Тестирование .NET web-сервиса без деплоя

Артем Сидорук

Kaspersky

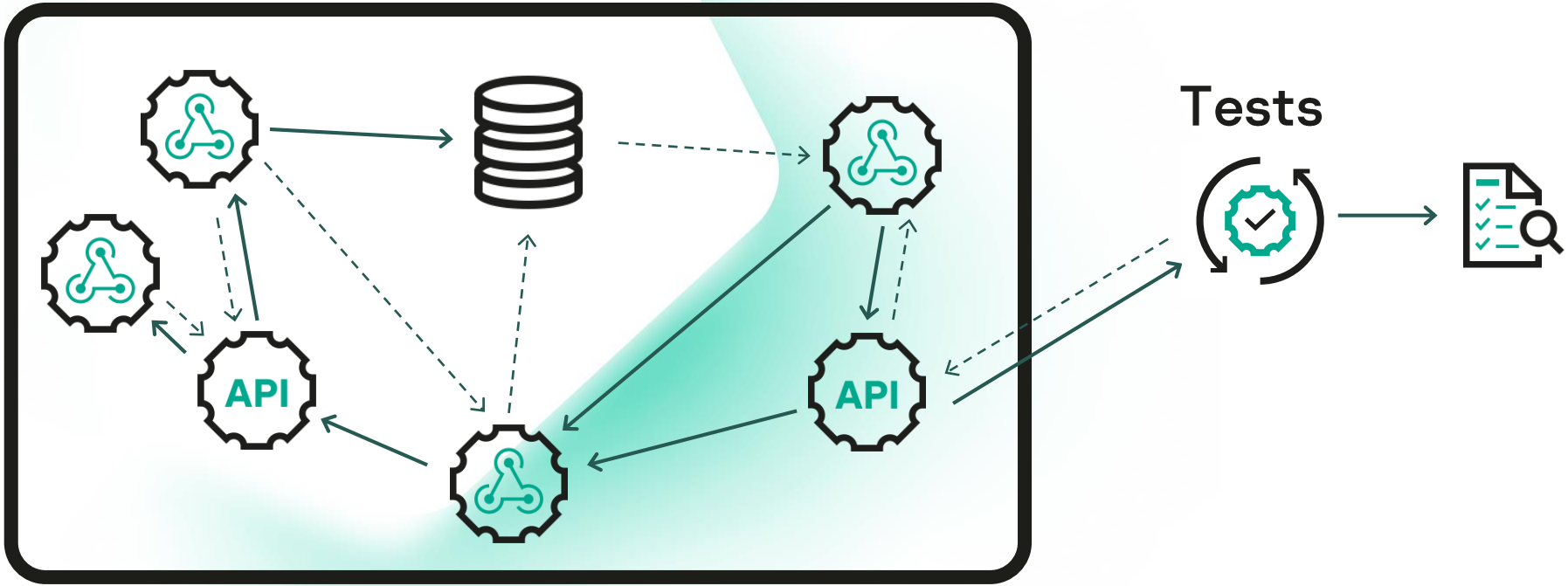
t.me@Sugrob57

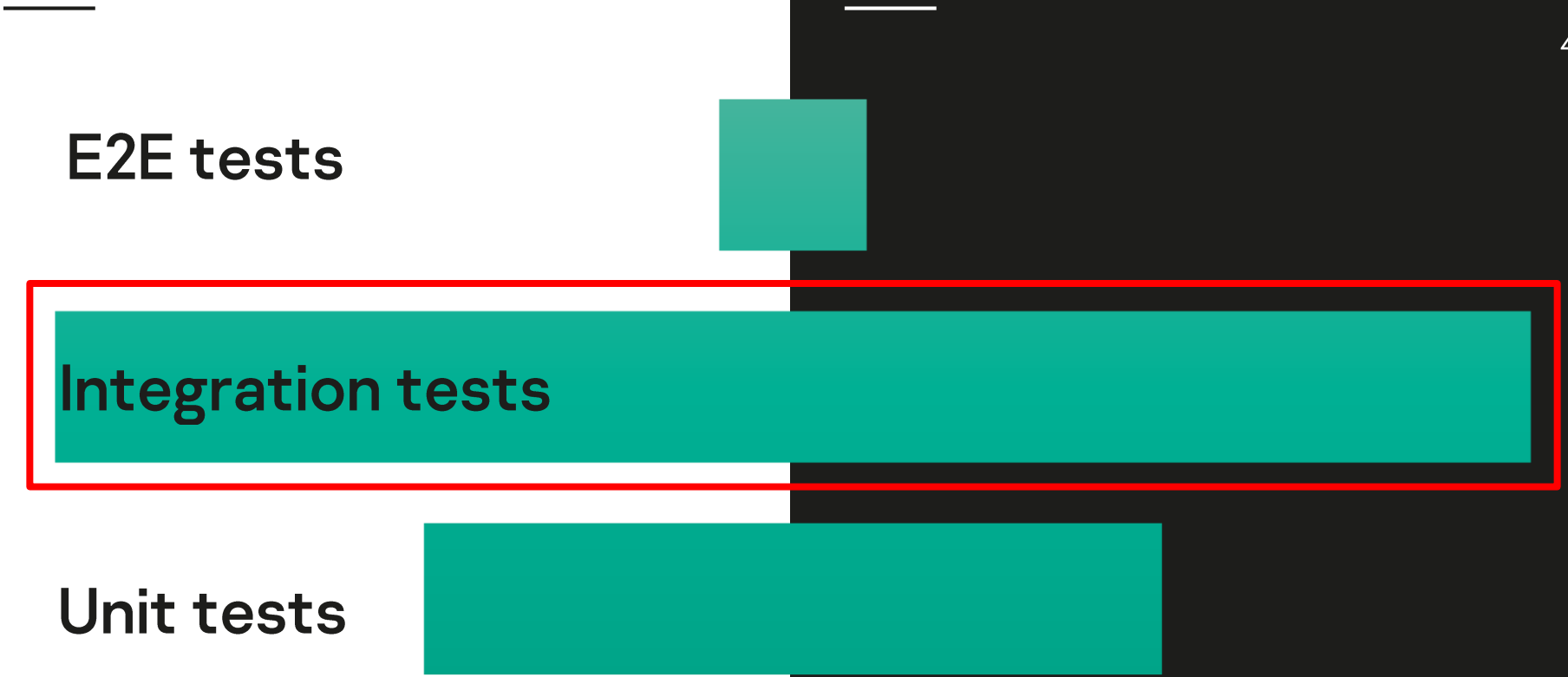
HEISENBUG

План доклада

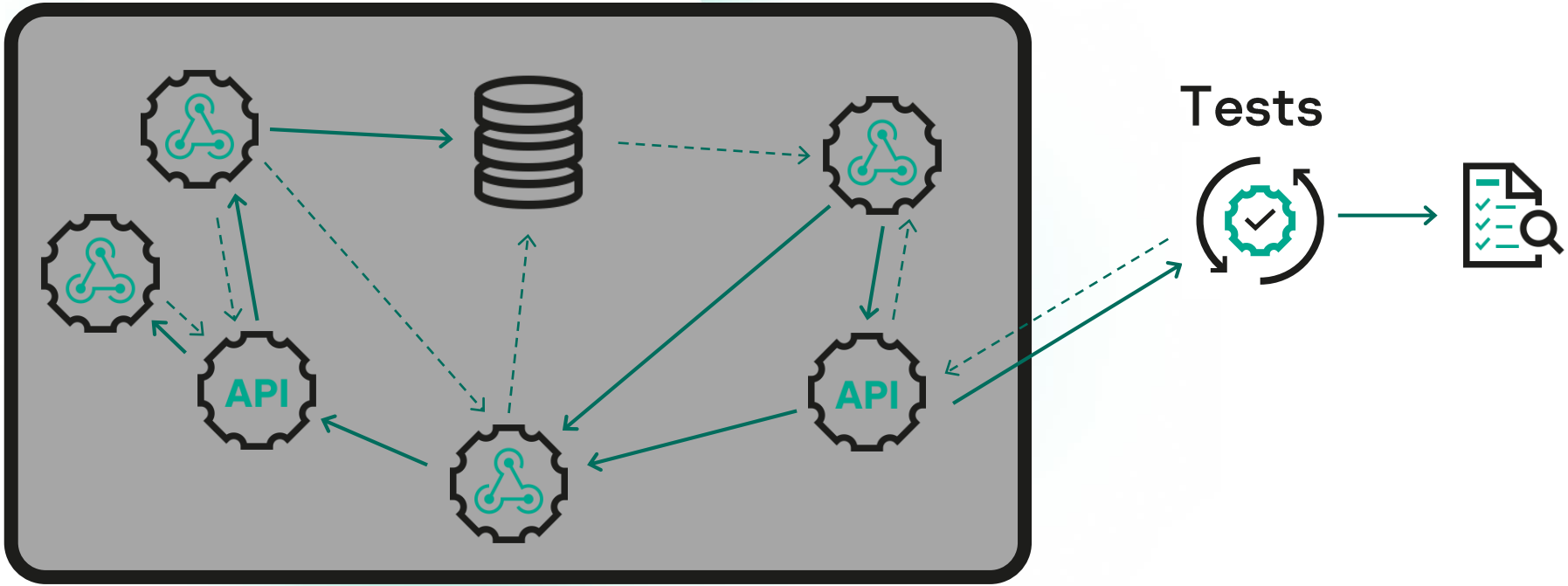
- Сфера применения
- Microsoft.AspNetCore.Mvc.Testing
- Примеры тестов
- Выводы и рекомендации

Integration tests

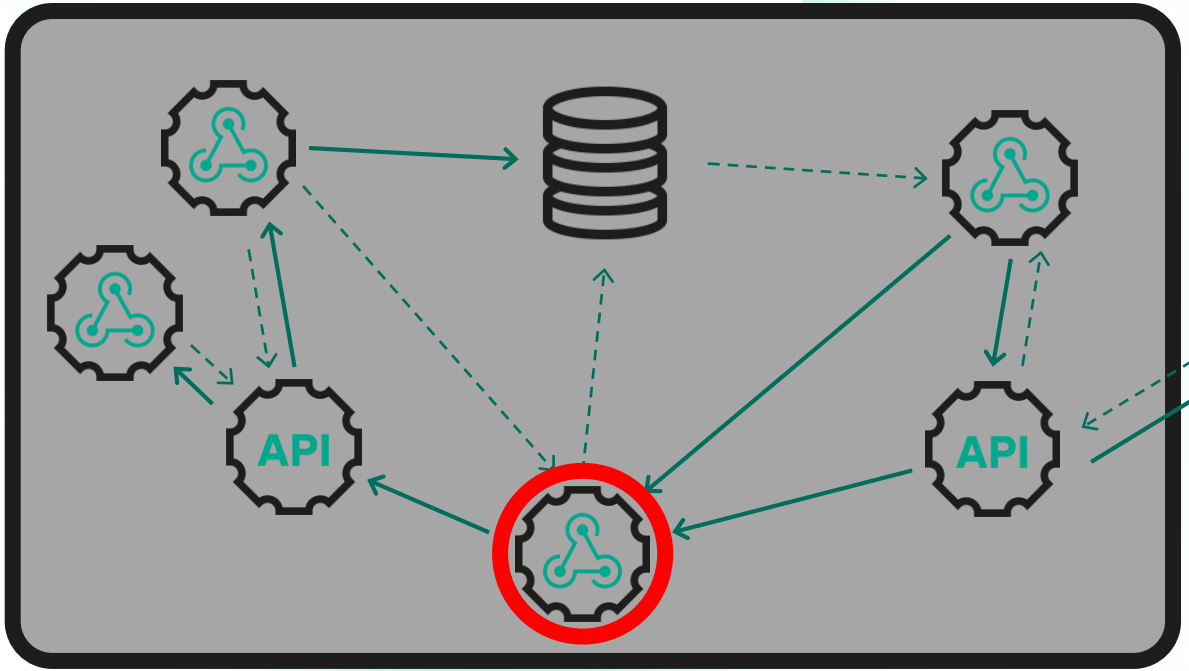


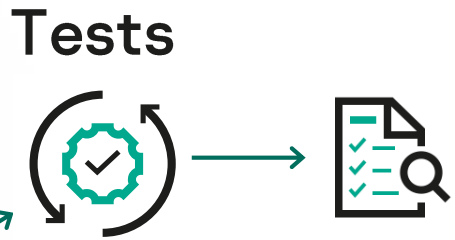

E2E tests
Integration tests
Unit tests

Integration tests

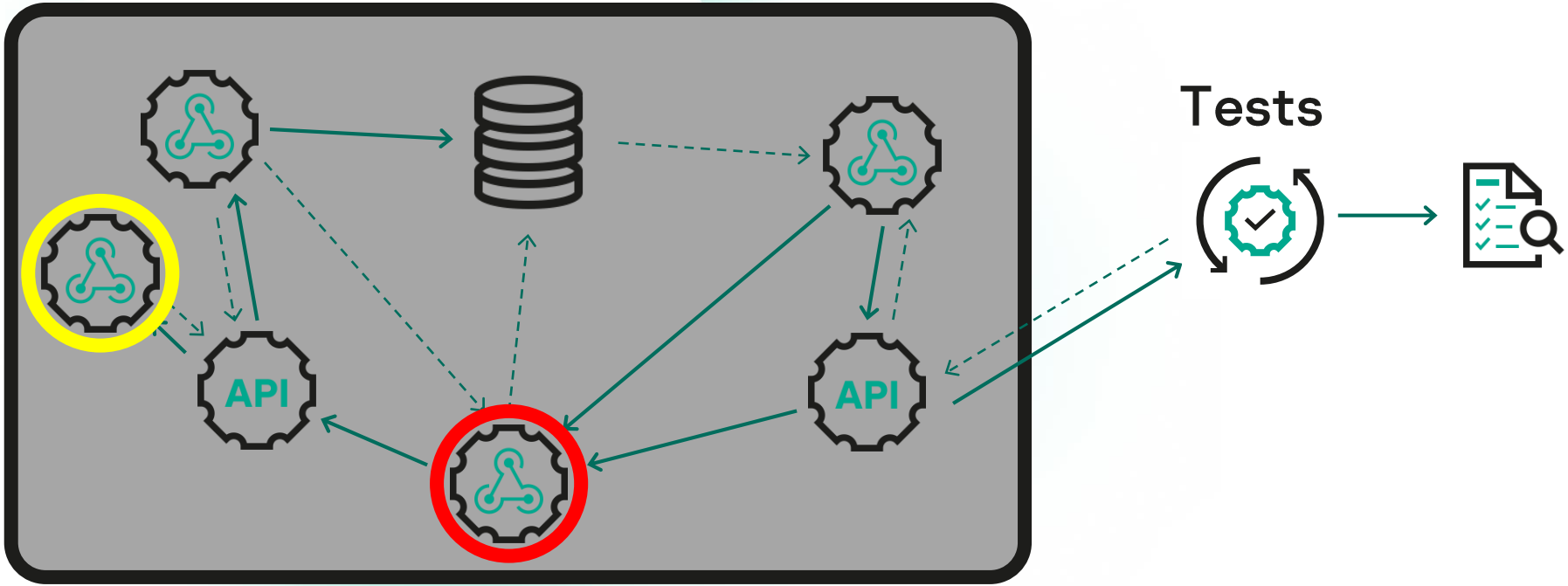


Integration tests

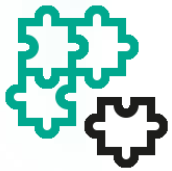




Integration tests

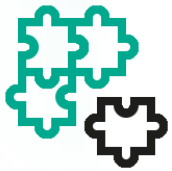


Integration tests



- **Отказ любого сервиса блокирует тесты**
- Если какой-либо из сервисов не работает – блокируется большое количество сценариев тестирования.

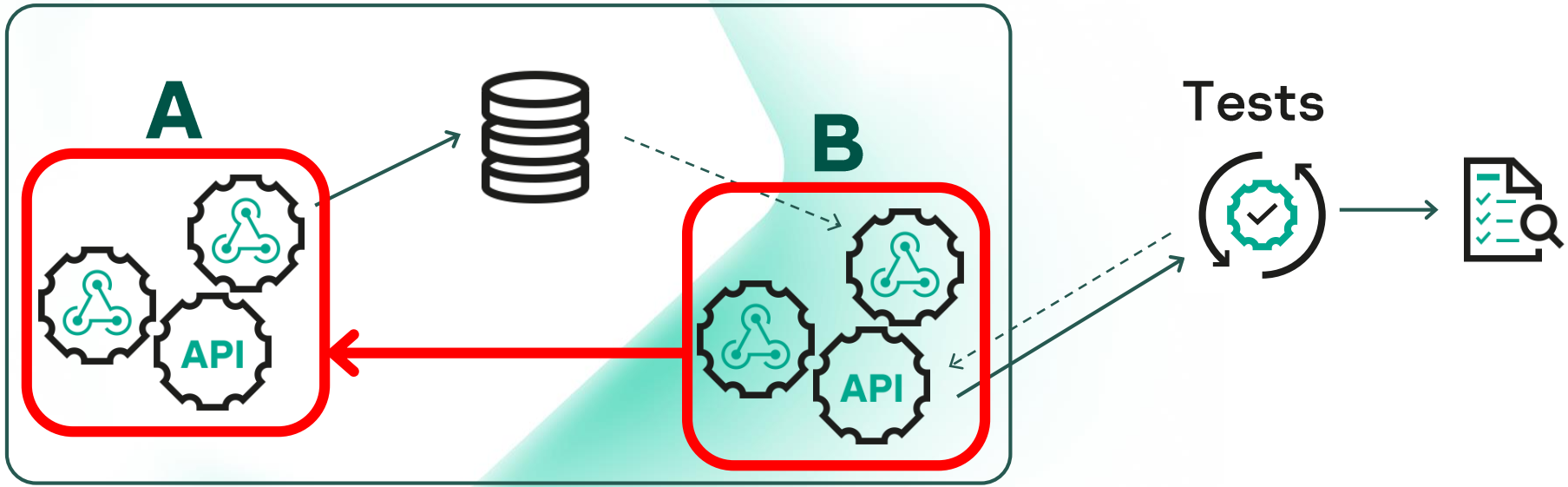
Integration tests



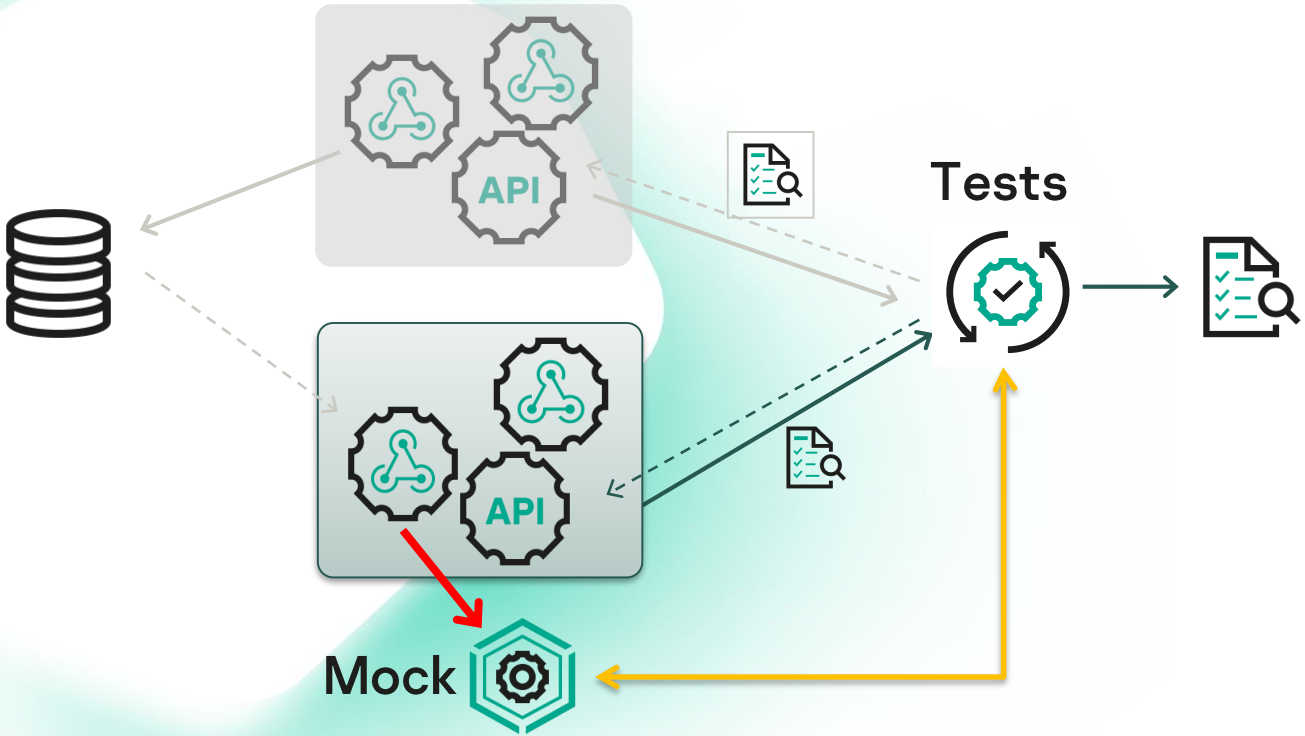
- **Отказ любого сервиса блокирует тесты**
- Если какой-либо из сервисов не работает – блокируется большое количество сценариев тестирования.

- **Долгая обратная связь**
- Изменения нужно задеплоить на интеграционную среду.
- Чтоб что-то проверить нужно пройти весь путь E2E-теста.

Service groups



L2 tests



https://www.youtube.com/watch?v=qrAvhCZMOmE

Классика

У нас

https://www.youtube.com/watch?v=srMAVpheQRw&t=1s

https://habr.com/ru/companies/kaspersky/articles/725348/

L2 tests

- **Деплой стал еще сложнее**
- Раньше просто деплоили все на среду – теперь деплоим выборочно, да еще и с явным созданием временных ресурсов

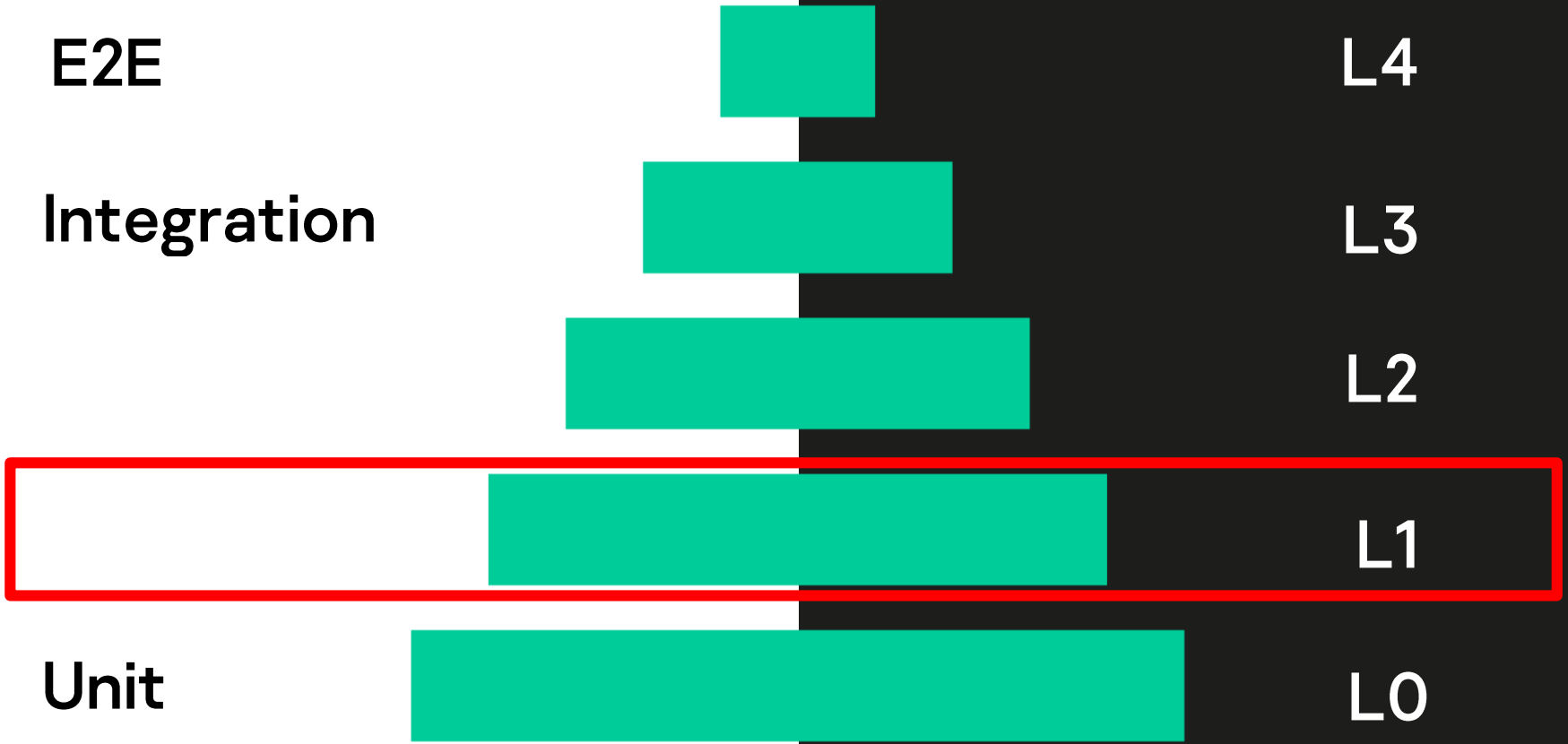
L2 tests

- **Деплой стал еще сложнее**
- Раньше просто деплоили все на среду – теперь деплоим выборочно, да еще и с явным созданием временных ресурсов

- **Локальный запуск нужно настраивать**
- Нужен локально установленный Docker + terraform чтобы что-то отлаживать локально. Можно. Работает. Но есть трудности.

Классика

У нас



L1 tests

L1 тесты – ваш выбор, если:

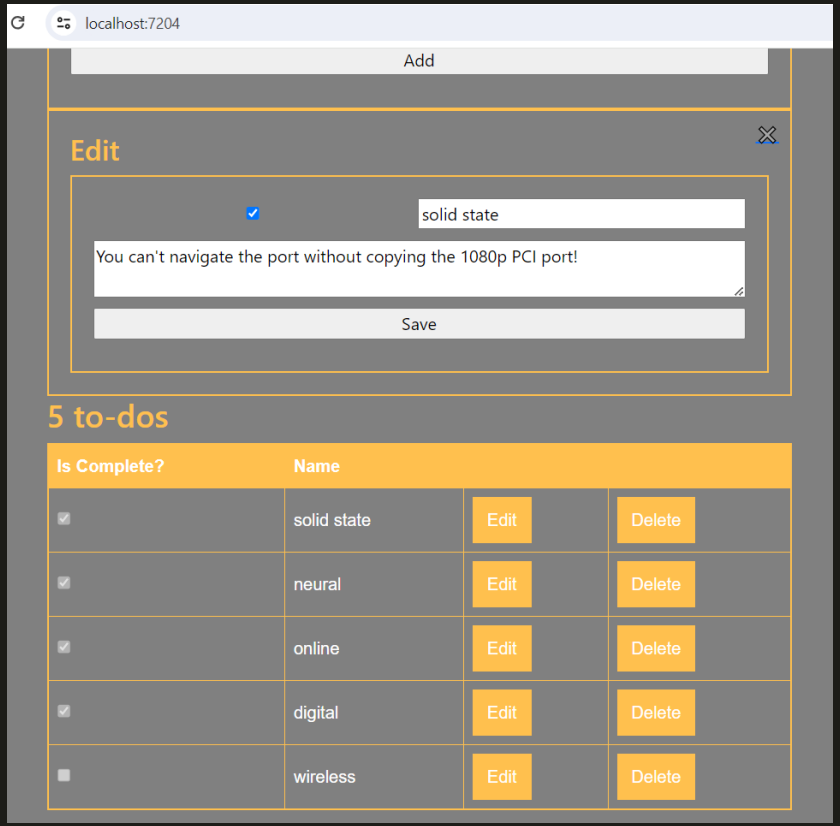
- **Нужна быстрая обратная связь**
- Я тестер, я не хочу ничего никуда деплоить, хочу запускать тесты.
- Я разработчик, я не хочу читать описание багов, хочу сразу дебажить код.

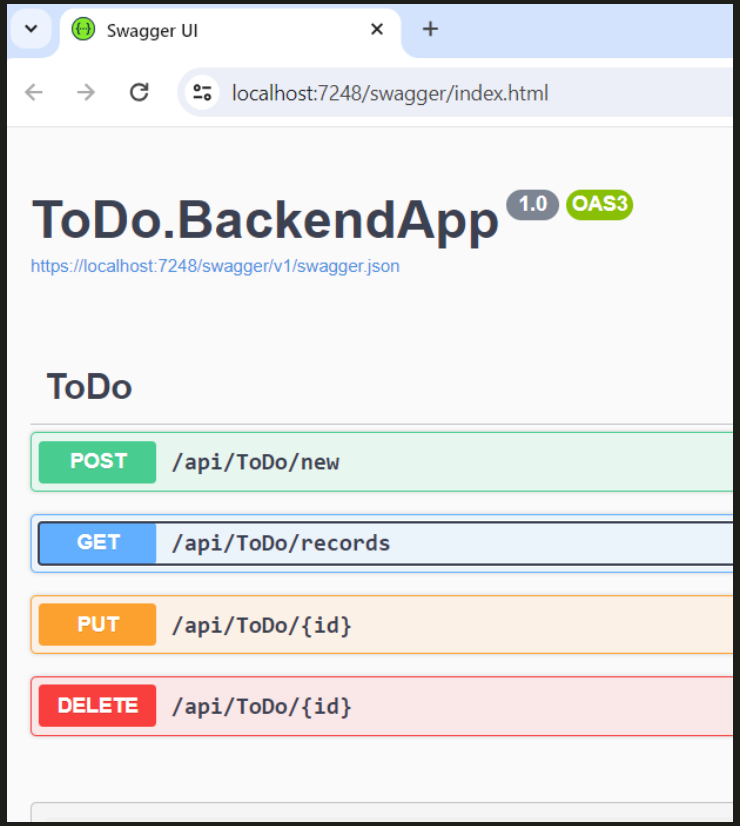
L1 тесты – ваш выбор, если:

- **Нужна быстрая обратная связь**
- Я тестер, я не хочу ничего никуда деплоить, хочу запускать тесты.
- Я разработчик, я не хочу читать описание багов, хочу сразу дебажить код.

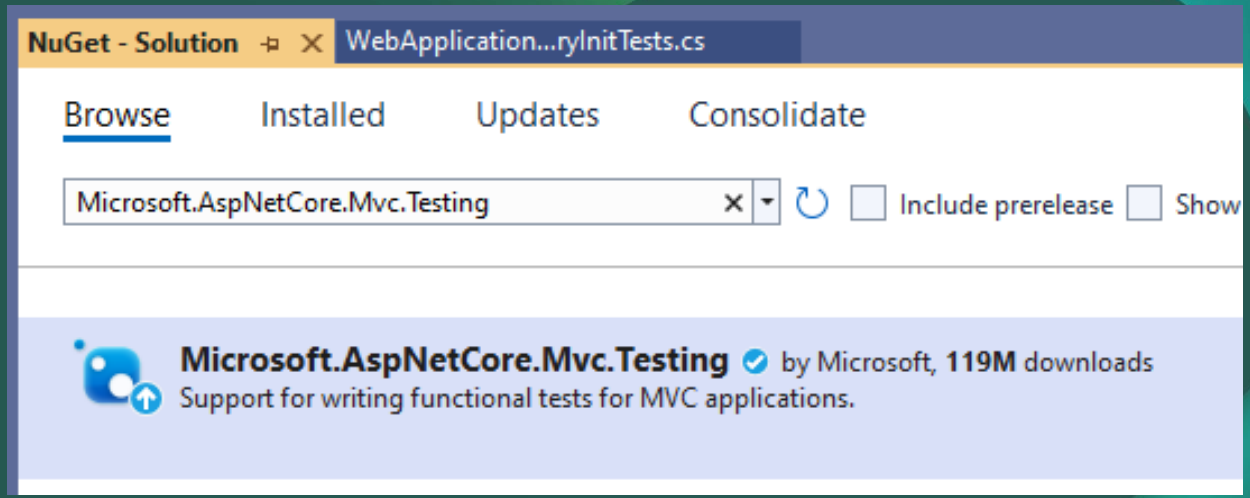
- **Есть сложности с подключением к реальным ресурсам**
- Есть реальная БД к которой не подключиться со своего ПК? Нет денег/возможности/желания держать персональные ресурсы для каждого? Нельзя подключаться, чтобы не «воровать» сообщения из общей очереди? **L1-тестам не нужны реальные ресурсы.**

Services


localhost:7204
Add
Edit
solid state
You can't navigate the port without copying the 1080p PCI port!
Save
5 to-dos
Is Complete?
Name
solid state
Edit
Delete
neural
Edit
Delete
online
Edit
Delete
digital
Edit
Delete
wireless
Edit
Delete


Swagger UI
localhost:7248/swagger/index.html
ToDo.BackendApp
1.0
OAS3
https://localhost:7248/swagger/v1/swagger.json
ToDo
POST
/api/ToDo/new
GET
/api/ToDo/records
PUT
/api/ToDo/{id}
DELETE
/api/ToDo/{id}

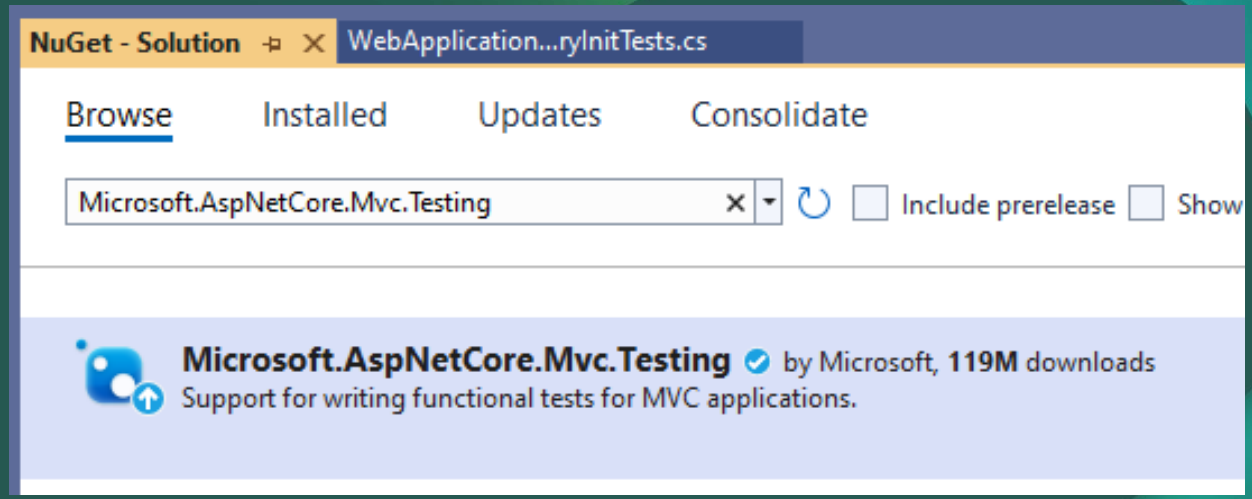
Microsoft.AspNetCore.Mvc.Testing



Примеры от MS: https://github.com/dotnet/AspNetCore.Docs.Samples/blob/main/fundamentals/minimal-apis/samples/MinApiTestsSample/IntegrationTests/TodoEndpointsV2Tests.cs

Реализация: https://github.com/dotnet/aspnetcore/tree/main/src/Hosting

Microsoft.AspNetCore.Mvc.Testing

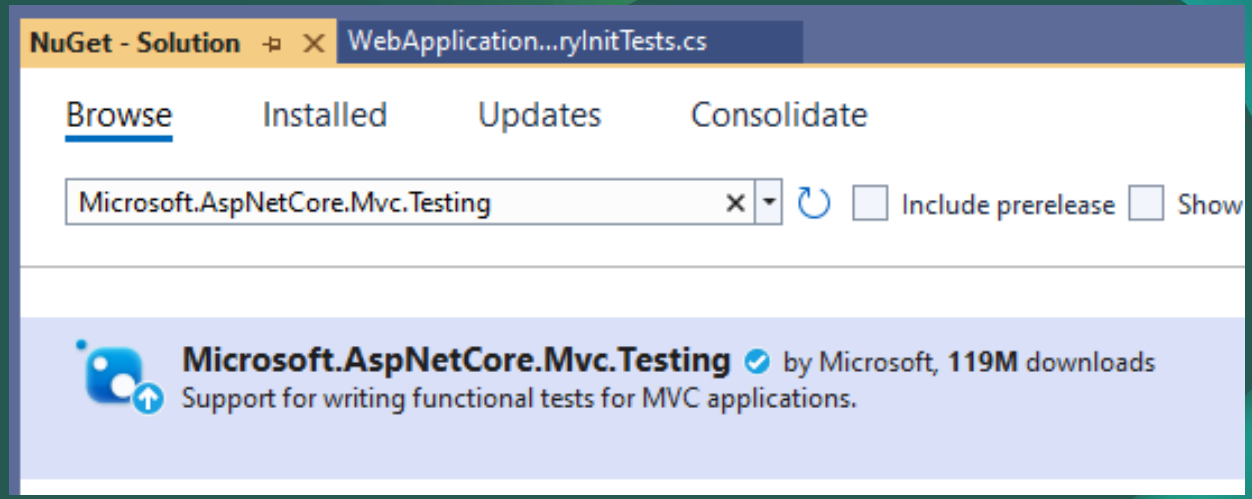


TestServer

Примеры от MS: https://github.com/dotnet/AspNetCore.Docs.Samples/blob/main/fundamentals/minimal-apis/samples/MinApiTestsSample/IntegrationTests/TodoEndpointsV2Tests.cs

Реализация: https://github.com/dotnet/aspnetcore/tree/main/src/Hosting

Microsoft.AspNetCore.Mvc.Testing



TestServer

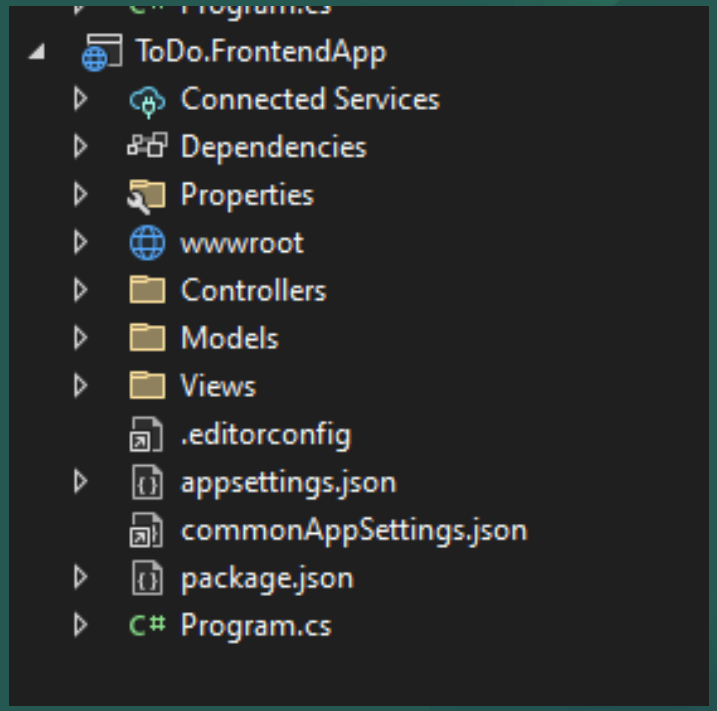
TestHost

Примеры от MS: https://github.com/dotnet/AspNetCore.Docs.Samples/blob/main/fundamentals/minimal-apis/samples/MinApiTestsSample/IntegrationTests/TodoEndpointsV2Tests.cs

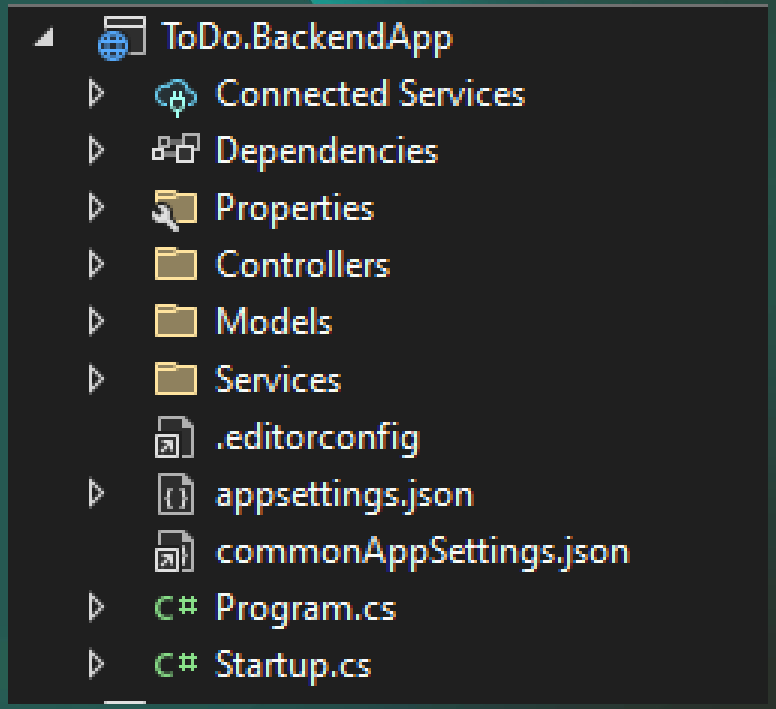
Реализация: https://github.com/dotnet/aspnetcore/tree/main/src/Hosting

Solution

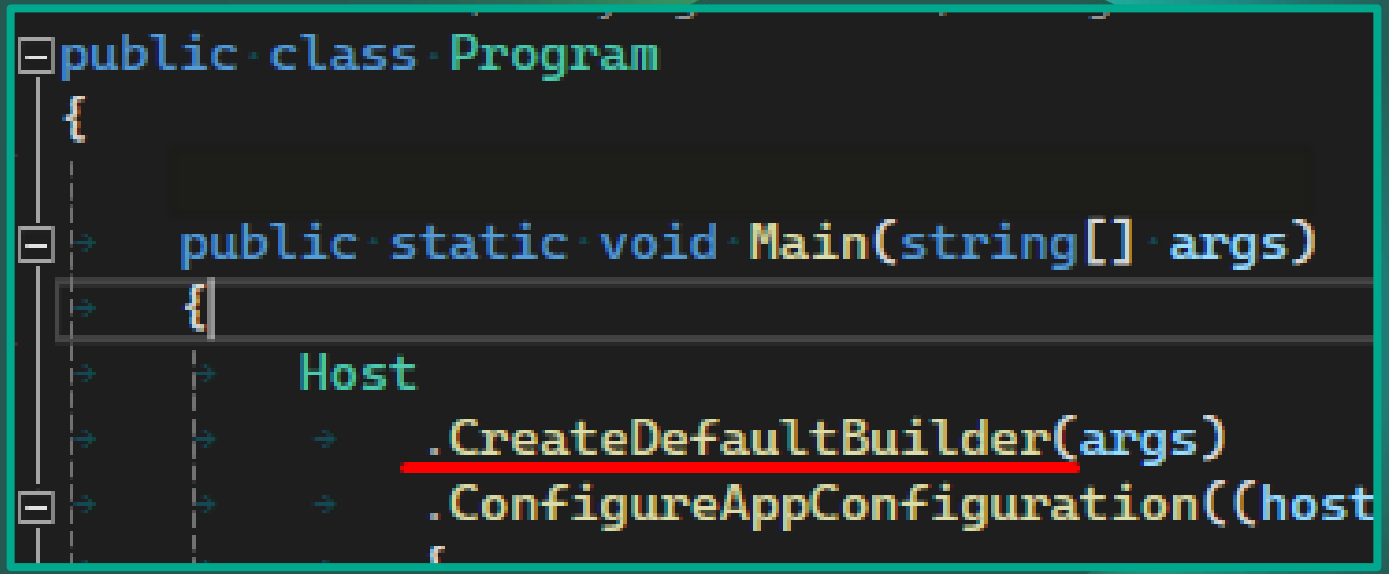
Todo.FrontendApp



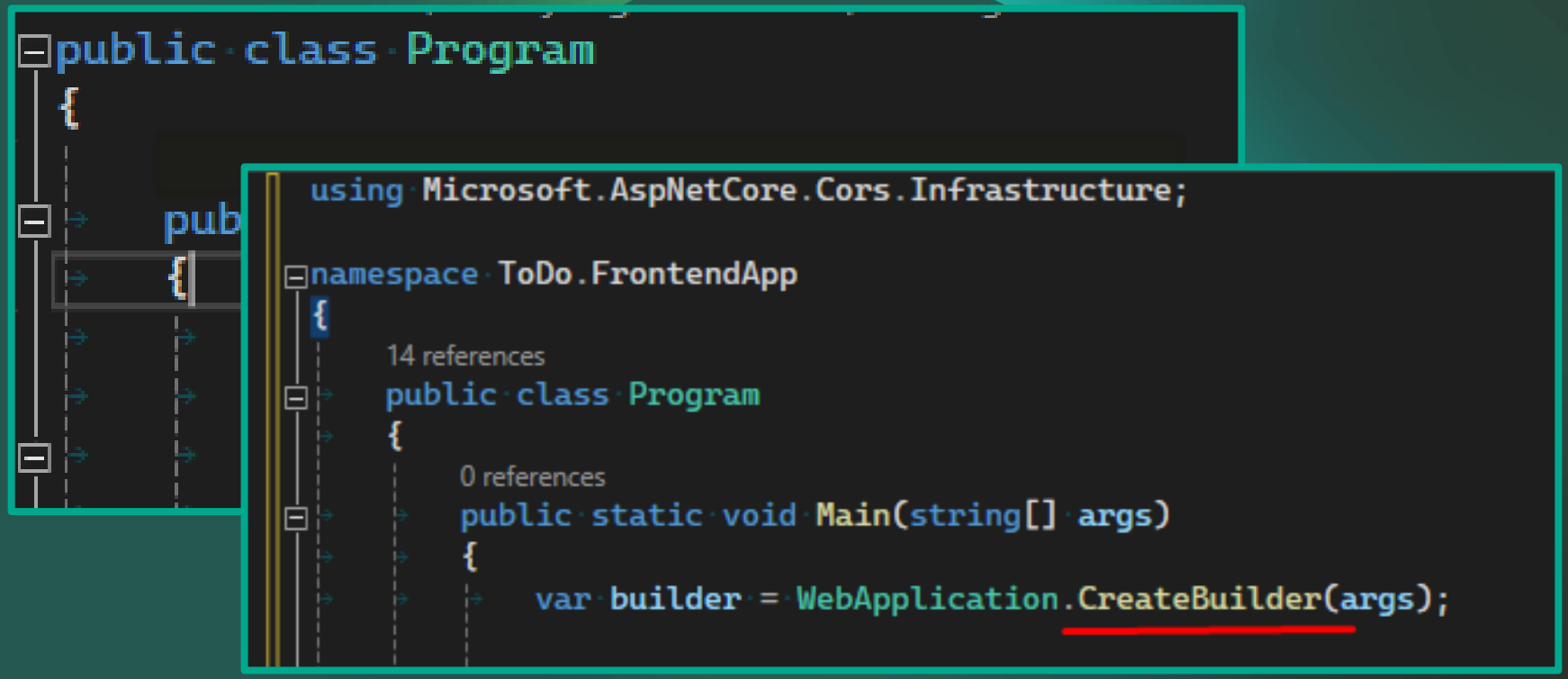
ToDo.BackendApp



Microsoft.AspNetCore.Mvc



Microsoft.AspNetCore.Mvc



First test


```
using ToDo.BackendApp;
using Microsoft.AspNetCore.Mvc.Testing;

[Test]
public void DefaultFactory_InternalHttpClient_ResponseOk()
{
        // arrange
        var factory = new WebApplicationFactory<Program>();
        var httpClient = factory.CreateClient();

        // act
        var response = await httpClient.GetStringAsync("api/todo/records");
        var records = response.FromJson<List<Todo>>();

        // assert
       records.Count.Should().Be(5);
}
```

First test


```
using ToDo.BackendApp;
using Microsoft.AspNetCore.Mvc.Testing;

[Test]
public void DefaultFactory_InternalHttpClient_ResponseOk()
{
        // arrange
        var factory = new WebApplicationFactory<Program>();
        var httpClient = factory.CreateClient();

        // act
        var response = await httpClient.GetStringAsync("api/todo/records");
        var records = response.FromJson<List<Todo>>();

        // assert
       records.Count.Should().Be(5);
}
```

First test


```
using ToDo.BackendApp;
using Microsoft.AspNetCore.Mvc.Testing;

[Test]
public void DefaultFactory_InternalHttpClient_ResponseOk()
{
        // arrange
        var factory = new WebApplicationFactory<Program>();
        var httpClient = factory.CreateClient();

        // act
        var response = await httpClient.GetStringAsync("api/todo/records");
        var records = response.FromJson<List<Todo>>();

        // assert
       records.Count.Should().Be(5);
}
```

First test


```
using ToDo.BackendApp;
using Microsoft.AspNetCore.Mvc.Testing;

[Test]
public void DefaultFactory_InternalHttpClient_ResponseOk()
{
        // arrange
        var factory = new WebApplicationFactory<Program>();
        var httpClient = factory.CreateClient();

        // act
        var response = await httpClient.GetStringAsync("api/todo/records");
        var records = response.FromJson<List<Todo>>();

        // assert
       records.Count.Should().Be(5);
}
```

First test


```
using ToDo.BackendApp;
using Microsoft.AspNetCore.Mvc.Testing;

[Test]
public void DefaultFactory_InternalHttpClient_ResponseOk()
{
        // arrange
        var factory = new WebApplicationFactory<Program>();
        var httpClient = factory.CreateClient();

        // act
        var response = await httpClient.GetStringAsync("api/todo/records");
        var records = response.FromJson<List<Todo>>();

        // assert
       records.Count.Should().Be(5);
}
```

First test


```
using ToDo.BackendApp;
using Microsoft.AspNetCore.Mvc.Testing;

[Test]
public void DefaultFactory_InternalHttpClient_ResponseOk()
{
        // arrange
        var factory = new WebApplicationFactory<Program>();
        var httpClient = factory.CreateClient();

        // act
        var response = await httpClient.GetStringAsync("api/todo/records");
        var records = response.FromJson<List<Todo>>();

        // assert
       records.Count.Should().Be(5);
}
```

First test


```
using ToDo.BackendApp;
using Microsoft.AspNetCore.Mvc.Testing;

[Test]
public void DefaultFactory_InternalHttpClient_ResponseOk()
{
        // arrange
        var factory = new WebApplicationFactory<Program>();
        var httpClient = factory.CreateClient();

        // act
        var response = await httpClient.GetStringAsync("api/todo/records");
        var records = response.FromJson<List<Todo>>();

        // assert
       records.Count.Should().Be(5);
}
```

#1 Доступ по сети

- **Уже есть описанные контракты API и готовый клиент к тестируемому сервису**

#1 Доступ по сети

- **Уже есть описанные контракты API и готовый клиент к тестируемому сервису**

- **Это Frontend-приложение и нужно писать Web-тесты**
- Например на Selenium или Playwright

External httpClient negative test


```
[Test]
public void DefaultFactory_ExternalHttpClient_RequestFailed()
{
        // arrange
        var factory = new WebApplicationFactory<Program>();
        var internalHttpClient = factory.CreateClient();
        var extHttpClient = new RestHttpClient();

        // act
        Action act = () => extHttpClient
            .Get<List<Todo>>($"{internalHttpClient.BaseAddress}/api/todo/records");

        // assert
       act.Should()
          .Throw<RestHttpClientException>("external httpClient not configured");
}
```

External httpClient negative test


```
[Test]
public void DefaultFactory_ExternalHttpClient_RequestFailed()
{
        // arrange
        var factory = new WebApplicationFactory<Program>();
        var internalHttpClient = factory.CreateClient();
        var extHttpClient = new RestHttpClient();

        // act
        Action act = () => extHttpClient
            .Get<List<Todo>>($"{internalHttpClient.BaseAddress}/api/todo/records");

        // assert
       act.Should()
          .Throw<RestHttpClientException>("external httpClient not configured");
}
```

External httpClient negative test


```
[Test]
public void DefaultFactory_ExternalHttpClient_RequestFailed()
{
        // arrange
        var factory = new WebApplicationFactory<Program>();
        var internalHttpClient = factory.CreateClient();
        var extHttpClient = new RestHttpClient();

        // act
        Action act = () => extHttpClient
            .Get<List<Todo>>($"{internalHttpClient.BaseAddress}/api/todo/records");

        // assert
       act.Should()
          .Throw<RestHttpClientException>("external httpClient not configured");
}
```

External httpClient negative test


```
[Test]
public void DefaultFactory_ExternalHttpClient_RequestFailed()
{
        // arrange
        var factory = new WebApplicationFactory<Program>();
        var internalHttpClient = factory.CreateClient();
        var extHttpClient = new RestHttpClient();

        // act
        Action act = () => extHttpClient
            .Get<List<Todo>>($"{internalHttpClient.BaseAddress}/api/todo/records");

        // assert
       act.Should()
          .Throw<RestHttpClientException>("external httpClient not configured");
}
```

Custom factory


```
public class CustomWebApplicationFactory<TStartup>
        : WebApplicationFactory<TStartup>
        where TStartup : class
{
    protected override IHost CreateHost(IHostBuilder builder)...

    protected override void ConfigureWebHost(IWebHostBuilder builder)
    {
        // do something;
    }
}
```

Custom factory


```
public class CustomWebApplicationFactory<TStartup>
        : WebApplicationFactory<TStartup>
        where TStartup : class
{
    protected override IHost CreateHost(IHostBuilder builder)...

    protected override void ConfigureWebHost(IWebHostBuilder builder)
    {
        // do something;
    }
}
```

Custom factory


```
public class CustomWebApplicationFactory<TStartup>
        : WebApplicationFactory<TStartup>
        where TStartup : class
{
    protected override IHost CreateHost(IHostBuilder builder)...

    protected override void ConfigureWebHost(IWebHostBuilder builder)
    {
        // do something;
    }
}
```

Custom factory


```
public class CustomWebApplicationFactory<TStartup>
        : WebApplicationFactory<TStartup>
        where TStartup : class
{
    protected override IHost CreateHost(IHostBuilder builder)...

    protected override void ConfigureWebHost(IWebHostBuilder builder)
    {
        // do something;
    }
}
```

Custom factory


```
public class CustomWebApplicationFactory<TStartup>
        : WebApplicationFactory<TStartup>
        where TStartup : class
{
    protected override IHost CreateHost(IHostBuilder builder)...

    protected override void ConfigureWebHost(IWebHostBuilder builder)
    {
        // do something;
    }
}
```

```
// assert
records.Count.Should().BeGreaterThan(0);
}
```

WebApplication - UseKestrel


```
public class CustomWebApplicationFactory<TStartup>
        : WebApplicationFactory<TStartup>
        where TStartup : class
{
      protected override IHost CreateHost(IHostBuilder builder)...

      protected override void ConfigureWebHost(IWebHostBuilder builder)
      {
          builder.UseKestrel();

          builder.UseSetting("urls", "https://0.0.0.0:0");
      }
}
```

WebApplication - UseKestrel


```
public class CustomWebApplicationFactory<TStartup>
        : WebApplicationFactory<TStartup>
        where TStartup : class
{
      protected override IHost CreateHost(IHostBuilder builder)...

      protected override void ConfigureWebHost(IWebHostBuilder builder)
      {
          builder.UseKestrel();

          builder.UseSetting("urls", "https://0.0.0.0:0");
      }
}
```

WebApplication - UseKestrel


```
public class CustomWebApplicationFactory<TStartup>
        : WebApplicationFactory<TStartup>
        where TStartup : class
{
      protected override IHost CreateHost(IHostBuilder builder)...

      protected override void ConfigureWebHost(IWebHostBuilder builder)
      {
          builder.UseKestrel();

          builder.UseSetting("urls", "https://0.0.0.0:0");
      }
}
```

WebApplication - UseKestrel


```
public class CustomWebApplicationFactory<TStartup>
        : WebApplicationFactory<TStartup>
        where TStartup : class
{
      protected override IHost CreateHost(IHostBuilder builder)...

      protected override void ConfigureWebHost(IWebHostBuilder builder)
      {
          builder.UseKestrel();

          builder.UseSetting("urls", "https://0.0.0.0:0");
      }
}
```

External httpClient positive test


```
[Test]
public void CustomFactory_ExternalHttpClient_ResponseOk()
{
        // arrange
        var factory = new CustomWebApplicationFactory<Program>();
        var baseAddress = factory.ServerAddress;
        var extHttpClient = new RestHttpClient();

        // act
        var records = extHttpClient
            .Get<List<Todo>>($"{baseAddress}/api/todo/records");

        // assert
        records.Should().Be(5);
}
```

External httpClient positive test


```
[Test]
public void CustomFactory_ExternalHttpClient_ResponseOk()
{
        // arrange
        var factory = new CustomWebApplicationFactory<Program>();
        var baseAddress = factory.ServerAddress;
        var extHttpClient = new RestHttpClient();

        // act
        var records = extHttpClient
            .Get<List<Todo>>($"{baseAddress}/api/todo/records");

        // assert
        records.Should().Be(5);
}
```

External httpClient positive test


```
[Test]
public void CustomFactory_ExternalHttpClient_ResponseOk()
{
        // arrange
        var factory = new CustomWebApplicationFactory<Program>();
        var baseAddress = factory.ServerAddress;
        var extHttpClient = new RestHttpClient();

        // act
        var records = extHttpClient
            .Get<List<Todo>>($"{baseAddress}/api/todo/records");

        // assert
        records.Should().Be(5);
}
```

External httpClient positive test


```
[Test]
public void CustomFactory_ExternalHttpClient_ResponseOk()
{
        // arrange
        var factory = new CustomWebApplicationFactory<Program>();
        var baseAddress = factory.ServerAddress;
        var extHttpClient = new RestHttpClient();

        // act
        var records = extHttpClient
            .Get<List<Todo>>($"{baseAddress}/api/todo/records");

        // assert
        records.Should().Be(5);
}
```

External httpClient positive test


```
[Test]
public void CustomFactory_ExternalHttpClient_ResponseOk()
{
        // arrange
        var factory = new CustomWebApplicationFactory<Program>();
        var baseAddress = factory.ServerAddress;
        var extHttpClient = new RestHttpClient();

        // act
        var records = extHttpClient
            .Get<List<Todo>>($"{baseAddress}/api/todo/records");

        // assert
        records.Should().Be(5);
}
```

#1 Доступ по сети

- Включить использование сетевых настроек
  - UseKestrel()

#1 Доступ по сети

- Включить использование сетевых настроек
  - UseKestrel()
- Сконфигурировать сеть так как нужно

#1 Доступ по сети

- **Включить использование сетевых настроек**
  - **UseKestrel()**

- **Сконфигурировать сеть так как нужно**
  - Если `WebApplication.CreateBuilder()`
  - то добавить вызов – `UseSetting("urls", "https://localhost:1234");`

#1 Доступ по сети

- **Включить использование сетевых настроек**
  - **UseKestrel()**

- **Сконфигурировать сеть так как нужно**
  - Если WebApplication.CreateBuilder()
  - то добавить вызов - UseSetting("urls", "https://localhost:1234");
  - Если Host.CreateDefaultBuilder()
  - обновить значение порта в IConfiguration

#2 Подмена зависимостей проекта

- **Если нет возможности использовать реальные интеграции**
- Стоит стремиться исключать все реальные интеграции

#2 Подмена зависимостей проекта

- **Если нет возможности использовать реальные интеграции**
- Стоит стремиться исключать все реальные интеграции

- **Если сервис имеет строки подключения к каким-либо внешним ресурсам**
- Базы данных, очереди, хранилища секретов и т.п.

Service descriptors


```
Public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IStorage, TodoDatabaseStorage>();

    services.AddTransient<IDbContext, TodoDbContext>();


        ...
});
```

```
namespace BackendApi.L1Tests.Tests
{
    [TestFixture]
    [Parallelizable(ParallelScope.All)]
    0 references
    public class WebApplicationFactoryInitTests : TestBase
    {
        var response = await httpClient.GetStringAsync(endpoint);
        var records = response.FromJson<List<Todo>>();

        // assert
        records.Count.Should().BeGreaterThan(0);
    }
```

Service descriptors


```
Public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IStorage, TodoDatabaseStorage>();

    services.AddTransient<IDbContext, TodoDbContext>();


        ...
});
```

Service descriptors


```
Public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IStorage, TodoDatabaseStorage>();

    services.AddTransient<IDbContext, TodoDbContext>();


        ...
});
```

Mock service


```
public class ToDoDatabaseMock : IStorage
{
        public T GetRecords<T>()
        {
            ...
        }


        ...
});
```

```
namespace BackendApi.L1Tests.Tests
{
    [TestFixture]
    [Parallelizable(ParallelScope.All)]
        var response = await httpClient.GetStringAsync(endpoint);
        var records = response.FromJson<List<Todo>>();

        // assert
        records.Count.Should().BeGreaterThan(0);
    }
```

Mock service


```
public class ToDoDatabaseMock : IStorage
{
        public T GetRecords<T>()
        {
            ...
        }


        ...
});
```

```
namespace BackendApi.L1Tests.Tests
{
    [TestFixture]
    [Parallelizable(ParallelScope.All)]
        var response = await httpClient.GetStringAsync(endpoint);
        var records = response.FromJson<List<Todo>>();

        // assert
        records.Count.Should().BeGreaterThan(0);
    }
```

Mock service


```
public class ToDoDatabaseMock : IStorage
{
        public T GetRecords<T>()
        {
            ...
        }


        ...
});
```

```
namespace BackendApi.L1Tests.Tests
{
    [TestFixture]
    [Parallelizable(ParallelScope.All)]
        var response = await httpClient.GetStringAsync(endpoint);
        var records = response.FromJson<List<Todo>>();

        // assert
        records.Count.Should().BeGreaterThan(0);
    }
```

Service descriptors


```
namespace BackendApi.L1Tests.Tests
{
    [TestFixture]
    [Parallelizable(ParallelScope.All)]
    0 references
    public class WebApplicationFactoryInitTests : TestBase
    {
        // act
        var endpoint = "https://localhost:9999/api/ToDo/records";
        var response = await httpClient.GetStringAsync(endpoint);
        var records = response.FromJson<List<Todo>>();

        // assert
        records.Count.Should().BeGreaterThan(0);
    }
```

```
Public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IStorage, TodoDatabaseStorage>();

    services.AddTransient<IDbContext, TodoDbContext>();
    //...
});
```

Service descriptors

TodoDatabaseMock

```
Public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IStorage, TodoDatabaseStorage>();

    services.AddTransient<IDbContext, TodoDbContext>();
    //...
});
```

```
namespace BackendApi.L1Tests.Tests
{
    [TestFixture]
    [Parallelizable(ParallelScope.All)]
    0 references
    public class WebApplicationFactoryInitTests : TestBase
    {
        // act
        var endpoint = "https://localhost:9999/api/ToDo/records";
        var response = await httpClient.GetStringAsync(endpoint);
        var records = response.FromJson<List<Todo>>();

        // assert
        records.Count.Should().BeGreaterThan(0);
    }
```

Custom factory


```
public class CustomWebApplicationFactory<TStartup>
        : WebApplicationFactory<TStartup>
        where TStartup : class
{
      protected override IHost CreateHost(IHostBuilder builder)...

      protected override void ConfigureWebHost(IWebHostBuilder builder)
      {
          builder.ConfigureServices(services =>
          {
             // do something;
          });
      }
}
```

Custom factory


```
public class CustomWebApplicationFactory<TStartup>
        : WebApplicationFactory<TStartup>
        where TStartup : class
{
      protected override IHost CreateHost(IHostBuilder builder)...

      protected override void ConfigureWebHost(IWebHostBuilder builder)
      {
          builder.ConfigureServices(services =>
          {
             // do something;
          });
      }
}
```

Custom factory


```
public class CustomWebApplicationFactory<TStartup>
        : WebApplicationFactory<TStartup>
        where TStartup : class
{
      protected override IHost CreateHost(IHostBuilder builder)...

      protected override void ConfigureWebHost(IWebHostBuilder builder)
      {
          builder.ConfigureServices(services =>
          {
              // do something;
          });
      }
}
```

Replace service descriptor


```
builder.ConfigureServices(services =>
    {
        var dbServiceDescriptor = services
            .SingleOrDefault(
            d => d.ServiceType == typeof(IStorage));

        services.Remove(dbServiceDescriptor);
        services
           .AddSingleton<IStorage>(
                new TodoDatabaseMock(testTodos));
   });
```

Replace service descriptor


```
builder.ConfigureServices(services =>
    {
        var dbServiceDescriptor = services
            .SingleOrDefault(
            d => d.ServiceType == typeof(IStorage));

        services.Remove(dbServiceDescriptor);
        services
           .AddSingleton<IStorage>(
                new TodoDatabaseMock(testTodos));
   });
```

Replace service descriptor


```
builder.ConfigureServices(services =>
    {
        var dbServiceDescriptor = services
            .SingleOrDefault(
            d => d.ServiceType == typeof(IStorage));

        services.Remove(dbServiceDescriptor);
        services
           .AddSingleton<IStorage>(
                new TodoDatabaseMock(testTodos));
   });
```

Replace service descriptor


```
builder.ConfigureServices(services =>
    {
        var dbServiceDescriptor = services
            .SingleOrDefault(
            d => d.ServiceType == typeof(IStorage));

        services.Remove(dbServiceDescriptor);
        services
           .AddSingleton<IStorage>(
                new TodoDatabaseMock(testTodos));
   });
```

Replace service descriptor


```
builder.ConfigureServices(services =>
    {
        var dbServiceDescriptor = services
            .SingleOrDefault(
            d => d.ServiceType == typeof(IStorage));

        services.Remove(dbServiceDescriptor);
        services
           .AddSingleton<IStorage>(
                new TodoDatabaseMock(testTodos));
   });
```

Replace service descriptor


```
builder.ConfigureServices(services =>
    {
        var dbServiceDescriptor = services
            .SingleOrDefault(
            d => d.ServiceType == typeof(IStorage));

        services.Remove(dbServiceDescriptor);
        services
           .AddSingleton<IStorage>(
                new TodoDatabaseMock(testTodos));
   });
```

Replace dependency test sample


```
    var todosCount = 15;
    var testTodos = FakeDataGenerators.GetTestTodos(todosCount);

    var factory = new CustomHostFactory<Program>(services =>
    {
        var dbContextDescriptor = services
            .SingleOrDefault(d => d.ServiceType == typeof(IStorageContext<Todo>));

        services.Remove(dbContextDescriptor);
        services.AddSingleton<IStorageContext<Todo>>(new TodoContextMock(testTodos));
    });

    WebAppFactory = factory;
    var httpClient = factory.CreateClient();

    // act
    var response = await httpClient.GetStringAsync("api/ToDo/records");
    var records = response.FromJson<List<Todo>>();

    // assert
    records.Should().HaveCount(todosCount);
    records.Should().BeEquivalentTo(testTodos);
}
```

Replace dependency test sample


```
    var todosCount = 15;
    var testTodos = FakeDataGenerators.GetTestTodos(todosCount);

    var factory = new CustomHostFactory<Program>(services =>
    {
        var dbContextDescriptor = services
            .SingleOrDefault(d => d.ServiceType == typeof(IStorageContext<Todo>));

        services.Remove(dbContextDescriptor);
        services.AddSingleton<IStorageContext<Todo>>(new TodoContextMock(testTodos));
    });

    WebAppFactory = factory;
    var httpClient = factory.CreateClient();

    // act
    var response = await httpClient.GetStringAsync("api/ToDo/records");
    var records = response.FromJson<List<Todo>>();

    // assert
    records.Should().HaveCount(todosCount);
    records.Should().BeEquivalentTo(testTodos);
}
```

Replace dependency test sample


```
    var todosCount = 15;
    var testTodos = FakeDataGenerators.GetTestTodos(todosCount);

    var factory = new CustomHostFactory<Program>(services =>
    {
        var dbContextDescriptor = services
            .SingleOrDefault(d => d.ServiceType == typeof(IStorageContext<Todo>));

        services.Remove(dbContextDescriptor);
        services.AddSingleton<IStorageContext<Todo>>(new TodoContextMock(testTodos));
    });

    WebAppFactory = factory;
    var httpClient = factory.CreateClient();

    // act
    var response = await httpClient.GetStringAsync("api/ToDo/records");
    var records = response.FromJson<List<Todo>>();

    // assert
    records.Should().HaveCount(todosCount);
    records.Should().BeEquivalentTo(testTodos);
}
```

Replace dependency test sample


```
var todosCount = 15;
var testTodos = FakeDataGenerators.GetTestTodos(todosCount);

var factory = new CustomHostFactory<Program>(services =>
{
    var dbContextDescriptor = services
        .SingleOrDefault(d => d.ServiceType == typeof(IStorageContext<Todo>));

    services.Remove(dbContextDescriptor);
    services.AddSingleton<IStorageContext<Todo>>(new TodoContextMock(testTodos));
});

WebAppFactory = factory;
var httpClient = factory.CreateClient();

// act
var response = await httpClient.GetStringAsync("api/ToDo/records");
var records = response.FromJson<List<Todo>>();

// assert
records.Should().HaveCount(todosCount);
records.Should().BeEquivalentTo(testTodos);
}
```

Replace dependency test sample


```
var todosCount = 15;
var testTodos = FakeDataGenerators.GetTestTodos(todosCount);

var factory = new CustomHostFactory<Program>(services =>
{
    var dbContextDescriptor = services
        .SingleOrDefault(d => d.ServiceType == typeof(IStorageContext<Todo>));

    services.Remove(dbContextDescriptor);
    services.AddSingleton<IStorageContext<Todo>>(new TodoContextMock(testTodos));
});

WebAppFactory = factory;
var httpClient = factory.CreateClient();

// act
var response = await httpClient.GetStringAsync("api/ToDo/records");
var records = response.FromJson<List<Todo>>();

// assert
records.Should().HaveCount(todosCount);
records.Should().BeEquivalentTo(testTodos);
}
```

Replace dependency test sample


```
    var todosCount = 15;
    var testTodos = FakeDataGenerators.GetTestTodos(todosCount);

    var factory = new CustomHostFactory<Program>(services =>
    {
        var dbContextDescriptor = services
            .SingleOrDefault(d => d.ServiceType == typeof(IStorageContext<Todo>));

        services.Remove(dbContextDescriptor);
        services.AddSingleton<IStorageContext<Todo>>(new TodoContextMock(testTodos));
    });

    WebAppFactory = factory;
    var httpClient = factory.CreateClient();

    // act
    var response = await httpClient.GetStringAsync("api/ToDo/records");
    var records = response.FromJson<List<Todo>>();

    // assert
    records.Should().HaveCount(todosCount);
    records.Should().BeEquivalentTo(testTodos);
}
```

Replace dependency test sample


```
    var todosCount = 15;
    var testTodos = FakeDataGenerators.GetTestTodos(todosCount);

    var factory = new CustomHostFactory<Program>(services =>
    {
        var dbContextDescriptor = services
            .SingleOrDefault(d => d.ServiceType == typeof(IStorageContext<Todo>));

        services.Remove(dbContextDescriptor);
        services.AddSingleton<IStorageContext<Todo>>(new TodoContextMock(testTodos));
    });

    WebAppFactory = factory;
    var httpClient = factory.CreateClient();

    // act
    var response = await httpClient.GetStringAsync("api/ToDo/records");
    var records = response.FromJson<List<Todo>>();

    // assert
    records.Should().HaveCount(todosCount);
    records.Should().BeEquivalentTo(testTodos);
}
```

Replace dependency test sample


```
    var todosCount = 15;
    var testTodos = FakeDataGenerators.GetTestTodos(todosCount);

    var factory = new CustomHostFactory<Program>(services =>
    {
        var dbContextDescriptor = services
            .SingleOrDefault(d => d.ServiceType == typeof(IStorageContext<Todo>));

        services.Remove(dbContextDescriptor);
        services.AddSingleton<IStorageContext<Todo>>(new TodoContextMock(testTodos));
    });

    WebAppFactory = factory;
    var httpClient = factory.CreateClient();

    // act
    var response = await httpClient.GetStringAsync("api/ToDo/records");
    var records = response.FromJson<List<Todo>>();

    // assert
    records.Should().HaveCount(todosCount);
    records.Should().BeEquivalentTo(testTodos);
}
```

#2 Подмена зависимостей проекта

- Найти ServiceDescriptor нужно сервиса

#2 Подмена зависимостей проекта

- Найти ServiceDescriptor нужно сервиса
- Создать MockService для нужного интерфейса

#2 Подмена зависимостей проекта

- Найти ServiceDescriptor нужно сервиса
- Создать MockService для нужного интерфейса
- Удалить старый ServiceDescriptor

#2 Подмена зависимостей проекта

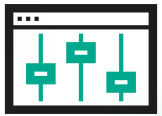
- Найти ServiceDescriptor нужно сервиса
- Создать MockService для нужного интерфейса
- Удалить старый ServiceDescriptor
- Добавить новый, который будет ссылаться на MockService

#3 Обновить значения

- **Если нужно обновить значения в конфигурации**
- Например, заменить ссылку на реальный сервис ссылкой на мок.
- Например, менять порт для запуска Web-тестов

#3 Обновить значения

- **Если нужно обновить значения в конфигурации**
- Например, заменить ссылку на реальный сервис ссылкой на мок.
- Например, менять порт для запуска Web-тестов



- **Если нет необходимости подменять сервис, но нужно поменять ему значения**
- Например, чтобы сократить сценарий тест-кейса.

Service descriptors


```
Public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IStorage, TodoDatabaseStorage>();

    services.AddTransient<IDbContext, TodoDbContext>();

        ...
});
```

https://habr.com/ru/companies/kaspersky/articles/757980/

https://www.youtube.com/watch?v=fofhl0ZyhNU

Service descriptors


```
Public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IStorage, TodoDatabaseStorage>();

    services.AddTransient<IDbContext, TodoDbContext>();


        ...
});
```

https://habr.com/ru/companies/kaspersky/articles/757980/

https://www.youtube.com/watch?v=fofhl0ZyhNU

Service descriptors


```
namespace BackendApi.L1Tests.Tests
{
    [TestFixture]
    [Parallelizable(ParallelScope.All)]
    0 references
    public class WebApplicationFactoryInitTests : TestBase
    {
        var response = await httpClient.GetStringAsync(endpoint);
        var records = response.FromJson<List<Todo>>();
        records.Count.Should().BeGreaterThan(0);
    }
```

```
Public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IStorage, TodoDatabaseStorage>();

    services.AddTransient<IDbContext, TodoDbContext>();

        ...
});
```

https://habr.com/ru/companies/kaspersky/articles/757980/

https://www.youtube.com/watch?v=fofhl0ZyhNU

Service descriptors


```
Public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IStorage, TodoDatabaseStorage>();

    services.AddTransient<IDbContext, TodoDbContext>();


        ...
});
```

https://habr.com/ru/companies/kaspersky/articles/757980/

https://www.youtube.com/watch?v=fofhl0ZyhNU

Update dependency values test sample


```
31
32          var mock = new WireMockProvider();
33          var emailServiceMock = mock.GetMock<MailingAppMappings>();
34          emailServiceMock.MockSendEmailRequest(email);
35
36          var factory = new CustomHostFactory<Program>(services =>
37          {
38              var configuration = services.BuildServiceProvider().GetService<IConfiguration>();
39              configuration["AppSettings:SmtpServerAddress"] = mock.ServerAddress;
40              configuration["AppSettings:NotificationsEmail"] = email;
41          });
42          WebAppFactory = factory;
43          var httpClient = factory.CreateClient();
44
45          // act
46          var response = await httpClient.PostAsJsonAsync("api/ToDo/new", newTodo);
47
48          var record = await response.Content.ReadFromJsonAsync<Todo>();
49
50          // assert
51          response.IsSuccessStatusCode.Should().BeTrue();
52          record.Should().BeEquivalentTo(newTodo);
53          mock.Server.LogEntries.Where(x => x.RequestMessage.Body.Contains(email)).Should().HaveCount(1);
54
55          mock.Dispose();
56      }
```

Update dependency values test sample


```
31
32          var mock = new WireMockProvider();
33          var emailServiceMock = mock.GetMock<MailingAppMappings>();
34          emailServiceMock.MockSendEmailRequest(email);
35
36          var factory = new CustomHostFactory<Program>(services =>
37          {
38              var configuration = services.BuildServiceProvider().GetService<IConfiguration>();
39              configuration["AppSettings:SmtpServerAddress"] = mock.ServerAddress;
40              configuration["AppSettings:NotificationsEmail"] = email;
41          });
42          WebAppFactory = factory;
43          var httpClient = factory.CreateClient();
44
45          // act
46          var response = await httpClient.PostAsJsonAsync("api/ToDo/new", newTodo);
47
48          var record = await response.Content.ReadFromJsonAsync<Todo>();
49
50          // assert
51          response.IsSuccessStatusCode.Should().BeTrue();
52          record.Should().BeEquivalentTo(newTodo);
53          mock.Server.LogEntries.Where(x => x.RequestMessage.Body.Contains(email)).Should().HaveCount(1);
54
55          mock.Dispose();
56      }
```

Update dependency values test sample


```
builder.ConfigureServices(services =>
    {
        var config = services
            .BuildServiceProvider()
            .GetService<IConfiguration>();

        config["AppSettings:SmtpServerAddress"]
            = mock.ServerAddress;
    });
```

```
        mock.Server.LogEntries.Where(x => x.RequestMessage.Body.Contains(email)).Should().HaveCount(1);

        mock.Dispose();
    }
```

Update dependency values test sample


```
builder.ConfigureServices(services =>
    {
        var config = services
            .BuildServiceProvider()
            .GetService<IConfiguration>();

        config["AppSettings:SmtpServerAddress"]
            = mock.ServerAddress;
    });
```

```
        mock.Server.LogEntries.Where(x => x.RequestMessage.Body.Contains(email)).Should().HaveCount(1);

        mock.Dispose();
    }
```

Update dependency values test sample


```
builder.ConfigureServices(services =>
    {
        var config = services
            .BuildServiceProvider()
            .GetService<IConfiguration>();

        config["AppSettings:SmtpServerAddress"]
            = mock.ServerAddress;
    });
```

```
        mock.Server.LogEntries.Where(x => x.RequestMessage.Body.Contains(email)).Should().HaveCount(1);

        mock.Dispose();
    }
```

Update dependency values test sample


```
builder.ConfigureServices(services =>
    {
        var config = services
            .BuildServiceProvider()
            .GetService<IConfiguration>();

        config["AppSettings:SmtpServerAddress"]
            = mock.ServerAddress;
    });
```

```
        mock.Server.LogEntries.Where(x => x.RequestMessage.Body.Contains(email)).Should().HaveCount(1);

        mock.Dispose();
    }
```

Update dependency values test sample


```
builder.ConfigureServices(services =>
    {
        var config = services
            .BuildServiceProvider()
            .GetService<IConfiguration>();

        config["AppSettings:SmtpServerAddress"]
            = mock.ServerAddress;
    });
```

```
        mock.Server.LogEntries.Where(x => x.RequestMessage.Body.Contains(email)).Should().HaveCount(1);

        mock.Dispose();
    }
```

Update dependency values test sample


```
31
32          var mock = new WireMockProvider();
33          var emailServiceMock = mock.GetMock<MailingAppMappings>();
34          emailServiceMock.MockSendEmailRequest(email);
35
36          var factory = new CustomHostFactory<Program>(services =>
37          {
38              var configuration = services.BuildServiceProvider().GetService<IConfiguration>();
39              configuration["AppSettings:SmtpServerAddress"] = mock.ServerAddress;
40              configuration["AppSettings:NotificationsEmail"] = email;
41          });
42          WebAppFactory = factory;
43          var httpClient = factory.CreateClient();
44
45          // act
46          var response = await httpClient.PostAsJsonAsync("api/ToDo/new", newTodo);
47
48          var record = await response.Content.ReadFromJsonAsync<Todo>();
49
50          // assert
51          response.IsSuccessStatusCode.Should().BeTrue();
52          record.Should().BeEquivalentTo(newTodo);
53          mock.Server.LogEntries.Where(x => x.RequestMessage.Body.Contains(email)).Should().HaveCount(1);
54
55          mock.Dispose();
56      }
```

#3 Обновить значения

- Найти ServiceDescriptor нужно сервиса

#3 Обновить значения

- Найти ServiceDescriptor нужно сервиса
- Проверить его Lifestyle

#3 Обновить значения

- Найти ServiceDescriptor нужно сервиса
- Проверить его Lifestyle
- Если Singleton – обновить значение

#3 Обновить значения

- Найти ServiceDescriptor нужно сервиса
- Проверить его Lifestyle
- Если Singleton – обновить значение
- Если не Singleton

  - либо подменить сервис целиком
  - либо использовать Proxy/Decorator pattern (см. проект с примерами)

#4 Использование http-моков

- **Если хочется протестировать как можно больше, и не подменять сервисы**
- И если интеграция выполняется по http-протоколу. В идеале – REST API

#4 Использование http-моков

- **Если хочется протестировать как можно больше, и не подменять сервисы**
- И если интеграция выполняется по http-протоколу. В идеале – REST API

- **Можно использовать библиотеку Wiremock.NET**
- Умеет подниматься на случайном порту.
- Легко взаимодействовать из кода.
- GitHub: https://github.com/WireMock-Net/WireMock.Net

Start wiremock server


```
public class WireMockProvider : IDisposable

2 references | 2/2 passing
public WireMockProvider()
{
    Server = WireMockServer.Start(new WireMockServerSettings
```

```
public WireMockProvider()
{
        Server = WireMockServer.Start();
}
```

```
    Server.Stop();
}

2 references | 2/2 passing
public string ServerAddress => Server.Url ?? Server.Urls.First();

5 references
private WireMockServer Server { get; set; }
```

Update dependency


```
var request = Request.Create()
        .WithPath(new WildcardMatcher($"/api/email/*", true))
        .UsingPost();

Server
    .Given(request)
    .RespondWith(Response.Create()
        .WithStatusCode(201)
        .WithBodyAsJson(new
         {
              Status = "Success"
         }));
```

Update dependency


```
var request = Request.Create()
        .WithPath(new WildcardMatcher($"/api/email/*", true))
        .UsingPost();

Server
    .Given(request)
    .RespondWith(Response.Create()
        .WithStatusCode(201)
        .WithBodyAsJson(new
         {
              Status = "Success"
         }));
```

Update dependency


```
var request = Request.Create()
        .WithPath(new WildcardMatcher($"/api/email/*", true))
        .UsingPost();

Server
    .Given(request)
    .RespondWith(Response.Create()
        .WithStatusCode(201)
        .WithBodyAsJson(new
         {
              Status = "Success"
         }));
```

Update dependency


```
var request = Request.Create()
        .WithPath(new WildcardMatcher($"/api/email/*", true))
        .UsingPost();

Server
    .Given(request)
    .RespondWith(Response.Create()
        .WithStatusCode(201)
        .WithBodyAsJson(new
         {
              Status = "Success"
         }));
```

Update dependency


```
var request = Request.Create()
        .WithPath(new WildcardMatcher($"/api/email/*", true))
        .UsingPost();

Server
    .Given(request)
    .RespondWith(Response.Create()
        .WithStatusCode(201)
        .WithBodyAsJson(new
         {
              Status = "Success"
         }));
```

Use http mock test sample


```
31
32          var mock = new WireMockProvider();
33          var emailServiceMock = mock.GetMock<MailingAppMappings>();
34          emailServiceMock.MockSendEmailRequest(email);
35
36          var factory = new CustomHostFactory<Program>(services =>
37          {
38              var configuration = services.BuildServiceProvider().GetService<IConfiguration>();
39              configuration["AppSettings:SmtpServerAddress"] = mock.ServerAddress;
40              configuration["AppSettings:NotificationsEmail"] = email;
41          });
42          WebAppFactory = factory;
43          var httpClient = factory.CreateClient();
44
45          // act
46          var response = await httpClient.PostAsJsonAsync("api/ToDo/new", newTodo);
47
48          var record = await response.Content.ReadFromJsonAsync<Todo>();
49
50          // assert
51          response.IsSuccessStatusCode.Should().BeTrue();
52          record.Should().BeEquivalentTo(newTodo);
53          mock.Server.LogEntries.Where(x => x.RequestMessage.Body.Contains(email)).Should().HaveCount(1);
54
55          mock.Dispose();
56      }
```

Use http mock test sample


```
var mock = new WireMockProvider();
mock.MockSendEmailRequest(email);
```

```
31
32          va
33          va
34          em
35
36          var factory = new CustomHostFactory<Program>(services =>
37          {
38              var configuration = services.BuildServiceProvider().GetService<IConfiguration>();
39              configuration["AppSettings:SmtpServerAddress"] = mock.ServerAddress;
40              configuration["AppSettings:NotificationsEmail"] = email;
41          });
42          WebAppFactory = factory;
43          var httpClient = factory.CreateClient();
44
45          // act
46          var response = await httpClient.PostAsJsonAsync("api/ToDo/new", newTodo);
47
48          var record = await response.Content.ReadFromJsonAsync<Todo>();
49
50          // assert
51          response.IsSuccessStatusCode.Should().BeTrue();
52          record.Should().BeEquivalentTo(newTodo);
53          mock.Server.LogEntries.Where(x => x.RequestMessage.Body.Contains(email)).Should().HaveCount(1);
54
55          mock.Dispose();
56      }
```

Use http mock test sample


```
var mock = new WireMockProvider();
mock.MockSendEmailRequest(email);
```

```
var factory = new CustomHostFactory<Program>(services =>
{
    var configuration = services
        .BuildServiceProvider()
        .GetService<IConfiguration>();
    configuration["AppSettings:SmtpServerAddress"] =
         mock.ServerAddress;
});
```

```
mock.Server.LogEntries.Where(x => x.RequestMessage.Body.Contains(email)).Should().HaveCount(1);

mock.Dispose();
```

Use http mock test sample


```
var mock = new WireMockProvider();
mock.MockSendEmailRequest(email);
```

```
var factory = new CustomHostFactory<Program>(services =>
{
    var configuration = services
        .BuildServiceProvider()
        .GetService<IConfiguration>();
    configuration["AppSettings:SmtpServerAddress"] =
         mock.ServerAddress;
});
```

```
mock.Server.LogEntries.Where(x => x.RequestMessage.Body.Contains(email)).Should().HaveCount(1);

mock.Dispose();
```

Use http mock test sample


```
var mock = new WireMockProvider();
mock.MockSendEmailRequest(email);
```

```
var factory = new CustomHostFactory<Program>(services =>
{
    var configuration = services
        .BuildServiceProvider()
        .GetService<IConfiguration>();
    configuration["AppSettings:SmtpServerAddress"] =
        mock.ServerAddress;
});
```

```
mock.Server.LogEntries.Where(x => x.RequestMessage.Body.Contains(email)).Should().HaveCount(1);

mock.Dispose();
```

Use http mock test sample


```
31
32          var mock = new WireMockProvider();
33          var emailServiceMock = mock.GetMock<MailingAppMappings>();
34          emailServiceMock.MockSendEmailRequest(email);
35
36          var factory = new CustomHostFactory<Program>(services =>
37          {
38              var configuration = services.BuildServiceProvider().GetService<IConfiguration>();
39              configuration["AppSettings:SmtpServerAddress"] = mock.ServerAddress;
40              configuration["AppSettings:NotificationsEmail"] = email;
41          });
42          WebAppFactory = factory;
43          var httpClient = factory.CreateClient();
44
45          // act
46          var response = await httpClient.PostAsJsonAsync("api/ToDo/new", newTodo);
47
48          var record = await response.Content.ReadFromJsonAsync<Todo>();
49
50          // assert
51          response.IsSuccessStatusCode.Should().BeTrue();
52          record.Should().BeEquivalentTo(newTodo);
53          mock.Server.LogEntries.Where(x => x.RequestMessage.Body.Contains(email)).Should().HaveCount(1);
54
55          mock.Dispose();
56      }
```

- # #4 Использование http-моков

  - Создать и запустить MockServer

- # #4 Использование http-моков

  - Создать и запустить MockServer
  - Настроить маппинги для ответов

• #4 Использование http-моков

- Создать и запустить MockServer
- Настроить маппинги для ответов
- Подменить ссылку в конфигурации сервиса

#5 Нужны Web-тесты

- **Если нужно тестировать javaScript-код**
- Простая загрузка GET-запросом страниц не активирует выполнение JS, для этого нужны WebDriver'ы (Selenium, Playwright ...)

#5 Нужны Web-тесты

- **Если нужно тестировать javaScript-код**
- Простая загрузка GET-запросом страниц не активирует выполнение JS, для этого нужны WebDriver'ы (Selenium, Playwright ...)

- **Быстрые тесты для мелких операций**
- Все локально, потому быстро. Хорошо подходит для тестов на снятие скриншотов, тестирования локализаций, разметки и т.п.

Selenium test


```
[Test]
public async Task MultiAppFactory_RunTwoApplications_DataLoaded()
{
    // arrange
    var factory = new MultiAppFactory<FrontProgram, BackProgram>();
    var url = $"{factory.FrontendAddress}Home";
    // act
    WebBrowser = new SeleniumChromeWebBrowser();
    WebBrowser.GoToUrl(url);

    var page = await WebBrowser.GetPage<IndexPage>();
    var todoList = page.TableRecords;

    // assert
    todoList.Count().Should().Be(5);
}
```

Selenium test


```
[Test]
public async Task MultiAppFactory_RunTwoApplications_DataLoaded()
{
    // arrange
    var factory = new MultiAppFactory<FrontProgram, BackProgram>();
    var url = $"{factory.FrontendAddress}Home";
    // act
    WebBrowser = new SeleniumChromeWebBrowser();
    WebBrowser.GoToUrl(url);

    var page = await WebBrowser.GetPage<IndexPage>();
    var todoList = page.TableRecords;

    // assert
    todoList.Count().Should().Be(5);
}
```

Selenium test


```
[Test]
public async Task MultiAppFactory_RunTwoApplications_DataLoaded()
{
    // arrange
    var factory = new MultiAppFactory<FrontProgram, BackProgram>();
    var url = $"{factory.FrontendAddress}Home";
    // act
    WebBrowser = new SeleniumChromeWebBrowser();
    WebBrowser.GoToUrl(url);

    var page = await WebBrowser.GetPage<IndexPage>();
    var todoList = page.TableRecords;

    // assert
    todoList.Count().Should().Be(5);
}
```

Selenium test


```
[Test]
public async Task MultiAppFactory_RunTwoApplications_DataLoaded()
{
    // arrange
    var factory = new MultiAppFactory<FrontProgram, BackProgram>();
    var url = $"{factory.FrontendAddress}Home";
    // act
    WebBrowser = new SeleniumChromeWebBrowser();
    WebBrowser.GoToUrl(url);

    var page = await WebBrowser.GetPage<IndexPage>();
    var todoList = page.TableRecords;

    // assert
    todoList.Count().Should().Be(5);
}
```

Selenium test


```
[Test]
public async Task MultiAppFactory_RunTwoApplications_DataLoaded()
{
    // arrange
    var factory = new MultiAppFactory<FrontProgram, BackProgram>();
    var url = $"{factory.FrontendAddress}Home";
    // act
    WebBrowser = new SeleniumChromeWebBrowser();
    WebBrowser.GoToUrl(url);

    var page = await WebBrowser.GetPage<IndexPage>();
    var todoList = page.TableRecords;

    // assert
    todoList.Count().Should().Be(5);
}
```

Selenium test


```
[Test]
public async Task MultiAppFactory_RunTwoApplications_DataLoaded()
{
    // arrange
    var factory = new MultiAppFactory<FrontProgram, BackProgram>();
    var url = $"{factory.FrontendAddress}Home";
    // act
    WebBrowser = new SeleniumChromeWebBrowser();
    WebBrowser.GoToUrl(url);

    var page = await WebBrowser.GetPage<IndexPage>();
    var todoList = page.TableRecords;

    // assert
    todoList.Count().Should().Be(5);
}
```

Selenium test


```
[Test]
public async Task MultiAppFactory_RunTwoApplications_DataLoaded()
{
    // arrange
    var factory = new MultiAppFactory<FrontProgram, BackProgram>();
    var url = $"{factory.FrontendAddress}Home";
    // act
    WebBrowser = new SeleniumChromeWebBrowser();
    WebBrowser.GoToUrl(url);

    var page = await WebBrowser.GetPage<IndexPage>();
    var todoList = page.TableRecords;

    // assert
    todoList.Count().Should().Be(5);
}
```

Selenium test


```
[Test]
public async Task MultiAppFactory_RunTwoApplications_DataLoaded()
{
    // arrange
    var factory = new MultiAppFactory<FrontProgram, BackProgram>();
    var url = $"{factory.FrontendAddress}Home";
    // act
    WebBrowser = new SeleniumChromeWebBrowser();
    WebBrowser.GoToUrl(url);

    var page = await WebBrowser.GetPage<IndexPage>();
    var todoList = page.TableRecords;

    // assert
    todoList.Count().Should().Be(5);
}
```

Playwright test


```
[Test]
public async Task MultiAppFactory_RunTwoApplications_DataLoaded()
{
    // arrange
    var factory = new MultiAppFactory<FrontProgram, BackProgram>();
    var url = $"{factory.FrontendAddress}Home";
    // act
    WebBrowser = new PlaywrightChromeWebBrowser();
    WebBrowser.GoToUrl(url);

    var page = await WebBrowser.GetPage<IndexPage>();
    var todoList = page.TableRecords;

    // assert
    todoList.Count().Should().Be(5);
}
```

Playwright test


```
[Test]
public async Task MultiAppFactory_RunTwoApplications_DataLoaded()
{
    // arrange
    var factory = new MultiAppFactory<FrontProgram, BackProgram>();
    var url = $"{factory.FrontendAddress}Home";
    // act
    WebBrowser = new PlaywrightChromeWebBrowser();
    WebBrowser.GoToUrl(url);

    var page = await WebBrowser.GetPage<IndexPage>();
    var todoList = page.TableRecords;

    // assert
    todoList.Count().Should().Be(5);
}
```

- # #5 Нужны Web-тесты

  - Включить доступ по сети

- # #5 Нужны Web-тесты

  - Включить доступ по сети
  - Запустить или замокать бэкенд апи

- # #5 Нужны Web-тесты

  - Включить доступ по сети
  - Запустить или замокать бэкенд апи
  - Запустить WebDriver

L1-тесты. Плюсы

L1-тесты. Плюсы

- **Проще запустить**
- Не нужно ничего деплоить.
- Не нужно ничего устанавливать локально – все запускается прямо из IDE.
- Нет внешних зависимостей.

L1-тесты. Плюсы

- **Проще запустить**
- Не нужно ничего деплоить.
- Не нужно ничего устанавливать локально – все запускается прямо из IDE.
- Нет внешних зависимостей.

- **Обратная связь стала быстрее**
- Можно запускать код приложений в режиме дебага.
- Не нужно подготавливать ресурсы на развернутой среде.

L1-тесты. Плюсы

- **Проще запустить**
- Не нужно ничего деплоить.
- Не нужно ничего устанавливать локально – все запускается прямо из IDE.
- Нет внешних зависимостей.

- **Обратная связь стала быстрее**
- Можно запускать код приложений в режиме дебага.
- Не нужно подготавливать ресурсы на развернутой среде.

- **Понятнее всем**
- Автотесты перестали быть «отдельной системой».
- Разработчики могут отлаживать свой код такими тестами.
- Тестировщики могут раньше приступать к написанию тестов.

L1-тесты. Плюсы

- **Проще запустить**
- Не нужно ничего деплоить.
- Не нужно ничего устанавливать локально – все запускается прямо из IDE.
- Нет внешних зависимостей.

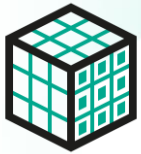
- **Обратная связь стала быстрее**
- Можно запускать код приложений в режиме дебага.
- Не нужно подготавливать ресурсы на развернутой среде.

- **Понятнее всем**
- Автотесты перестали быть «отдельной системой».
- Разработчики могут отлаживать свой код такими тестами.
- Тестировщики могут раньше приступать к написанию тестов.

- **Больше тестов**
- Можно заменить что угодно на моки.
- Можно реализовать любое поведение.

L1-тесты. Недостатки

L1-тесты. Недостатки



- **Стало сложнее**
- Написание таких тестов требует от тестировщика больше навыков.
- Вероятно, для написания мок-сервисов понадобится помощь разработчиков.

L1-тесты. Недостатки

- **Стало сложнее**
- Написание таких тестов требует от тестировщика больше навыков.
- Вероятно, для написания мок-сервисов понадобится помощь разработчиков.

- **Мы стали доверчивее, тестируем не все.**
- Заменили внешние библиотеки на моки – перестали их тестировать.
- Нет интеграций – не проверяем внешние контракты.
- Нет деплоя – не проверяем особенностей конкретной среды развертывания.

L1-тесты. Недостатки

- **Стало сложнее**
- Написание таких тестов требует от тестировщика больше навыков.
- Вероятно, для написания мок-сервисов понадобится помощь разработчиков.

- **Мы стали доверчивее, тестируем не все.**
- Заменили внешние библиотеки на моки – перестали их тестировать.
- Нет интеграций – не проверяем внешние контракты.
- Нет деплоя – не проверяем особенностей конкретной среды развертывания.

- **Всегда с чистого листа**
- Тесты всегда стартуют на измененной системе. Нет старых данных, которые вероятно остались в реальной системе. Нужно не забывать и явно думать о таких тест-кейсах.

L1-тесты. Недостатки

- **Стало сложнее**
- Написание таких тестов требует от тестировщика больше навыков.
- Вероятно, для написания мок-сервисов понадобится помощь разработчиков.

- **Мы стали доверчивее, тестируем не все.**
- Заменили внешние библиотеки на моки – перестали их тестировать.
- Нет интеграций – не проверяем внешние контракты.
- Нет деплоя – не проверяем особенностей конкретной среды развертывания.

- **Всегда с чистого листа**
- Тесты всегда стартуют на измененной системе. Нет старых данных, которые вероятно остались в реальной системе. Нужно не забывать и явно думать о таких тест-кейсах.

- **Нужно писать в два раза больше тестов**
- Сначала мокаем сервис «слева», потом мокаем севрис «справа» - раньше это был один интеграционный тест.
- Нужно писать тесты на старые данные – раньше они просто были.

L1-тесты. Рекомендации

L1-тесты. Рекомендации

- **Подменяйте только внешние библиотеки**
- Если подменять свои библиотеки, то часть кода останется непротестированной.

L1-тесты. Рекомендации

- **Подменяйте только внешние библиотеки**
- Если подменять свои библиотеки, то часть кода останется непротестированной.

- **L1-тесты не отменяют более высокоуровневых тестов**
- Но, чем больше у вас будет L1-тестов, тем меньше их останется на интеграционном уровне.

L1-тесты. Рекомендации

- **Подменяйте только внешние библиотеки**
- Если подменять свои библиотеки, то часть кода останется непротестированной.

- **L1-тесты не отменяют более высокоуровневых тестов**
- Но, чем больше у вас будет L1-тестов, тем меньше их останется на интеграционном уровне.

- **Пишите всей командой**
- Разработчики смогут быстро найти сервисы которые можно подменить.
- Тестировщики смогут значительно повысить читаемость и понятность тестов.

L1-тесты. Рекомендации

- **Подменяйте только внешние библиотеки**
- Если подменять свои библиотеки, то часть кода останется непротестированной.

- **L1-тесты не отменяют более высокоуровневых тестов**
- Но, чем больше у вас будет L1-тестов, тем меньше их останется на интеграционном уровне.

- **Пишите всей командой**
- Разработчики смогут быстро найти сервисы которые можно подменить.
- Тестировщики смогут значительно повысить читаемость и понятность тестов.

- **Попробуйте внедрить TDD (TestDrivenDevelopment)**
- L1-тесты позволяют организовать процесс разработки в формате:
  1 – написали один тест, как требование. 2 – разработчик реализует то, что ожидает тест. 3 – тестировщик дописывает больше тестов + исправляет исходный.

L1-тесты. Рекомендации

- **Подменяйте только внешние библиотеки**
- Если подменять свои библиотеки, то часть кода останется непротестированной.

- **L1-тесты не отменяют более высокоуровневых тестов**
- Но, чем больше у вас будет L1-тестов, тем меньше их останется на интеграционном уровне.

- **Пишите всей командой**
- Разработчики смогут быстро найти сервисы которые можно подменить.
- Тестировщики смогут значительно повысить читаемость и понятность тестов.

- **Попробуйте внедрить TDD (TestDrivenDevelopment)**
- L1-тесты позволяют организовать процесс разработки в формате:
  1 – написали один тест, как требование. 2 – разработчик реализует то, что ожидает тест. 3 – тестировщик дописывает больше тестов + исправляет исходный.

- **По возможности используйте базовый InMemory подход**
- Несмотря на то, что L1-тесты довольно быстрые, лучше воздержаться от включения везде Kestrel и использования реальной сети. Тестирование в памяти значительно быстрее.

https://careers.kaspersky.ru/tech/stack/SDET

https://github.com/Sugrob57/TestServer_samples

kaspersky

t.me@Sugrob57