab-b30e-de7e30fcad66",
47       "quantity": 1,
48       "price": 81990,
49       "discount": 0,
50       "warehouse": "4b142e1f-8651-41ab-b30e-de7e30fsklad1"
51     }
52   ]
53 }
54 }
```

# Testing

- The whole system was covered with tests on PHPUnit
- We put tests on each route, each class, each method
- For API testing we have used PHPUnit with some kind of JSON generator, based on FakerPHP (really awesome thing)

## An test example

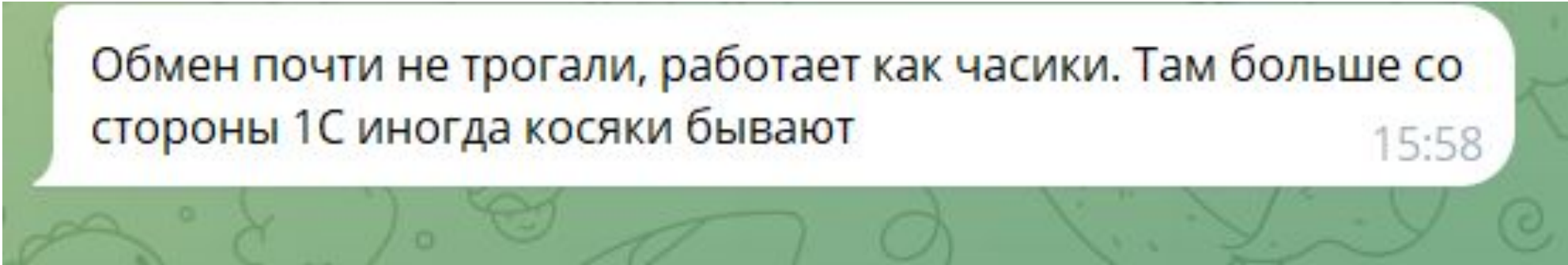
```
1 <?php
2
3 use Bitrix\Main\Loader;
4 use Api\Main;
5 use Api\Tools;
6 use PHPUnit\Framework\TestCase;
7
8 final class VendorsClassTest extends TestCase
9 {
10     public Api\Vendors $vendor;
11     public Api\FieldsMap $mapper;
12
13     protected function setUp(): void
14     {
15         Loader::includeModule('api');
16
17         $shop_id = 'shop_test';
18
19         $config = Main::$container->get('Api\Config');
20
21         $this->mapper = Main::$container->get('Api\FieldsMap');
22
23         $site_id = $config->getSiteByShopId($shop_id);
24         $config->set('IBLOCK_ID', $config->options[$site_id . '__VENDORS_IBLOCK_ID']);
25
26         $this->vendor = Main::$container->make('Api\Vendors', [
27             'cfg' => $config
28         ]);
29     }
30
31     public function testCanCreateVendorElement()
32     {
33
34         $vendor_item = [
35             "NAME" => $this->randString(),
36             "XML_ID" => $this->randGuid(),
37         ];
38
39         $res = $this->vendor->create($vendor_item);
40         $this->assertTrue($res['status']);
41     }
42
43     public function testCantCreateVendorElement()
44     {
45         $vendor_item = [
46             "NAME" => $this->randString(),
47             "XML_ID" => $this->faker->randomNumber(),
48         ];
49
50         $res = $this->vendor->create($vendor_item);
51         $this->assertFalse($res['status']);
52     }
53
54     protected function tearDown(): void
55     {
56         if (isset($this->vendor->iblock->element_id) && is_int($this->vendor->iblock->element_id)) {
57             $this->vendor->iblock->deleteById($this->vendor->iblock->element_id);
58         }
59     }
60 }
```

# What about the performance?

- We were limited by default processors. Yeah, we could just put the data in DB and it would be faster, but on the other side a lot of problems could be discovered. I don't want to make my head ache.
- So, our DiHouseExchange Component is using standard MODX processors like object/create, object/update, etc.
- 500 product items with full bunch of data except the pictures were created just in 5 minutes.
- The full launch was made after me, but, as I know, everything is fine.

# After 14 months after I left the company it's still working fine

And this is why I can say we've done it well.

A screenshot of a text message on a green background with faint white doodle patterns. The message is contained in a white rounded rectangle with a drop shadow. The text is in Russian and reads: "Обмен почти не трогали, работает как часики. Там больше со стороны 1С иногда косяки бывают". The time "15:58" is displayed in the bottom right corner of the message bubble.

Обмен почти не трогали, работает как часики. Там больше со стороны 1С иногда косяки бывают

15:58

# Thank you!

My Telegram: @zahod5277

My Instagram: zahod5277

My E-mail: [zahod5277@mail.ru](mailto:zahod5277@mail.ru)

LinkedIn: [www.linkedin.com/in/zahod5277](http://www.linkedin.com/in/zahod5277)