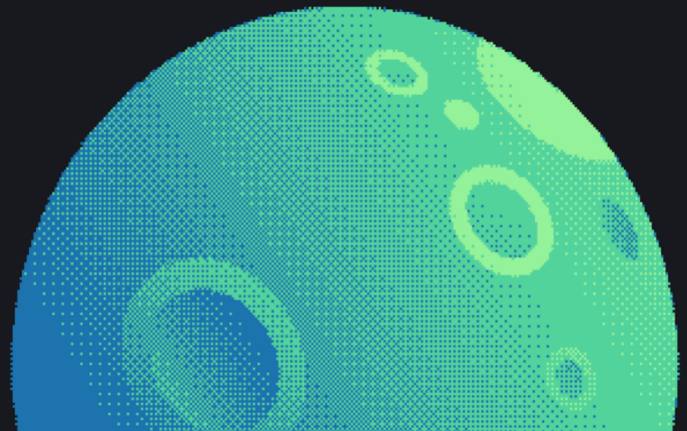
DODO ENGINEERING

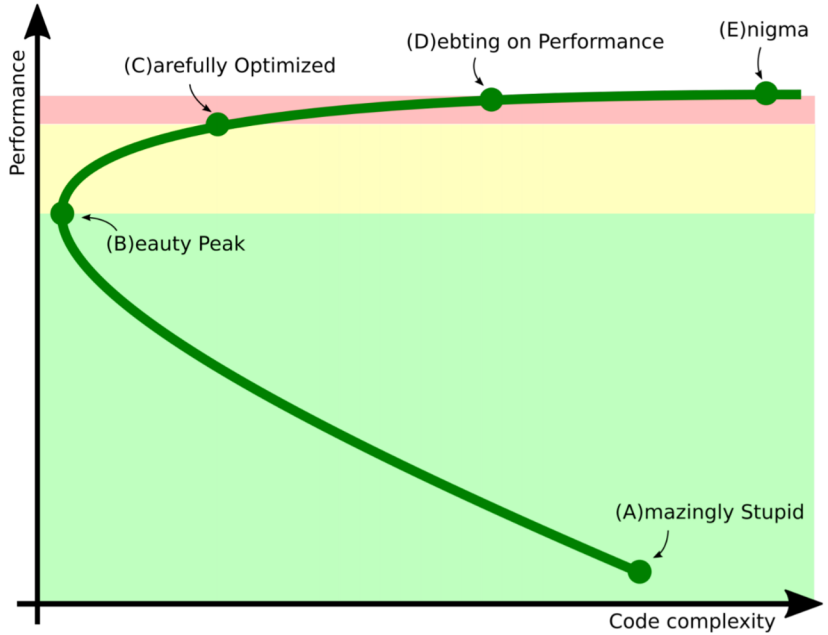
Антон Оникийчук(@holyslon)
Андрей Парамонов(@PrivetAndrey)

Вы кеши продаёте? Показываем!



Про перформанс, но без байтиков (почти)

Крупно: Кривая им. Ш:



Slide 8/66. «Keynote: Performance», Aleksey Shipilëv, 2017, D:20170404035127+02'00'

redhat.

Алексей Шипилёв — Перформанс: Что В Имени Тебе Моём? (https://shipilev.net/#perf-keynote)

Dodo Brands – это про IT

Dodo Brands – компания,
которая развивает три бренда:
Додо Пицца, Дринкит
и Донер 42, на базе единой
инфраструктуры и самописной
цифровой платформы Dodo IS.
Dodo Engineering – команда
разработки.

15 стран мира

26 млрд выручки
за 2020

700+ точек питания

16 млн клиентов

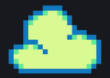
600+ человек
в Dodo Brands

Dodo IS

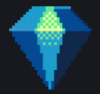
3000 запросов в секунду

200–250 заказов в минуту

470 – максимальная нагрузка заказов в минуту

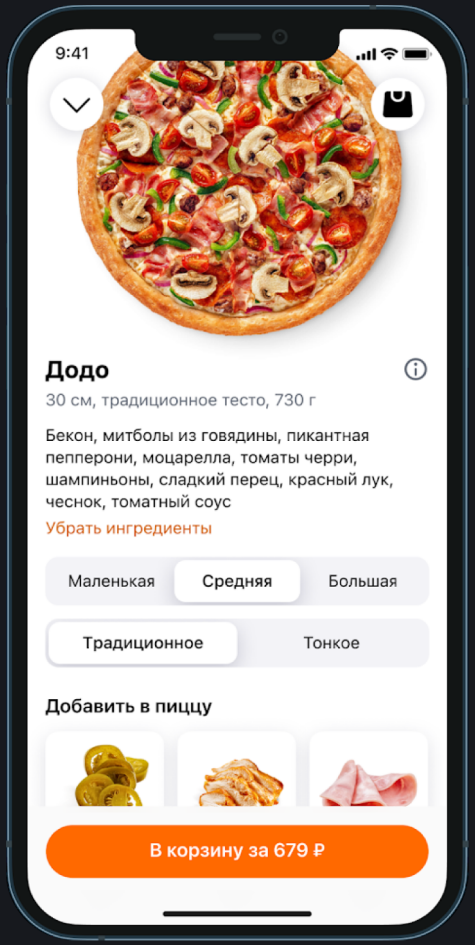


2 датацентра



40+ сервисов

160+ человек в Dodo Engineering



Мобильное приложение

– принимает 40% заказов

scan ME

MAPI — это BFF

Menu/Get

Запрашивается при старте приложения

Запрос 1 меню к MAPI приводит к N запросам в downstream сервисам

Response:

- 1 Mb JSON
- 40k строк
- 9 уровней вложенности

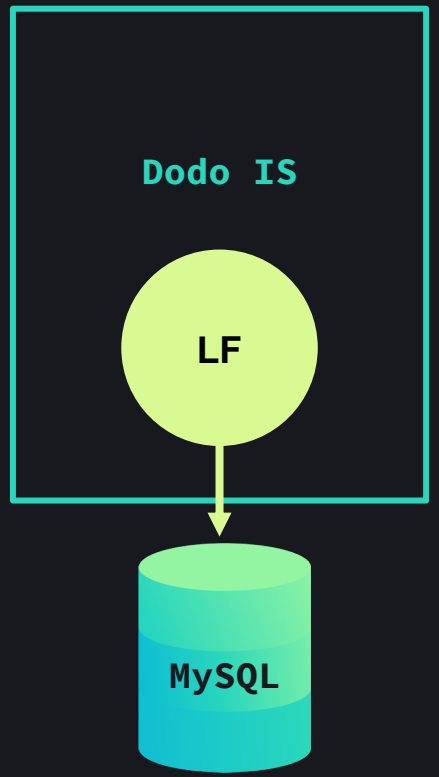
DODO IS 2019

MySQL на страну (МОНОЛИТ)

DODO IS 2019

MySQL на страну (МОНОЛИТ)

LF + In Memory Cache
на доменную сущность

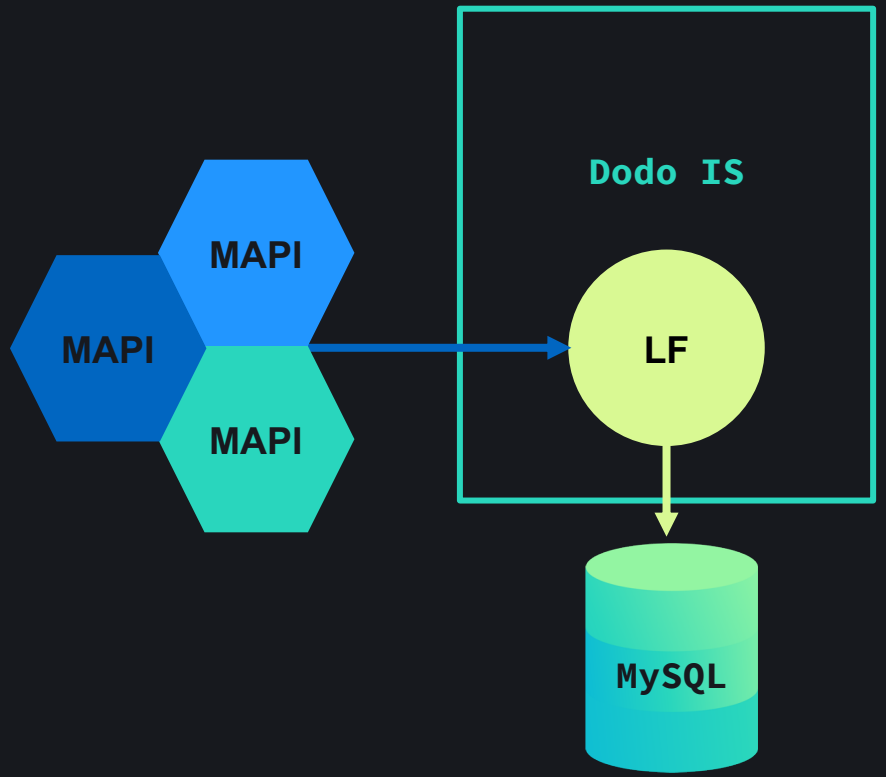


DODO IS 2019

MySQL на страну (МОНОЛИТ)

LF + In Memory Cache
на доменную сущность

MAPI + In Memory Cache
на доменную сущность



Проблема актуальности данных

1 час LF

2 часа

1 час МАПИ

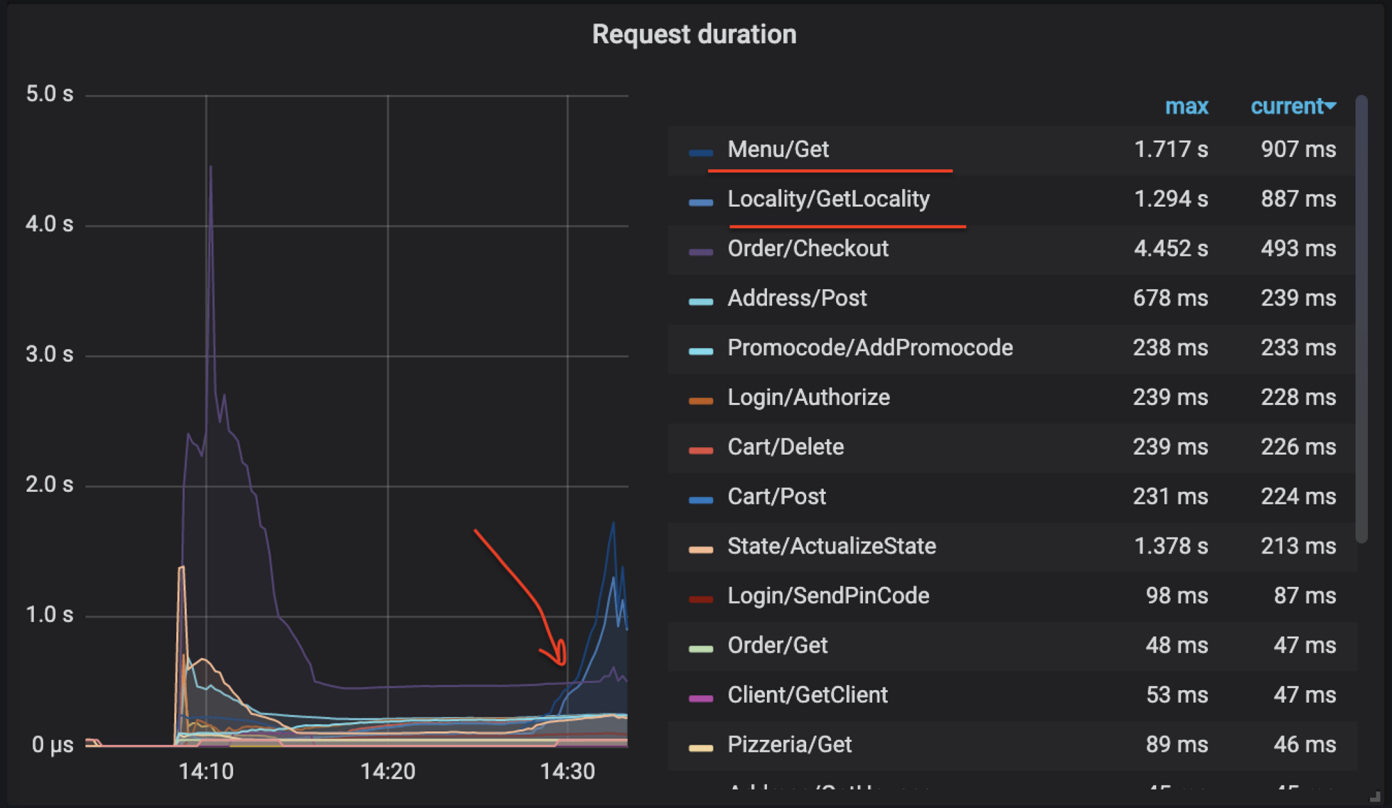
Меню МАПИ

Работает

Задержка при обновлении
2 часа

Целостно в любой момент времени

Проблема


Request duration
5.0 s
4.0 s
3.0 s
2.0 s
1.0 s
0 µs
14:10
14:20
14:30
max
current
Menu/Get 1.717 s 907 ms
Locality/GetLocality 1.294 s 887 ms
Order/Checkout 4.452 s 493 ms
Address/Post 678 ms 239 ms
Promocode/AddPromocode 238 ms 233 ms
Login/Authorize 239 ms 228 ms
Cart/Delete 239 ms 226 ms
Cart/Post 231 ms 224 ms
State/ActualizeState 1.378 s 213 ms
Login/SendPinCode 98 ms 87 ms
Order/Get 48 ms 47 ms
Client/GetClient 53 ms 47 ms
Pizzeria/Get 89 ms 46 ms

В чем проблема?

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddNewtonsoftJson();
}
```

https://github.com/dotnet/aspnetcore/issues/17843

Решение: кеширование json

А) Кешировать в приложении

Б) Кешировать в сервисе

В) Кешировать на load balancer

Кеширование в HTTP

Cache-Control

ETag/If-None-Match

Last-Modified/If-Modified-Since

developer.mozilla.org

Проблема актуальности данных

1 час LF

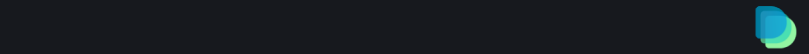
1 час МАПИ

1 час Nginx

2 часа

3 часа

Кеширование в HTTP



Cache-Control: 300

~~ETag/If-None-Match~~

Last-Modified/If-Modified-Since

developer.mozilla.org

Первый запрос

```
Request:
GET /menu
...

Response:
200 OK
Cache-Control: 300
Last-Modified: Wed, 21 Oct 2015 07:28:00 GMT
{
    ...
}
```

Последующие запросы

```
Request:
GET /menu
If-Modified-Since: Wed, 21 Oct 2015 07:28:00 GMT
...

Response:
304 Not Modified
Cache-Control: 300
Last-Modified: Wed, 21 Oct 2015 07:28:00 GMT
```

Настраиваем кеширование на nginx

```
location ~* ^/api/v1/menu$ {
    proxy_cache_revalidate on;
    proxy_cache_use_stale error timeout updating http_500;

        proxy_cache_lock on;
    proxy_cache_background_update on;
}
```

Как это выглядит в коде?

```
public interface ICache
{
    Task<(T Value, DateTime AddTime)> GetWithLastModified<T>(string key);
}

public struct CacheItem
{
        public object Value { get; }
        public DateTime AddTime { get; }
        public CacheItem(object value, DateTime addTime)
        {
                Value = value;
                AddTime = addTime;
        }
}
```

Как используем

```
async Task<MenuResponse> BuildMenu(MenuRequest request, CancellationToken ct)
{
    var (menu, addTime) = await menuService.GetPizzeriaMenu(...);

    if (addTime > request.IfModifiedSince) {
        return new MenuResponse(BuildMenu(menu), addTime)
    }
    return new MenuResponse(addTime);
}
```

Меню МАПИ

Работает

Задержка при обновлении
2 часа

Целостно в любой момент времени

Можем раздать сколько угодно*

Меню отдается за 50 мс

САЙТ

Статические страницы

- SSR рендеринг из Core 2.1
- Node находится в одном контейнере с сайтом
- AddResponseCaching/UseResponseCaching
- Vary: Platform + Locality

Response:

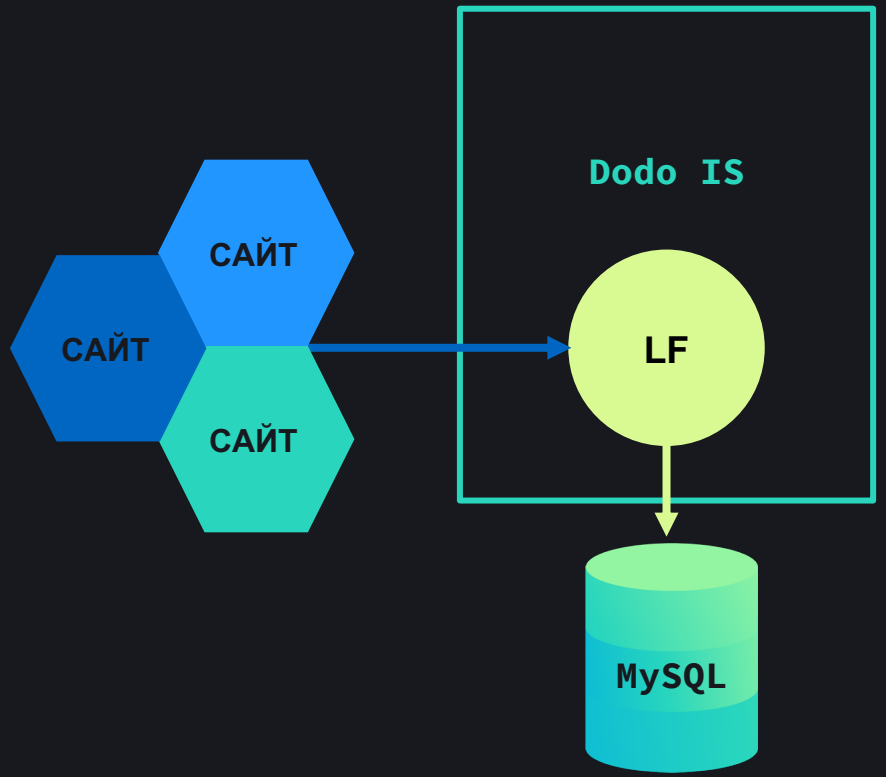
- 4 Mb HTML

DODO IS 2019

MySQL на страну (МОНОЛИТ)

LF + In Memory Cache на доменную сущность

Сайт + In Memory Cache на ответ LF



Проблема актуальности данных

1 час LF

2 часа

1 час Сайт

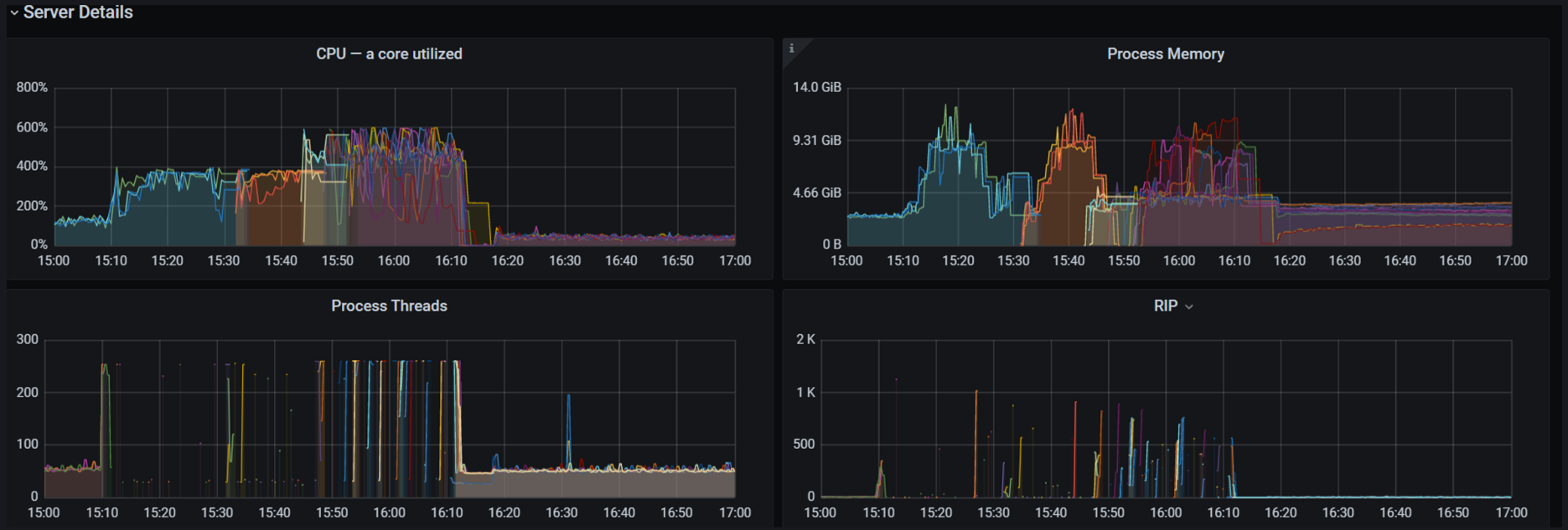
Страницы сайта

Работает

Задержка при обновлении
2 часа

Целостно в любой момент времени

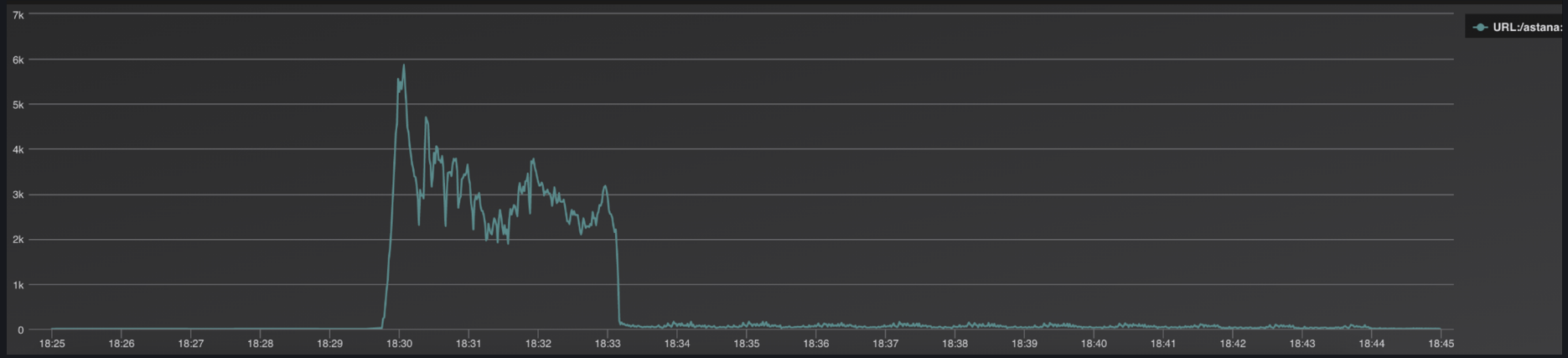
Можем раздать сколько угодно*



Как проблему увидел дежурный

Последствия

- Сайт перестал отвечать на запросы
- Пришлось перегружать поды Сайта
- Пришлось отключать весь трафик на Сайт пока поды не поднялись
- Сайт под нагрузкой не поднимался :(



Атака на Астану

Что было на самом деле

- Атакующий перебирал значения куки locality
- Кеш протыкался
- На каждый запрос мы генерили 4мб страницу :(
- dotnet умирал тк не успевал разгребать таски

Быстрое решение: request limiter

- Спасает от атаки
- Отдает 500 атакующему

- Отдает 500 нормальным пользователям :(

Нормальное решение

- Кеши на nginx
- Сам конфиг, в котором количество ключей для кэширования конечно
- Тесты на конфиг (тк он был достаточно сложный)

Ключ кэширования

```
set $cache_key $scheme$host$proxy_host$uri$client_type;
```

- scheme - http/https 2 значения
- host - dodopizza.[ru,by…]
- uri - количество локаций + количество пиццерий * 3
  ограниченное множество
- client_type - тип клиента мобильный или настольный 2 значения

Про client_type

Определяем на основе хидера UserAgent с помощью регулярки

```
map $http_user_agent $client_type {
        default "desktop";
        "~*mobi" "mobile";
}
```

А как избавились от куки Locality?

Теперь мы читаем ее прямо в Nginx конфиге и делаем редирект именно там

```
location = / {
    if ($redirect_to_location) {
                return 302 /$cookie_locality;
        }
    proxy_pass http://backend;
        proxy_cache_key $cache_key;
}

map "$cookie_locality" $redirect_to_location {
        default 0;
        "^(.+)$" 1;
}
```

Тестовое окружение

- Простой http сервер на GO
- Тесты на GO
- Запускаем все с помощью docker-compose

```
> docker-compose up \
            --exit-code-from test \
            --abort-on-container-exit \
            --build \
            --force-recreate
```

Что нужно от test suite

- Отдавать redirect
- Отдавать body
- cookie

Страницы сайта

Работает

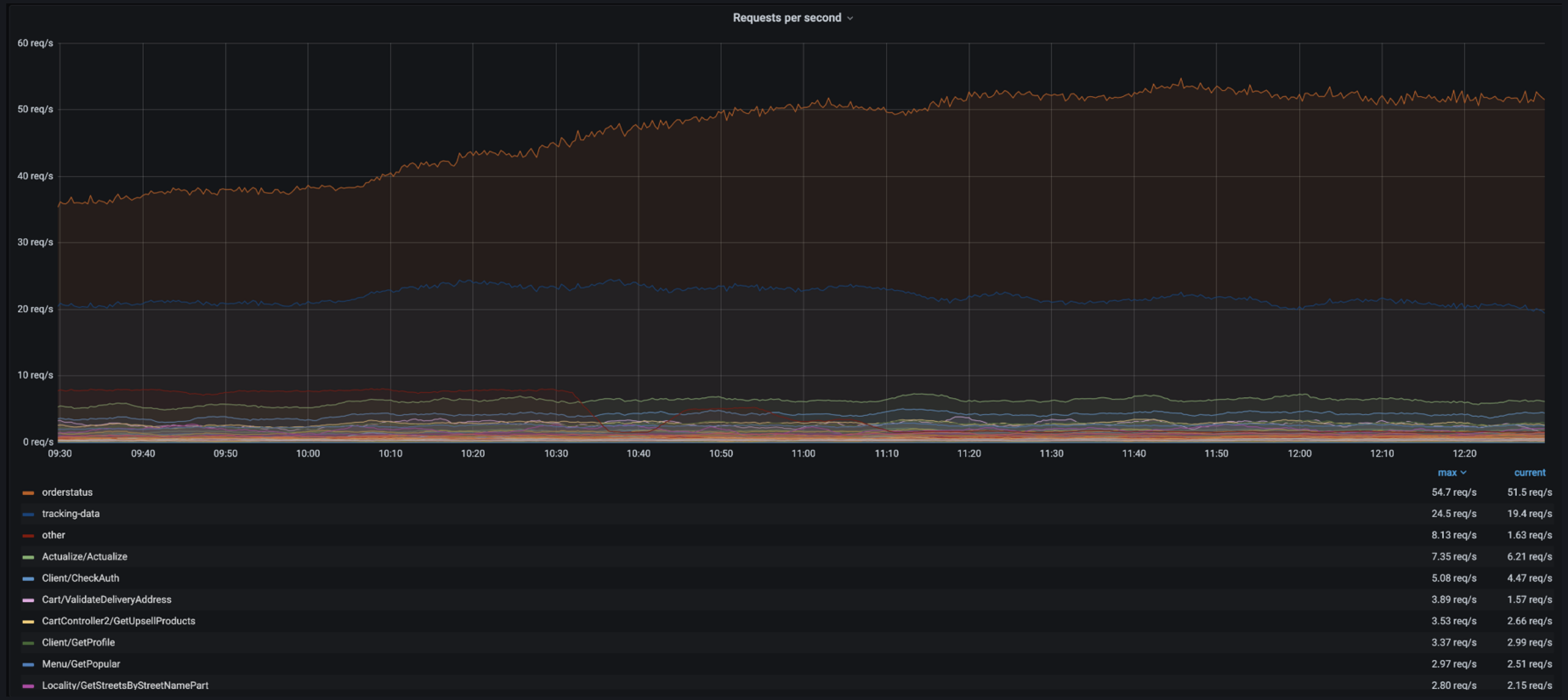
Задержка при обновлении
2 часа

Целостно в любой момент времени

Можем раздать сколько угодно*

Не страдает от пробития кеша

Что делать с быстро изменяющимися данными?



Топ запросов к dodopizza.[ru,by…] сейчас

ДЕМО

САЙТ

TrackingData

- Один из самых частых запросов сайта
- Ходит в LF
- Кешируется на 5 минут
- Полится раз в 30 секунд с клиента

Как работала tracking-data

- Трекер генерирует событие на изменение статуса продукта
- Монолит ловит событие и пишет его в orders_composition
- Сайт ходит в ЛФ, который достает данные из orders_composition и отдает сайту.

  Сайт кеширует у себя ответ монолита на 5 минут, но в кеше участвует UUID заказа

Проблемы:

- “Лишние” походы в базу монолита
- Крайне не эффективное кэширование

Tracking data

Работает

Задержка при обновлении 5 минут

Неэффективное кеширование

Большая нагрузка на базу монолита

А как лучше?

Подписаться на события трекера самостоятельно

Агрегировать продукты по одной пиццерии

Переупорядочивание на стороне клиента

Работать только с идентификаторами продуктов

Что такое cooking info?

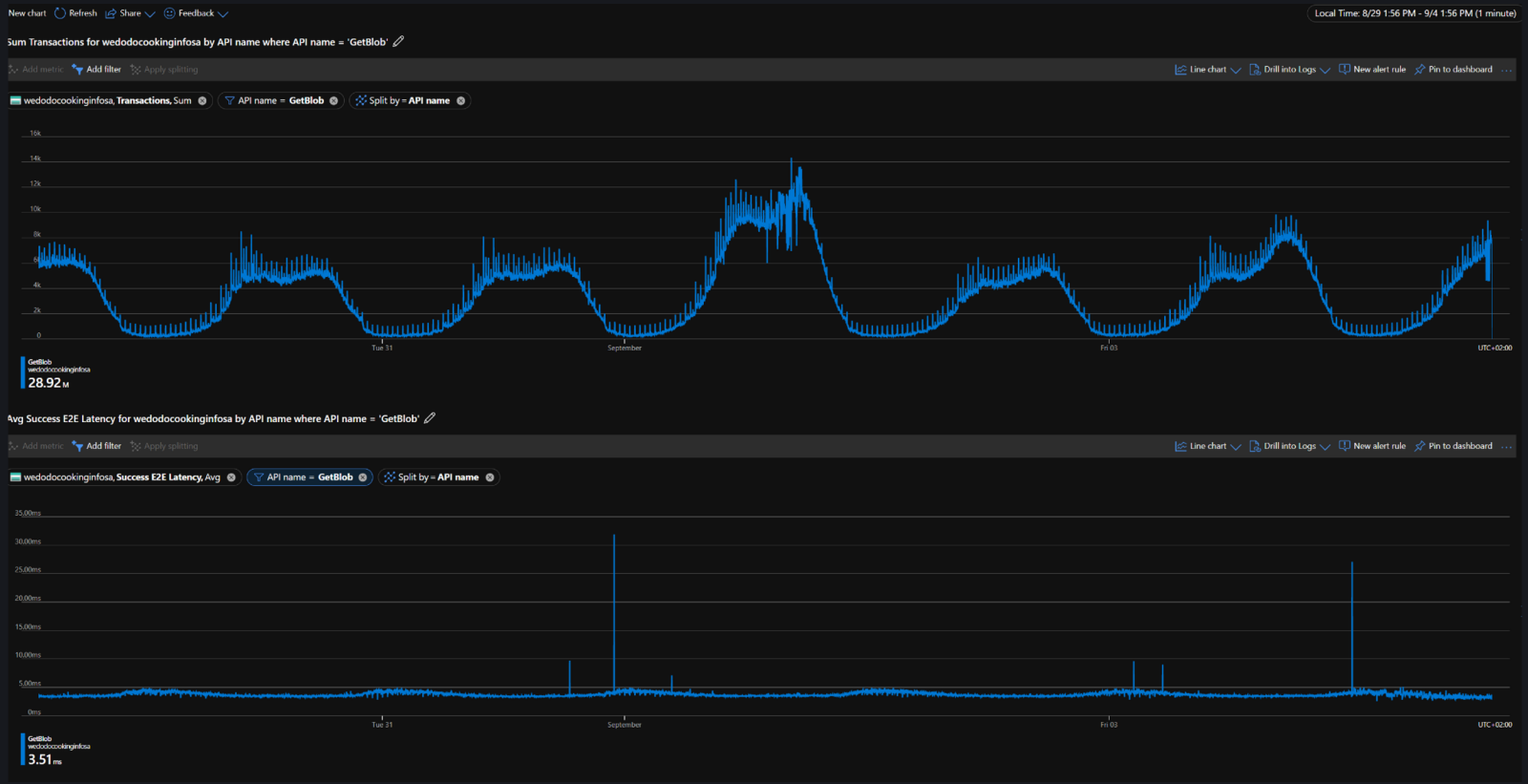
Сервис отвечающий на один вопрос: “Что сейчас готовится в пиццерии?”

- Ключ хранения данных UUID пиццерии
- Один JSON документ со стадиями приготовления продуктов
- Можем хранить в Blob storage и обновлять при приходе эвента

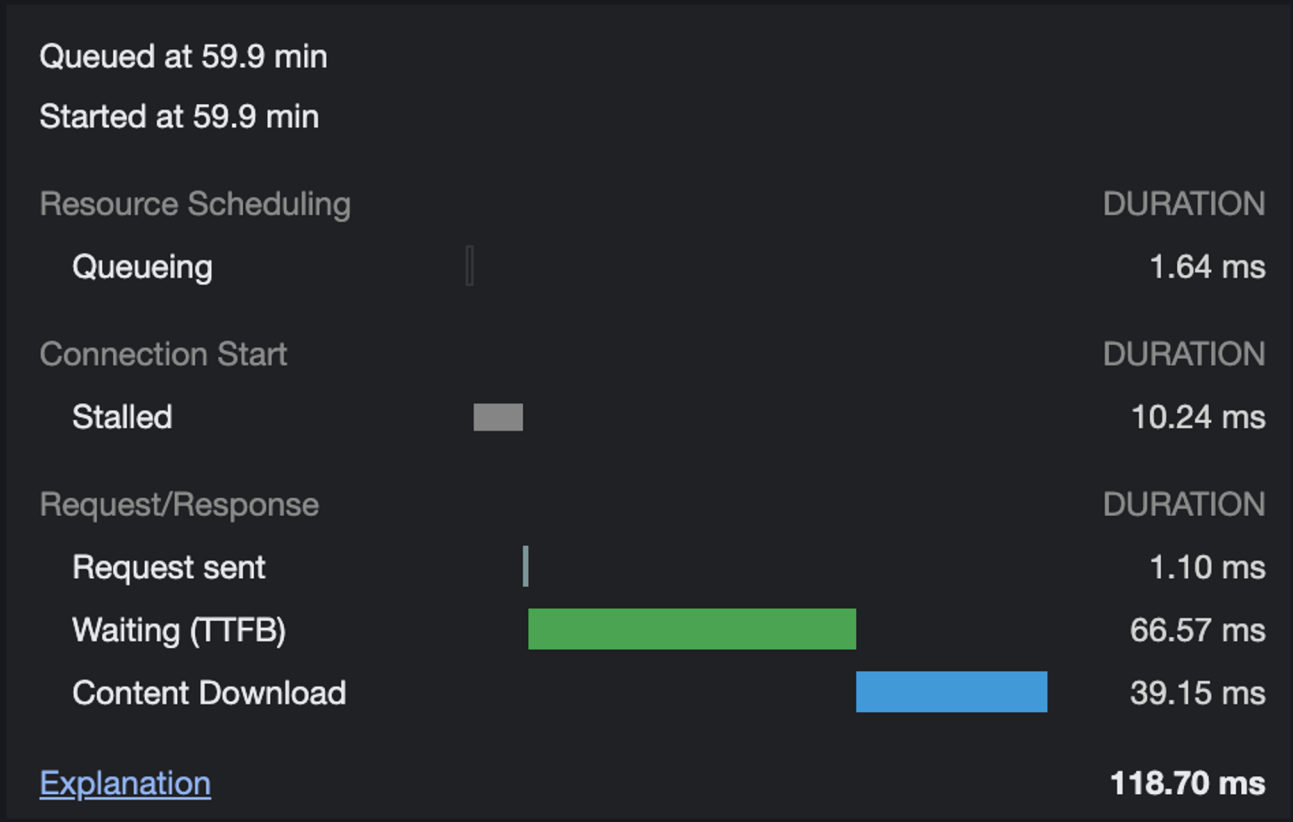
Почему Blob storage

- Отлично подходит для хранения по одному ключу
- Есть локи, можно писать атомарно
- Простой
- Очень быстро отдает данные
- ОЧЕНЬ дешево отдает данные
- Абсолютное масштабирование (платим за запросы и хранящиеся данные)

По сути заменяет нам все кеши.

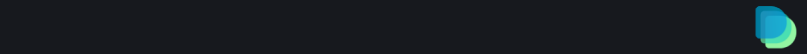


Метрики blob storage cooking info



Результаты tracking-data

Tracking data



Работает

Реактивно

Можем раздавать сколько угодно

А что если сделать то же самое с меню МАПИ?

Меню МАПИ

Работает

Задержка при обновлении
2 часа

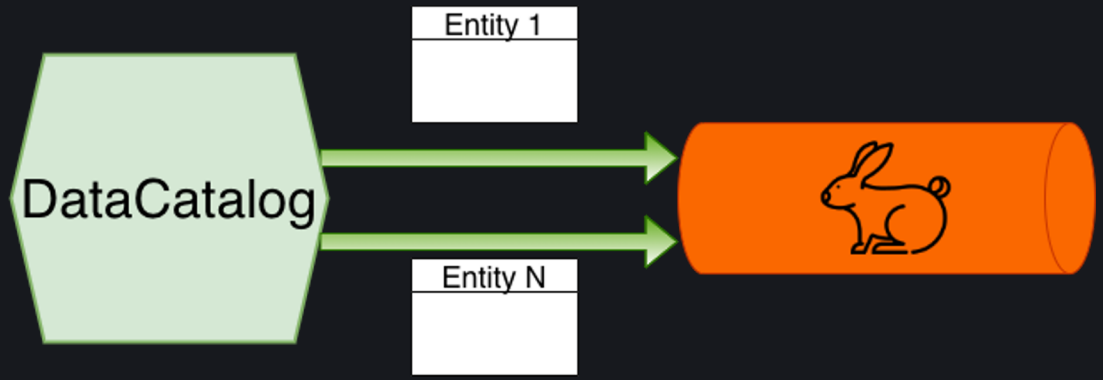
Целостно в любой момент времени

Можем раздать сколько угодно*

Меню отдается за 50 мс

DataCatalog – source of truth для всех данных, необходимых для построения меню

DataCatalog, как это работает



Событие DataCatalog

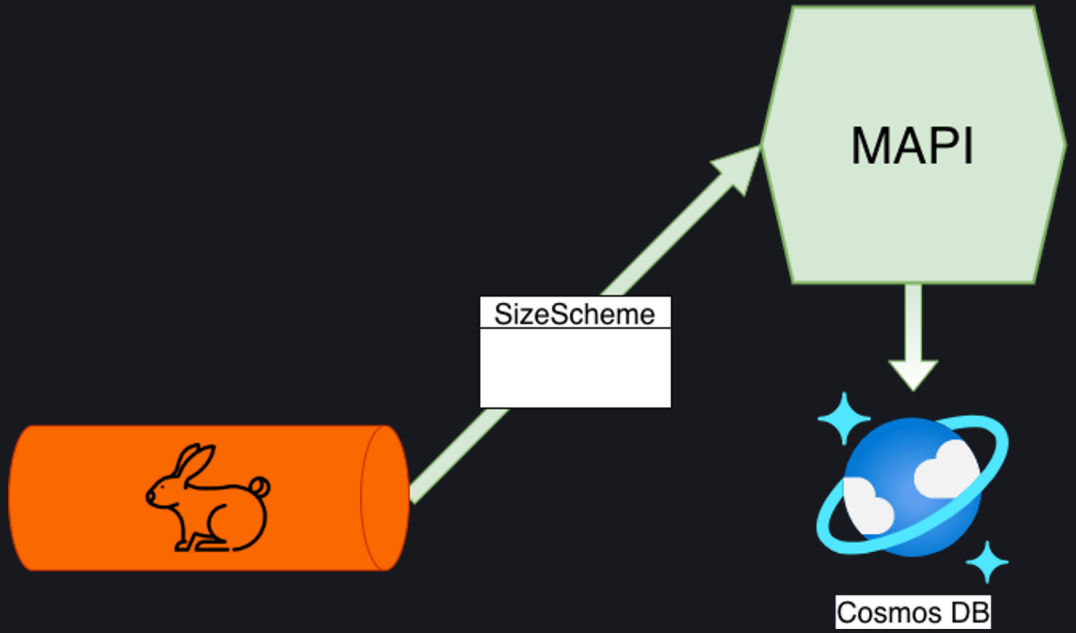
```
public class SizeScheme
{
    public UUId Id { get; set;  }
    public int CountryId { get; set; }
    public string Name { get; set; }
    public ProductSize[] Sizes { get; set; }
    public int Version { get; set; }
    public UUId MonolithId { get; set; }
    public DateTime UpdatedAtUtc { get; set; }
}
```

Событие DataCatalog

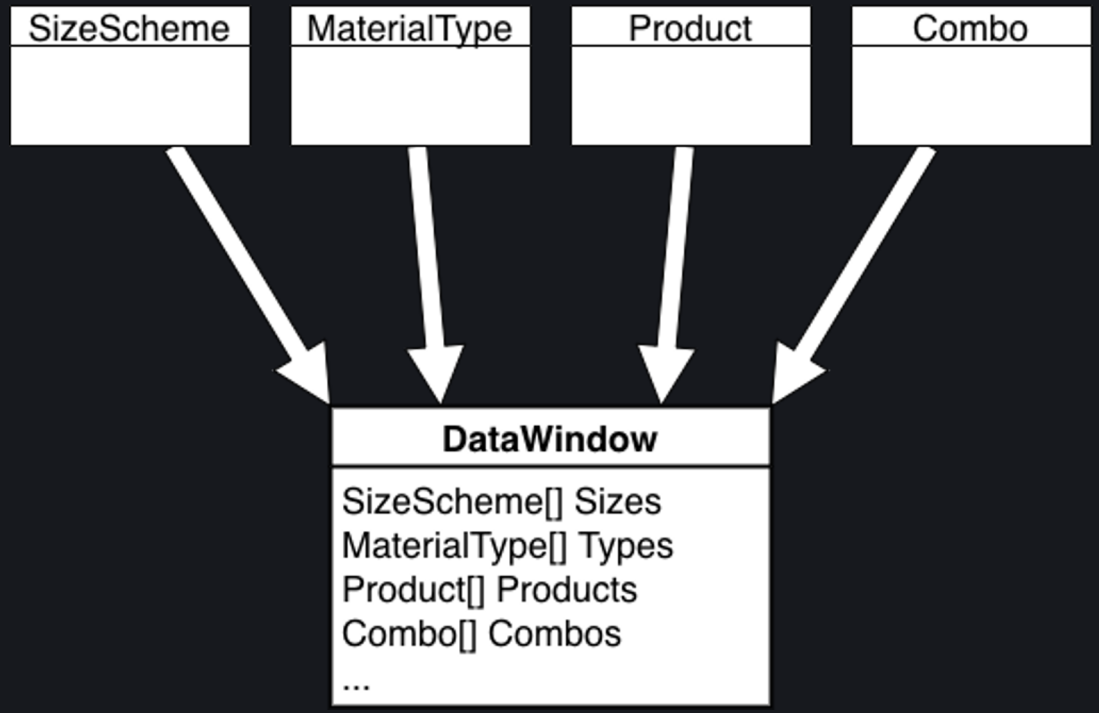
```
public class SizeScheme
{
    public UUId Id { get; set;  }
    public int CountryId { get; set; }

    public int Version { get; set; }
}
```

DMS - data materialization service

Как DMS работает с событием DC



Строим Data Window



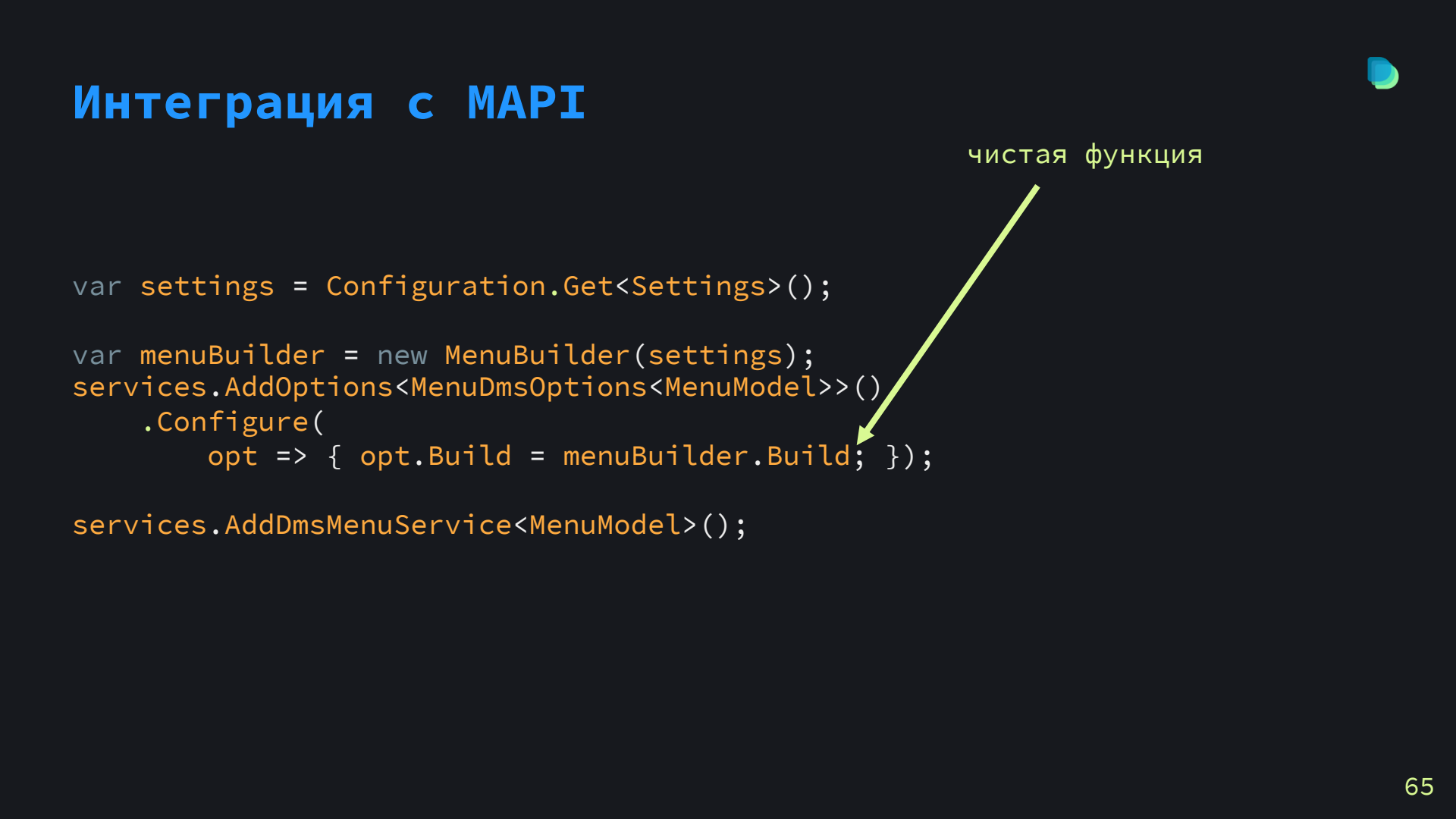
Строим витрину для меню

```
public class MenuDataWindow
{
    public UUId Id { get; }

    public int Version { get; }

    public UUId? LocationId { get; }
    public Country? Country { get; }
    public MenuType Type { get; }
    public LocationType LocationType { get; }
    public Currency? Currency { get; }
    public Popular? Popular { get; }
    public IReadOnlyDictionary<UUId, ComboTemplate> ComboTemplates { get; }
    public IReadOnlyDictionary<UUId, MetaProduct> MetaProducts { get; }
    public IReadOnlyDictionary<UUId, Product> Products { get; }
    public IReadOnlyDictionary<UUId, SizeScheme> SizeSchemes { get; }
    public IReadOnlyDictionary<UUId, MaterialType> MaterialTypes { get; }
    public DataCatalog.Contracts.Menu.v1.Menu? Menu { get; }
}
```

Интеграция с MAPI

```
var settings = Configuration.Get<Settings>();

var menuBuilder = new MenuBuilder(settings);
services.AddOptions<MenuDmsOptions<MenuModel>>()
    .Configure(
        opt => { opt.Build = menuBuilder.Build; });

services.AddDmsMenuService<MenuModel>();
```

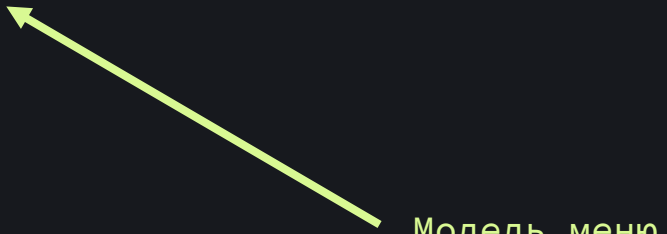
Интеграция с MAPI




```
var settings = Configuration.Get<Settings>();

var menuBuilder = new MenuBuilder(settings);
services.AddOptions<MenuDmsOptions<MenuModel>>()
    .Configure(
        opt => { opt.Build = menuBuilder.Build; });

services.AddDmsMenuService<MenuModel>();
```

Интеграция с MAPI

```
var settings = Configuration.Get<Settings>();

var menuBuilder = new MenuBuilder(settings);
services.AddOptions<MenuDmsOptions<MenuView>>()
    .Configure(
        opt => { opt.Build = menuBuilder.Build; });

services.AddDmsMenuService<MenuView>();
```

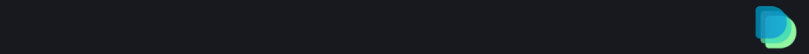


Работа с 2 вьюхами меню

```
var settings = Configuration.Get<Settings>();

...

var menuV2Builder = new MenuV2Builder(settings);
services.AddOptions<MenuDmsOptions<MenuViewV2>>()
    .Configure(
        opt => { opt.Build = menuV2Builder.Build; });

services.AddDmsMenuService<MenuViewV2>();
```

Новая модель меню

Кеширование в HTTP



`Cache-Control: 300`

`ETag/If-None-Match`

~~`Last-Modified/If-Modified-Since`~~

developer.mozilla.org

Как эффективно кэшировать

```
async Task<MenuResponse> BuildMenu(MenuRequest request, CancellationToken ct)
{
    var (menuModel, etag) = await _dmsBuilder.GetPizzeriaMenu(
            new MenuDescriptor
            {
                                ...
                Etag = request.Etag,
            });
    }
    if (etag != request.Etag) {
        return new MenuResponse(menuModel, etag)
    }
    return new MenuResponse(etag);
}
```

Меню МАПИ

Работает

Целостно в любой момент времени

Можем раздать сколько угодно*

Меню отдается за 50 мс

Реактивно

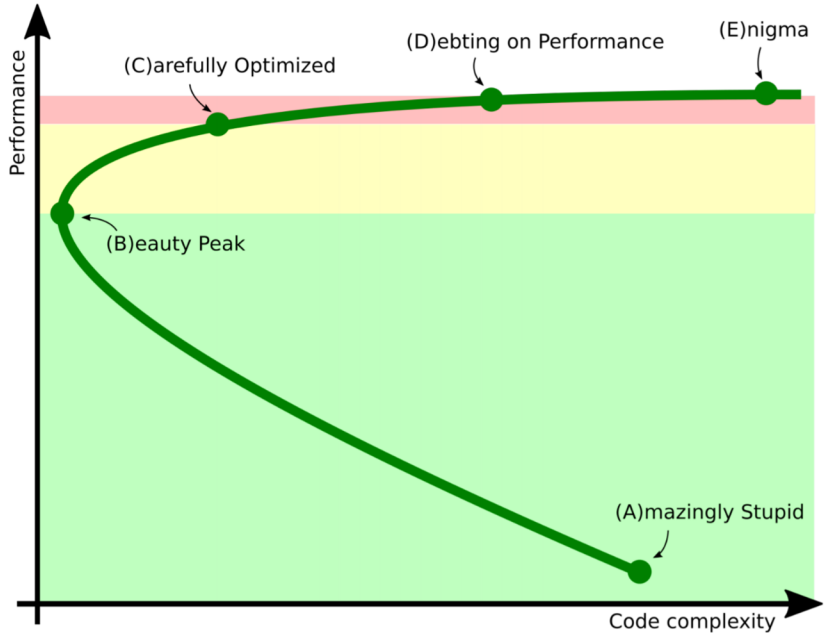
Выводы

Мы не можем позволить себе “просто” отдавать json

Материализация данных - подходит нам лучше всего

Кеширование - часть стратегии материализации

Крупно: Кривая им. Ш:



Slide 8/66. «Keynote: Performance», Aleksey Shipilëv, 2017, D:20170404035127+02'00'

redhat.

Алексей Шипилёв — Перформанс: Что В Имени Тебе Моём? (https://shipilev.net/#perf-keynote)