TDD в кровавом энтерпрайзе

История одного моста

Сначала вопрос

Как можно тестировать?

Результат – только видимые извне итоги работы системы, не обладая знаниями о реализации.

Поведение – система имеет зависимости, взаимодействие с которыми можно отследить и протестировать.

Контракты, интерфейсы, поверхность – аналогично результату.

Немного теории тестирования

Когда тестировать

В ОБЩЕМ СЛУЧАЕ

Высокая стоимость ошибки

Поведение не очевидно

Требуется подтверждение результата работы

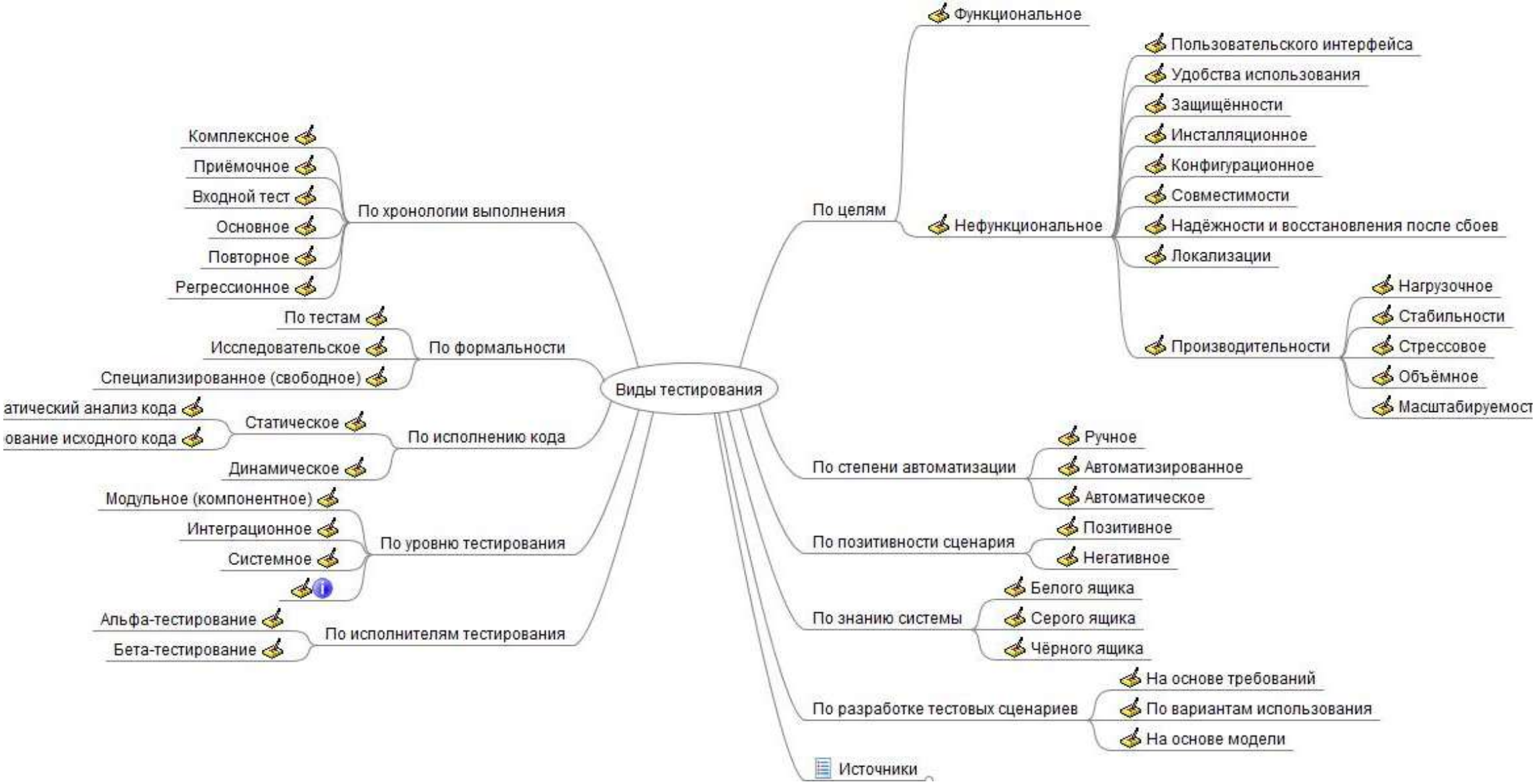
АВТОМАТИЗИРУЯ

Высокая частота релизов

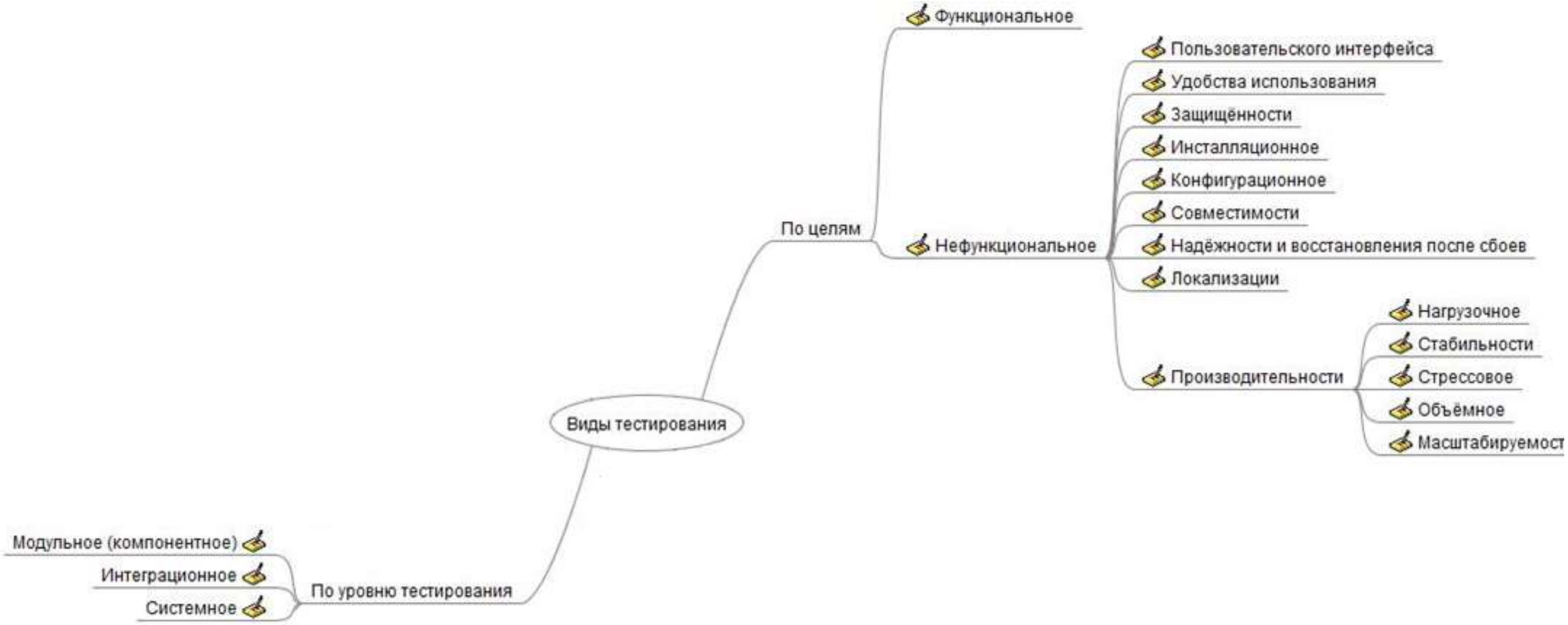
Контракты стабилизированы

CI

Виды тестирования



https://github.com/polarnik/TypesOfTesting


Виды тестирования
По целям
Функциональное
Нефункциональное
Пользовательского интерфейса
Удобства использования
Защищённости
Инсталляционное
Конфигурационное
Совместимости
Надёжности и восстановления после сбоев
Локализации
Производительности
Нагрузочное
Стабильности
Стрессовое
Объёмное
Масштабируемост
По уровню тестирования
Модульное (компонентное)
Интеграционное
Системное

https://github.com/polarnik/TypesOfTesting

Основные артефакты

Тест план

- Назначение данного плана
- Что тестируется (система, модуль, билд, …)
- Описание рисков
- Тестируемые фичи
- Не тестируемые фичи
- Стратегия тестирования
- Критерии успешного и не успешного завершения тестирования
- Описание окружения
- …

Зачем нужен план?

- Определяет цель тестирования
- Описывает стратегию и необходимый объём работ по тестированию
- Определяет объект тестирования
- Позволяет определить покрытие функциональных требований
- Ограничивает скоуп тестирования
- Описывает исходы тестирования и условия успешного прохождения тестирования

Тест кейс

- Название
- Тестируемые фичи
- Определение входных данных
- Определение результата выполнения
- Серьёзность и приоритет бага
- ...

Баги

СЕРЬЁЗНОСТЬ

Блокирующий

Критический

Значительный

Незначительный

Тривиальный

ПРИОРИТЕТ

Высокий

Средний

Низкий

Практика

11 СЛАЙДОВ БЕЗ СТРОЧКИ КОДА – ГДЕ ЖЕ ТУТ TEST DRIVEN?

Проблема

Когда закончить писать код?

Test-driven development требует определить критерий полноты выполнения требования до того, как приступать к реализации.

Определим Объект тестирования

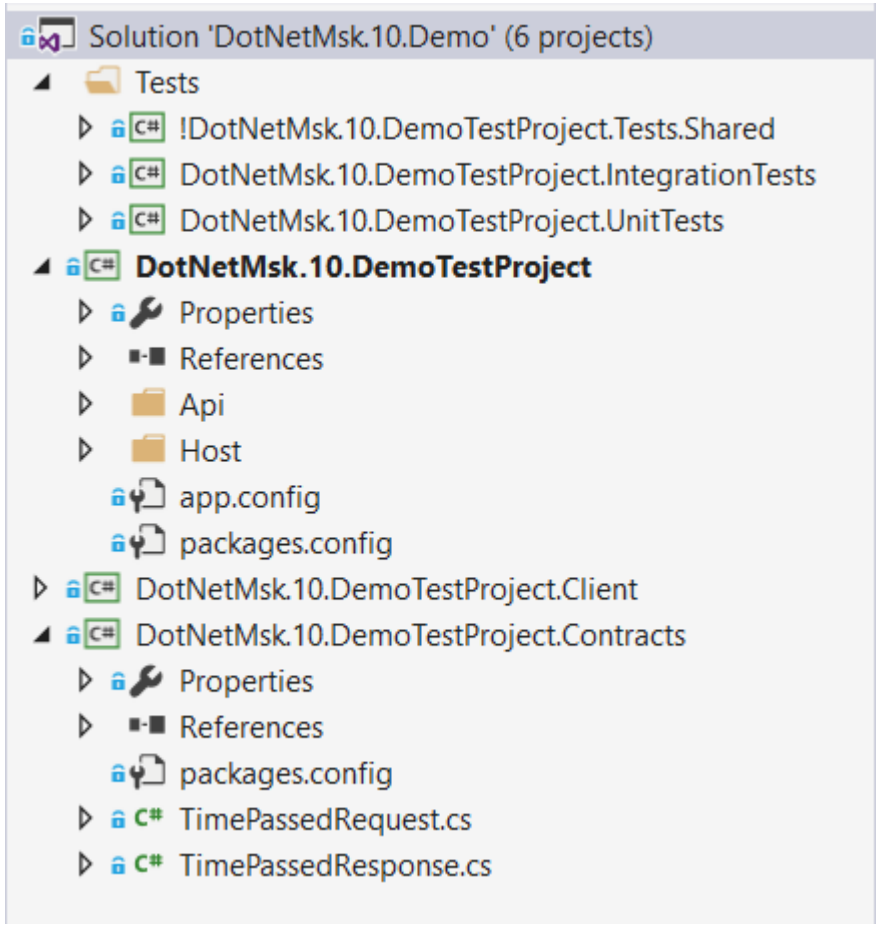
Web API, принимающий на вход дату и возвращающий интервал времени, прошедший (или предшествующий) этой дате с Unix Epoch

Тестируем:

- Обработку входного параметра времени с указанием часового пояса
- Возможный интервал дат

Не тестируем:

- Обработку входного параметра без указания часового пояса
- Даты вне интервала допустимых значений



- DotNetMsk.10.DemoTestProject.Contracts – DTO контракты нашего сервиса
- DotNetMsk.10.DemoTestProject.Client – реализация клиента
- DotNetMsk.10.DemoTestProject – веб-сервис. Его мы и будем тестировать
- DotNetMsk.10.DemoTestProject.IntegrationTests, DotNetMsk.10.DemoTestProject.UnitTests – проекты с тестами
- DotNetMsk.10.DemoTestProject.Tests.Shared – общий код для тестов. В нём находятся SeverityAttribute и PriorityAttribute классы

Первый тест

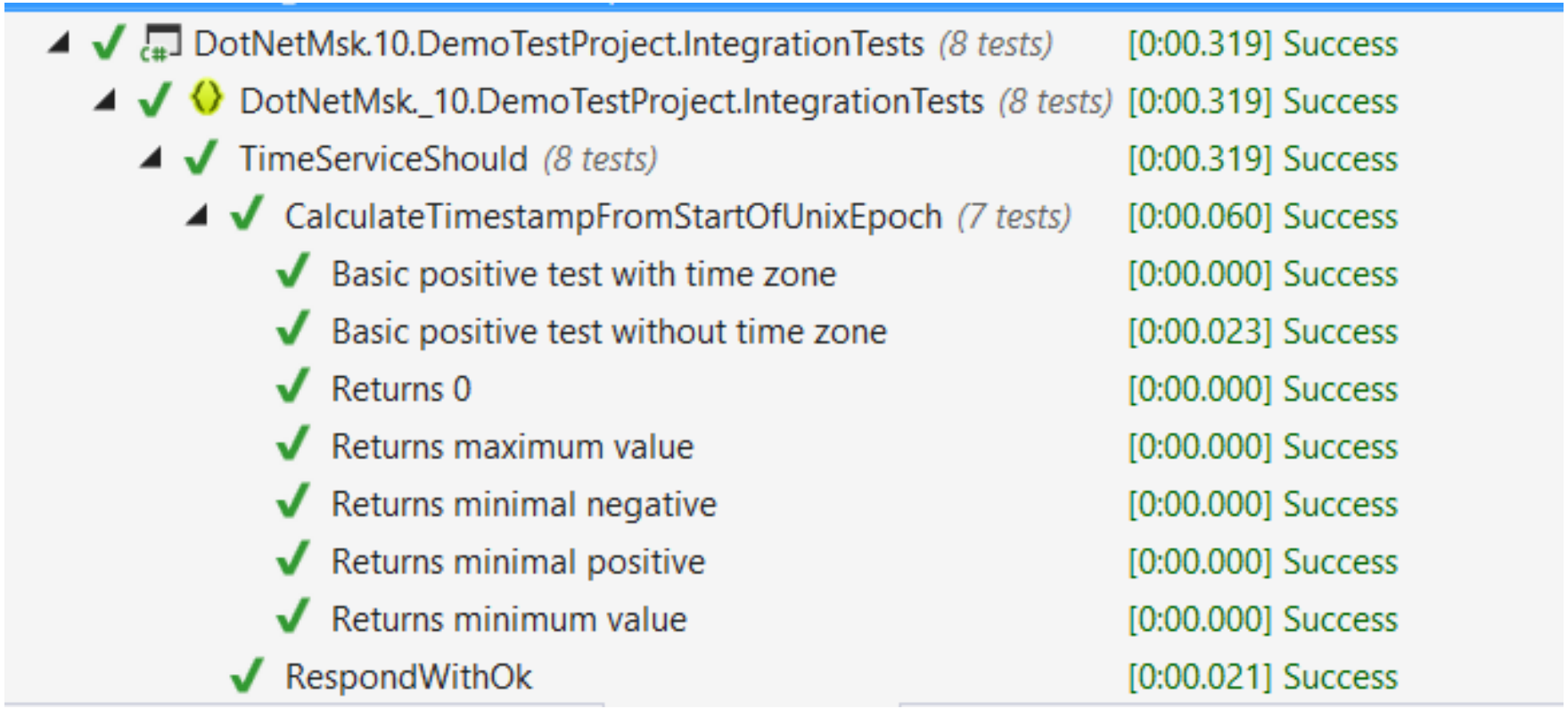
```
[TestFixture]
public class TimeServiceShould
{
    private readonly string _baseUrl;

    public TimeServiceShould()
    {
        _baseUrl = ConfigurationManager.AppSettings["api.baseUrl"];

        _baseUrl.Should().NotBeNullOrWhiteSpace();
    }

    private ITimerServiceClient Client => new TimerServiceClient(_baseUrl);

    [Test]
    [Severity(Severity.Blocker)]
    public async Task RespondWithOk()
    {
        var response = await new RestClient(_baseUrl).ExecuteTaskAsync(new
              RestRequest(TimerServiceClient.GetTimePassedRoute, Method.HEAD));

        response.Should().NotBeNull();
        response.Headers.Should().NotBeNullOrEmpty();
        response.Headers.Should().Contain(p => p.Name.Equals("Allow",
              StringComparison.InvariantCultureIgnoreCase));
    }
}
```

Тестируем основной функционал

```
[Severity(Severity.Critical)]
[TestCaseSource(nameof(CalculateTimestampFromStartOfUnixEpoch_DataSource))]
public async Task<string> CalculateTimestampFromStartOfUnixEpoch(string
      tillDate)
{
    return (await Client.GetTimePassedTillDateAsync(DateTime.Parse(tillDate)))
       .ToString();
}
```

Запускаем – всё красное

Добавим немного кода ☺



Http stubs

Карманный веб-сервер

```
[TestFixture]
public class TimerClientShould : IDisposable
{
    private readonly IHttpServer _httpMock;
    private readonly string _baseUrl;

    public TimerClientShould()
    {
        _httpMock = HttpMockServer.LaunchTimeStub(ConfigurationManager.AppSettings["stub.baseUrl"]);
        _baseUrl = ConfigurationManager.AppSettings["stub.baseUrl"];

        _baseUrl.Should().NotBeNullOrWhiteSpace();
    }

    [Severity(Severity.Critical)]
    [TestCase("2017-01-01T00:00:00.0000000Z", Description = "Basic positive test without time zone",
              ExpectedResult = "17166.21:00:00")]
    public async Task<string> CalculateTimestampFromStartOfUnixEpoch(string tillDate)
    {
        var sut = new TimerServiceClient(_baseUrl);

        return (await sut.GetTimePassedTillDateAsync(DateTime.Parse(tillDate))).ToString();
    }

    public void Dispose() => _httpMock?.Dispose();
}
```

```
public static class HttpMockServer
{
    public static IHttpServer LaunchTimeStub(string baseAddress)
    {
        var result = new HttpServer(new Uri(baseAddress));

        //Setup error routes for test purposes
        result.SetupErrorStubMethod(HttpStatusCode.BadRequest)
            .SetupErrorStubMethod(HttpStatusCode.Unauthorized)
            .SetupErrorStubMethod(HttpStatusCode.NotFound)
            .SetupErrorStubMethod(HttpStatusCode.UnsupportedMediaType)
            .SetupErrorStubMethod(HttpStatusCode.InternalServerError)
            .SetupErrorStubMethod(HttpStatusCode.NotImplemented);

        //Setup actual methods
        result.Stub(rf => rf.Post($"/{TimerServiceClient.GetTimePassedTillDateRoute}"))
            .Return(JsonConvert.SerializeObject(new TimePassedResponse
            {
                TimePassed = TimeSpan.Parse("17166.21:00:00"),
                StartingPoint = new DateTime(1970, 1, 1)
            }))
            .AsContentType("application/json")
            .WithStatus(HttpStatusCode.OK);

        result.Start();

        return result;
    }
}
```

Обо мне

.NET C# разработчик, тим лид

Skype: lleyte

Email: leyte@live.ru

Facebook: https://facebook.com/an.zaytsev

GitHub repo: https://github.com/zaytsevand/DotNetMsk