Formación de Testing en .NET

Nafarroako Gobernua
Gobierno de Navarra

hiberus©

La compañía hiperespecializada en las TIC

Tema 5: Escenarios complejos

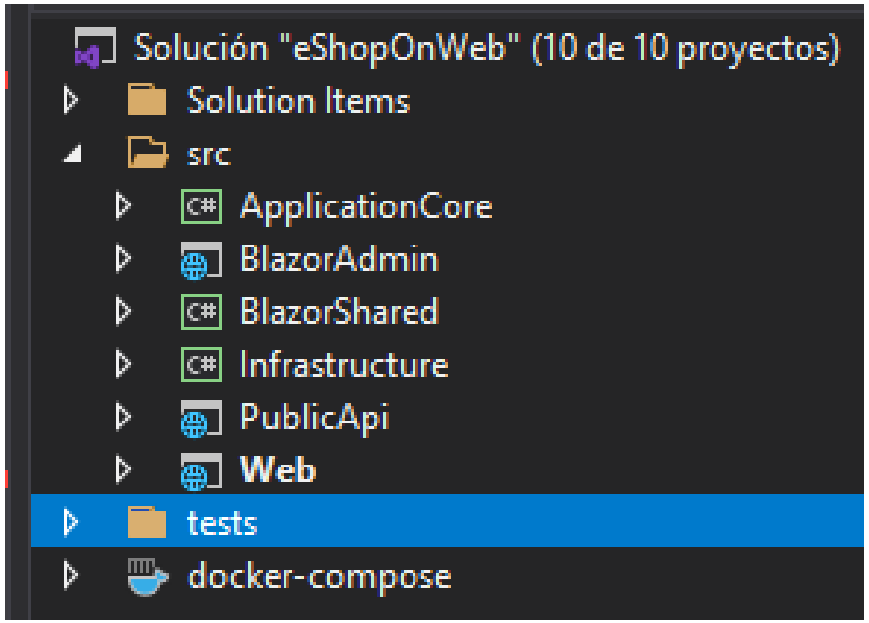
Nafarroako Gobernua
Gobierno de Navarra

www.hiberus.com

Real-World Test

N-Layer Arquitecture

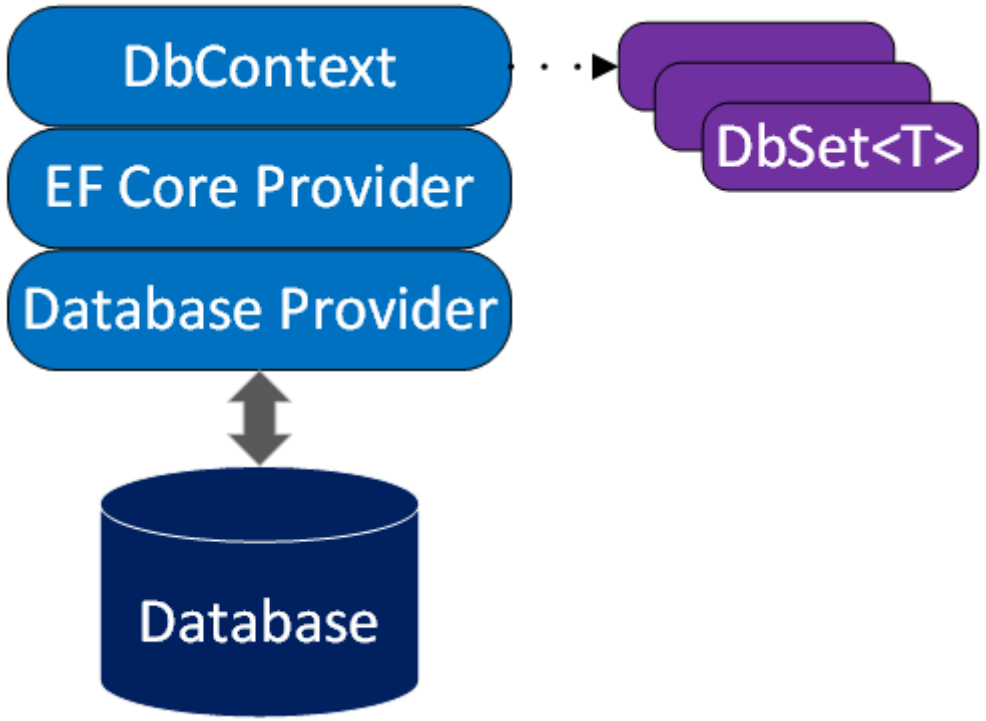
https://github.com/dotnet-architecture/eShopOnWeb



Entity Framework

Entity Framework

Simplifying DB environments



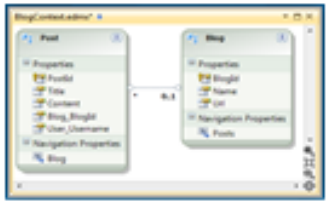
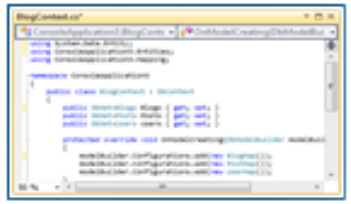
- **Database First**

- **Model First**

- **Code First**

Entity Framework

Simplifying DB environments

	Designer Centric	Code Centric
		
New Database	**Model First** - Create .edmx model in designer - Generate database from -edmx - Classes auto-generate from .edmx	**Code First** - Define dasses & mapping in code - Database auto-created at runtime
Existing Database	**Database First** - Reverse engineer .edmx model - Classes auto-generate from .edmx	**Code First** - Define dasses & mapping in code (Reverse engineer tools available)

Entity Framework

Simplifying DB environments

#somoshiberus

```
[Table("SyncTable")]
6 referencias | 0 cambios | 0 autores, 0 cambios
public class AppSyncVersionEntity : BaseIDEntity
{
    [Required]
    [Column("username", TypeName = Constantes.SQL_NVARCHAR_50)]
    4 referencias | 0 cambios | 0 autores, 0 cambios
    public string UserName { get; set; }

    [Required]
    [Column("syncname", TypeName = Constantes.SQL_NVARCHAR_50)]
    3 referencias | 0 cambios | 0 autores, 0 cambios
    public string SyncName { get; set; }

    [Required]
    [Column("version", TypeName = Constantes.SQL_BIGINT)]
    3 referencias | 0 cambios | 0 autores, 0 cambios
    public long Version { get; set; }
}
```

Clean Architecture

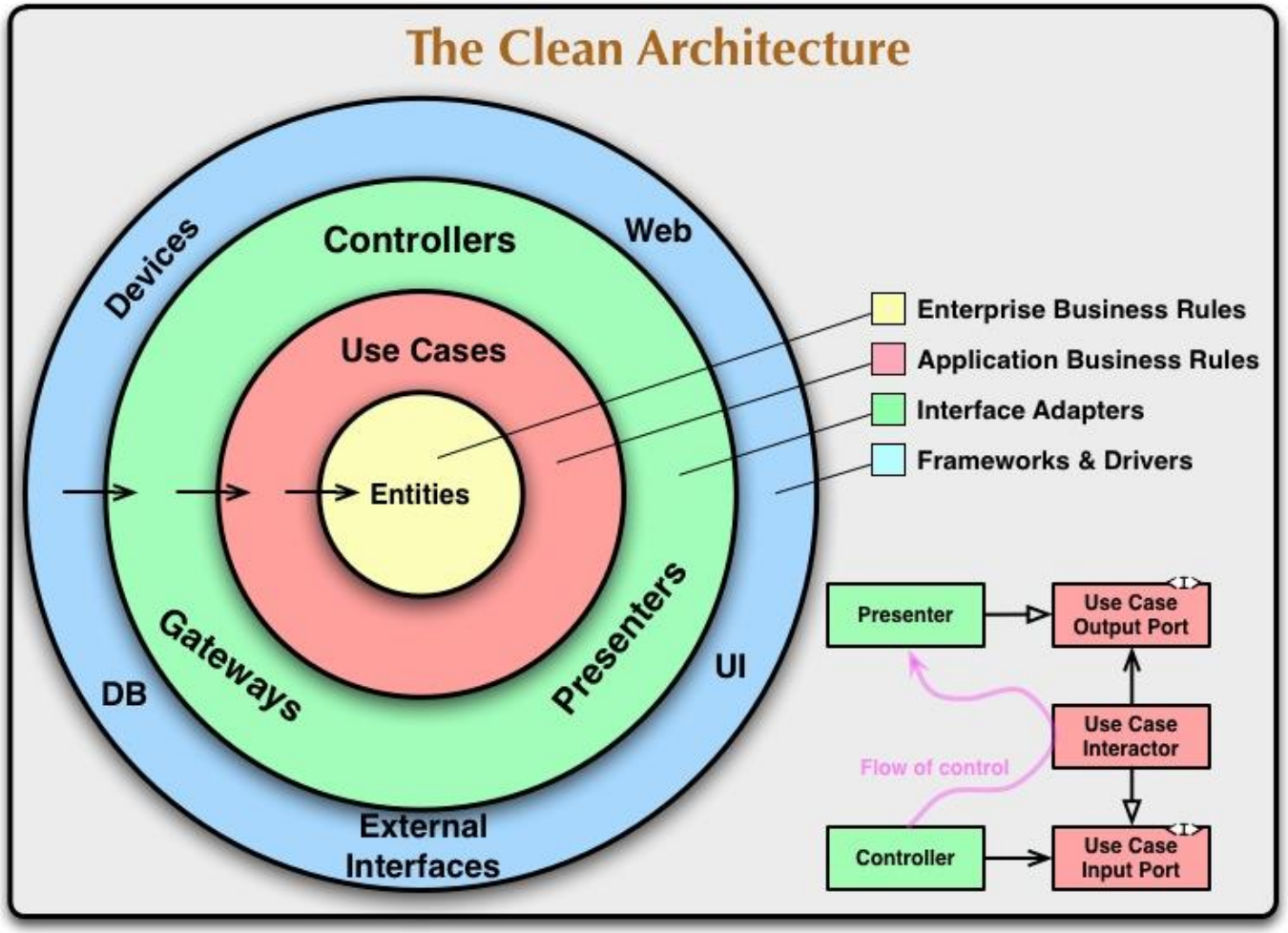
SOLID Principles

- *Single Responsibility Principle*
- *Open/Closed Principle*
- *Liskov Substitution Principle*
- *Interface Segregation Principle*
- *Dependency Inversion Principle*

```
public class BasketViewModelService : IBasketViewModelService
{
    private readonly IRepository<Basket> _basketRepository;
    private readonly IUriComposer _uriComposer;
    private readonly IBasketQueryService _basketQueryService;
    private readonly IRepository<CatalogItem> _itemRepository;

    0 referencias
    public BasketViewModelService(IRepository<Basket> basketRepository,
        IRepository<CatalogItem> itemRepository,
        IUriComposer uriComposer,
        IBasketQueryService basketQueryService)
    {
        _basketRepository = basketRepository;
        _uriComposer = uriComposer;
        _basketQueryService = basketQueryService;
        _itemRepository = itemRepository;
    }

    5 referencias
    public async Task<BasketViewModel> GetOrCreateBasketForUser(string userName)
    {
        var basketSpec = new BasketWithItemsSpecification(userName);
        var basket = (await _basketRepository.GetBySpecAsync(basketSpec));

        if (basket == null)
        {
            return await CreateBasketForUser(userName);
        }
        var viewModel = await Map(basket);
        return viewModel;
    }
```

Clean Architecture

Clean Principles



Clean Architecture

Entities & UseCases C# ApplicationCore

#somoshiberus

```
namespace Microsoft.eShopWeb.ApplicationCore.Entities;

17 referencias
public class CatalogBrand : BaseEntity, IAggregateRoot
{
    4 referencias
    public string Brand { get; private set; }
    5 referencias
    public CatalogBrand(string brand)
    {
        Brand = brand;
    }
}
```

```
public interface IBasketService
{
    8 referencias | 0/6 pasando
    Task TransferBasketAsync(string anonymousId,
    4 referencias | 0/2 pasando
    Task<Basket> AddItemToBasket(string username,
    6 referencias | 0/3 pasando
    Task<Basket> SetQuantities(int basketId, Dict
    3 referencias | 0/1 pasando
    Task DeleteBasketAsync(int basketId);
}
```

```
public class BasketService : IBasketService
{
    private readonly IRepository<Basket> _basketRepository;
    private readonly IAppLogger<BasketService> _logger;

    12 referencias | 0/12 pasando
    public BasketService(IRepository<Basket> basketRepository,
        IAppLogger<BasketService> logger)
    {
        _basketRepository = basketRepository;
        _logger = logger;
    }

    4 referencias | 0/2 pasando
    public async Task<Basket> AddItemToBasket(string username, int catalogItemId, decimal price, int quantity = 1)
    {
        var basketSpec = new BasketWithItemsSpecification(username);
        var basket = await _basketRepository.GetBySpecAsync(basketSpec);

        if (basket == null)
        {
            basket = new Basket(username);
            await _basketRepository.AddAsync(basket);
        }

        basket.AddItem(catalogItemId, price, quantity);

        await _basketRepository.UpdateAsync(basket);
        return basket;
    }

    3 referencias | 0/1 pasando
    public async Task DeleteBasketAsync(int basketId)
    {
        var basket = await _basketRepository.GetByIdAsync(basketId);
        await _basketRepository.DeleteAsync(basket);
    }
```

Clean Architecture

DB/external configuration C# Infrastructure

#somoshiberus

```
public class CatalogContext : DbContext
{
    3 referencias
    public CatalogContext(DbContextOptions<CatalogContext> options) : base(options)
    {
    }

    1 referencia
    public DbSet<Basket> Baskets { get; set; }
    2 referencias
    public DbSet<CatalogItem> CatalogItems { get; set; }
    2 referencias
    public DbSet<CatalogBrand> CatalogBrands { get; set; }
    2 referencias
    public DbSet<CatalogType> CatalogTypes { get; set; }
    3 referencias | 0/2 pasando
    public DbSet<Order> Orders { get; set; }
    0 referencias
    public DbSet<OrderItem> OrderItems { get; set; }
    0 referencias
    public DbSet<BasketItem> BasketItems { get; set; }

    0 referencias
    protected override void OnModelCreating(ModelBuilder builder)
    {
        base.OnModelCreating(builder);
        builder.ApplyConfigurationsFromAssembly(Assembly.GetExecutingAssembly());
    }
```

```
public class BasketConfiguration : IEntityTypeConfiguration<Basket>
{
    0 referencias
    public void Configure(EntityTypeBuilder<Basket> builder)
    {
        var navigation = builder.Metadata.FindNavigation(nameof(Basket.Items));
        navigation.SetPropertyAccessMode(PropertyAccessMode.Field);

        builder.Property(b => b.BuyerId)
            .IsRequired()
            .HasMaxLength(256);
    }
}
```

```
public partial class FixBuyerId : Migration
{
    0 referencias
    protected override void Up(MigrationBuilde
    {
        migrationBuilder.AlterColumn<string>(
            name: "BuyerId",
            table: "Orders",
            type: "nvarchar(256)",
            maxLength: 256,
            nullable: false,
            defaultValue: "",
            oldClrType: typeof(string),
            oldType: "nvarchar(max)",
            oldNullable: true);

        migrationBuilder.AlterColumn<string>(
            name: "BuyerId",
            table: "Baskets",
            type: "nvarchar(256)",
            maxLength: 256,
            nullable: false,
            oldClrType: typeof(string),
            oldType: "nvarchar(40)",
            oldMaxLength: 40);
    }
```

```
public class CatalogContextSeed
{
    3 referencias
    public static async Task SeedAsync(CatalogContext catalogContext,
        ILogger logger,
        int retry = 0)
    {
        var retryForAvailability = retry;
        try
        {
            if (catalogContext.Database.IsSqlServer())
            {
                catalogContext.Database.Migrate();
            }

            if (!await catalogContext.CatalogBrands.AnyAsync())
            {
                await catalogContext.CatalogBrands.AddRangeAsync(
                    GetPreconfiguredCatalogBrands());

                await catalogContext.SaveChangesAsync();
            }

            if (!await catalogContext.CatalogTypes.AnyAsync())
            {
                await catalogContext.CatalogTypes.AddRangeAsync(
                    GetPreconfiguredCatalogTypes());

                await catalogContext.SaveChangesAsync();
            }

            if (!await catalogContext.CatalogItems.AnyAsync())
            {
                await catalogContext.CatalogItems.AddRangeAsync(
                    GetPreconfiguredItems());
```

https://docs.microsoft.com/en-US/ef/core/managing-schemas/migrations/

Clean Architecture

Controllers

PublicApi

#somoshiberus

List.cs | CatalogContext.cs | BasketService.cs | Program.cs

PublicApi > Microsoft.eShopWeb.PublicApi.CatalogBrandEndpoints.List

```
10
11  namespace Microsoft.eShopWeb.PublicApi.CatalogBrandEndpoints;
12
    1 referencia
13  public class List : EndpointBaseAsync
14      .WithoutRequest
15      .WithActionResult<ListCatalogBrandsResponse>
16  {
17      private readonly IRepository<CatalogBrand> _catalogBrandRepository;
18      private readonly IMapper _mapper;
19
        0 referencias
20      public List(IRepository<CatalogBrand> catalogBrandRepository,
21          IMapper mapper)
22      {
23          _catalogBrandRepository = catalogBrandRepository;
24          _mapper = mapper;
25      }
26
27      [HttpGet("api/catalog-brands")]
28      [SwaggerOperation(
29          Summary = "List Catalog Brands",
30          Description = "List Catalog Brands",
31          OperationId = "catalog-brands.List",
32          Tags = new[] { "CatalogBrandEndpoints" })
33      ]
        0 referencias
34      public override async Task<ActionResult<ListCatalogBrandsResponse>> HandleAsync(CancellationToken cancellationToken)
35      {
36          var response = new ListCatalogBrandsResponse();
37
38          var items = await _catalogBrandRepository.ListAsync(cancellationToken);
39
40          response.CatalogBrands.AddRange(items.Select(_mapper.Map<CatalogBrandDto>));
41
42          return Ok(response);
43      }
44  }
45
```

88 % | No se encontraron problemas.

Explorador de soluciones

Buscar en Explorador de soluciones (Ctrl+`)

- Solución "eShopOnWeb" (10 de 10 proyectos)
  - Solution Items
  - src
    - ApplicationCore
    - BlazorAdmin
    - BlazorShared
    - Infrastructure
    - PublicApi
      - Connected Services
      - Dependencias
      - Properties
      - AuthEndpoints
      - CatalogBrandEndpoints
        - CatalogBrandDto.cs
        - List.cs
      - CatalogItemEndpoints
        - CatalogItemDto.cs
        - Create.cs
        - Delete.cs
        - GetById.cs
        - ListPaged.cs
        - Update.cs
      - CatalogTypeEndpoints
        - CatalogTypeDto.cs
        - List.cs
      - Middleware
      - appsettings.json
      - BaseMessage.cs
      - BaseRequest.cs
      - BaseResponse.cs
      - CustomSchemaFilters.cs

Clean Architecture

Dependency Injection PublicApi

#somoshiberus

Program.cs

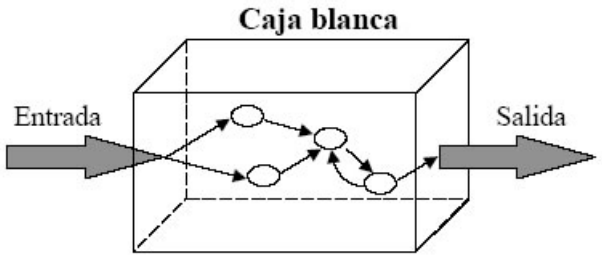
PublicApi

```
32
33      // use real database
34      // Requires LocalDB which can be installed with SQL Server Express 2016
35      // https://www.microsoft.com/en-us/download/details.aspx?id=54284
36      builder.Services.AddDbContext<CatalogContext>(c =>
37          c.UseSqlServer(builder.Configuration.GetConnectionString("CatalogConnection")));
38
39      // Add Identity DbContext
40      builder.Services.AddDbContext<AppIdentityDbContext>(options =>
41          options.UseSqlServer(builder.Configuration.GetConnectionString("IdentityConnection")));
42
43      builder.Services.AddIdentity<ApplicationUser, IdentityRole>()
44              .AddEntityFrameworkStores<AppIdentityDbContext>()
45              .AddDefaultTokenProviders();
46
47      builder.Services.AddScoped(typeof(IRepository<>), typeof(EfRepository<>));
48      builder.Services.AddScoped(typeof(IReadRepository<>), typeof(EfRepository<>));
49      builder.Services.Configure<CatalogSettings>(builder.Configuration);
50      builder.Services.AddSingleton<IUriComposer>(new UriComposer(builder.Configuration.Get<CatalogSettings>()));
51      builder.Services.AddScoped(typeof(IAppLogger<>), typeof(LoggerAdapter<>));
52      builder.Services.AddScoped<ITokenClaimsService, IdentityTokenClaimService>();
53
54      var configSection = builder.Configuration.GetRequiredSection(BaseUrlConfiguration.CONFIG_NAME);
55      builder.Services.Configure<BaseUrlConfiguration>(configSection);
56      var baseUrlConfig = configSection.Get<BaseUrlConfiguration>();
57
58      builder.Services.AddMemoryCache();
59
60      var key = Encoding.ASCII.GetBytes(AuthorizationConstants.JWT_SECRET_KEY);
61      builder.Services.AddAuthentication(config =>
62      {
63          config.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
64      })
65      .AddJwtBearer(config =>
66      {
67          config.RequireHttpsMetadata = false;
68          config.SaveToken = true;
69          config.TokenValidationParameters = new TokenValidationParameters
70          {
71              ValidateIssuerSigningKey = true,
72              IssuerSigningKey = new SymmetricSecurityKey(key),
73              ValidateIssuer = false,
74              ValidateAudience = false
75          };
76      });
77
```

Clean Architecture

Test definition

#somoshiberus



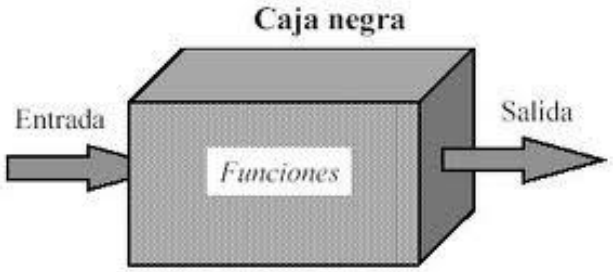

```
11
12          return Char.IsUpper(s[0]);
13      }
14
        0 references
15      public static bool StartsWithLower(this string s)
16      {
17          if (String.IsNullOrWhiteSpace(s))
18              return false;
```

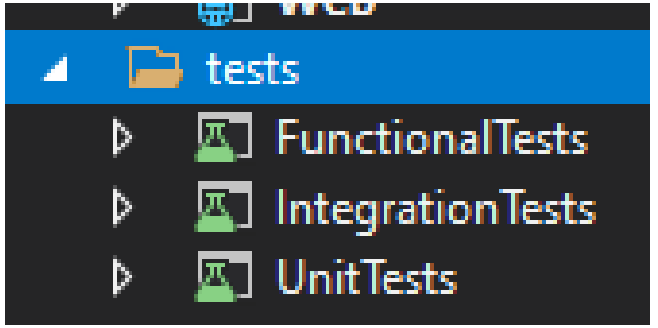
Test

TestHasEmbeddedSpaces

Run All | Debug All

```
24      {
25          foreach (var ch in s.Trim())
26          {
27              if (ch == ' ')
28                  return true;
29          }
30          return false;
31      }
32  }
33  }
```

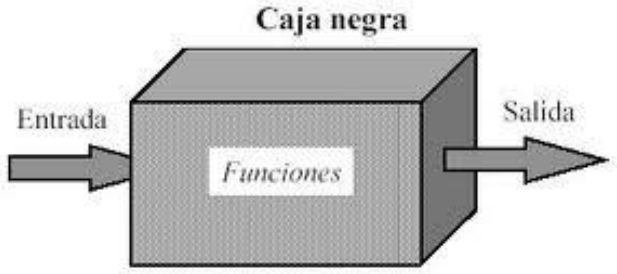


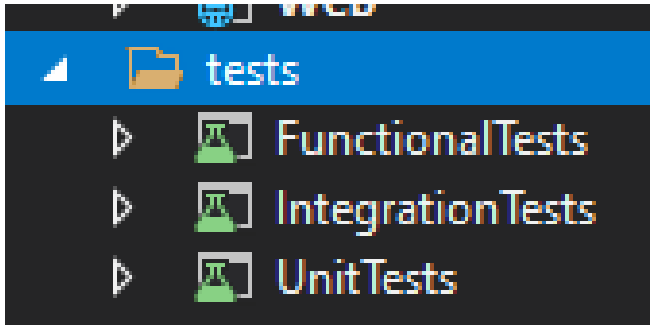


Clean Architecture

AAA: Arrange, Act, Assert

#somoshiberus



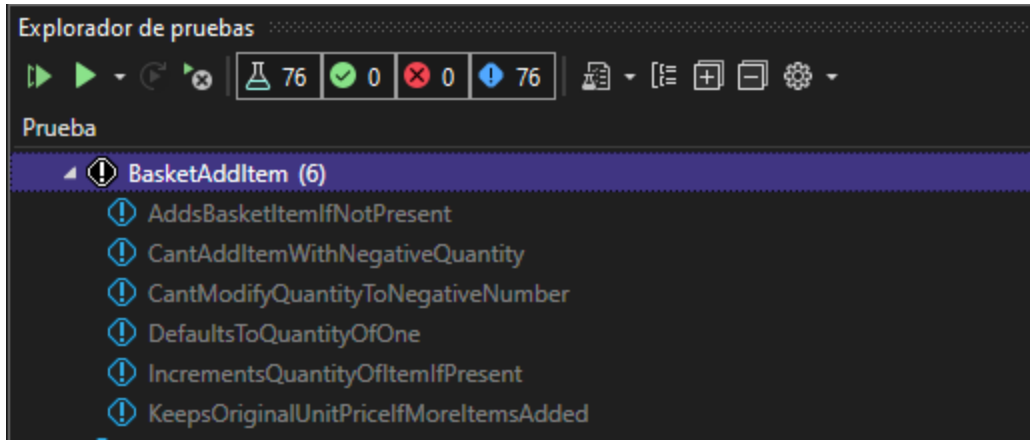



```
// arrange
var repository = Substitute.For<IClientRepository>();
var client = new Client(repository);

// act
client.Save();

// assert
mock.Received.SomeMethod();
```

Clean Architecture

xUnit: Naming and Categorization


```
public class BasketAddItem
{
    private readonly int _testCatalogItemId = 123;
    private readonly decimal _testUnitPrice = 1.23m;
    private readonly int _testQuantity = 2;
    private readonly string _buyerId = "Test buyerId";

    [Fact]
    0 referencias
    public void AddsBasketItemIfNotPresent()
    {
        var basket = new Basket(_buyerId);
        basket.AddItem(_testCatalogItemId, _testUnitPrice, _testQuantity);

        var firstItem = basket.Items.Single();
        Assert.Equal(_testCatalogItemId, firstItem.CatalogItemId);
        Assert.Equal(_testUnitPrice, firstItem.UnitPrice);
        Assert.Equal(_testQuantity, firstItem.Quantity);
    }
```

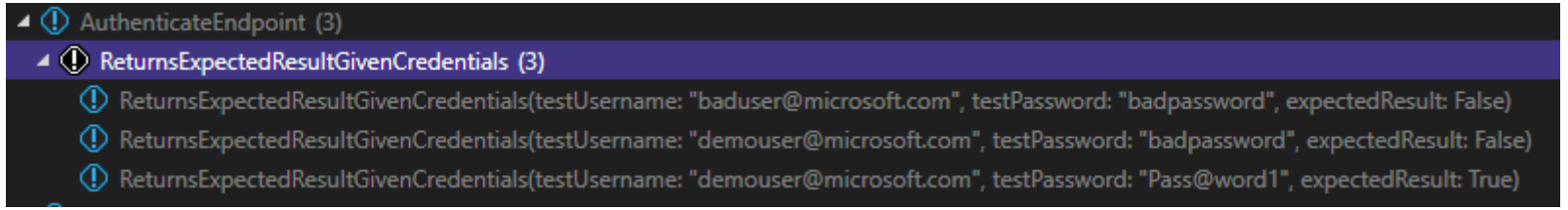


Clean Architecture

xUnit: Theories

#somoshiberus

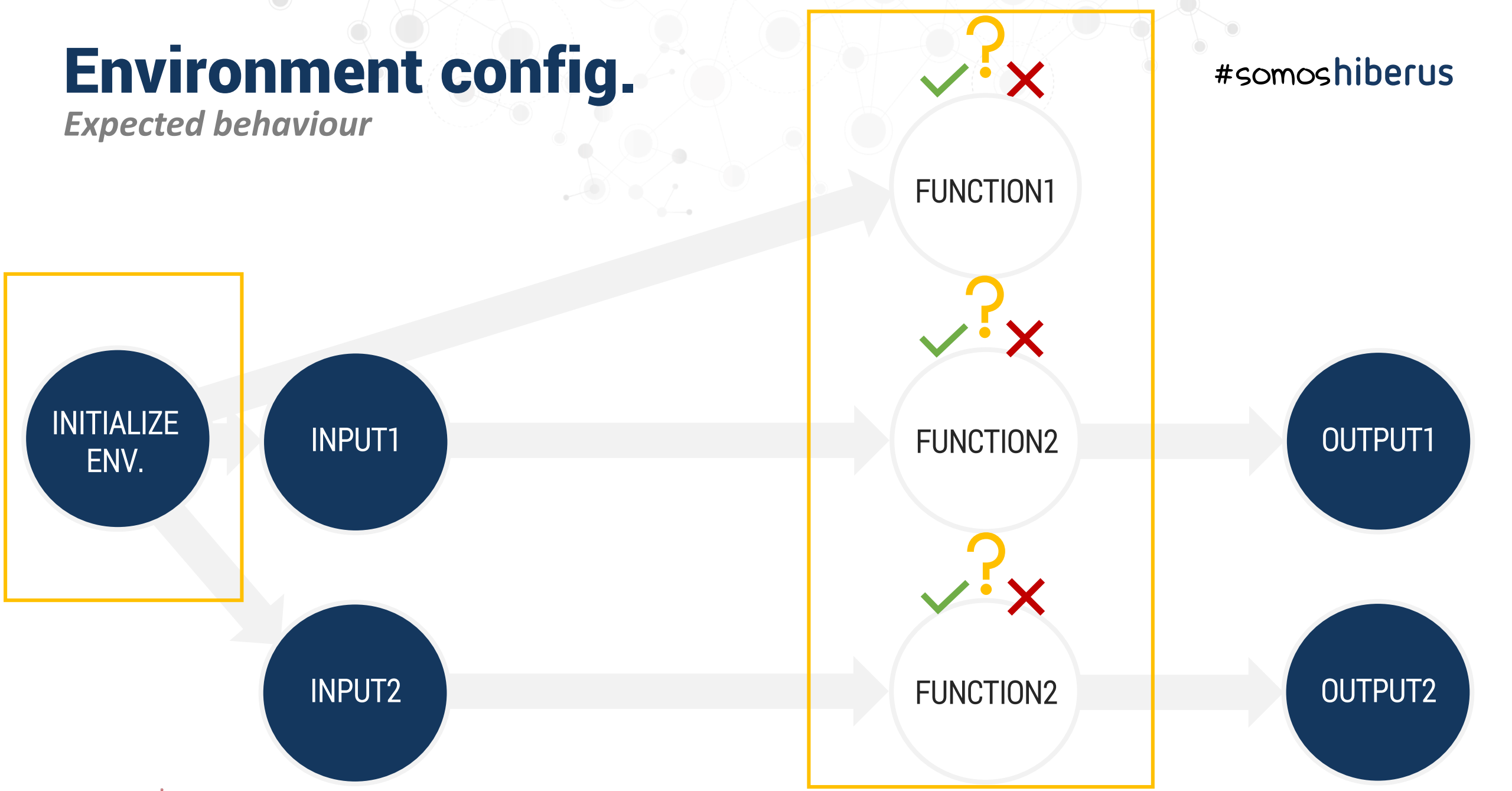
```
[Theory]
[InlineData("demouser@microsoft.com", AuthorizationConstants.DEFAULT_PASSWORD, true)]
[InlineData("demouser@microsoft.com", "badpassword", false)]
[InlineData("baduser@microsoft.com", "badpassword", false)]
0 referencias
public async Task ReturnsExpectedResultGivenCredentials(string testUsername, string testPassword, bool expectedResult)
{
    var request = new AuthenticateRequest()
    {
        Username = testUsername,
        Password = testPassword
    };
    var jsonContent = new StringContent(JsonSerializer.Serialize(request), Encoding.UTF8, "application/json");
    var response = await Client.PostAsync("api/authenticate", jsonContent);
    response.EnsureSuccessStatusCode();
    var stringResponse = await response.Content.ReadAsStringAsync();
    var model = stringResponse.FromJson<AuthenticateResponse>();

    Assert.Equal(expectedResult, model.Result);
}
```



Environment config.

Expected behaviour

#somoshiberus



Environment config.

xUnit: Fixtures

#somoshiberus

```
[Collection("Sequential")]
1 referencia
public class AuthenticateEndpoint : IClassFixture<TestApiApplication>
{
    JsonSerializerOptions _jsonOptions = new JsonSerializerOptions { P

    0 referencias
    public AuthenticateEndpoint(TestApiApplication factory)
    {
        Client = factory.CreateClient();
    }
```

```
public class TestApiApplication : WebApplicationFactory<Authenticate>
{
    private readonly string _environment = "Testing";

    0 referencias
    protected override IHost CreateHost(IHostBuilder builder)
    {
        builder.UseEnvironment(_environment);

        // Add mock/test services to the builder here
        builder.ConfigureServices(services =>
        {
            services.AddScoped(sp =>
            {
                // Replace SQLite with in-memory database for tests
                return new DbContextOptionsBuilder<CatalogContext>()
                .UseInMemoryDatabase("DbForPublicApi")
                .UseApplicationServiceProvider(sp)
                .Options;
            });
            services.AddScoped(sp =>
            {
                // Replace SQLite with in-memory database for tests
                return new DbContextOptionsBuilder<AppIdentityDbContext>()
                .UseInMemoryDatabase("IdentityDbForPublicApi")
                .UseApplicationServiceProvider(sp)
                .Options;
            });
        });
```

Environment config.

WebApplicationFactory

#somoshiberus

`TEntryPoint`

A type in the entry point assembly of the application. Typically the Startup or Program classes can be used.

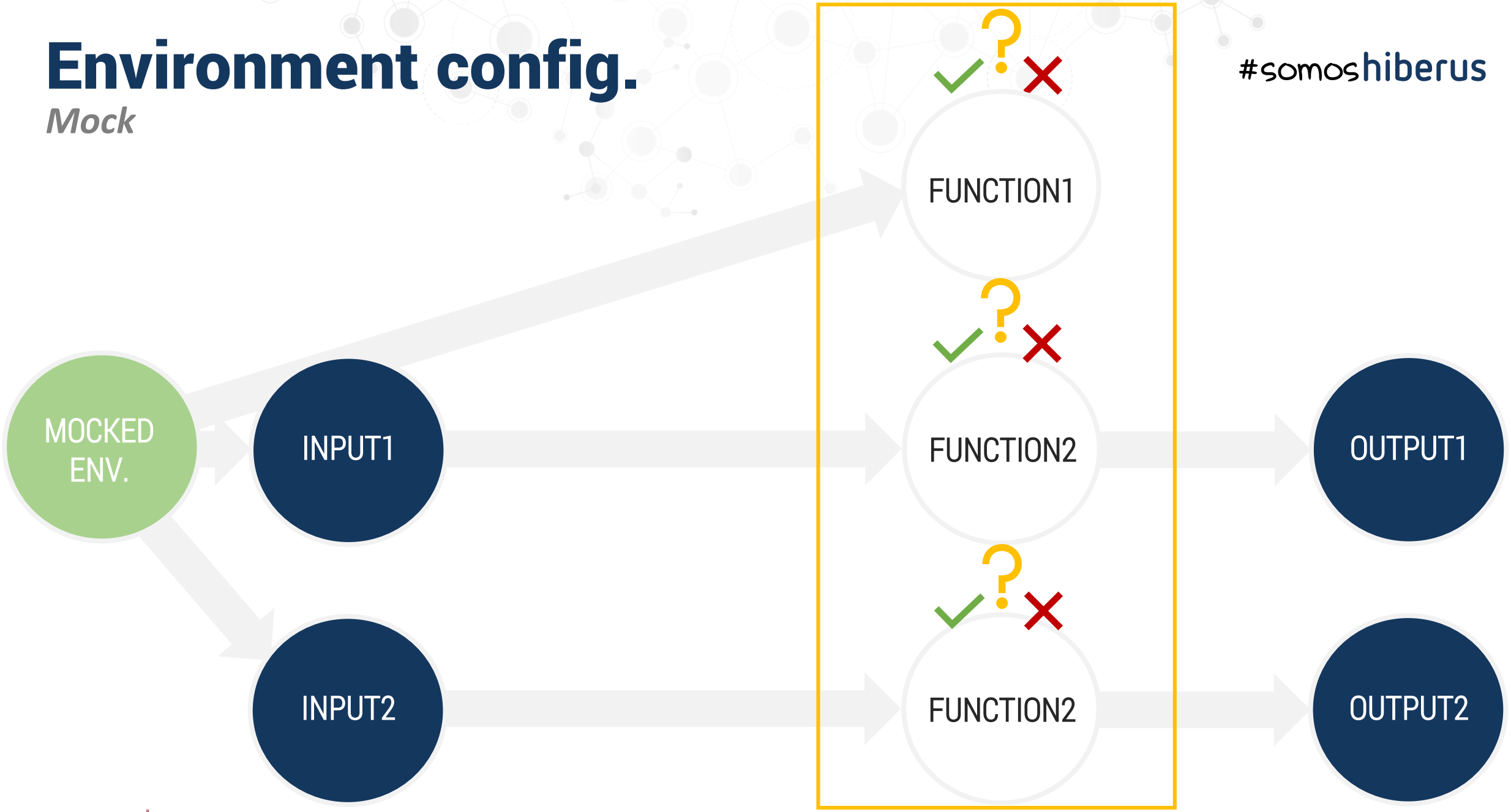
Constructors

WebApplicationFactory<TEntryPoint>()	Creates an instance of WebApplicationFactory<TEntryPoint>. This factory can be used to create a TestServer instance using the MVC application defined by `TEntryPoint` and one or more HttpClient instances used to send HttpRequestMessage to the TestServer. The WebApplicationFactory<TEntryPoint> will find the entry point class of `TEntryPoint` assembly and initialize the application by calling `IWebHostBuilder CreateWebHostBuilder(string [] args)` on `TEntryPoint`. This constructor will infer the application content root path by searching for a WebApplicationFactoryContentRootAttribute on the assembly containing the functional tests with a key equal to the `TEntryPoint` assembly FullName. In case an attribute with the right key can't be found, WebApplicationFactory<TEntryPoint> will fall back to searching for a solution file (*.sln) and then appending `TEntryPoint` assembly name to the solution directory. The application root directory will be used to discover views and content files. The application assemblies will be loaded from the dependency context of the assembly containing `TEntryPoint`. This means that project dependencies of the assembly containing `TEntryPoint` will be loaded as application assemblies.

Environment config.

Mock

#somoshiberus



Environment config.

Mock - Test Doubles

- **Dummy**
  Passed around, but not actually used (e.g. need to fill parameter list)
- **Fake**
  Have working implementation, but have shortcuts (not to use in prod.)
- **Stubs**
  Provide canned answers to calls, not responding if other inputs
- **Spies**
  Stubs with memory; e.g. mail service with count of sent messages
- **Mocks**
  Pre-programmed objects with expected behaviour

Environment config.

Mock - Test Doubles


```
public const string AUTH_KEY = "AuthKeyOfDoomThatMustBeAMinimumNumberOfBytes";
```

```
private readonly Mock<IRepository<Basket>> _mockBasketRepo = new();
```

```
_mockBasketRepo.Setup(x => x.GetBySpecAsync(It.IsAny<BasketWithItemsSpecification>(), default)).ReturnsAsync(basket);
```

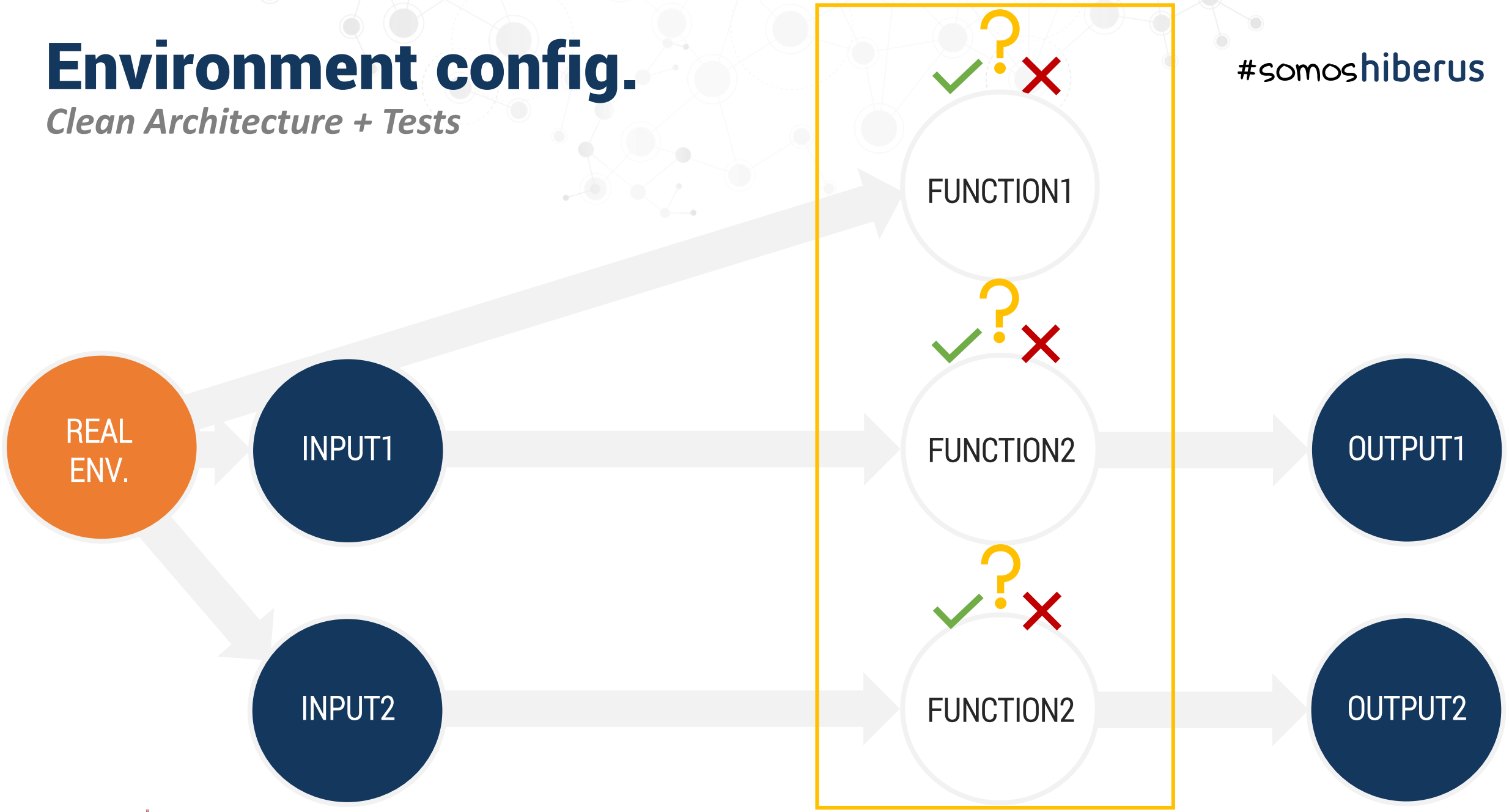
```
_mockBasketRepo.SetupSequence(x => x.GetBySpecAsync(It.IsAny<BasketWithItemsSpecification>(), default))
    .ReturnsAsync(anonymousBasket)
    .ReturnsAsync(userBasket);
```

```
_mockBasketRepo.Setup(x => x.GetBySpecAsync(It.IsAny<BasketWit

var basketService = new BasketService(_mockBasketRepo.Object,
```

Environment config.

Clean Architecture + Tests

#somoshiberus



Environment config.

Clean Architecture + Tests

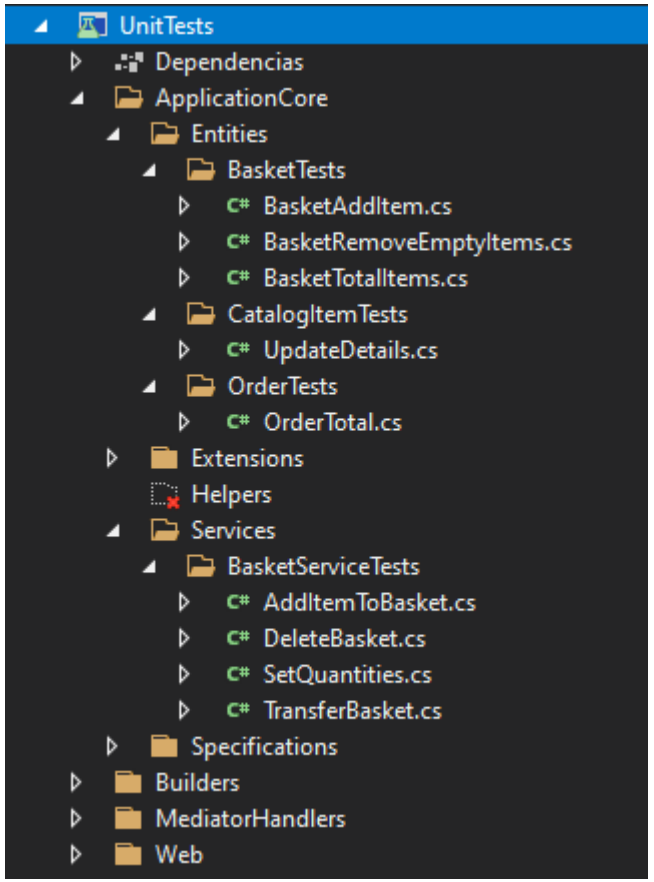
- **Unit Testing**
  Test that individual modules **issolated**
  are working as expected
- **Integration Testing**
  Test that the interaction of differentes modules **working together**
  are working as expected
- **Functional Testing**
  Test that **specific functionalities** of our system
  are working as expected

Environment config.

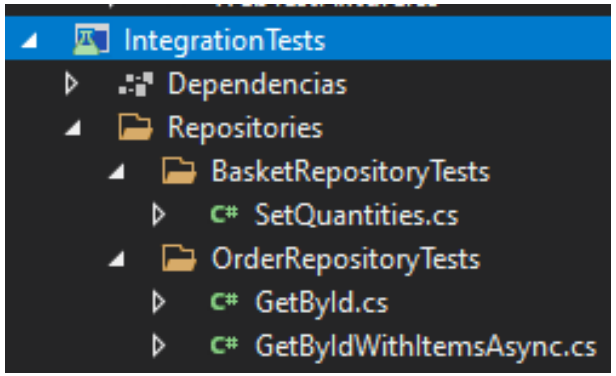
Clean Architecture + Tests

#somoshiberus

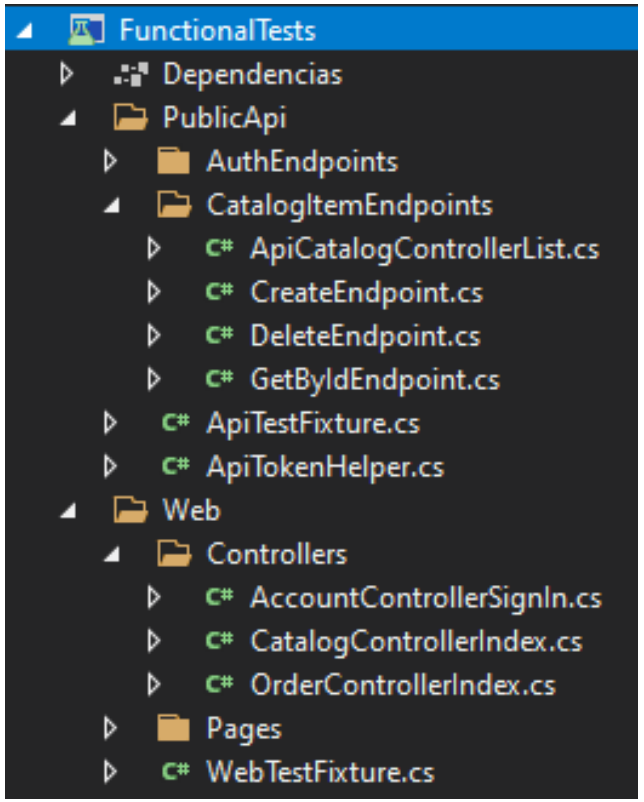
Unit Testing



Integration Testing



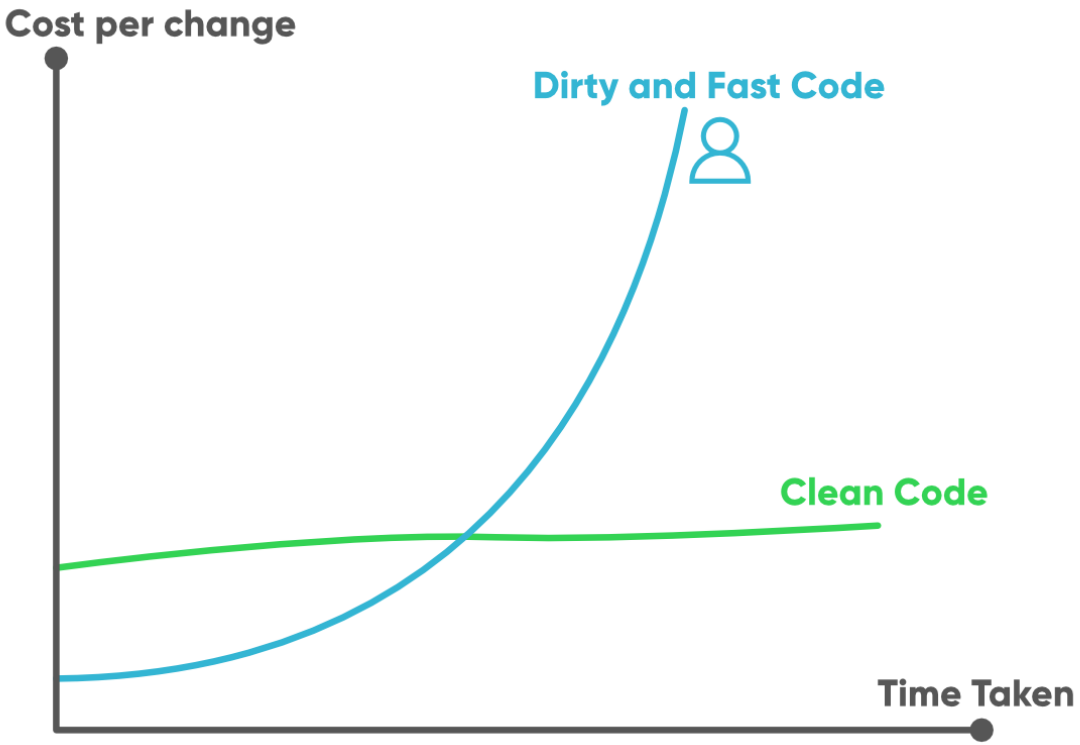
Functional Testing

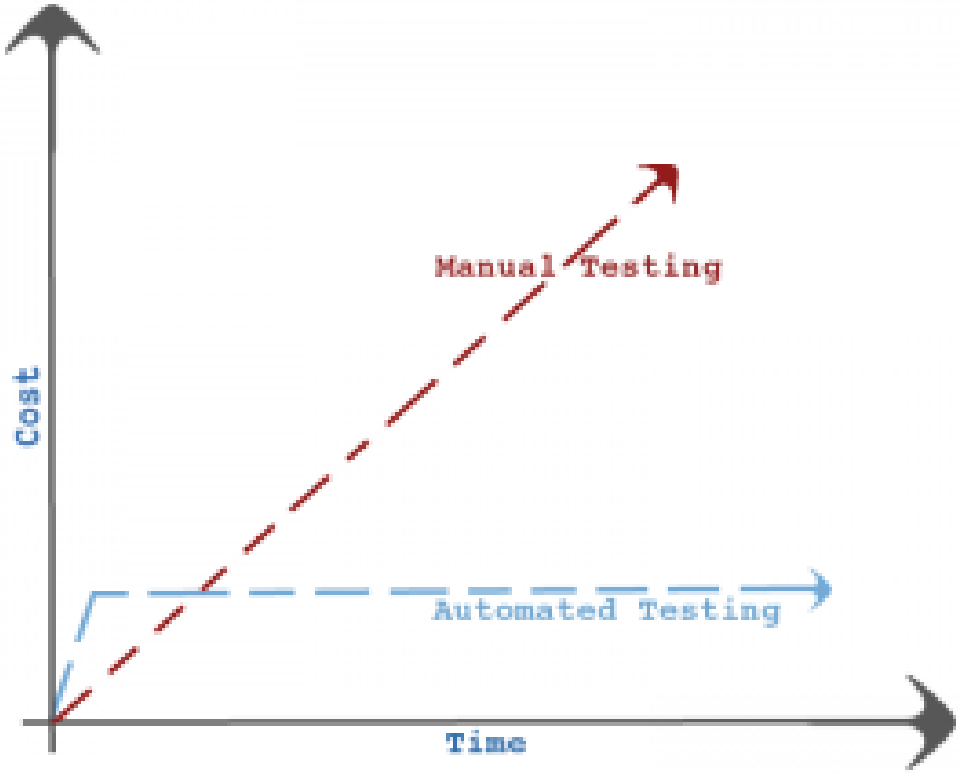


Environment config.

Benefits

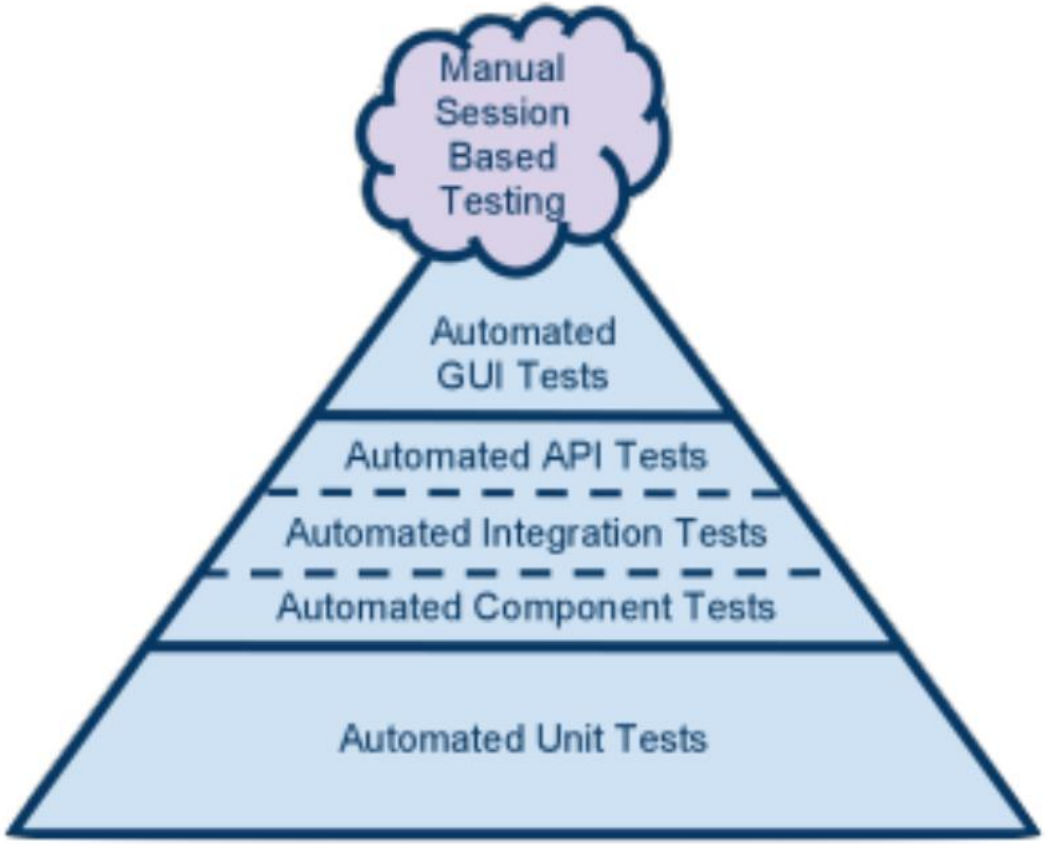
#somoshiberus





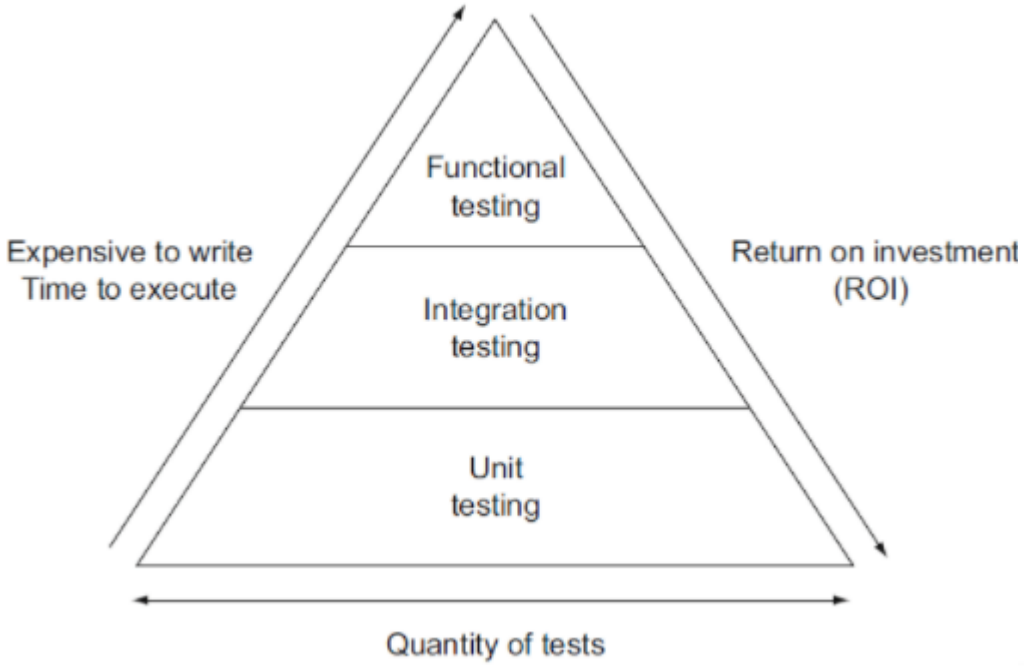
Environment config.

Benefits



Environment config.

Benefits



Environment config.

Unit, Integration & Functional Tests

Unit Test

```
namespace Microsoft.eShopWeb.UnitTests.ApplicationCore.Services.BasketServiceTests

0 referencias
public class SetQuantities
{
    private readonly int _invalidId = -1;
    private readonly Mock<IRepository<Basket>> _mockBasketRepo = new();

    [Fact]
    0 referencias
    public async Task ThrowsGivenInvalidBasketId()
    {
        var basketService = new BasketService(_mockBasketRepo.Object, null);

        await Assert.ThrowsAsync<BasketNotFoundException>(async () =>
            await basketService.SetQuantities(_invalidId, new System.Collections.G
    }
```

Environment config.

Unit, Integration & Functional Tests

Integration Test

```
namespace Microsoft.eShopWeb.IntegrationTests.Repositories.BasketRepositoryTests

1 referencia
public class SetQuantities
{
    private readonly CatalogContext _catalogContext;
    private readonly EfRepository<Basket> _basketRepository;
    private readonly BasketBuilder BasketBuilder = new BasketBuilder();

    0 referencias
    public SetQuantities()
    {
        var dbOptions = new DbContextOptionsBuilder<CatalogContext>()
            .UseInMemoryDatabase(databaseName: "TestCatalog")
            .Options;
        _catalogContext = new CatalogContext(dbOptions);
        _basketRepository = new EfRepository<Basket>(_catalogContext);
    }
```

```
[Fact]
0 referencias
public async Task RemoveEmptyQuantities()
{
    var basket = BasketBuilder.WithOneBasketItem();
    var basketService = new BasketService(_basketRepository, null);
    await _basketRepository.AddAsync(basket);
    _catalogContext.SaveChanges();

    await basketService.SetQuantities(BasketBuilder.BasketId, new Di

    Assert.Equal(0, basket.Items.Count);
}
```

Environment config.

Unit, Integration & Functional Tests

#somoshiberus

Functional Test

```
[Collection("Sequential")]
1 referencia
public class ApiCatalogControllerList : IClassFixture<TestApiApplication>
{
    0 referencias
    public ApiCatalogControllerList(TestApiApplication factory)
    {
        Client = factory.CreateClient();
    }

    3 referencias
    public HttpClient Client { get; }

    [Fact]
    0 referencias
    public async Task ReturnsFirst10CatalogItems()
    {
        var response = await Client.GetAsync("/api/catalog-items?pageSize=10");
        response.EnsureSuccessStatusCode();
        var stringResponse = await response.Content.ReadAsStringAsync();
        var model = stringResponse.FromJson<CatalogIndexViewModel>();

        Assert.Equal(10, model.CatalogItems.Count());
    }
```

```
public class TestApiApplication : WebApplicationFactory<Authenticate>
{
    private readonly string _environment = "Testing";

    0 referencias
    protected override IHost CreateHost(IHostBuilder builder)
    {
        builder.UseEnvironment(_environment);

        // Add mock/test services to the builder here
        builder.ConfigureServices(services =>
        {
            services.AddScoped(sp =>
            {
                // Replace SQLite with in-memory database for tests
                return new DbContextOptionsBuilder<CatalogContext>()
                .UseInMemoryDatabase("DbForPublicApi")
                .UseApplicationServiceProvider(sp)
                .Options;
            });
            services.AddScoped(sp =>
            {
                // Replace SQLite with in-memory database for tests
                return new DbContextOptionsBuilder<AppIdentityDbContext>()
                .UseInMemoryDatabase("IdentityDbForPublicApi")
                .UseApplicationServiceProvider(sp)
                .Options;
            });
        });
```

Environment config.

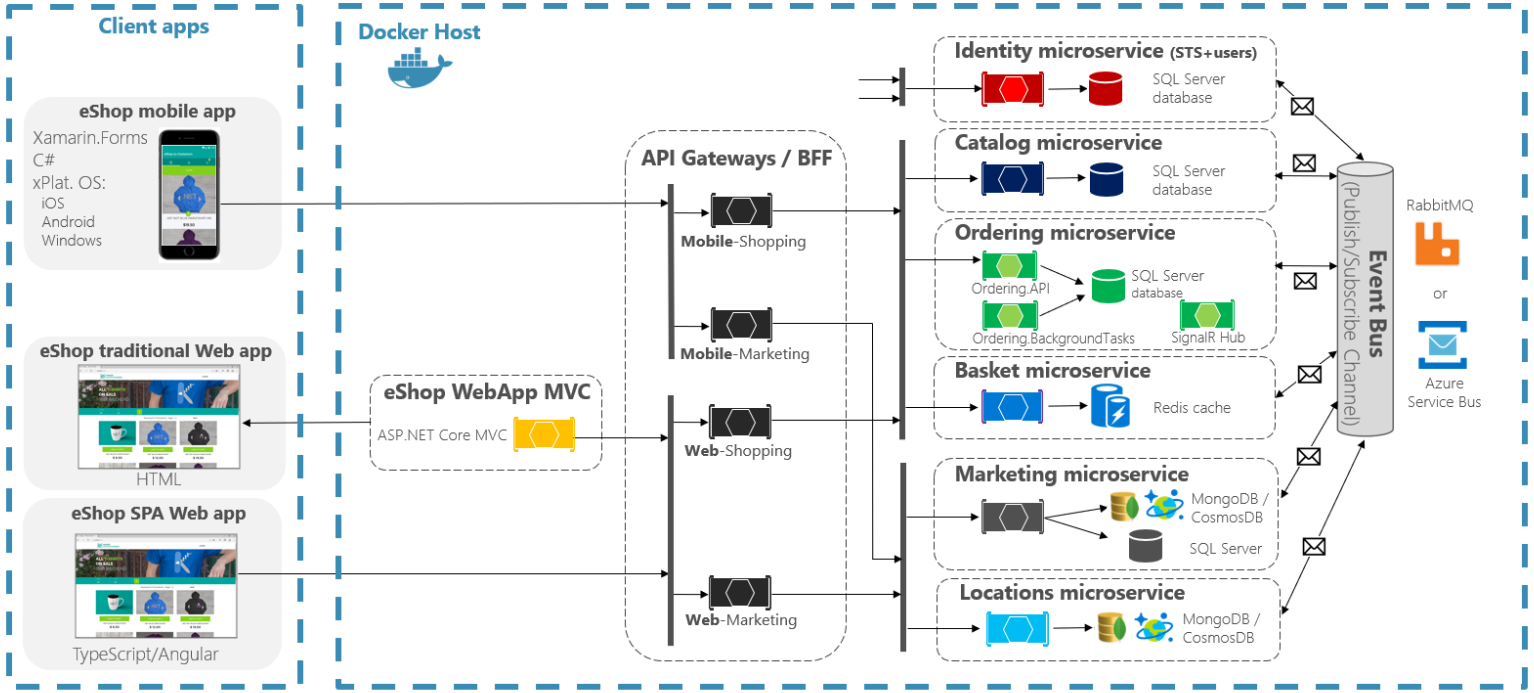
Unit, Integration & Functional Tests

#somoshiberus

https://github.com/dotnet-architecture/eShopOnContainers

eShopOnContainers reference application

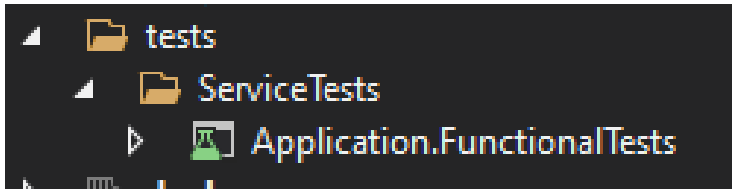
(Development environment architecture)

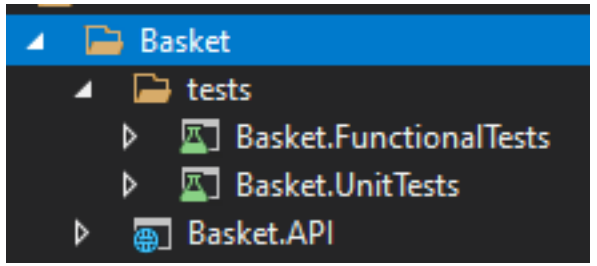


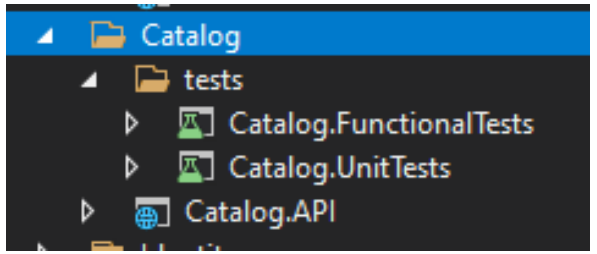
Environment config.

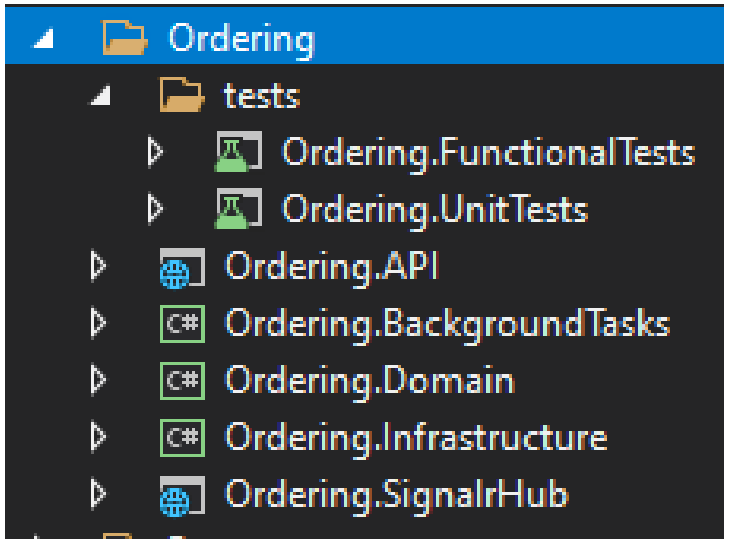
Unit, Integration & Functional Tests

https://github.com/dotnet-architecture/eShopOnContainers



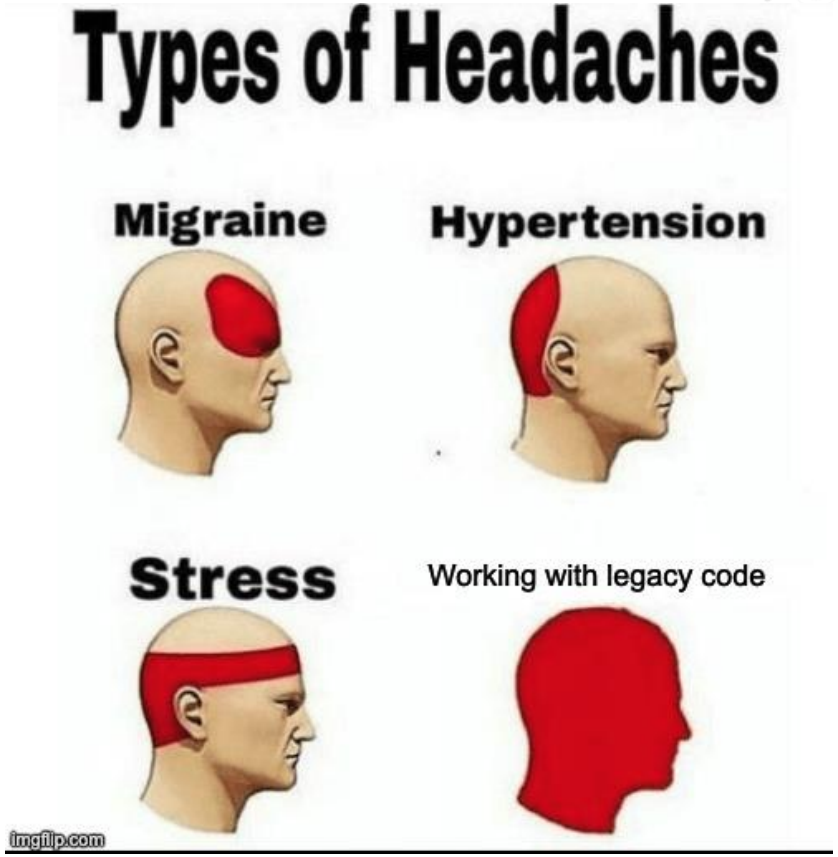






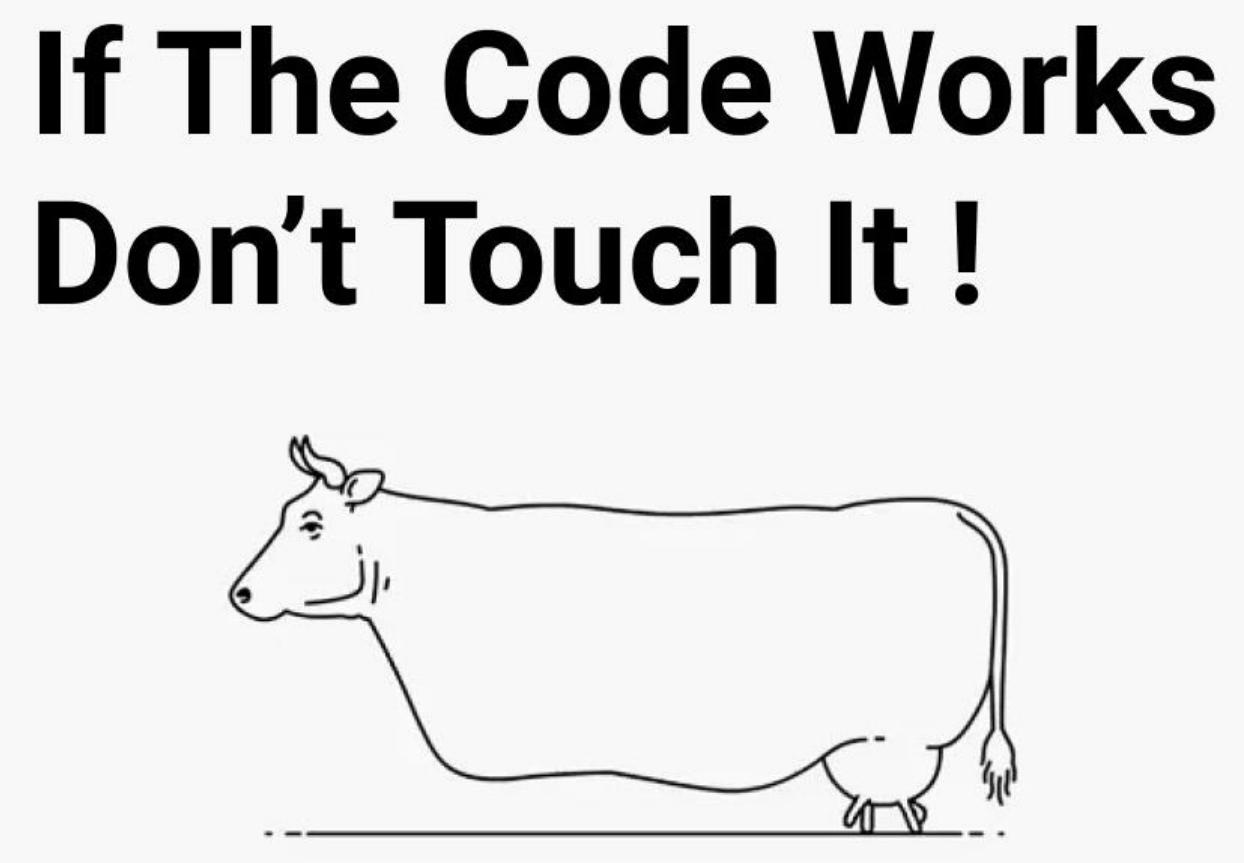
Pragmatic testing

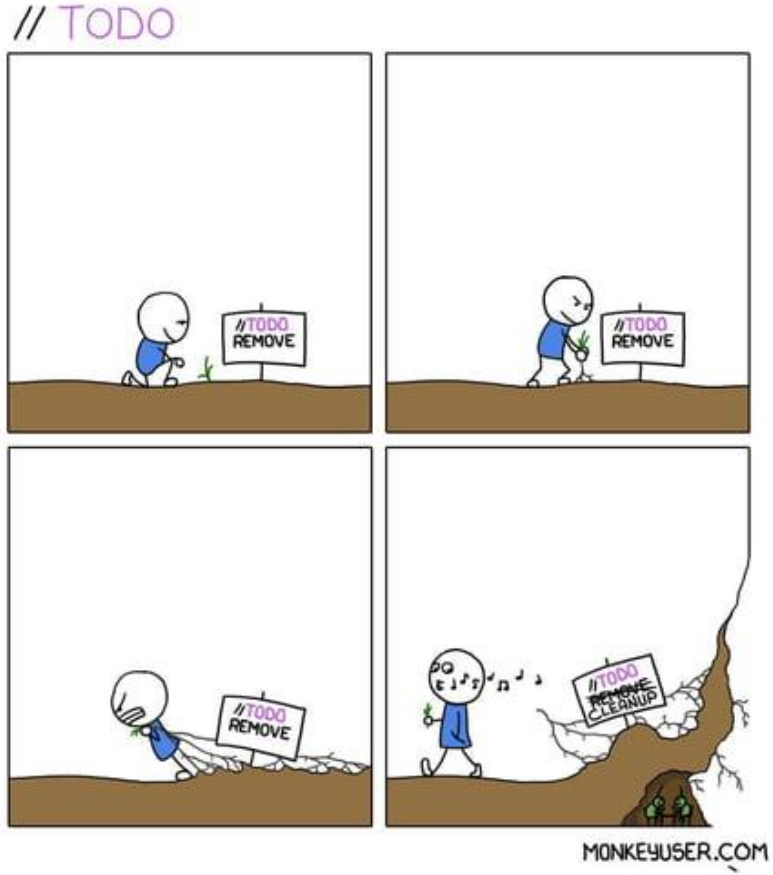
Legacy code



Pragmatic testing

Where to start?





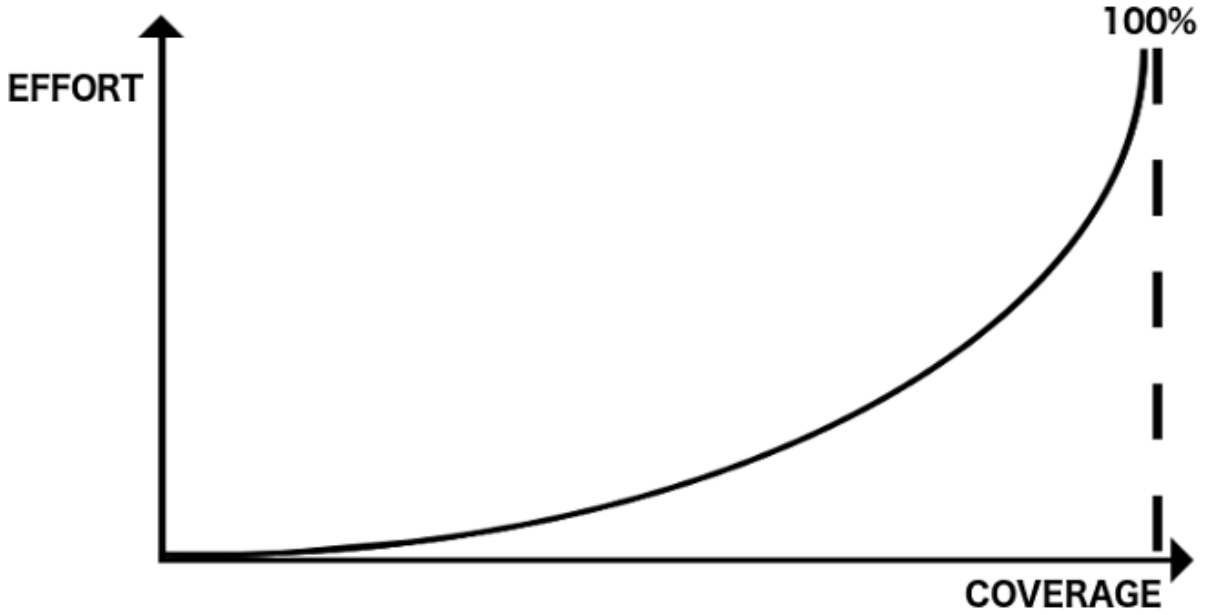
Pragmatic testing

Where to start?

#somoshiberus

Symbol	Coverage (%)	Uncover...
Total	100%	0/50
DotCover	100%	0/50
(1.0.0.0, net5.0)	100%	0/25
DotCover	100%	0/25
Calculator	100%	0/4
Tests	100%	0/21
(1.0.0.0, .NETCoreApp,Versi	100%	0/25
DotCover	100%	0/25
Calculator	100%	0/4
Tests	100%	0/21

ution | IL Viewer | Unit Tests Coverage | Datab



Pragmatic testing

Jimmy Bogard – "Holistic Testing"

Chasing coverage

"Chasing code coverage is a really bad idea.
If code doesn't change, then there's not a real reason to start adding test to it. The reason why I need to add it is because there is a bug there"

"I can have 100% code coverage and no one uses my product, or I can have 0% code coverage and it's a giant success, so there's no correlation between those two things"

"We're not paid to write test, we're paid to ship;
customers don't care about our coverage numbers"

Pragmatic testing

How to start?

#somoshiberus

REAL SYSTEM

Green = class in focus
Yellow = dependencies
Grey = other unrelated classes

CLASS IN UNIT TEST

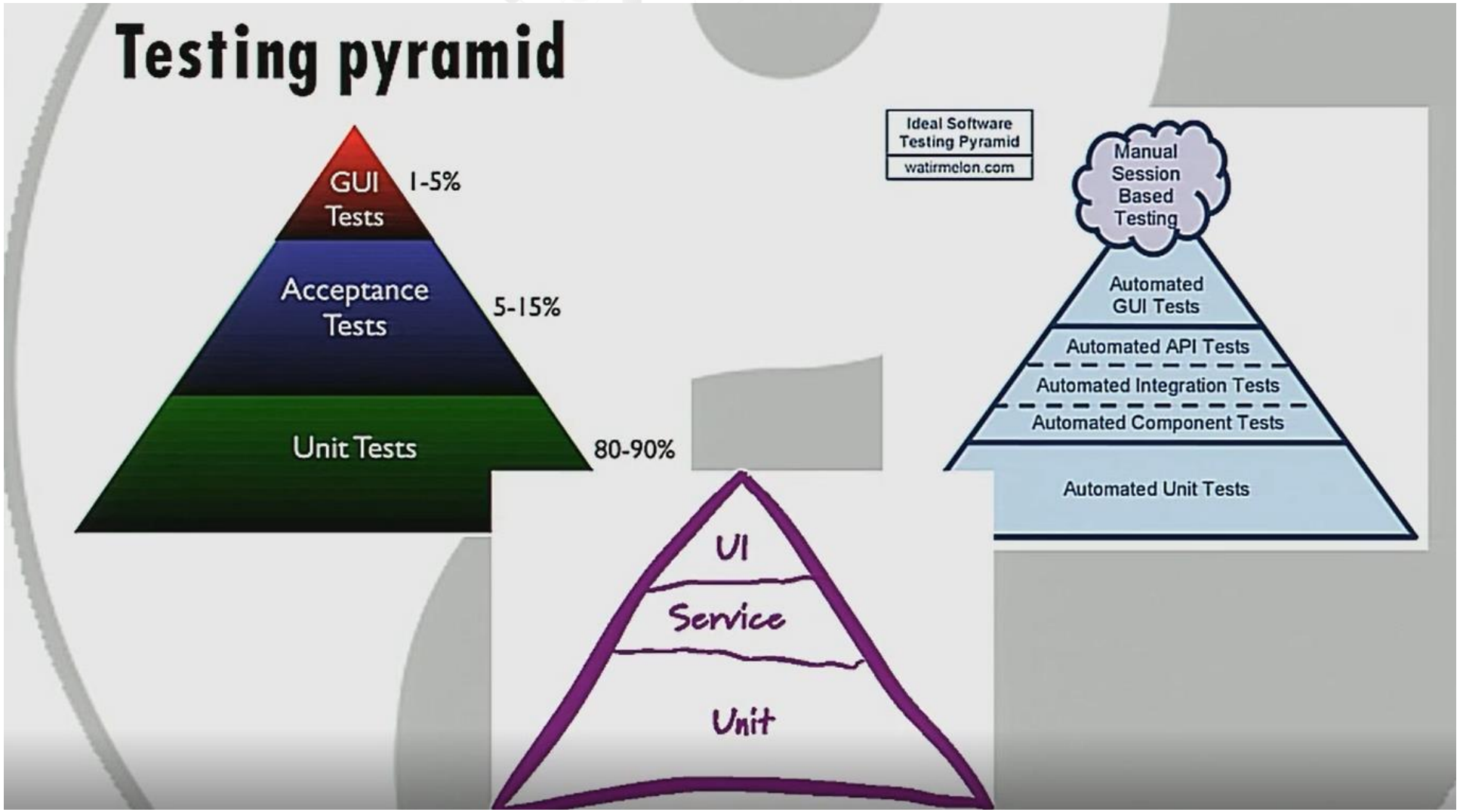
Green = class in focus
Yellow = mocks for the unit test

End-to-end Testing

Pragmatic testing

Jimmy Bogard – "Holistic Testing"

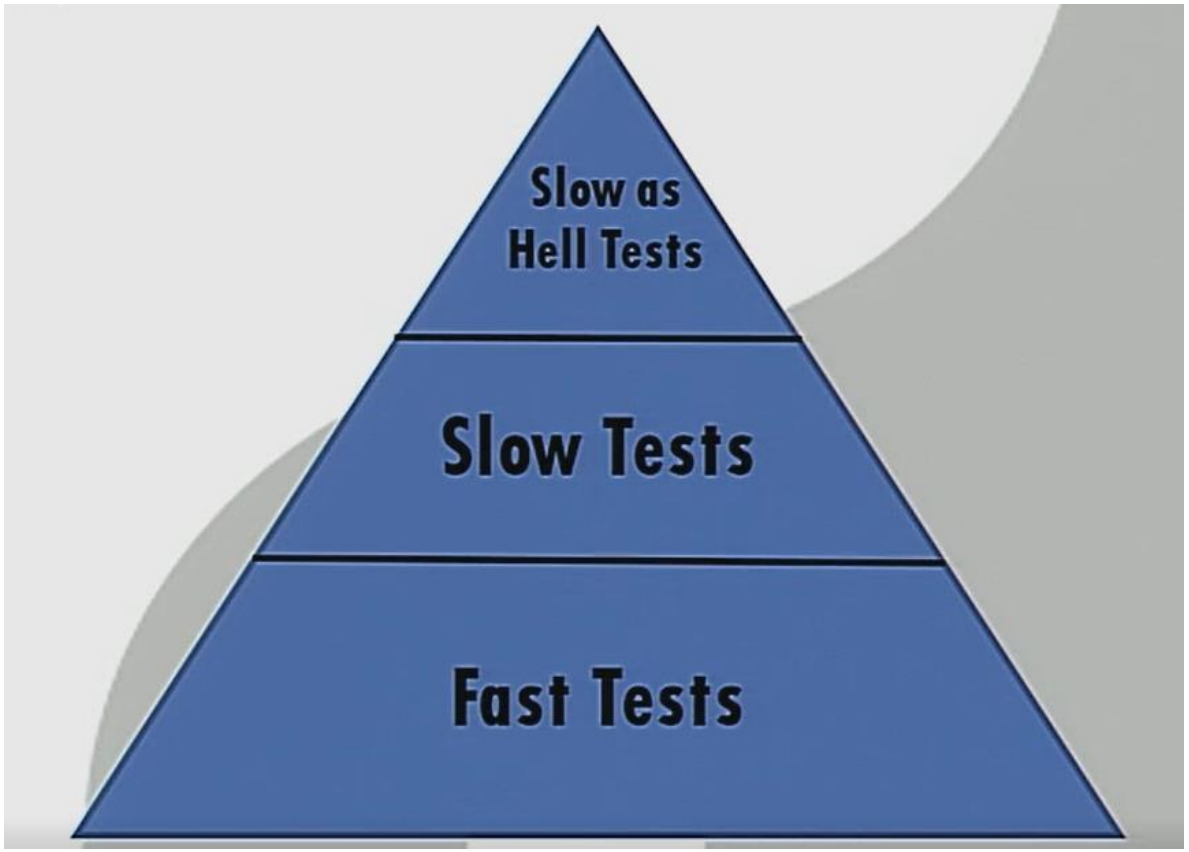
#somoshiberus



Pragmatic testing

Jimmy Bogard – "Holistic Testing"

Testing pyramid



Pragmatic testing

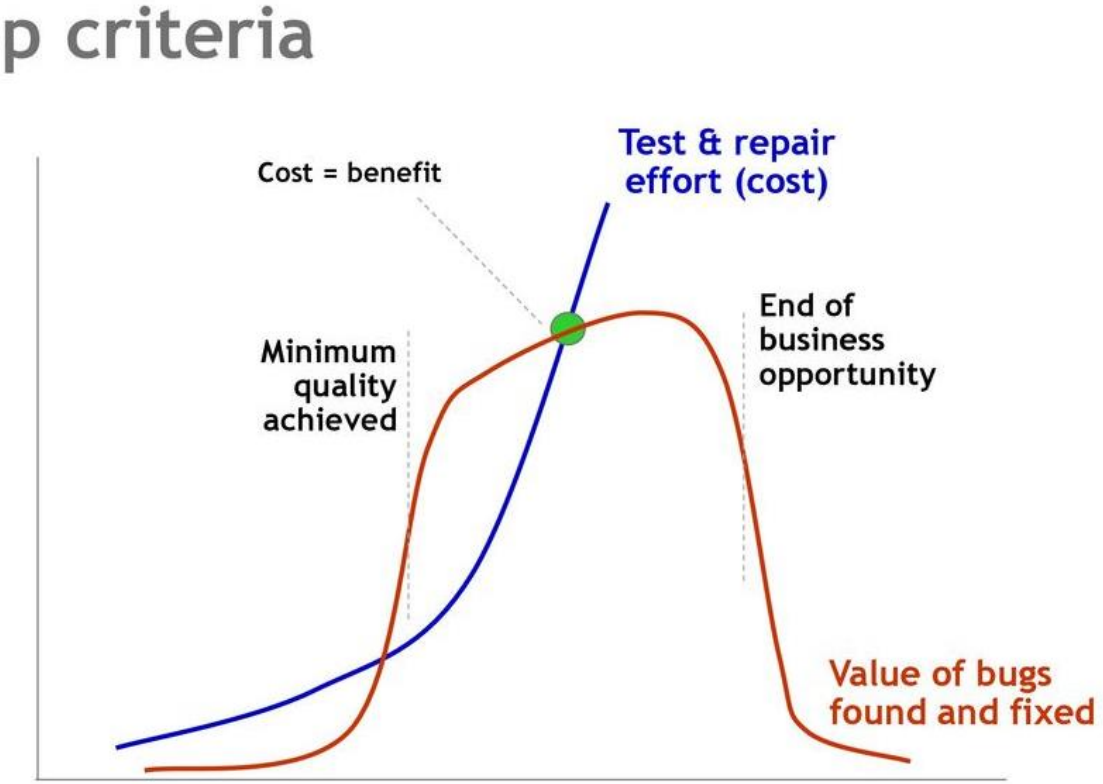
When to stop?

Testing Opportunity Cost =

Not developed Features -

Prevented Bugs

(discovered with tests)



Pragmatic testing

Principles

1. *Testing shows the presence of defects*
2. *Exhaustive testing is impossible*
3. *Early testing*
4. *Defect clustering*
5. *Pesticide paradox*
6. *Testing is context dependant*
7. *Absence-of-errors fallacy*

Pragmatic testing

Jimmy Bogard – "Holistic Testing"

Why do we test

"The ultimate goal here is to ship code, it's not to write tests; tests are just a means to the end of shipping code"

https://vimeo.com/68390508

hiberus© TECNOLOGIA

La compañía hiperespecializada en las TIC

www.hiberus.com