# Как вырастить opensource в банке

Райффайзен

## Обо мне

Константин Густов

архитектор,
Райффайзенбанк

10+ лет опыта в разработке

konst.gustov@gmail.com

# Темы

I. С чего все началось?

# Темы

I. С чего все началось?

II. Детали реализации

# Темы

I. С чего все началось?

II. Детали реализации

III. Продвижение

# Темы

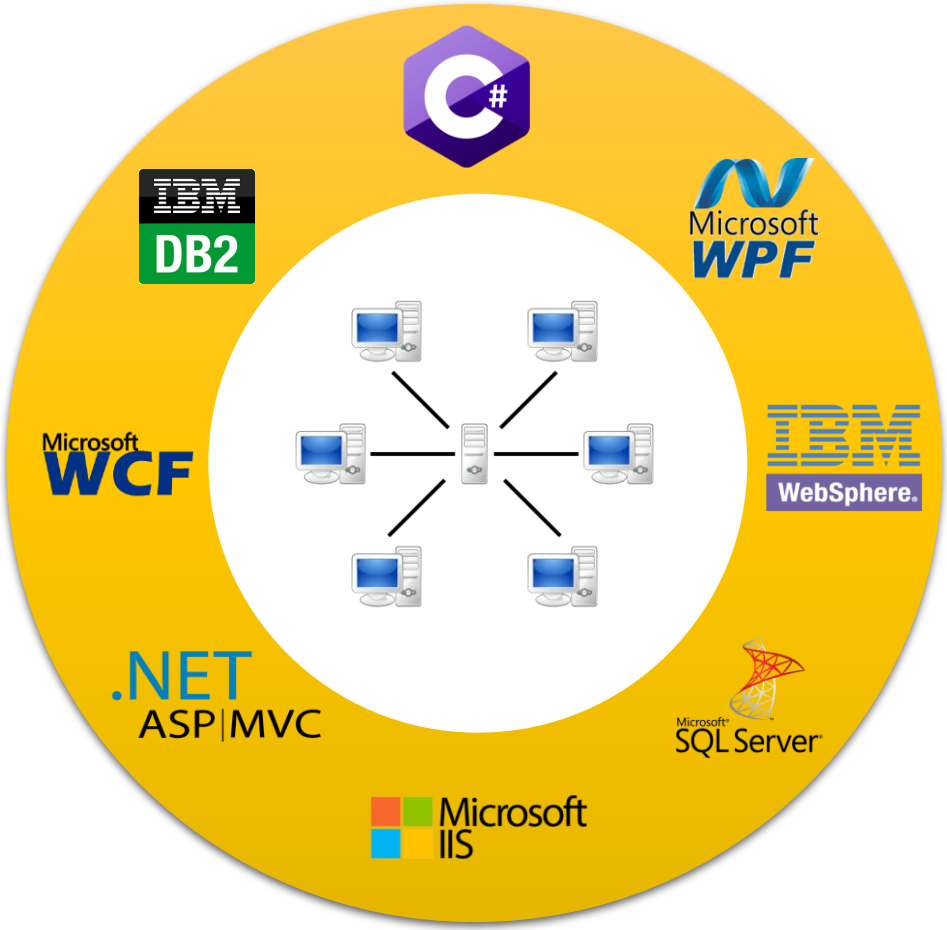
I. С чего все началось?

II. Детали реализации

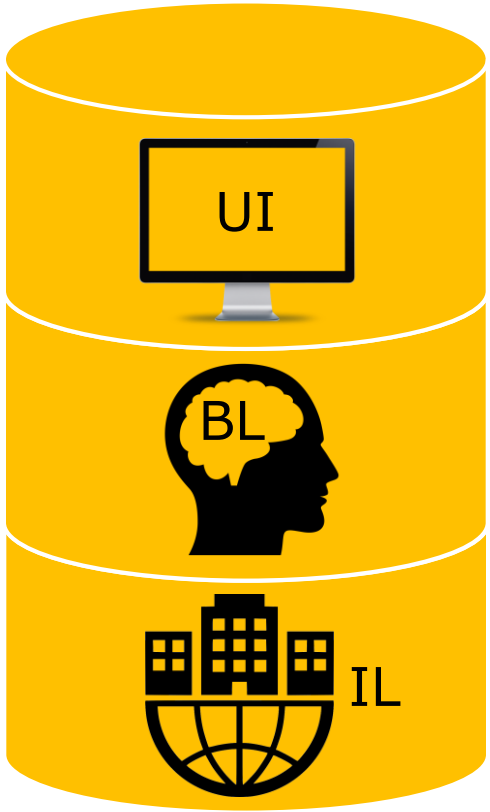
III. Продвижение

IV. Что дальше?

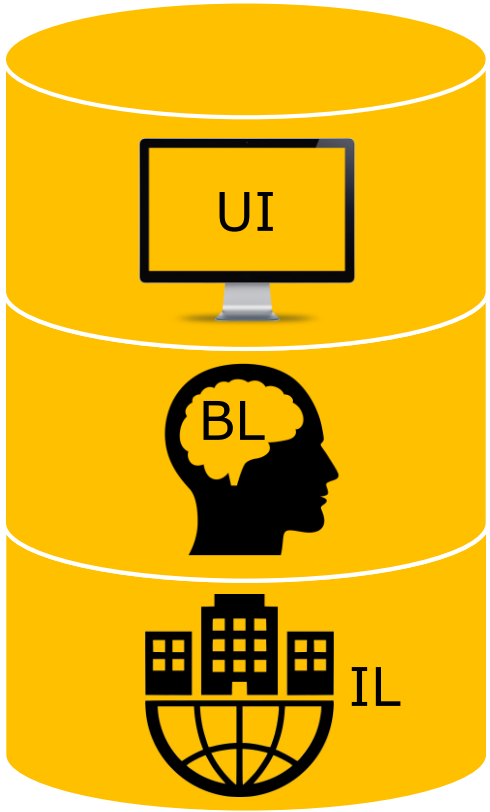
# Корпоративное приложение

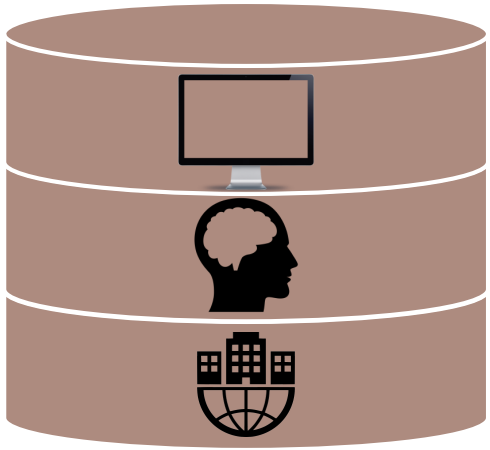


# Хранение исходных кодов

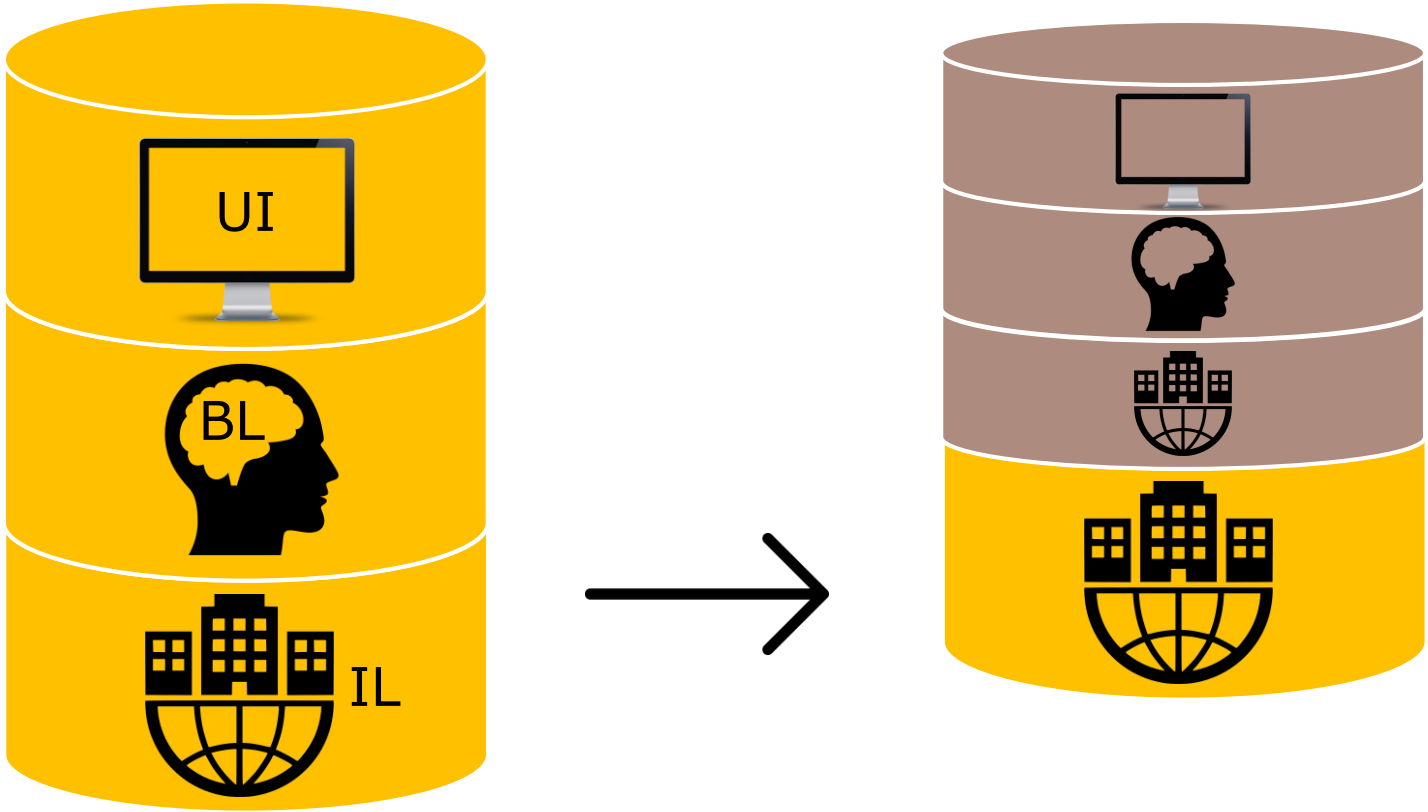


# Хранение исходных кодов



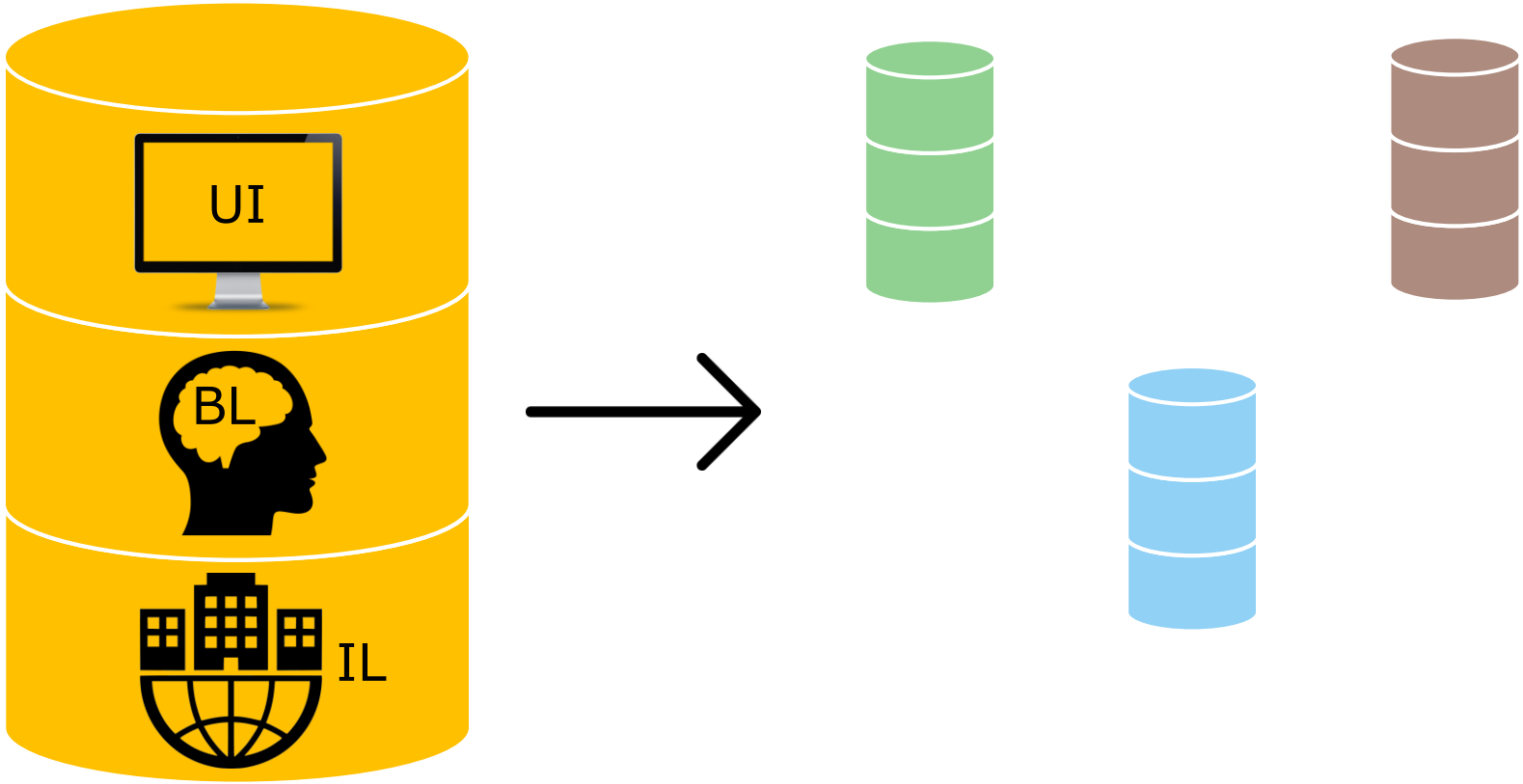
# Хранение исходных кодов



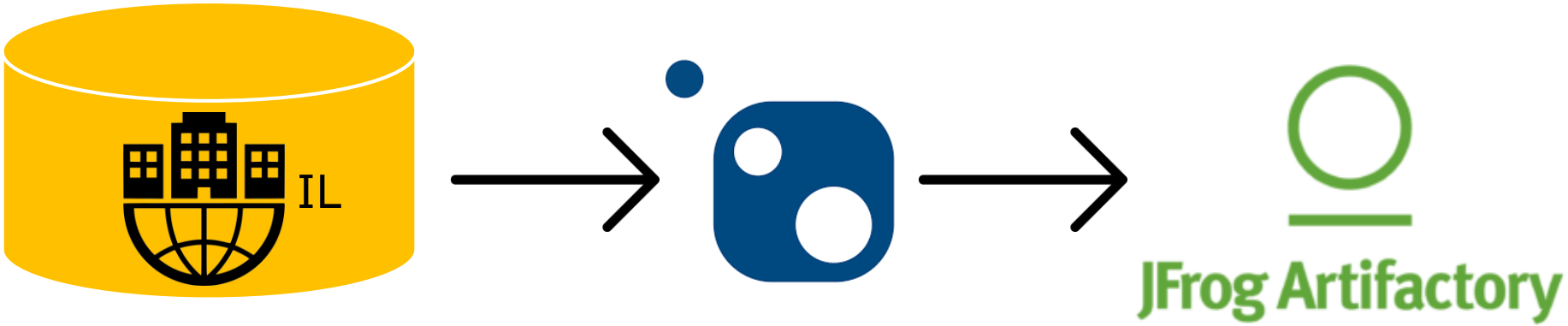
# Результат

# Результат

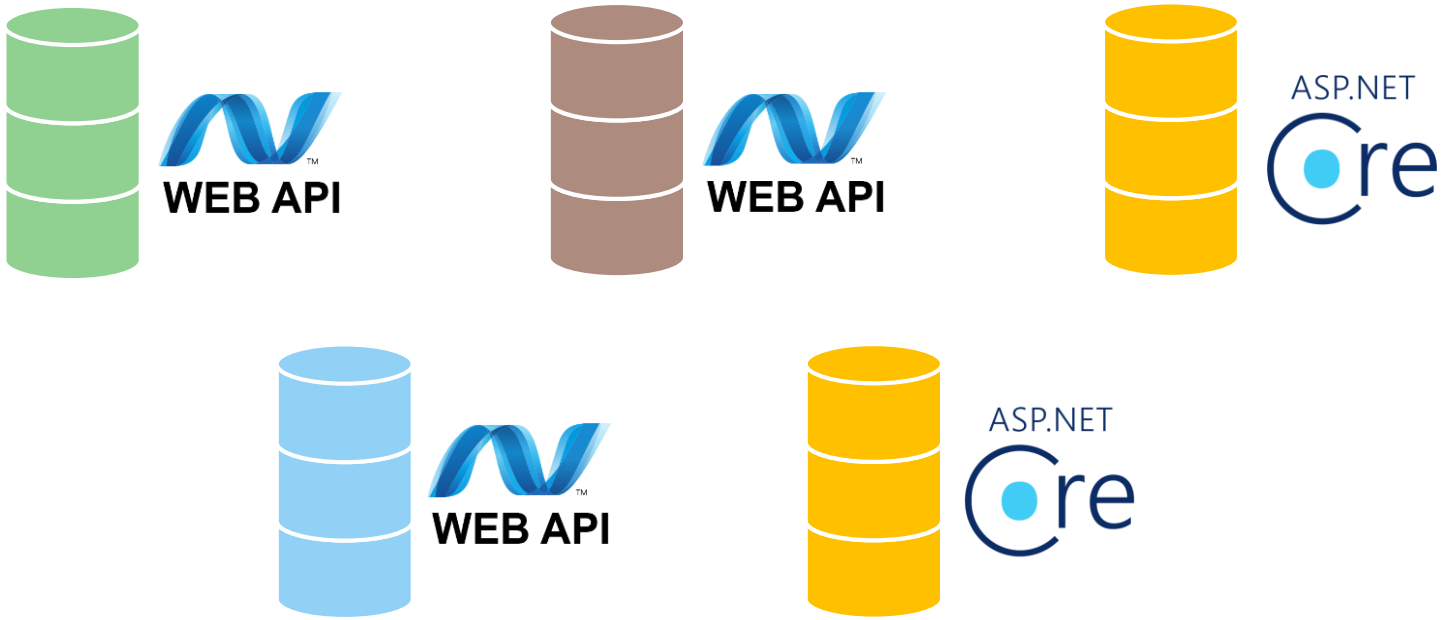
# Репозиторий на сервис



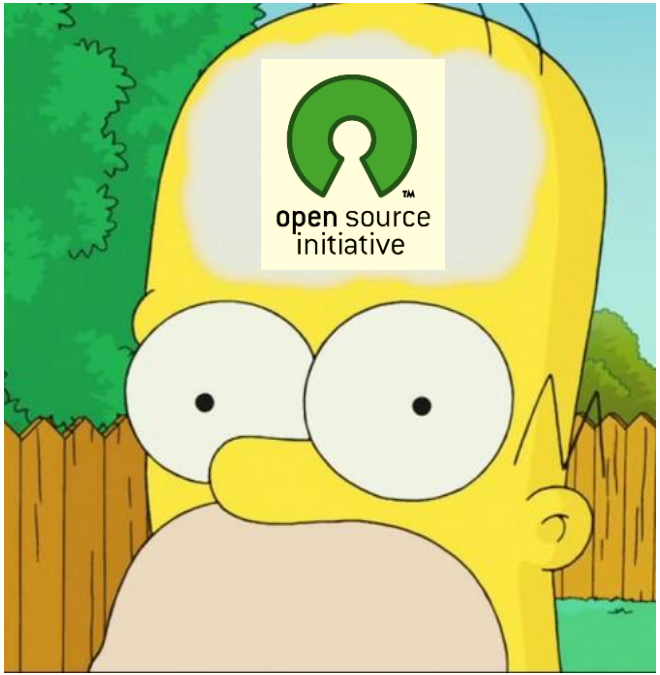
# Репозиторий для зависимостей



# Новые проекты

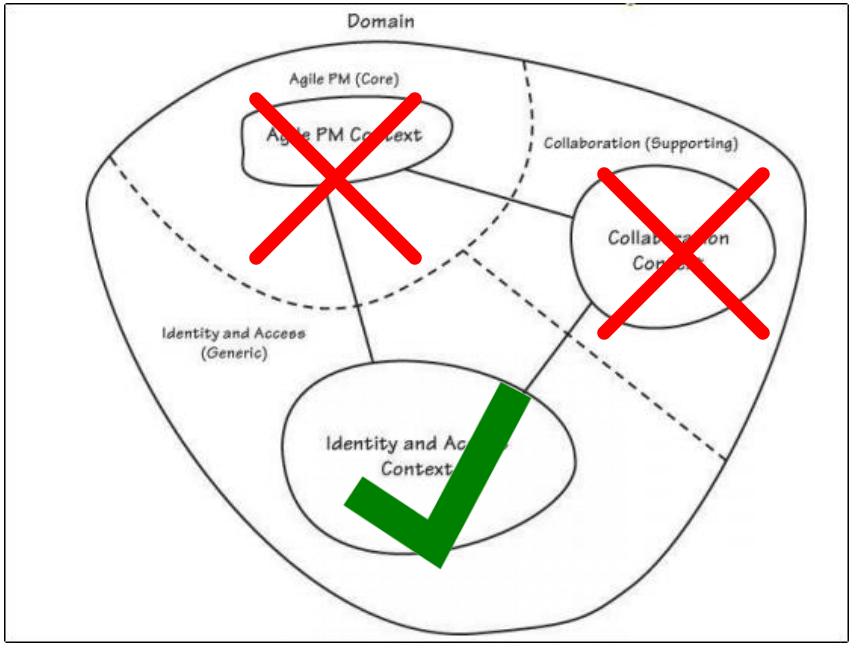


# От inner к open source



# Open source puzzle


Разработка должна быть свободна от ключевой бизнес-логики компании



# Детали реализации

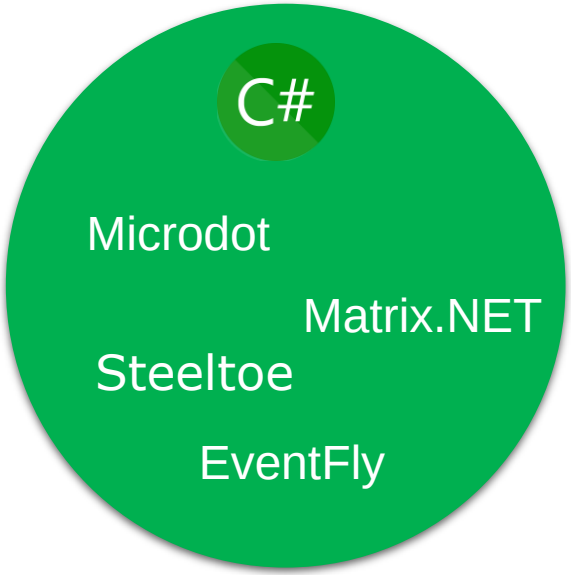
# Идея фреймворка

## Микросервисное шасси

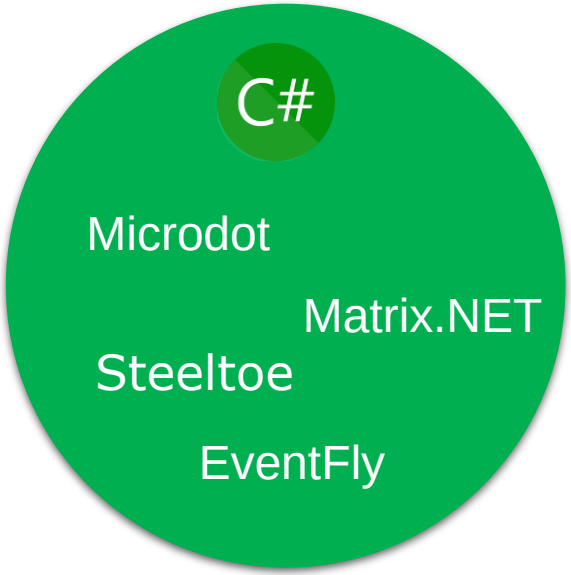
Каркас, который решает сквозные (комплексные) задачи

# Микросервисное шасси

# Микросервисное шасси

# Микросервисное шасси

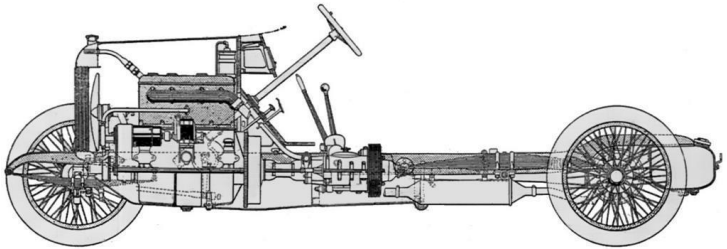




abp v1.0

Алексей Мерсон: ASP.NET Boilerplate: Framework по заветам Domain Driven Design https://youtu.beci1klCHQE6k

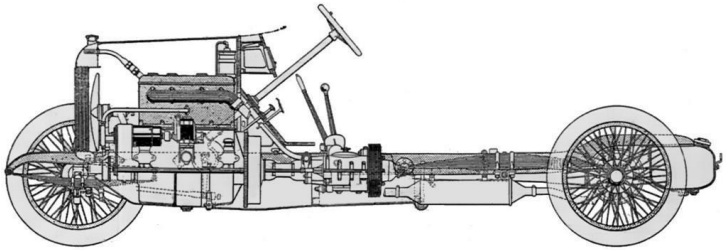
# Идея фреймворка

Микросервисное шасси



# Идея фреймворка

Микросервисное шасси

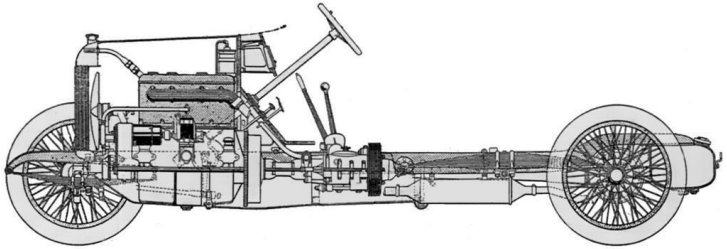


+

Domain driven design
tacticals

# Идея фреймворка

Микросервисное шасси



+

Domain driven design tacticals

+

Enterprise

# Основные возможности

Архитектура и тактические паттерны DDD

Конфигурируемость

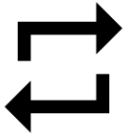
Модульность

Логирование и аудит

Наблюдаемость и метрики

Авторизация

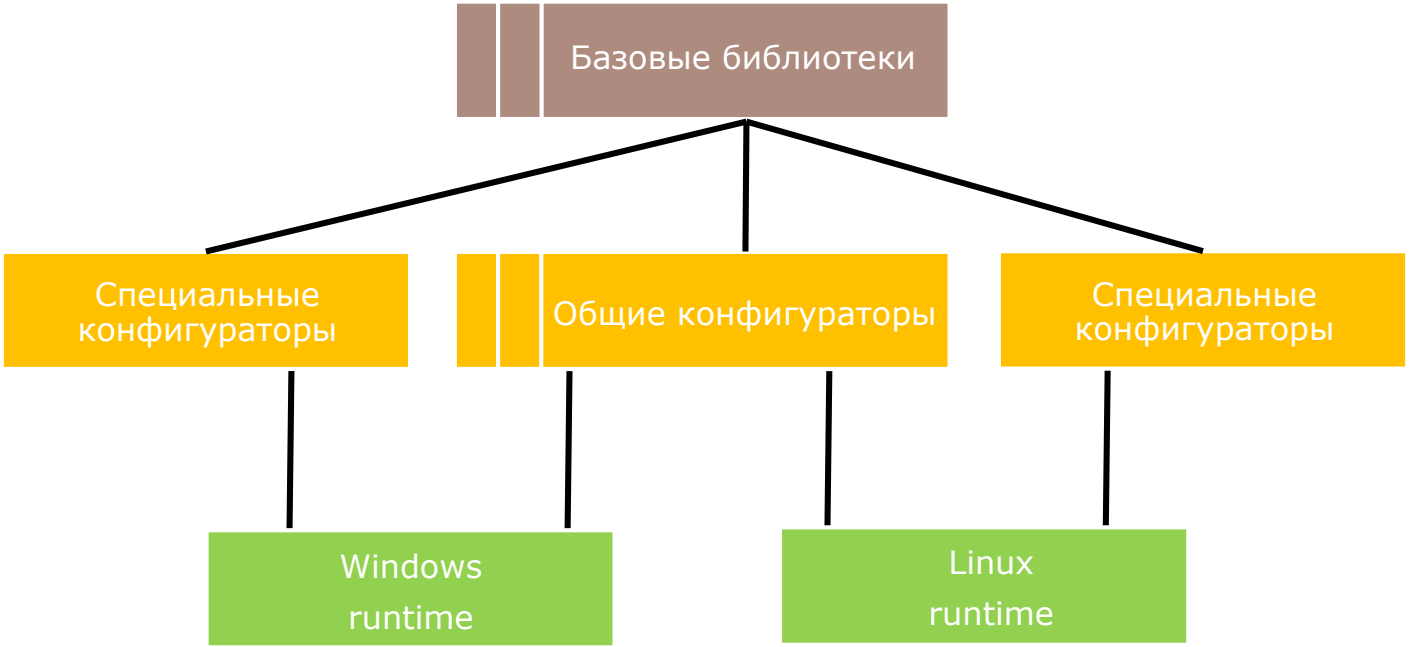
Транзакционность



Адаптеры к внешним источникам

CQRS & event sourcing

# Библиотеки web-сервисов



# Диагностика

Diagnostic

```
[Route("diagnostic")]
public class DiagnosticController : ControllerBase
{
  private readonly IHealthCheckingService _healthCheckingService;

  [HttpGet("ping")]
  [ProducesResponseType(200)]
  public IActionResult Ping()
  {
    return Ok();
  }

  [HttpGet("diagnose")]
  [ProducesResponseType(typeof(DiagnoseResult), 200)]
  [ProducesResponseType(typeof(DiagnoseResult), 503)]
  public async Task<ActionResult<DiagnoseResult>> Diagnose()
  {
    var result = await GetDiagnose();
    return result.HasErrors
        ? StatusCode((int)HttpStatusCode.ServiceUnavailable, result)
        : Ok(result);
  }
}
```

# Диагностика

Diagnostic

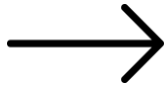
```
public class HealthCheckingService : IHealthCheckingService
{
  private readonly IEnumerable<IDiagnosticImplementor> _implementors;

  public HealthCheckingService(IEnumerable<IDiagnosticImplementor> implementors)
  {
    _implementors = implementors.ThrowIfNull(nameof(implementors));
  }

  public async Task<IEnumerable<DiagnosticInfo>> CheckHealthAsync()
  {
    var tasks = _implementors.Select(implementor => implementor.Diagnose())
                             .ToList();
    var result = (await Task.WhenAll(tasks).ConfigureAwait(false))
                        .SelectMany(diagnosticInfos => diagnosticInfos)

    return result;
  }
}
```

# Диагностика

```
public class OrmConnectionsChecker : IDiagnosticImplementor
{
  ...
  private DiagnosticInfo CheckProvider(ISessionFactoryProvider prov)
  {
    try
    {
      var sessionFactory = _sessionFactoryManager.GetSessionFactory(prov.Nick);
      using (var session = sessionFactory.OpenSession())
      {
        var allClassMetadata = session.SessionFactory.GetAllClassMetadata();
        foreach (var entity in allClassMetadata)
        {
          session.CreateCriteria(entity.Key).SetTimeout(DiagnosticTimeout).SetMaxResults(1)
                 .List();
        }
      }
      return new DiagnosticInfo($"DB: {prov.Nick}", string.Empty);
    }
    catch (Exception e) { ... }
  }
}
```

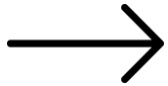
# Библиотеки web-сервисов

Logging

# Библиотеки web-сервисов

Logging

Jwt

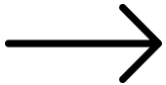
**JWT**

JSON Web Tokens

Microsoft.AspNetCore.Authentication.JwtBearer

# Библиотеки web-сервисов

Logging

Jwt

**JWT**

JSON Web Tokens

Microsoft.AspNetCore.Authentication.JwtBearer

Metrics

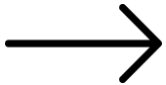
App.Metrics

# Запись метрик

```
private void ConfigurePrometheusMetrics(IServiceCollection services, IConfiguration configuration)
{
  var metrics = AppMetrics.CreateDefaultBuilder().OutputMetrics.AsPrometheusPlainText().Build();
  services.AddMetrics(metrics);

  var options = new MetricsWebHostOptions();
  options.EndpointOptions = endpointsOptions =>
  {
    endpointsOptions.MetricsTextEndpointOutputFormatter = metrics.OutputMetricsFormatters
                            .OfType<MetricsPrometheusTextOutputFormatter>().First();
  };

  services.AddMetricsReportingHostedService(options.UnobservedTaskExceptionHandler);
  services.AddMetricsEndpoints(options.EndpointOptions, configuration);
  services.AddMetricsTrackingMiddleware(options.TrackingMiddlewareOptions, configuration);

  services.AddSingleton<IStartupFilter, DefaultMetricsEndpointsStartupFilter>();
  services.AddMetricsTrackingMiddleware(configuration);
  services.AddSingleton<IStartupFilter, DefaultMetricsTrackingStartupFilter>();
}
```

# Библиотеки web-сервисов

Logging

Jwt

JWT

JSON Web Tokens

Microsoft.AspNetCore.Authentication.JwtBearer

Metrics

App.Metrics

Swagger

Swashbuckle.AspNetCore

# Запуск сервиса на Kestrel

```
BaseKestrelRunner.Configure()
                 .BuildWebHost(args)
                 .Run();
```

# Запуск сервиса на Kestrel

```
BaseKestrelRunner.Configure()
                 .BuildWebHost(args)
                 .Run();
```

```
BaseKestrelRunner.Configure()
                 .RegisterServices((sc, cfg) => serviceCollection.AddResponseCaching())
                 .ConfigureApp((b, cfg, env, container) => builder.UseResponseCaching())
                 .ConfigureApp((app, cfg, env, obj) => app.UseForwardedHeaders())
                 .BuildWebHost(args)
                 .Run();
```

# Основные возможности

Архитектура и тактические паттерны DDD

Конфигурируемость

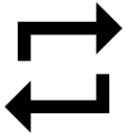
Модульность

Логирование и аудит

Наблюдаемость и метрики

Авторизация

Транзакционность



Адаптеры к внешним источникам

CQRS & event sourcing

# Конфигурация

```
{
  "webApiConfiguration": {
    "portNumber": 80,
    "httpsPort": 8080,
    "CertificatePath": "Cert1.crt"
  },
  "webApiEndpoints": [
    {
      "name": "anotherOneService",
      "url": "http://localhost:9090/",
      "timeout": 30,
      "authType": "jwt"
    }
  ],
  "metrics": {
    "enabled": true,
    "reporter": "prometheus"
  },
  "swagger": {
    "securityServiceUrl": "https://dot-next-server:8047",
    "requestMethod": "POST"
  },
}
```

# Основные возможности

Архитектура и тактические паттерны DDD

Конфигурируемость

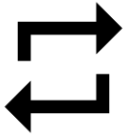
Модульность

Логирование и аудит

Наблюдаемость и метрики

Авторизация

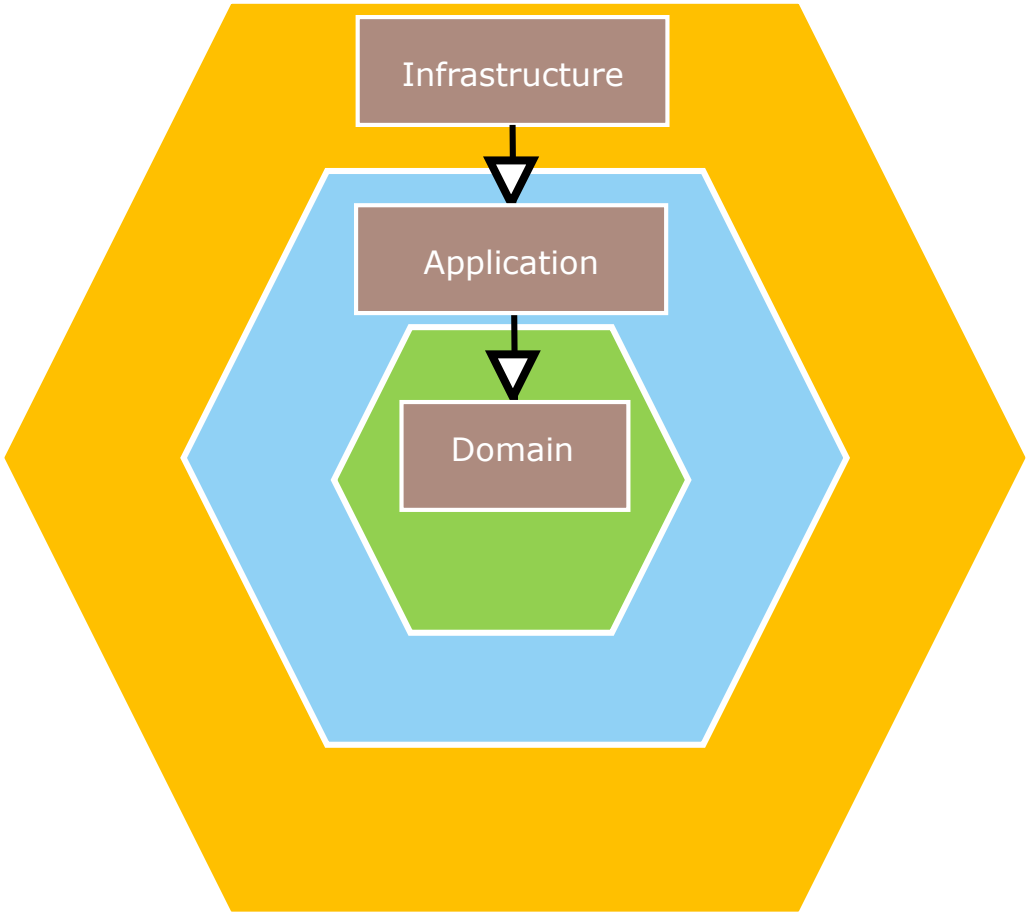
Транзакционность



Адаптеры к внешним источникам

CQRS & event sourcing

# Библиотеки по слоям



# Доменный слой

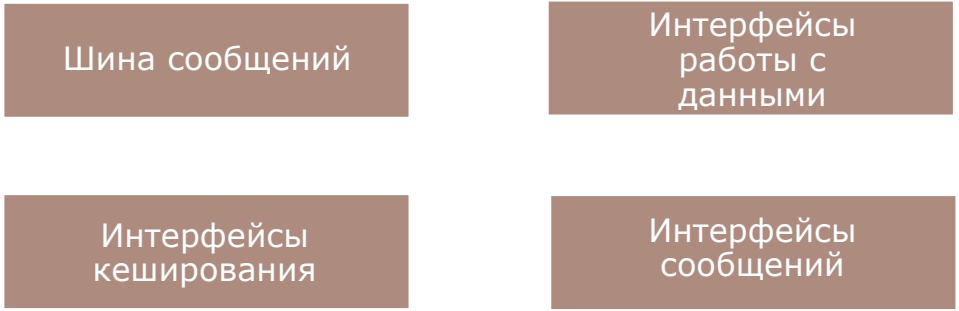
Доменные события

Сущности

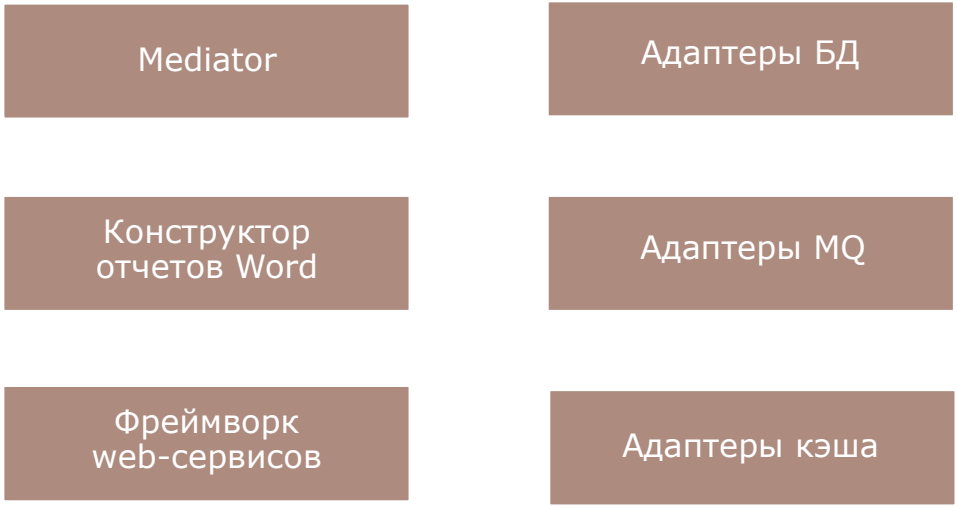
Спецификации

Фреймворк валидации

# Аппликационный слой



# Инфраструктурный слой



# Сущность

```
public class Card : IEntity<int>, IEventProvider
{
  private IEventCollector _eventCollector;
  ...
  public virtual void AddBlock([NotNull] BlockDetails blockDetails)
  {
    ...
    _eventCollector.CollectEvent(new CardBlocked(Id, blockDetails));
  }

  public virtual void SetCollector(IEventCollector sender)
  {
    _eventCollector = sender;
  }
}
```

# Use-Case

```
public async Task<BlockResult> HandleAsync(BlockCardRequest dataRequest, CancellationToken token)
{
  using (var uow = _entityFactoryService.Create())
  {
    var card = _entityFactoryService.Create<Card>()
                                    .GetAsync(dataRequest.Id);

    var result = await card.BlockCard(request.Reason);

    uow.CommitAsync();

    return result;
  }
}
```

# Основные возможности

Архитектура и тактические паттерны DDD

Конфигурируемость

Модульность

Логирование и аудит

Наблюдаемость и метрики

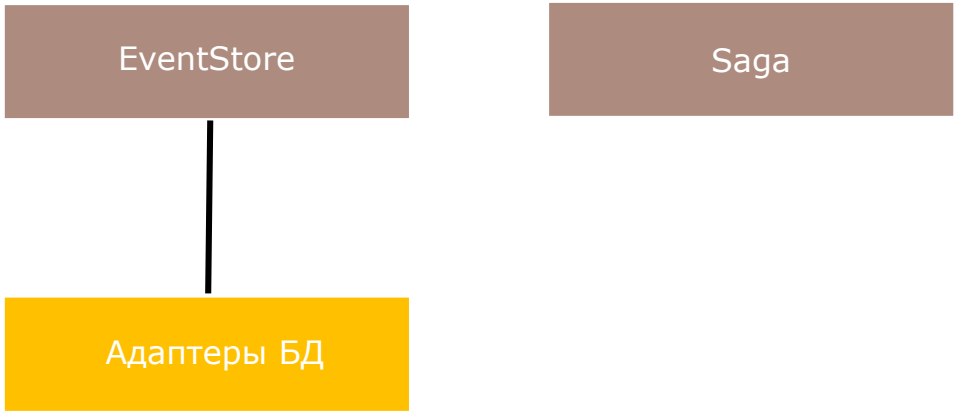
Авторизация

Транзакционность

Адаптеры к внешним источникам

CQRS & event sourcing

# Event sourcing



# Open source puzzle

Разработка должна иметь ценность, как минимум, для компании

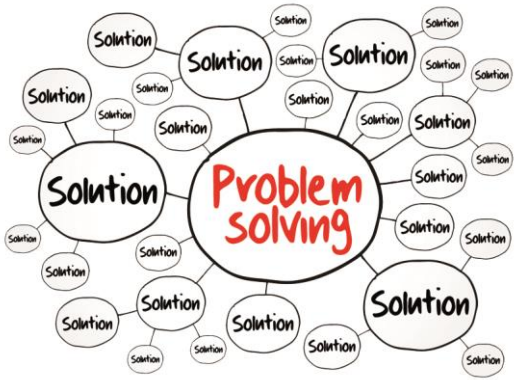
# Продвижение

## Идеи для продвижения

Продукт должен решать проблемы компании

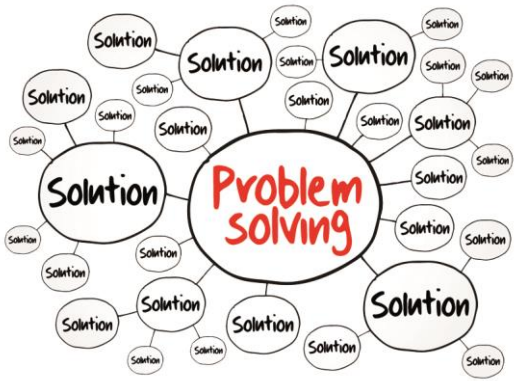
# Проблемы

- Большое количество новых проектов



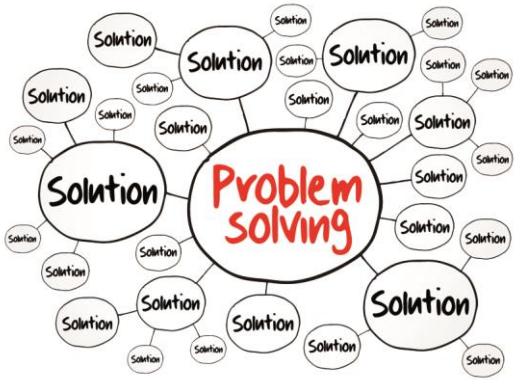
# Проблемы

- Большое количество новых проектов
- Переход на микросервисную архитектуру



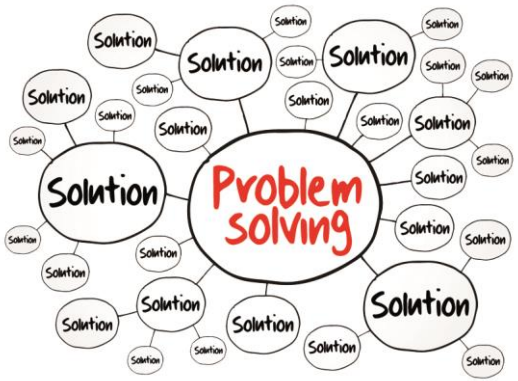
# Проблемы

- Большое количество новых проектов
- Переход на микросервисную архитектуру
- Соответствие стандартам



# Проблемы

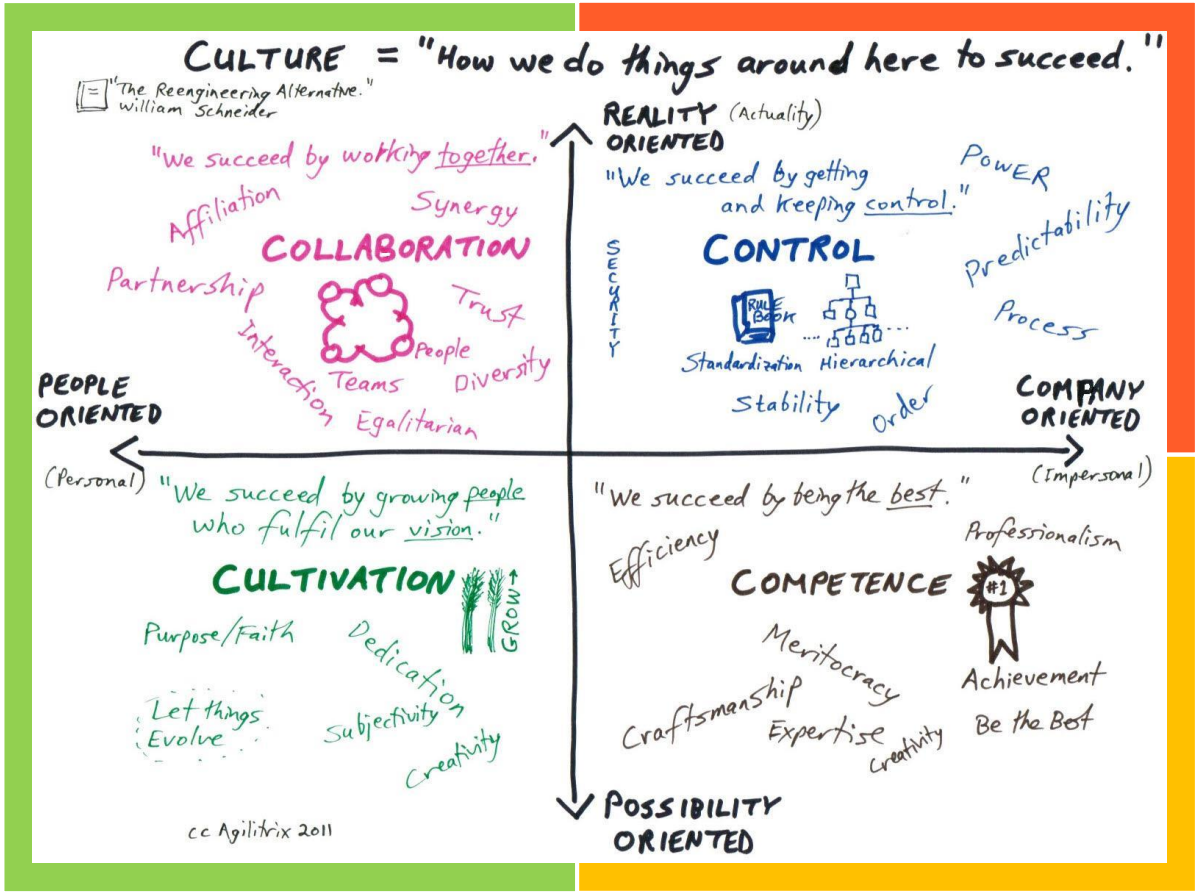
- Большое количество новых проектов
- Переход на микросервисную архитектуру
- Соответствие стандартам
- Внедрение контейнерных технологий



## Идеи для продвижения

Вокруг продукта должно быть сообщество

# Модель культуры по Шнайдеру



# Создание сообщества продукта

- Собрать людей вокруг продукта

# Создание сообщества продукта

- Собрать людей вокруг продукта
- Вовлечь коллег в разработку

# Создание сообщества продукта

- Собрать людей вокруг продукта
- Вовлечь коллег в разработку
- Помогать использовать продукт

# Создание сообщества продукта

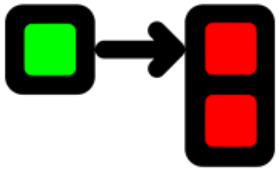
- Собрать людей вокруг продукта
- Вовлечь коллег в разработку
- Помогать использовать продукт
- Быть открытым для новых идей и мнений

## Идеи для продвижения

Нужно заботиться о качестве продукта

# Стек технологий


v3.1



EasyNetQ
v3.7.1


v5.2.6


v7.2.0

# Code review

**PERESADA Oleg** commented on a file 26 Dec 2019

Company.WebApi.Core.Configurators / Middleware / **RequestRegistrationMiddleware.cs** OUTDATED

```
10 +  /// </summary>
11 +  public class RequestRegistrationMiddleware
12 +  {
13 +    private readonly RequestDelegate _next;
14 +
15 +    protected RequestRegistrationMiddleware(RequestDelegate next)
16 +    {
17 +      _next = next;
18 +    }
19 +
20 +    public async Task Invoke(HttpContext context, RequestDelegate next)
```

**PERESADA Oleg**
А чем _next в конструкторе отличается в методе?
Reply · Create task · Create Jira issue · Like · 26 Dec 2019

**ATINK Dmitry**
Есть два варианта реализации middleware:

1. Через наследование интерфейса IMiddleware, в котором Invoke(HttpContext context, RequestDelegate next)
2. Без него, тогда RequestDelegate нужно прокинуть через конструктор

Здесь я убрал использование IMiddleware, но не почистил код.
fixed
Reply · Delete · Create task · Create Jira issue · Like · 2 days ago

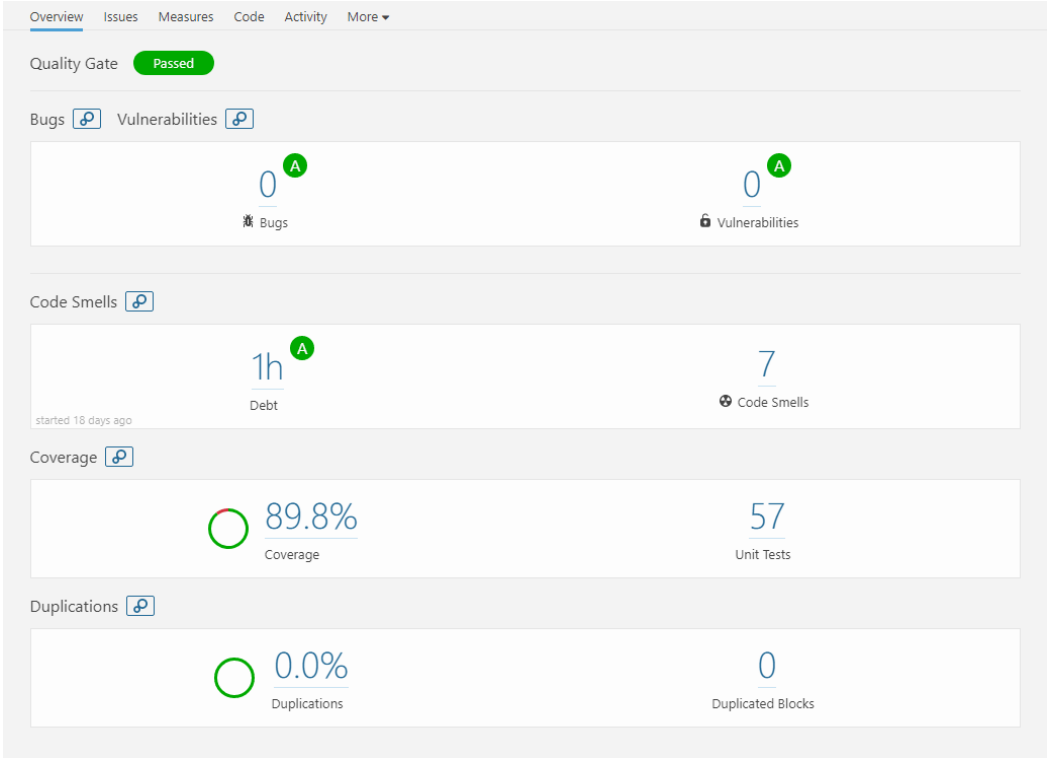
```
21 +    {
22 +      if (!context.Request.Headers.ContainsKey(CompanyHttpHeaders.RequestId))
23 +      {
24 +        context.Request.Headers.Add(CompanyHttpHeaders.RequestId, Guid.NewGuid().ToString("N"));
25 +      }
26 +
27 +      await _next(context);
28 +    }
29 +  }
30 + }
```

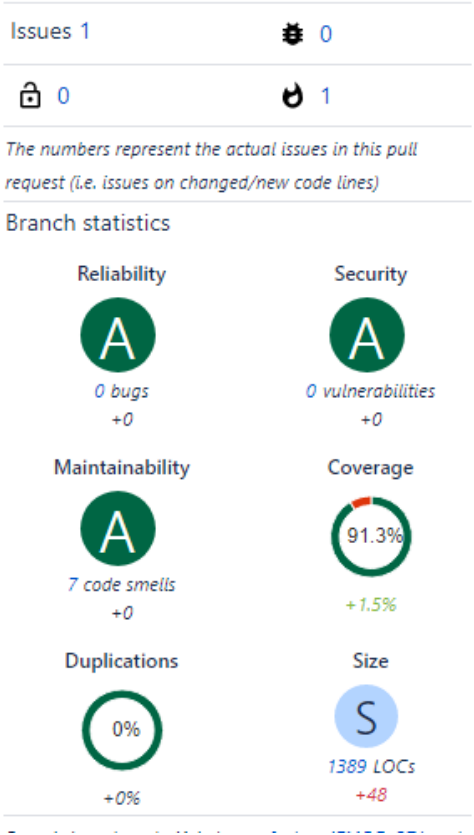
**PERESADA Oleg** commented on a file 30 Dec 2019

Company.WebApi.Core.Configurators.Kestrel / **KestrelConfigurator.cs** OUTDATED

```
38 +      var hostConfiguration = context.Configuration.GetSection(WebApiConfiguration.SectionName)
39 +                                    .Get<WebApiConfiguration>();
40 +
```

# Статический анализ кода

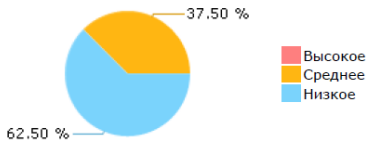


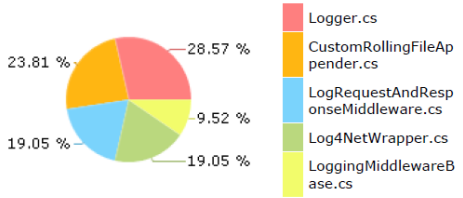
# Анализ уязвимостей

CHECKMARX

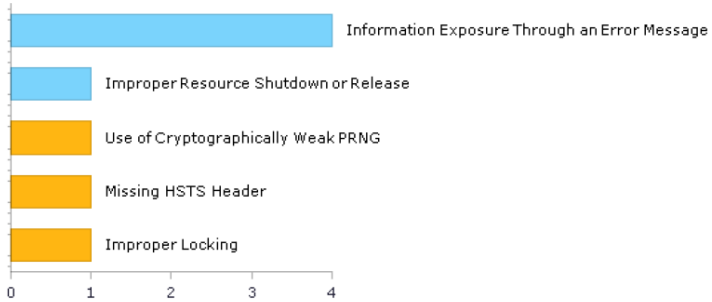
## Сводные данные по результату




## Самые уязвимые файлы

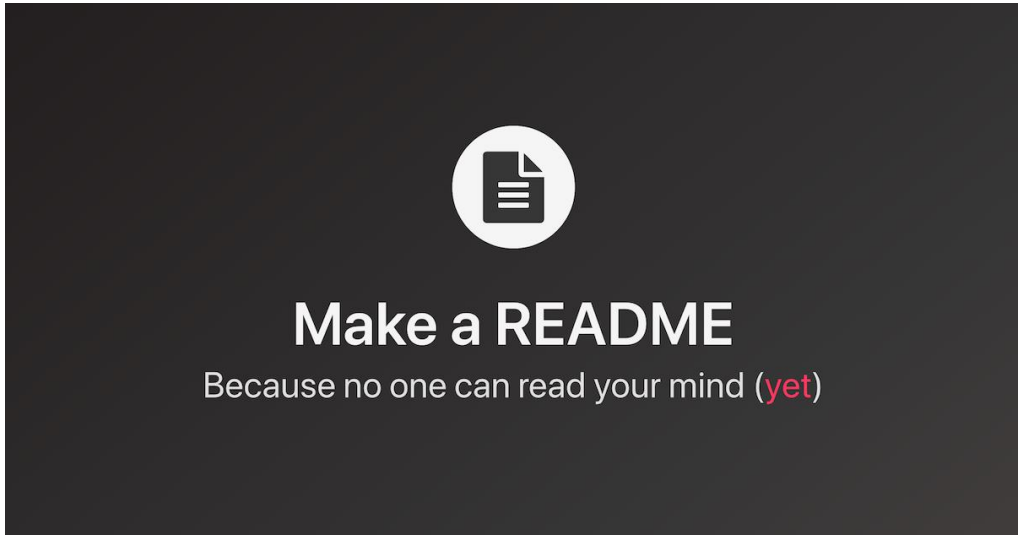



## 5 главных уязвимостей



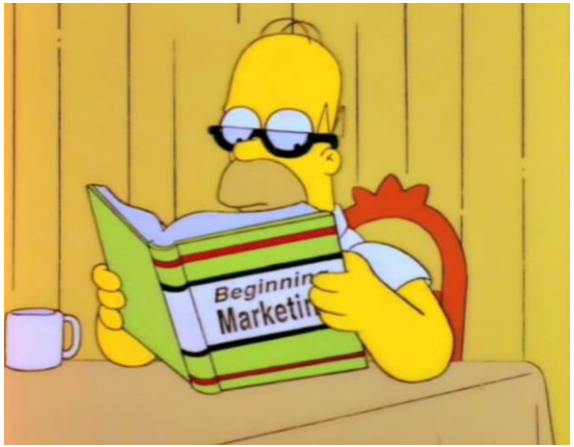
## Идеи для продвижения

Создать хорошую документацию



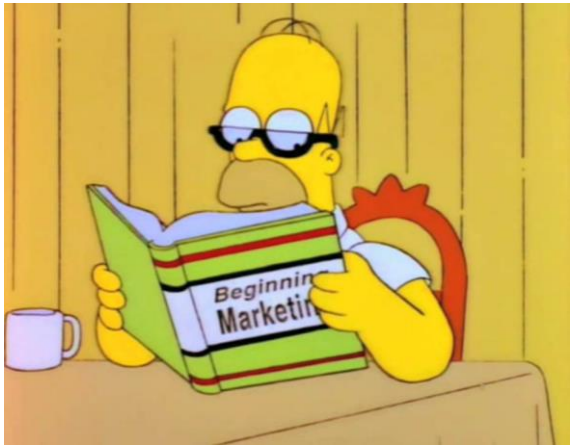
# Состав документации

- XML-документация



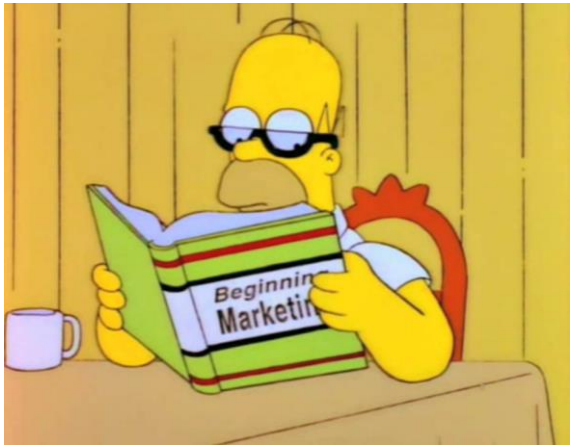
# Состав документации

- XML-документация
- Readme.md



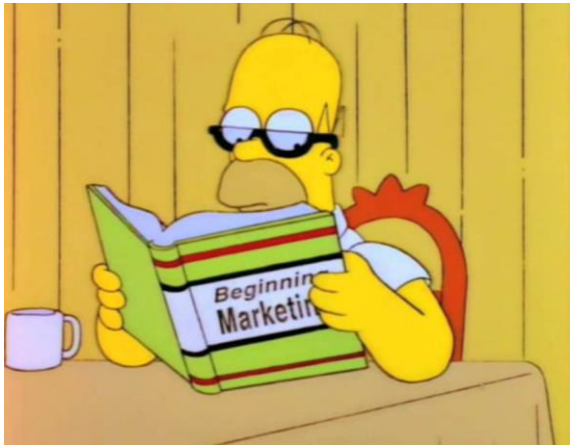
# Состав документации

- XML-документация
- Readme.md
- Страница в Confluence



# Состав документации

- XML-документация
- Readme.md
- Страница в Confluence
- Шаблоны сервисов

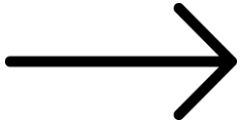


# Open source puzzle


Разработка должна использоваться сообществом

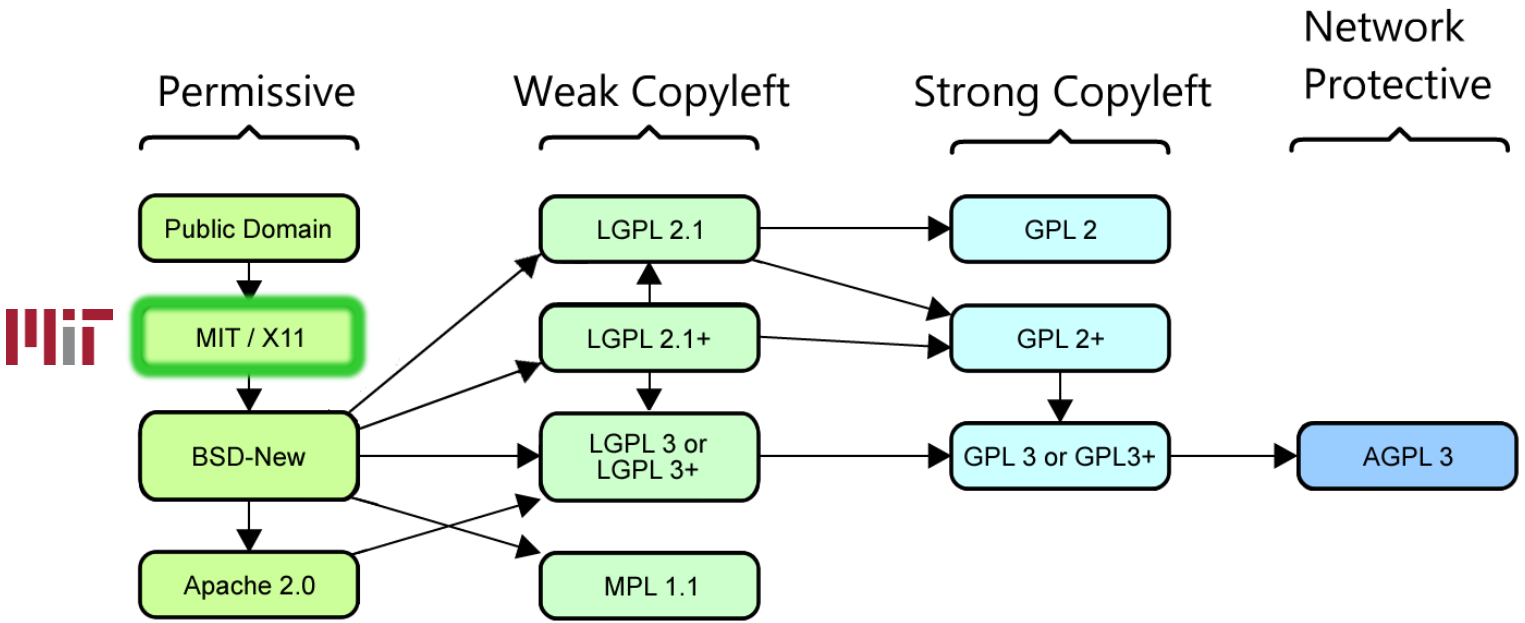
# Что дальше

# Переход от inner к open



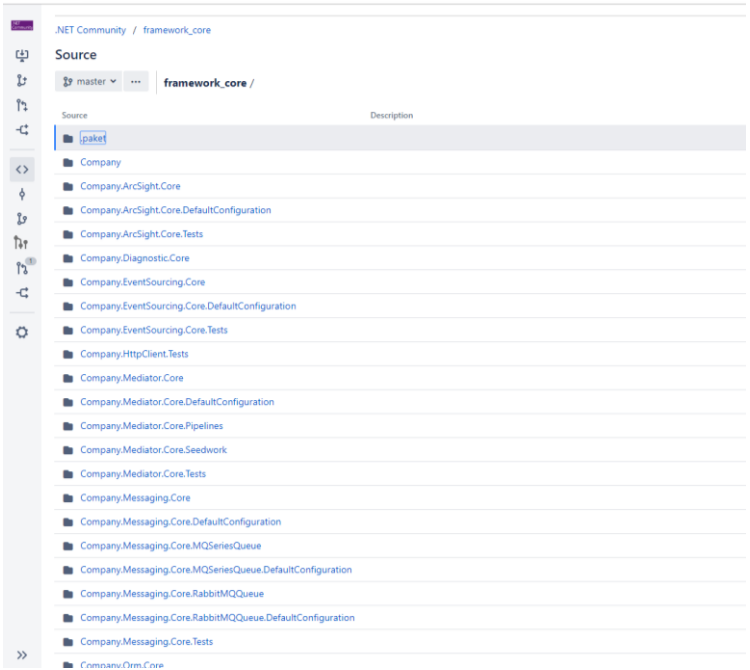
# Лицензия

# Лицензии


Permissive
Weak Copyleft
Strong Copyleft
Network Protective
Public Domain
MIT / X11
BSD-New
Apache 2.0
LGPL 2.1
LGPL 2.1+
LGPL 3 or LGPL 3+
MPL 1.1
GPL 2
GPL 2+
GPL 3 or GPL3+
AGPL 3

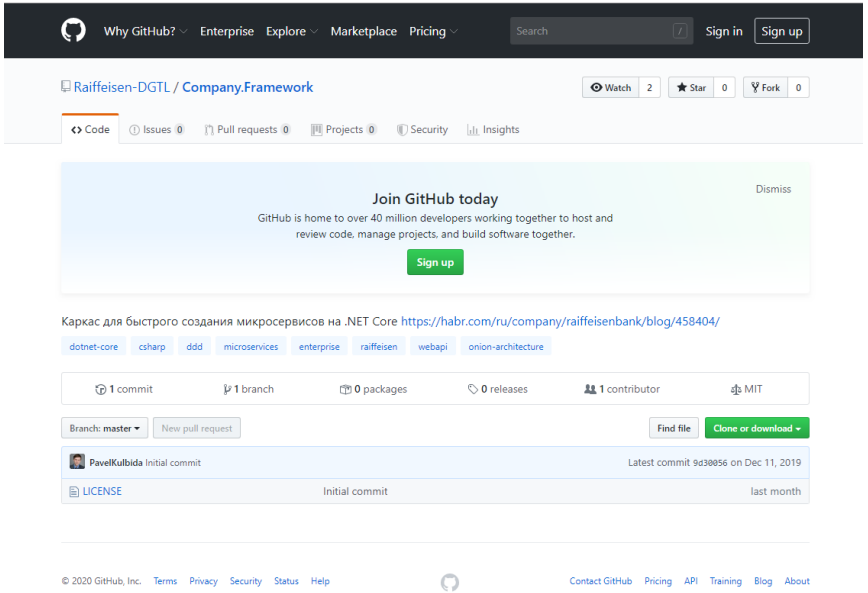
# Управление исходными кодами

# Управление исходными кодами

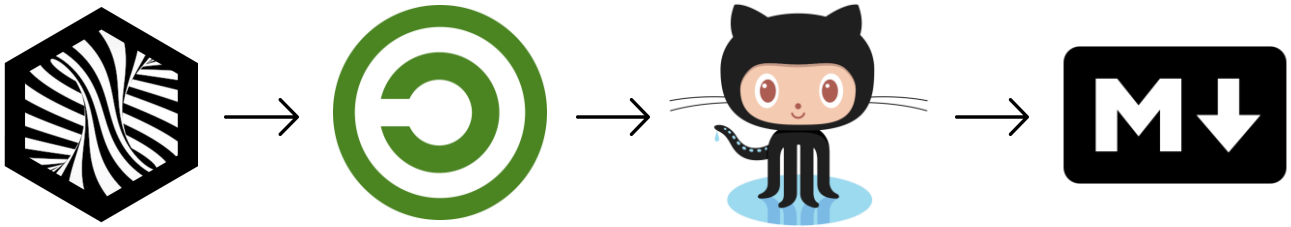



Bitbucket

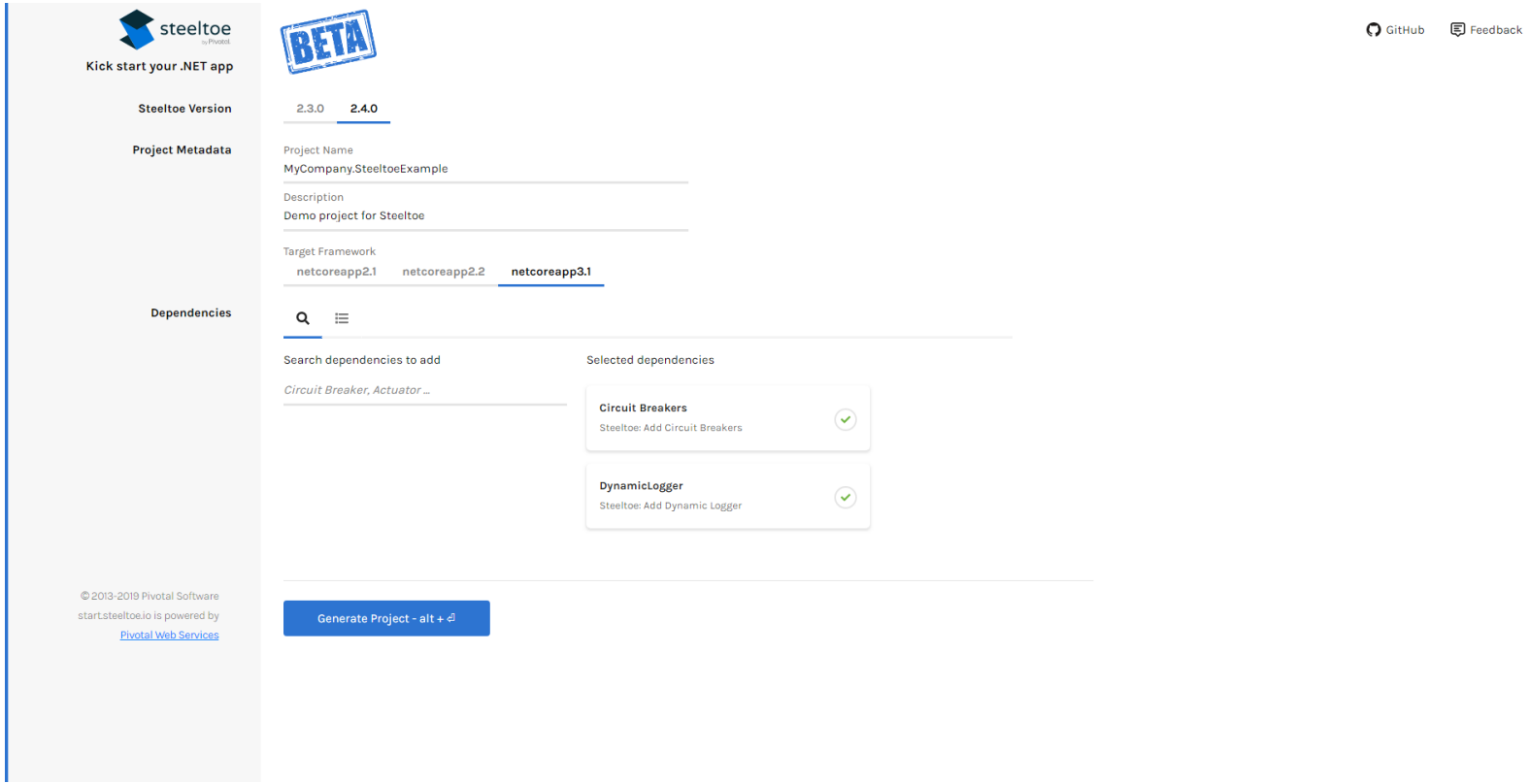
VS



# Документация

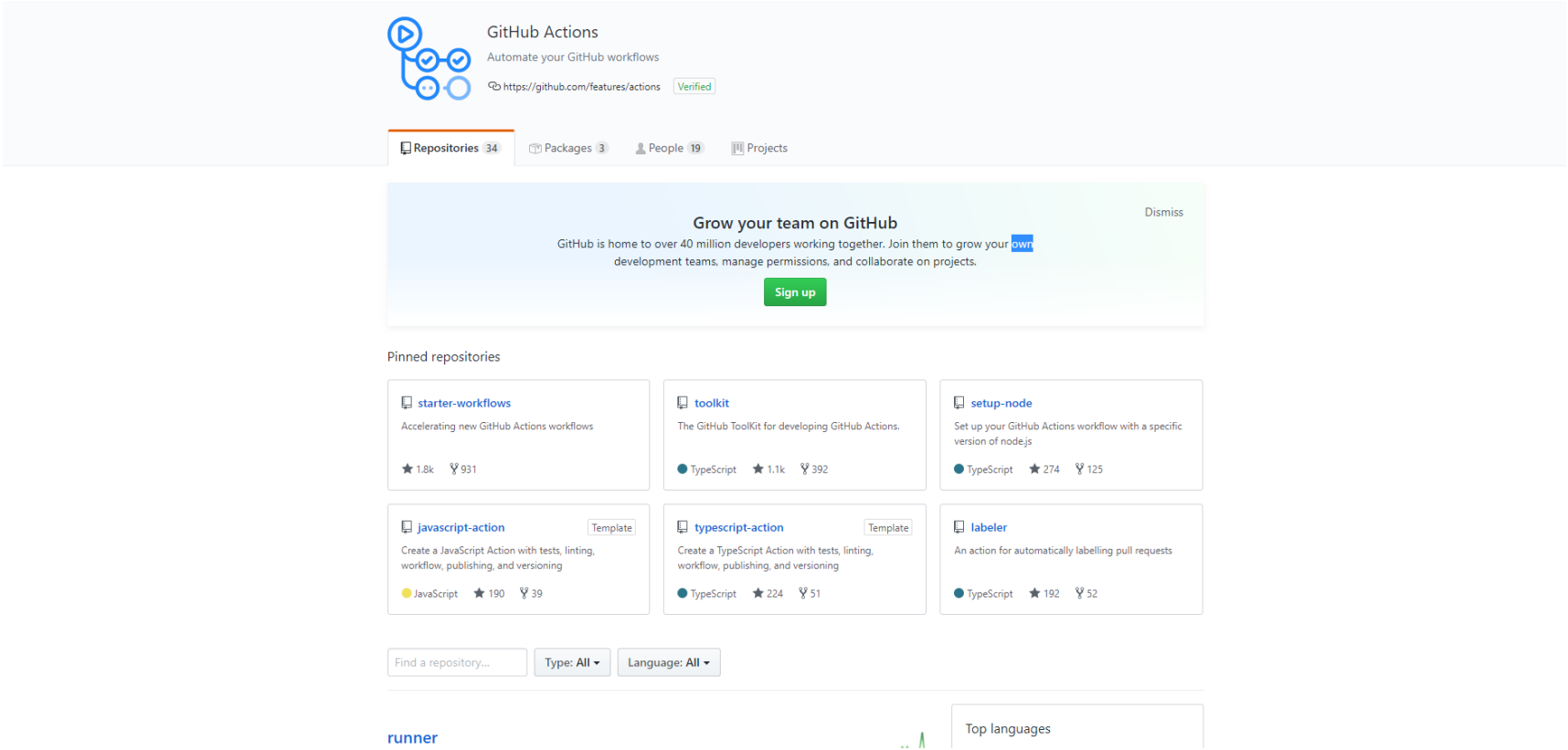


# Построение шаблонов

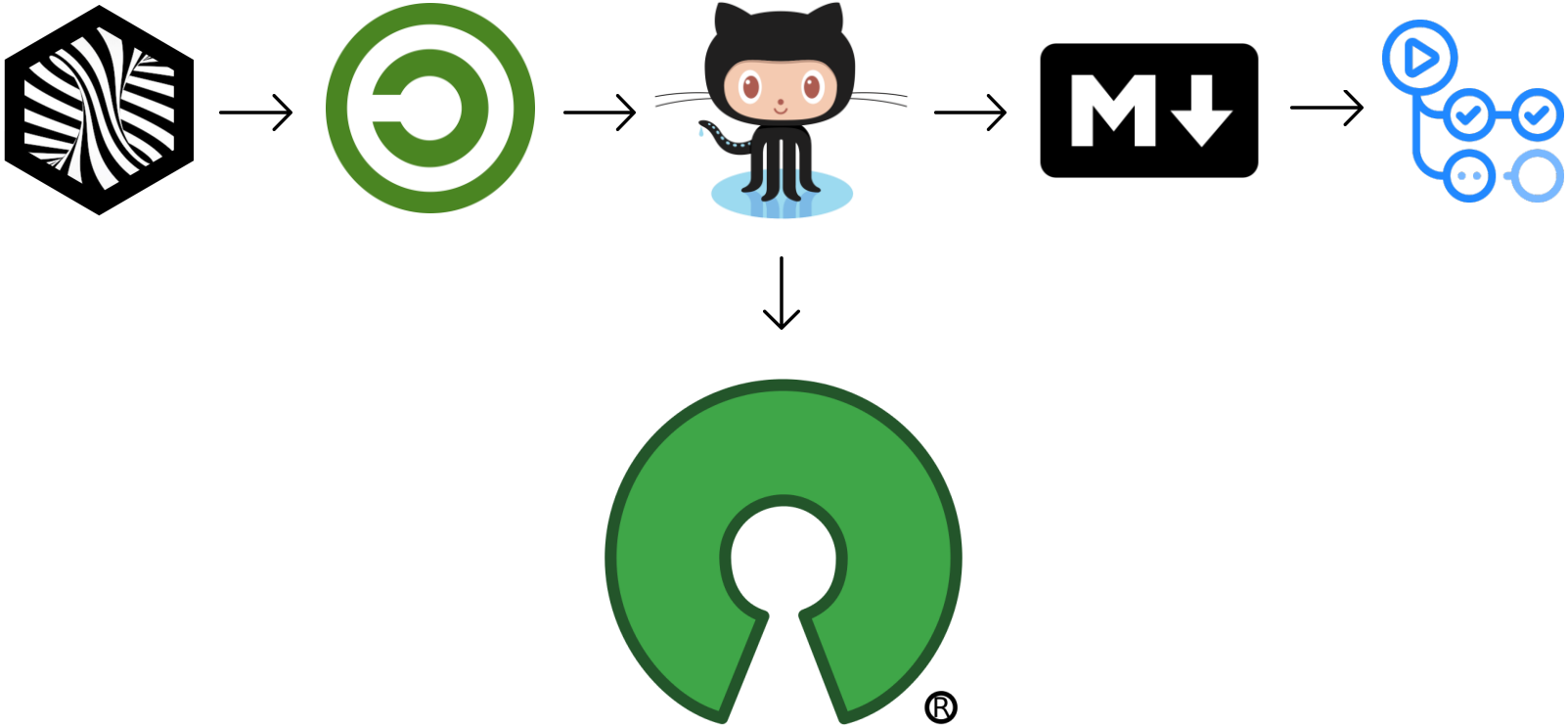


# Автоматизация процессов

# Автоматизация управления



# Open source



# Open source puzzle

Разработка должна быть подготовлена к публикации

# Open source puzzle


Разработка должна быть свободна от ключевой бизнес-логики компании

Разработка должна иметь ценность, как минимум, для компании

Разработка должна использоваться сообществом

Разработка должна быть подготовлена к публикации

## Резюме

- Внутри компаний есть потенциал для создания open source проектов
- Подходы inner source позволяют повысить качество внутренних продуктов
- Для развития open source требуется наличие внутреннего community
- Без хорошей документации привлечение внешних контрибьюторов почти невозможно
- Open source – логичный шаг развития agile-подходов в компании

# Спасибо!

konst.gustov@gmail.com