
Capacitación .NET

Sesión 01

AVANTICA
an encora company



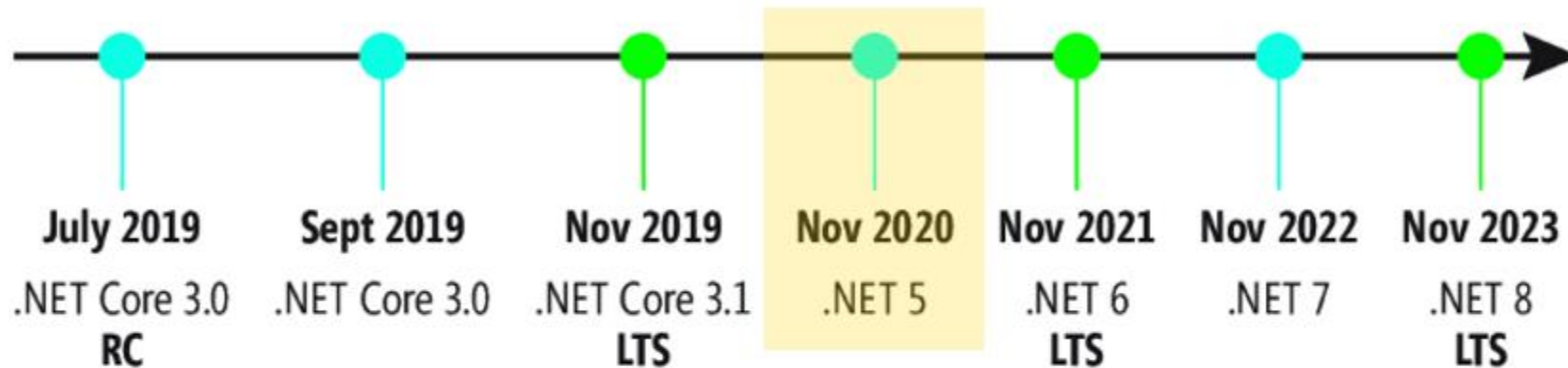
Temas

1. News on .NET 5
2. Clean Architecture
3. TDD y BDD
4. Entity Framework Core

Expone: Juan Alberto Carlos Vera

News en .NET 5

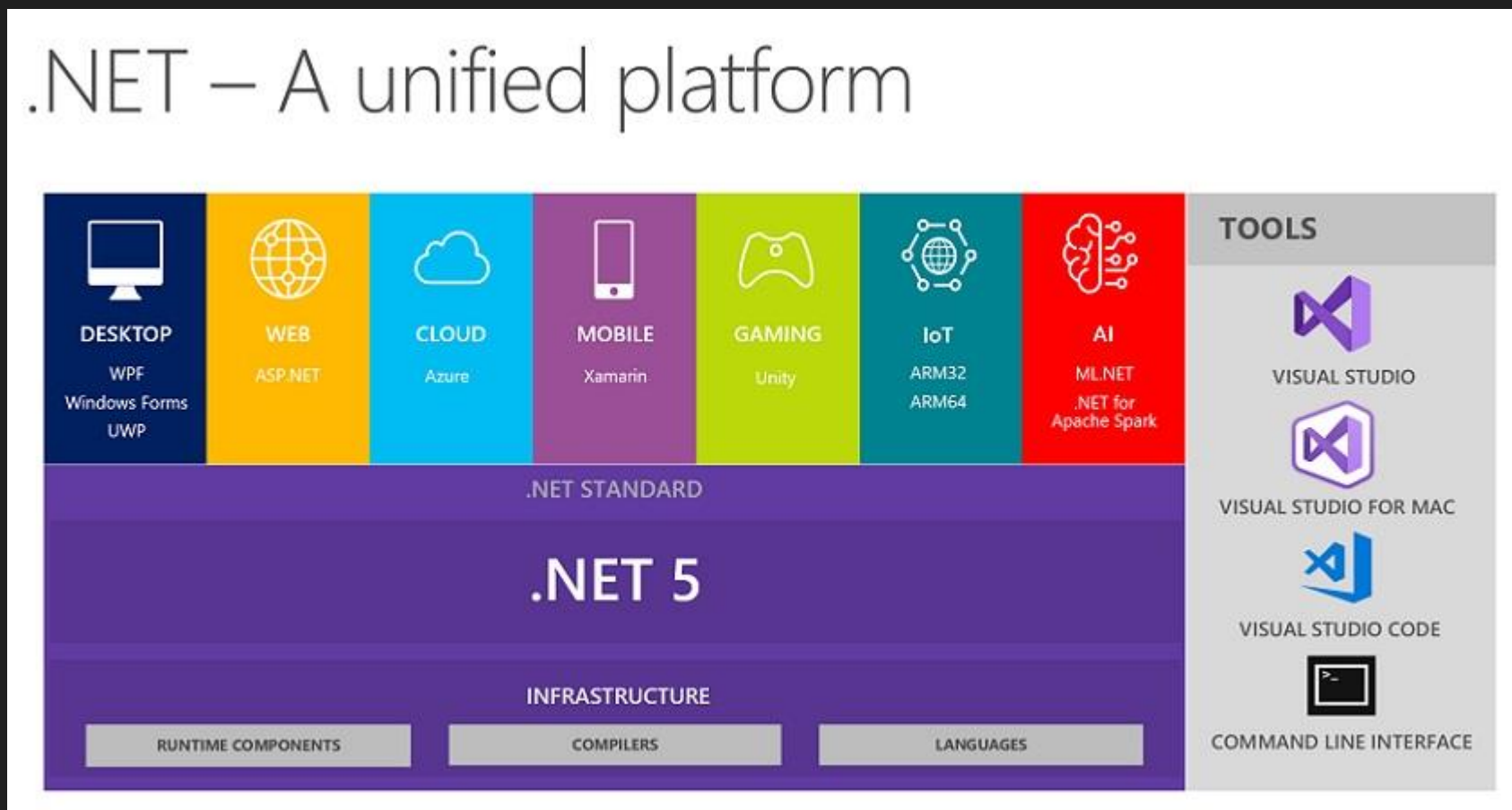
Microsoft ha decidido nombrar a la siguiente versión NET 5 en vez de NET Core 5 (saltándose la 4) esto es para, así, dejar .NET Framework como legacy y asumir que todo el mundo continúa en .NET 5.



- .NET Core 3.0 release in September
- .NET Core 3.1 = Long Term Support (LTS)
- .NET 5 release in November 2020
- Major release every year, LTS for even-numbered releases
- Predictable schedule, minor releases if needed

News en .NET 5

Una plataforma unificada.



News en .NET 5

1. Rendimiento

Mejoras de rendimiento en general para el lenguaje, e hicieron mucho detalle en las mejoras de la arquitectura arm, el serializador de json, las expresiones regulares y HTTP.

2. ClickOnce

Han devuelto ClickOnce para publicar aplicaciones de cliente, agregando mejoras en la experiencia de usuario.

3. Blazor

Cuando cambias código en visual studio, blazor se recarga automáticamente con los cambios, con lo que no debes parar el proyecto y volverlo a arrancar.

News en .NET 5

4. Xamarin.Forms

Al realizar un cambio en el código y lo podemos ver en la app móvil al instante.

5. gRPC

Es un framework de RPC (llamada a procedimiento remoto).

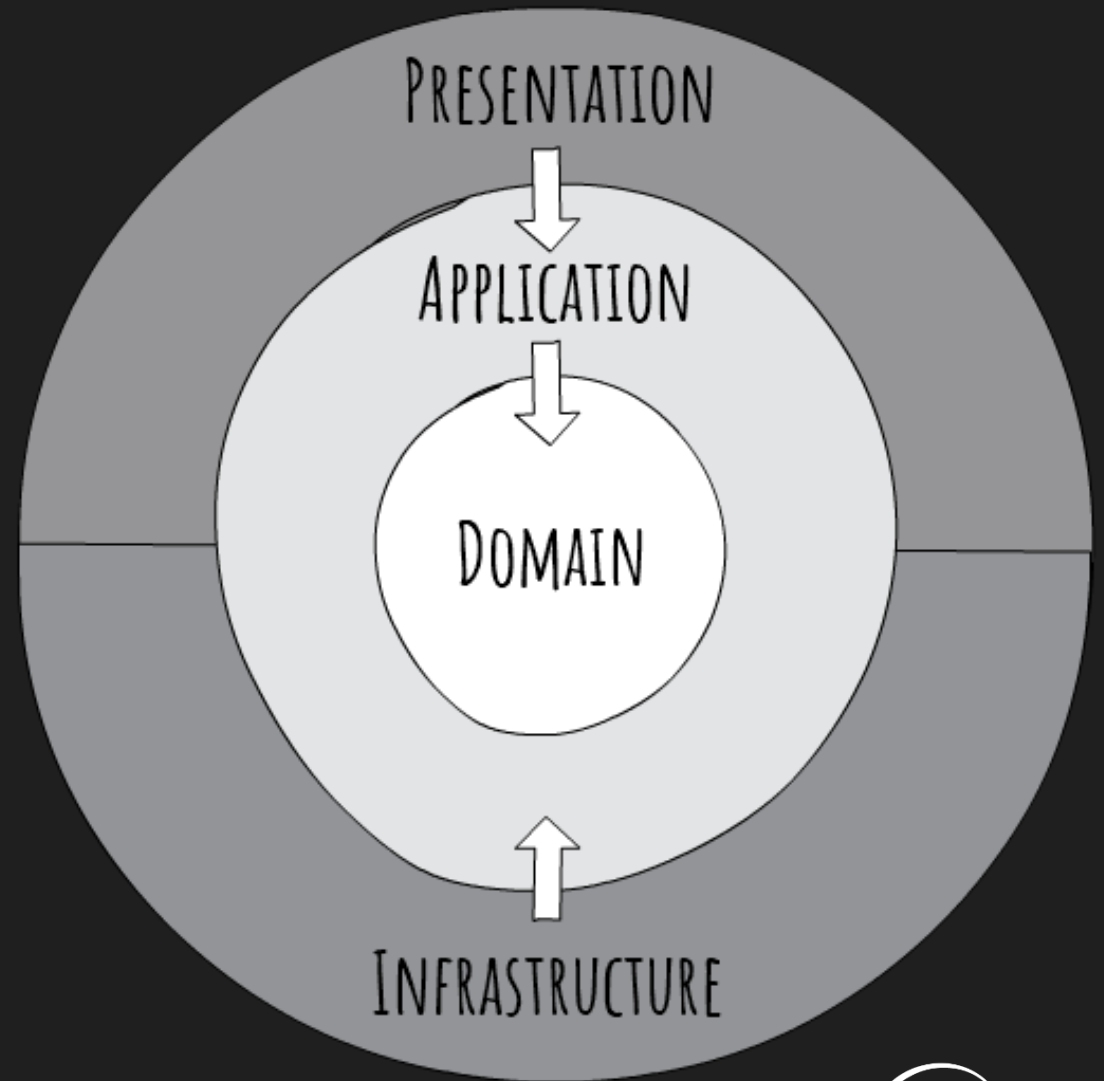
6. Project Tye

Es una herramienta que permite desarrollar, testear y desplegar microservicios y sistemas distribuidos de una forma mucho más sencilla.

Incluye orquestación de los microservicios y además permite desplegarlos en Kubernetes con muy poca configuración adicional.

Clean Architecture

Es una forma de diseñar y construir software propuesto por primera vez por Robert C. Martin (el tío Bob) en su libro del mismo nombre.



Clean Architecture

Capas

Domain

Contiene lógica de dominio, incluye entidades, enumeraciones, excepciones, interfaces, tipos y lógica específica de la capa de dominio.

Dependencia: No depende de nada externo.

Application

Contiene toda la lógica empresarial. Esta capa define interfaces que son implementadas por capas externas. Por ejemplo, si la aplicación necesita acceder a un servicio de notificación, se agregará una nueva interfaz a la aplicación y se creará la implementación dentro de la infraestructura.

Dependencia: Domain.

Clean Architecture

Infrastructure

Contiene clases para acceder a recursos externos como sistemas de archivos, servicios web, SMTP, etc. Estas clases deben basarse en interfaces definidas dentro de la capa de aplicación.

Presentation

Contiene las interfaces de usuario con las cuales interactúa. Tenga en cuenta que la dependencia de la infraestructura es solo para admitir la inyección de dependencia.

Dependencia: *Application, Infrastructure.*

Clean Architecture

Ventajas

Independiente de los frameworks

No requiere la existencia de alguna herramienta o framework.

Testable

Fácil de probar, no depende de nada externo, por lo que escribir pruebas automatizadas es mucho más fácil.

Clean Architecture

Independiente de la UI

La lógica se mantiene fuera de la UI, por lo que es fácil cambiar a otra tecnología; ahora mismo, es posible que esté usando Angular, pronto Vue, ¡eventualmente Blazor!

Independiente de la base de datos

Las preocupaciones sobre el acceso a los datos están claramente separadas, por lo que pasar de SQL Server a CosmosDB u otra es trivial.

Independiente de cualquier cosa externa

El core está completamente aislado del mundo exterior, esa es la diferencia entre un sistema que durará 3 años y uno que durará 20 años.

TDD y BDD

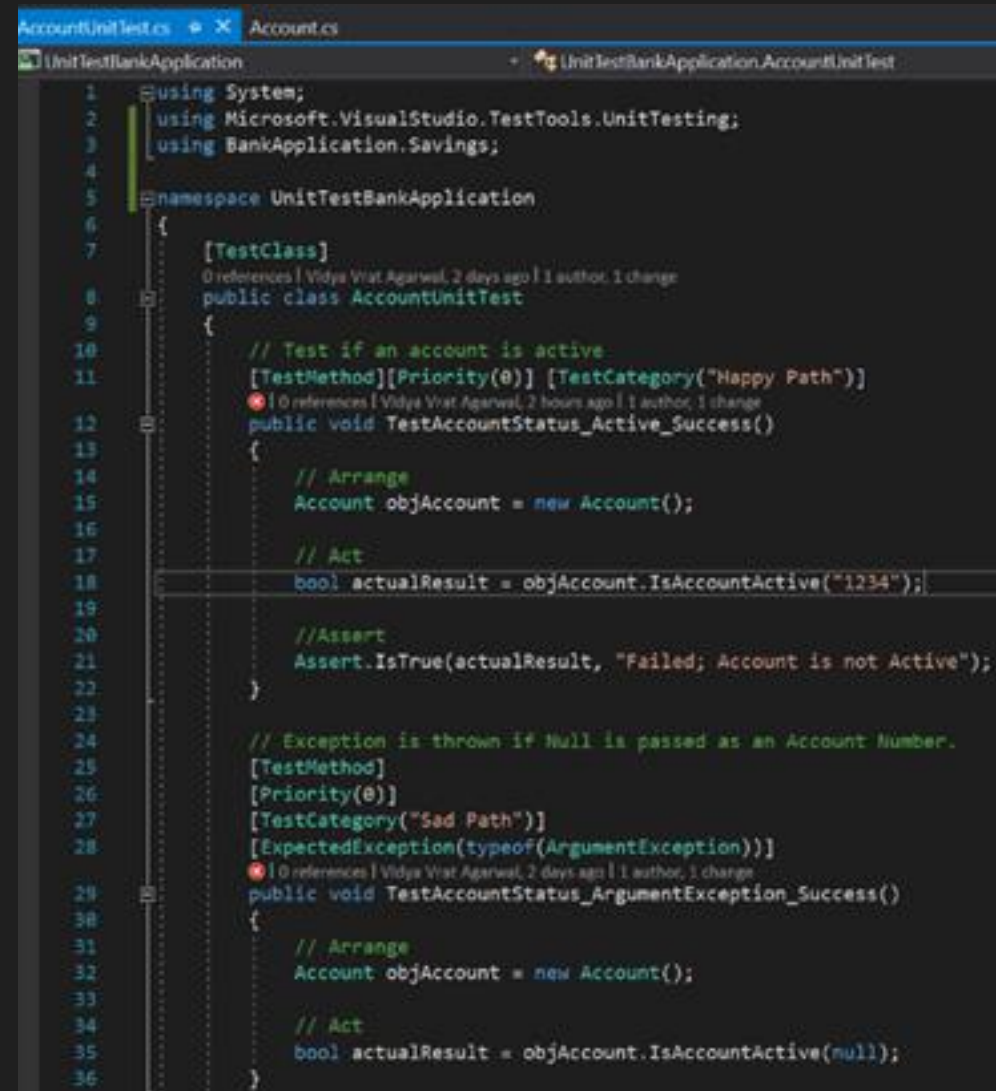
TDD

Test-driven development (Desarrollo guiado por pruebas)

Es un enfoque de desarrollo de software en que primero escribes los tests **(test-first)** y después utilizas estos tests para diseñar tu aplicación y obviamente desarrollarla.

Estructura de los tests:

- Configuración (arrange)
- Ejecución (act)
- Validación (assert)



```
1 using System;
2 using Microsoft.VisualStudio.TestTools.UnitTesting;
3 using BankApplication.Savings;
4
5 namespace UnitTestBankApplication
6 {
7     [TestClass]
8     public class AccountUnitTest
9     {
10         // Test if an account is active.
11         [TestMethod][Priority(0)] [TestCategory("Happy Path")]
12         public void TestAccountStatus_Active_Success()
13         {
14             // Arrange
15             Account objAccount = new Account();
16
17             // Act
18             bool actualResult = objAccount.IsAccountActive("1234");
19
20             //Assert
21             Assert.IsTrue(actualResult, "Failed; Account is not Active");
22         }
23
24         // Exception is thrown if Null is passed as an Account Number.
25         [TestMethod]
26         [Priority(0)]
27         [TestCategory("Sad Path")]
28         [ExpectedException(typeof(ArgumentException))]
29         public void TestAccountStatus_ArgumentException_Success()
30         {
31             // Arrange
32             Account objAccount = new Account();
33
34             // Act
35             bool actualResult = objAccount.IsAccountActive(null);
36         }
37     }
38 }
```

TDD y BDD

TDD

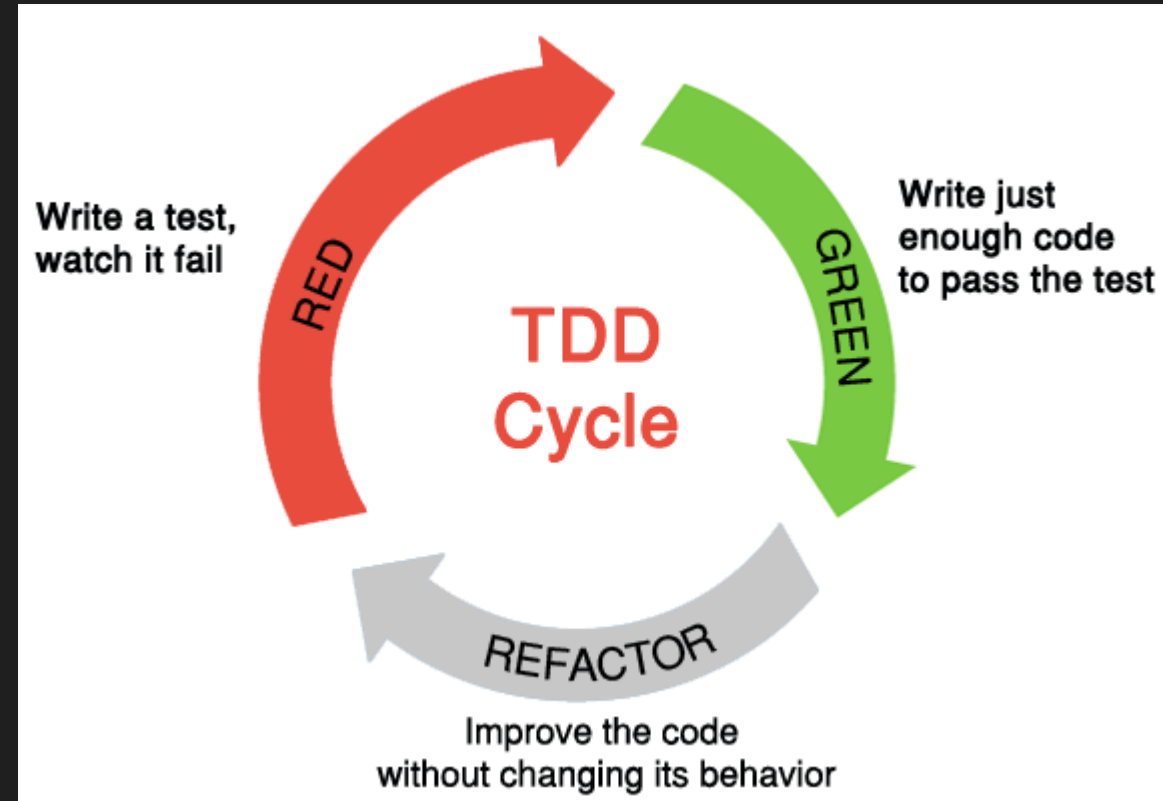
Test-driven development
(Desarrollo guiado por pruebas)

Enfoque red-green-refactor:

Red: Ejecutar los test y comprobar que fallan.

Green: Escribir el código y pasar la prueba.

Refactor: Mejorar y optimizar el código.

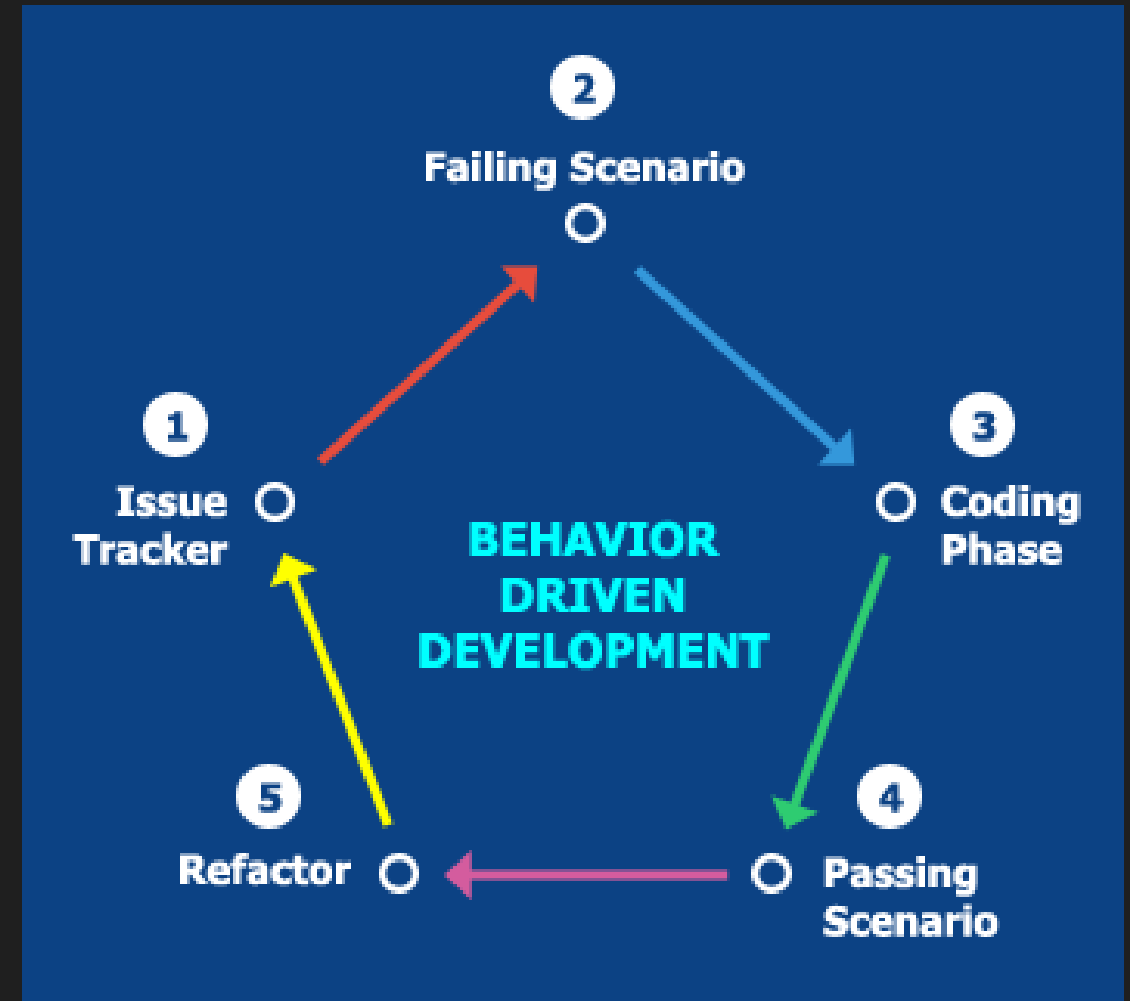


TDD y BDD

BDD

Behavior-driven development
(Desarrollo guiado por el
comportamiento)

Es una estrategia de desarrollo dirigido por comportamiento, se define en un idioma común entre todos los stakeholders, lo que mejora la comunicación entre equipos tecnológicos y no técnicos.



TDD y BDD

BDD

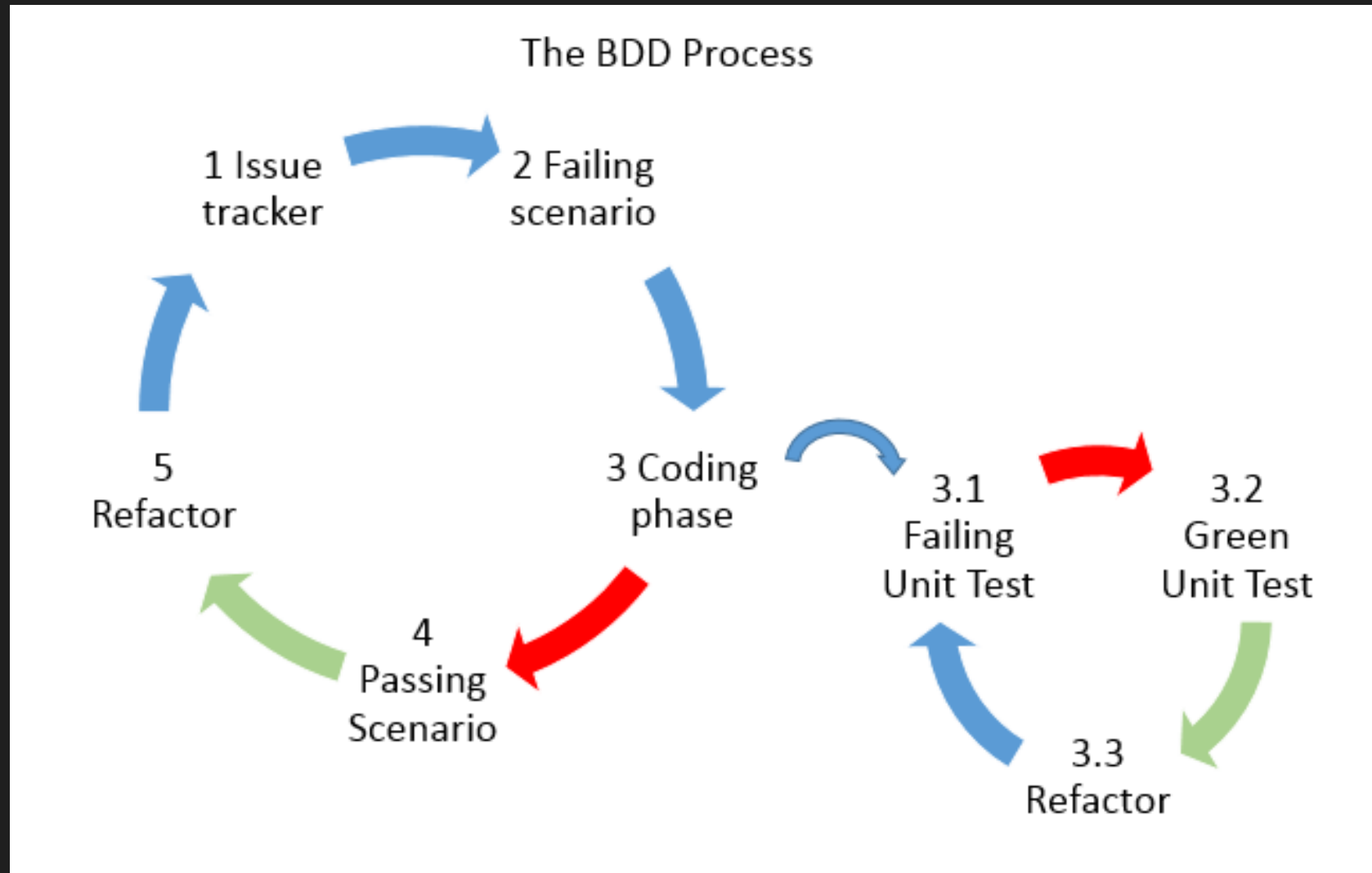
Behavior-driven development (Desarrollo guiado por el comportamiento)

Ciclo:

- Escribir historias de usuario.
- Escribir los escenarios.
- Automatizar pruebas de aceptación.

```
CheckLocationDataResponse.feature  ➦ ✕  
1  Feature: Returning location data based o  
2      As a consumer of the Zippopotam.us A  
3      I want to receive location data matc  
4      So I can use this data to auto-compl  
5  
6  Scenario: An existing country and zi  
7      Given the country code us and zi  
8      When I request the locations cor  
9      Then the response contains the p  
10  
11 Scenario: An existing country and zi  
12     Given the country code us and zi  
13     When I request the locations cor  
14     Then the response contains exact  
15  
16 Scenario: An existing country and zi  
17     Given the country code us and zi  
18     When I request the locations cor  
19     Then the response has status cod
```


TDD y BDD



Entity Framework Core

EF (Entity Framework)

Es el ORM oficial de Microsoft.


Permite acceder a una base de datos utilizando clases que representan cada una de las entidades/tablas de ésta, pudiendo realizar cualquier operación sobre los datos simplemente llamando a métodos de estas clases.

Entity Framework Core

EF 6	EF Core
✓ First released in 2008 with .NET Framework 3.5 SP1	✓ First released in June 2016 with .NET Core 1.0
✓ Stable and feature rich	✓ New and evolving
✓ Windows only	✓ Windows, Linux, OSX
✓ Works on .NET Framework 3.5+	✓ Works on .NET Framework 4.5+ and .NET Core
✓ Open-source	✓ Open-source

Demo





¡Iniciemos un proyecto juntos!



Contáctenos:

info@avantica.com

www.avantica.com

USA: +1 (650) 641 3134

Costa Rica: +506 4040 0700

Perú: +501 616 7676

Bolivia: +591 4 4067250

Colombia: +57 (2) 321 7000

