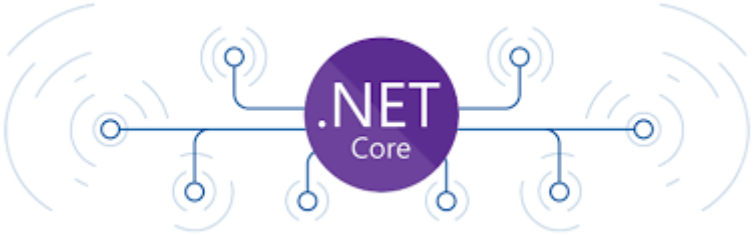
2021/2022(2)
IF184605 Framework-Based Programming

Lecture #4

.NET: Desktop, Web & Mobile App

Misbakhul Munir IRFAN SUBAKTI
司馬伊凡
Мисбакхул Мунир Ирфан Субакти

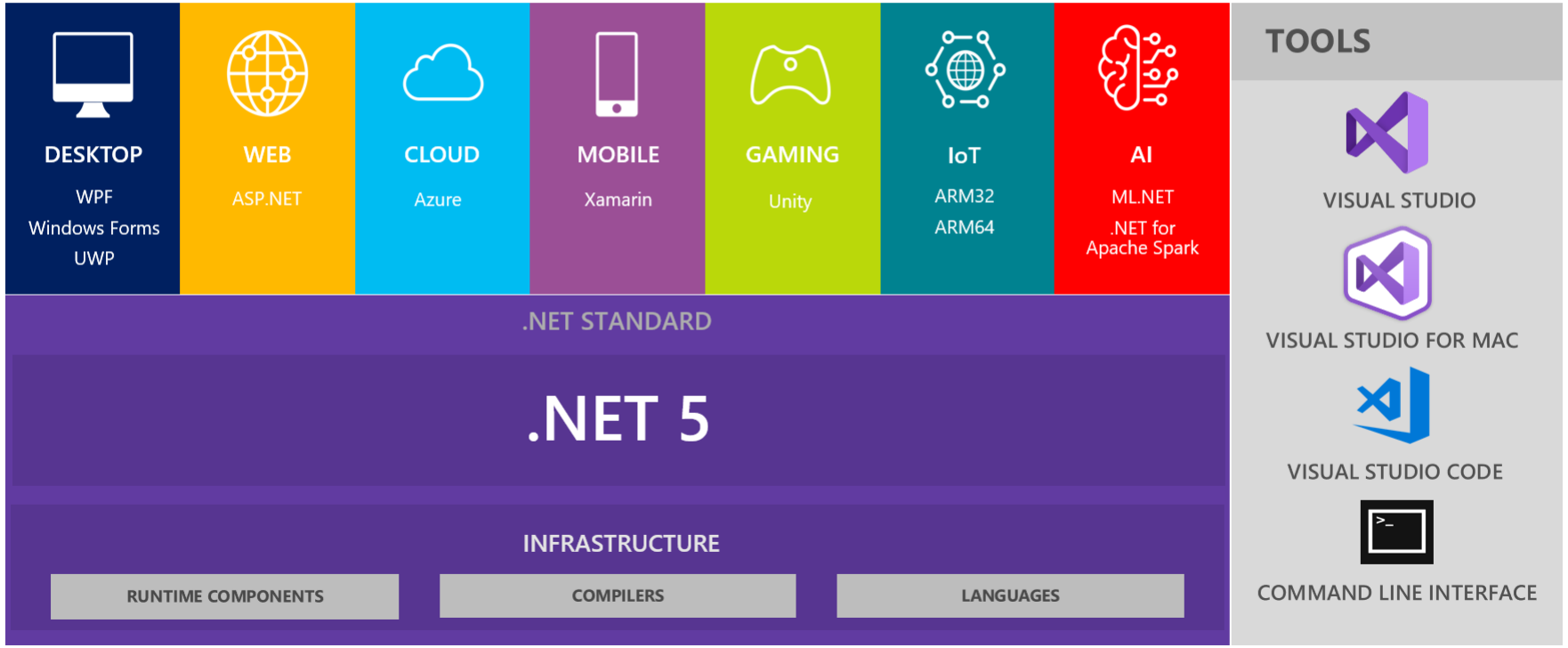


Desktop app

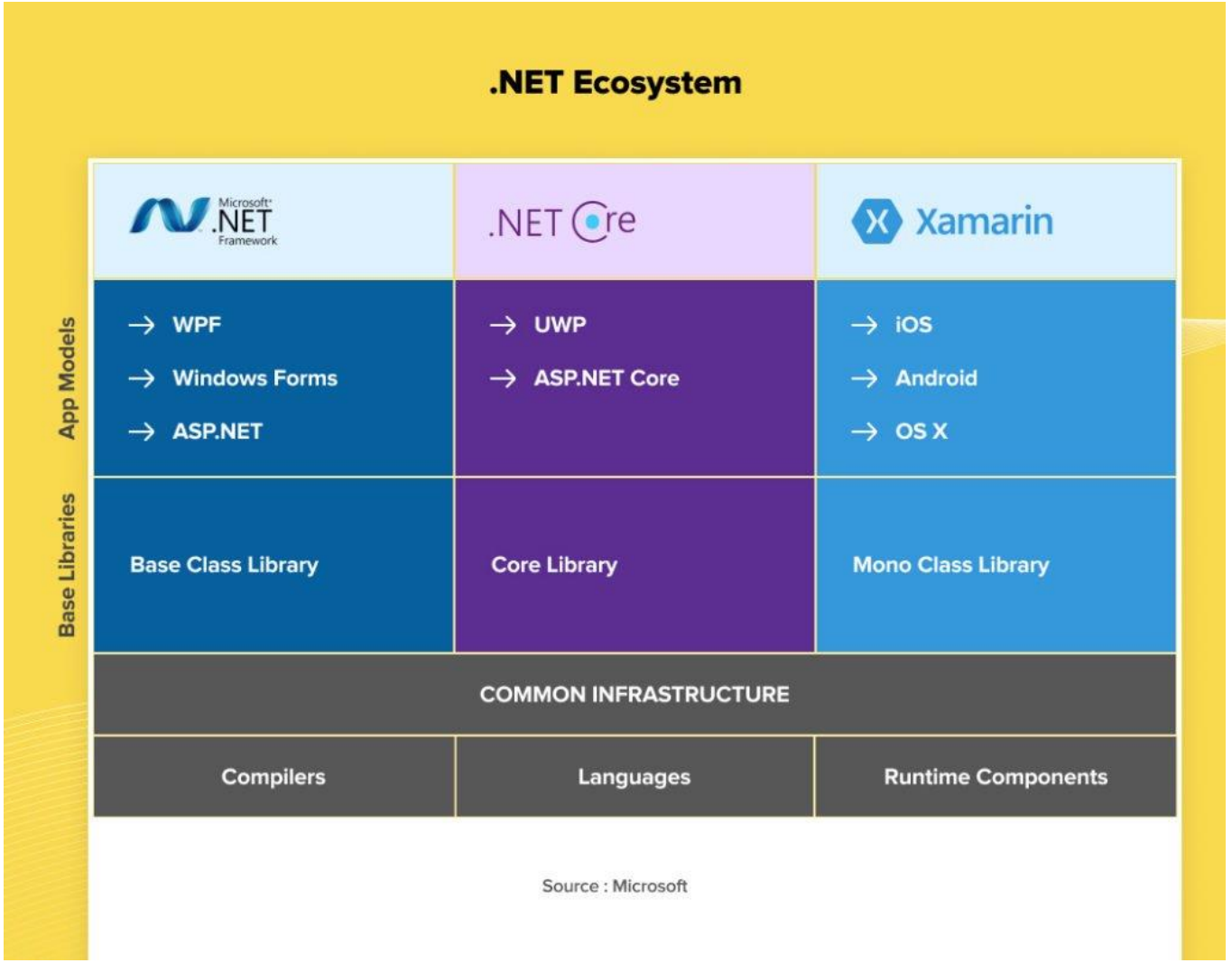
- .NET Core is one of Microsoft's efforts to embrace more developers by developing .NET technology that is open source and multi-platform OS (Operating System).
  - It means .NET Core not only can run on Windows but can also run on Linux and macOS operating systems.
- .NET technology is very potential to be portable, only in its development, it has grown very big.
  - It also has become too much fragmentation and only specifically running on Windows called .NET Framework.
- Along with the development of technology and the market, the need for deployment of applications to multi-device and multi-processor architectures will be increasingly needed.
  - At around 2014, Microsoft then took the essential features or the only core from the .NET Framework and then came up with .NET Core.

Building the apps

.NET – A unified platform



.NET: Multi-platform OS



Tutorial

- C# Full Project Tutorial (Supermarket Management System): link in here.
  - In this project, you will create a C# complete application of Supermarket management.
  - The application starts with a nice looking splash form which leads to a login form.
  - The login form will be connected to a database, there are two roles in the application, Admin and Seller.
  - If the *userID* and *password* match the Seller table entries, the user will log in to the Selling form.
  - From the Selling form, he can sell different products and print Bills.
  - If the user logs in as Admin, the user can go to the Product form.
  - He can add, edit, delete and view Product stock.
  - The Admin can manage the different Sellers.
  - He can add, edit or update the Sellers.
  - The Admin can manage the different Categories to which belong the different products.
  - The Admin can list all the sales of a given day.

Reference

- Windows Forms (WinForms): link in here.
  - Windows Forms (WinForms) is a UI framework for building Windows desktop applications. It is a .NET wrapper over Windows user interface libraries, such as User32 and GDI+. It also offers controls and other functionality that is unique to Windows Forms.
  - Windows Forms also provides one of the most productive ways to create desktop applications based on the visual designer provided in Visual Studio. It enables drag-and-drop of visual controls and other similar functions that make it easy to build desktop applications.

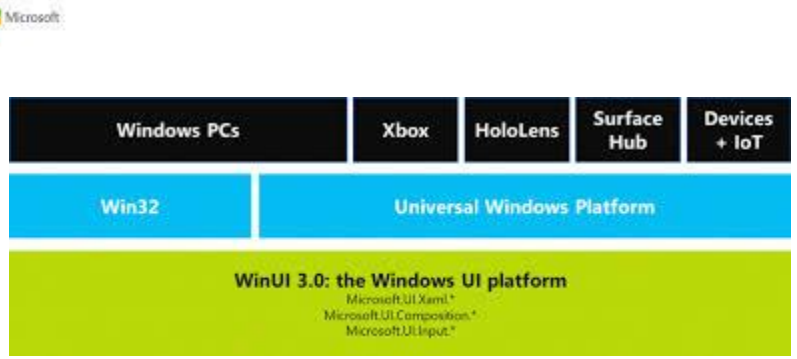
Reference (continued)

- Windows Presentation Foundation (WPF): link in here.
  - Windows Presentation Foundation (WPF) is a UI framework for building Windows desktop applications. WPF supports a broad set of application development features, including an application model, resources, controls, graphics, layout, data binding and documents. WPF uses the Extensible Application Markup Language (XAML) to provide a declarative model for application programming.
  - WPF applications are based on a vector graphics architecture. This enables applications to look great on high DPI monitors, as they can be infinitely scaled. WPF also includes a flexible hosting model, which makes it straightforward to host a video on a button, for example. The visual designer provided in Visual Studio makes it easy to build a WPF application, with drag-in-drop and/or direct editing of XAML markup.
  - As of .NET 6.0, WPF supports ARM64.
  - See the WPF Roadmap to learn about project priorities, status and ship dates.
  - WinForms is another UI framework for building Windows desktop applications that are supported on .NET (6.0.x/5.0.x/3.1.x). WPF and WinForms applications only run on Windows. They are part of the Microsoft.NET.Sdk.WindowsDesktop SDK. You are recommended to use the most recent version of Visual Studio to develop WPF and WinForms applications for .NET.
  - To build the WPF repo and contribute features and fixes for .NET 7.0, Visual Studio 2022 Preview is required.

Reference (continued)

- Universal Windows Platform (UWP) app samples: link in here.
  - This repo contains the samples that demonstrate the API usage patterns for the Universal Windows Platform (UWP) in the Windows Software Development Kit (SDK) for Windows 10. These code samples were created with the Universal Windows Platform templates available in Visual Studio, and are designed to run on desktop, mobile, and future devices that support the Universal Windows Platform.
- UWP Experiences – app samples: link in here.
  - The UWP App Experiences are beautiful, cross-device, feature-rich and functional app samples built to demonstrate realistic app scenarios on the UWP platform across desktop, Xbox, mobile, and more. Besides being open source on GitHub, each sample is published to the Windows Store for easier access for developers and each is accompanied by at least one blog post and short overview video.

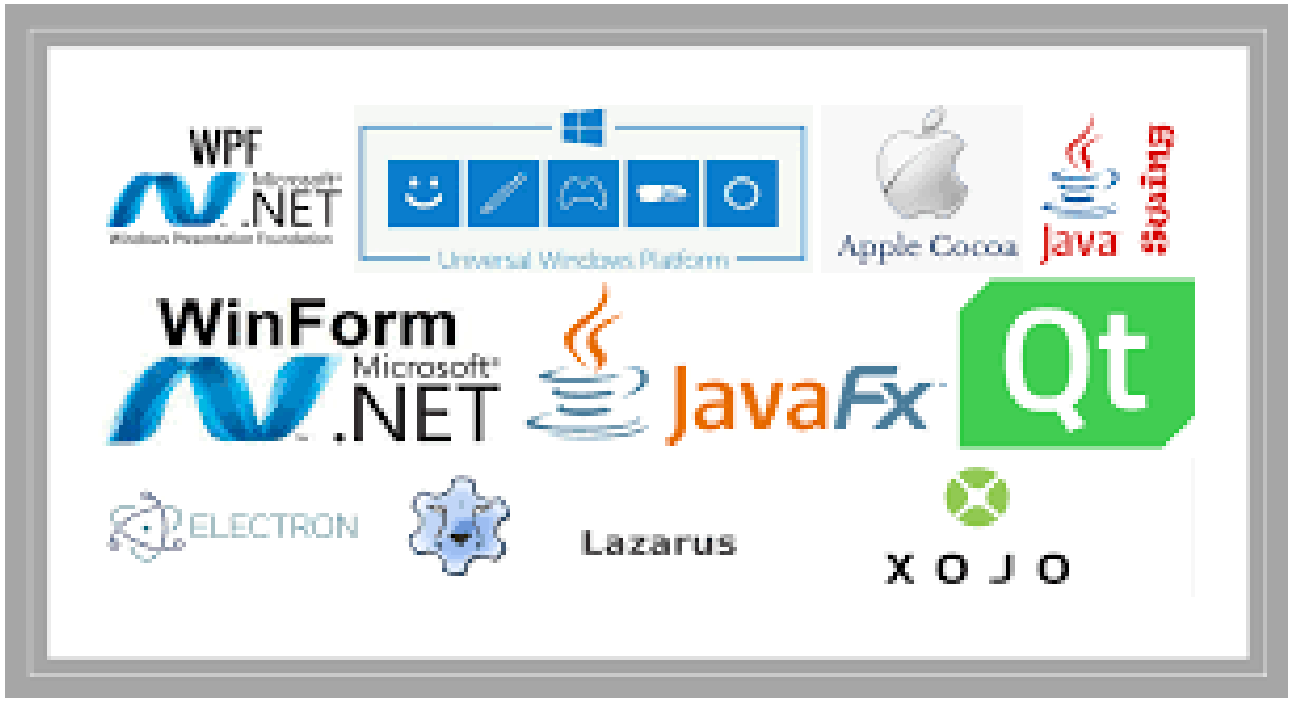
Reference (continued)

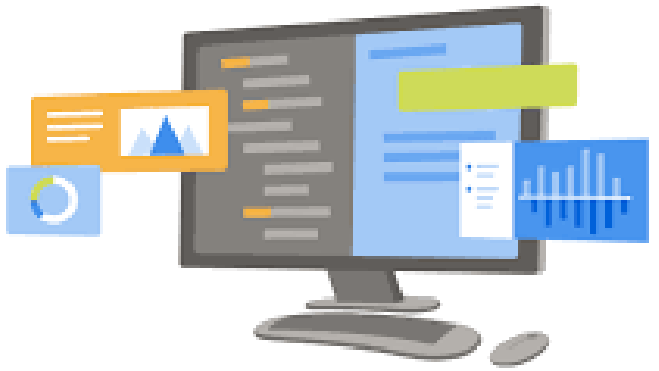


- Windows 10 UI XAML: link in here.
  - WinUI is a user interface layer that contains modern controls and styles for building Windows apps. As the native UI layer in Windows, it embodies Fluent Design, giving each Windows app the polished feel that customers expect.
  - WinUI 2 is a library of controls that provides official native Microsoft UI controls and features for Windows UWP apps. WinUI 2 can be used in any Windows 10 UWP XAML app, or a Xamarin.Forms app running on Windows 10 using native view embedding.
  - WinUI 3 is the next generation of the WinUI framework. It dramatically expands WinUI into a full UX framework, making WinUI available for all types of Windows apps – from Win32 to UWP – for use as the UI layer.

Reference (continued)

- Building .NET Desktop Applications with .NET Core and Windows 10: link in here.





Web app

- Intro to ASP.NET Core Razor Pages – From Start to Published: link in here.
  - ASP.NET Core is the web portion of .NET Core development.
  - Under this umbrella are three major components - Razor Pages, MVC, Blazor, and API.
  - All of these project types can co-exist on the same website without an issue, yet they all fill a different role.
  - In this video, we are going to look at Razor pages, the quickest and easiest way to create a powerful server-side web application in C#.
  - We will look at how to set a project up, how to capture and display data, and how to deploy it.

Reference

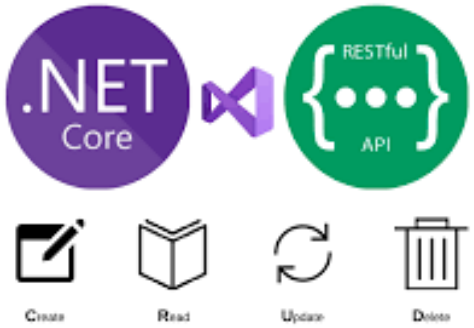
- ASP.NET Core for Beginners – Workshop: link in here.
  - This is a half-day workshop for developers who are completely new to .NET Core and ASP.NET.
  - Tutorial breakdown can be seen in the following.

Reference (continued)

VS Code Tutorial	Visual Studio Tutorial	Visual Studio for Mac Tutorial	Topics
Tutorial 1	VS Tutorial 1	VS for Mac Tutorial 1	Create a Razor Page using dotnet CLI and VS Code.
Tutorial 2	VS Tutorial 2	VS for Mac Tutorial 2	Add a model to an ASP.NET Core Razor Pages app
Tutorial 3	VS Tutorial 3	VS for Mac Tutorial 3	Update the generated pages
Tutorial 4	VS Tutorial 4	VS for Mac Tutorial 4	Adding search to a Razor Pages app

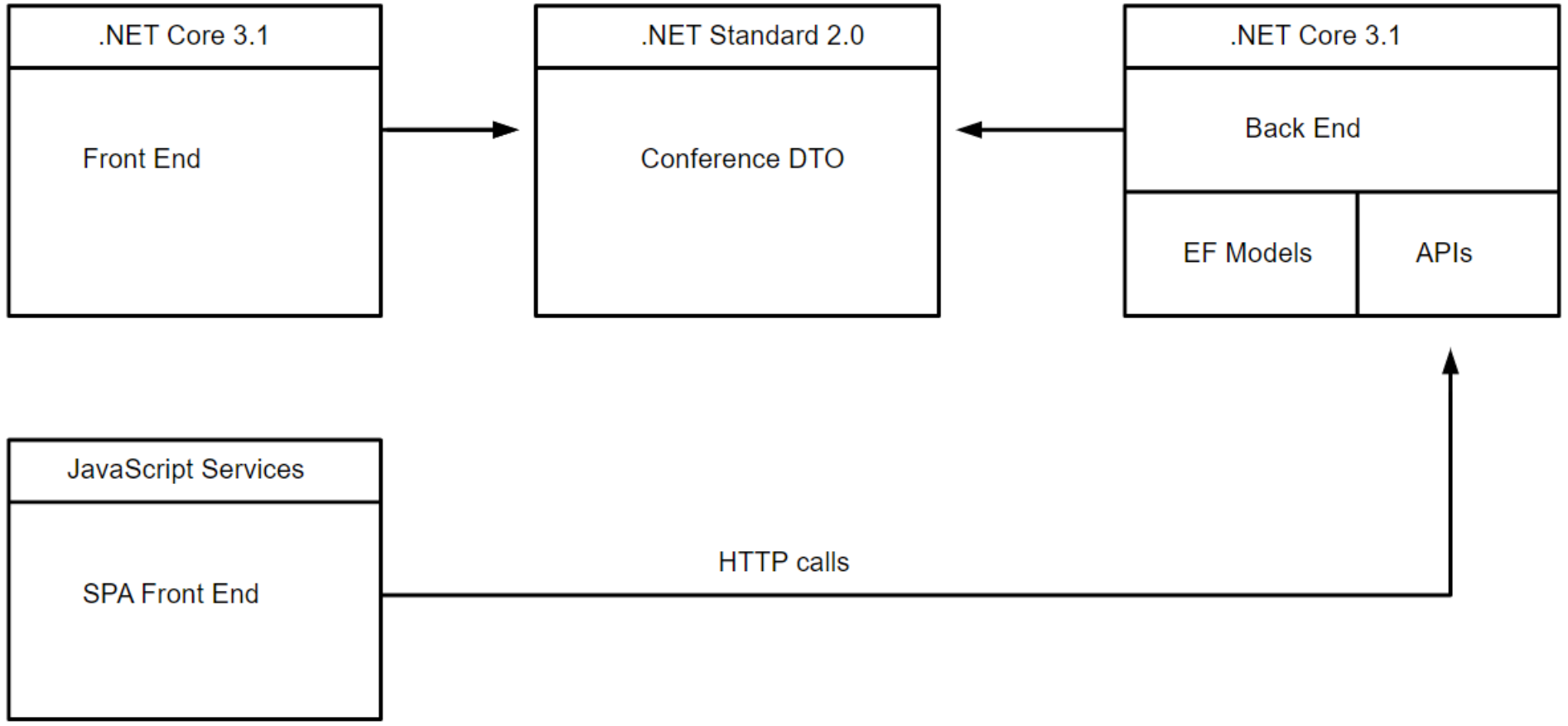
Reference (continued)

- ASP.NET Core – App Building Workshop: link in here.
  - In this workshop, you'll learn by building a full-featured ASP.NET Core application from scratch.
  - We'll start from File/New and build up to an API back-end application, a web front-end application, and a common library for shared data transfer objects using .NET Standard.



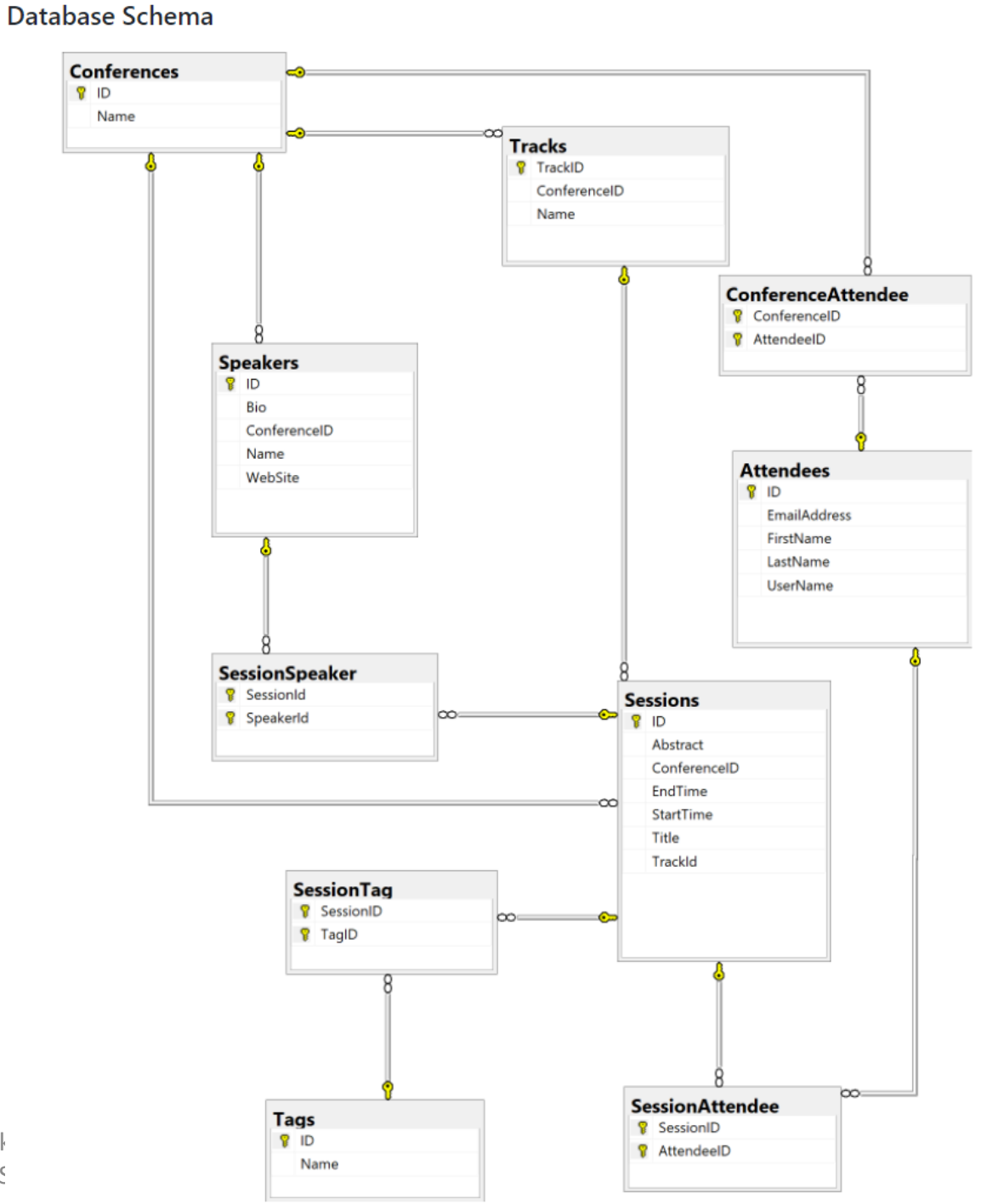
Reference (continued)

Application Architecture



Reference (continued)

- Database schema:

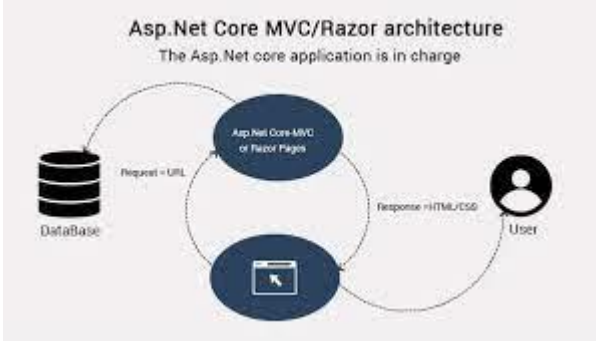


Reference (continued)

- Build Your First ASP.NET Core Web Application Workshop, Part 1: link in here.
  - The ASP.NET Core 3.1 workshop teaches you how to build your first application using a microservice architecture, view models, and razor pages.
  - In this first module, Jeff talks about the benefits of .NET Core, ASP.NET Core, and how you can get started on any operating system with any editor.
  - We build our first API and configure it as an OpenAPI endpoint with Swashbuckle.
- Build Your First ASP.NET Core Web Application Workshop, Part 2: link in here.
  - In this video, Jeff creates a .NET Standard project to manage a collection of Data Transfer Objects (DTO) that will be shared from the backend microservices to the frontend web application.

Reference (continued)

- Build Your First ASP.NET Core Web Application Workshop, Part 3: link in here.
  - The ASP.NET Core 3.1 workshop teaches you how to build your first application using a microservice architecture, view models, and razor pages.
  - Adding the ability to log in to our website. Authentication capabilities of ASP.NET Core
  - Creating an Admin website section and controlling authorization to that section of the site
  - Creating a TagHelper for authorization and controlling content presentation
  - Healthchecks and production readiness for our application
  - Using gRPC with ASP.NET Core

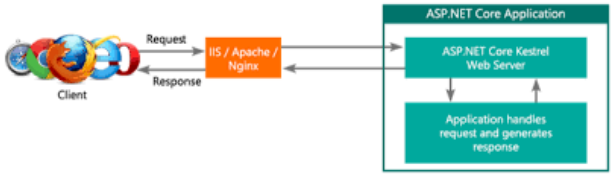


Reference (continued)

- ASP.NET Core: 01 - Learn MVC Basics Tutorial: link in here.
  - First in the series on building the Fasetto Word back-end server using ASP.NET Core as our server.
  - This video just jumps straight into the world of ASP.NET Core, .NET Core and MVC, covering all of the server spin-up, views, models, styles, configuration, settings and so on in one go, with the aim at explaining all areas of the ASP.NET Core MVC application in a single video, to give you a good basis of understanding to get started.
- ASP.NET Core: 02 - SQL Server Basics Learn Transact-SQL Management Studio: link in here.

Reference (continued)

- ASP.NET Core: 03 – Entity Framework Core Basics: link in here.
- ASP.NET Core: 04 – User Login Create Manage ASP.NET Core: link in here.
- ASP.NET Core: 05 – Client Login Token Authentication: link in here.
- ASP.NET Core: 06 – Email Verification: link in here.
- ASP.NET Core: 07 – Integrating Dna Framework: link in here.
- ASP.NET Core: 08 – User Profile API Calls: link in here.
- ASP.NET Core: 09 – Change User Password: link in here.
- ASP.NET Core: 09 – Publishing to Web Server: link in here.



Mobile app

- Xamarin is a cross-platform development tool that allows developers to build native iOS and Android apps, as well as Windows and Mac apps, using a single shared C# codebase. The company also allows developers to test applications on hundreds of devices through the Xamarin Cloud service, offers its own Xamarin Studio IDE, and runs live online classes with Xamarin University programs.

- Xamarin was originally a Microsoft-owned San Francisco-based software company founded in May 2011 by the engineers who made Mono, Xamarin.Android (formerly Mono for Android) and Xamarin.iOS (formerly MonoTouch), which are cross-platform implementations of the Common Language Infrastructure (CLI) and Common Language Specification (often called Microsoft .NET).

- With a C#-shared codebase, developers can use Xamarin tools to write native Android, iOS, and Windows apps with native user interfaces and share code across multiple platforms, including Windows, macOS, and Linux. According to Xamarin, over 1.4 million developers were using Xamarin products in 120 countries worldwide as of April 2017.

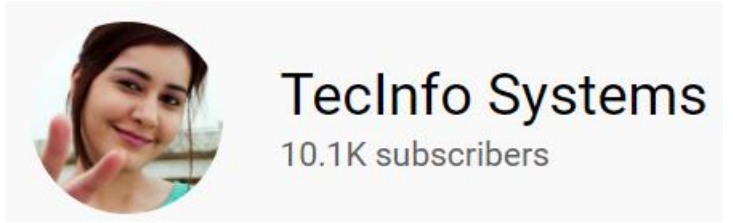
- The name Xamarin comes from the name of the Tamarin monkey, replacing the leading T with an X. This is in line with the naming theme used ever since Ximian was started. On February 24, 2016, Microsoft announced it had signed a definitive agreement to acquire Xamarin.

Xamarin: Updated

- In May 2020, Microsoft announced that *Xamarin.Forms*, a major component of its mobile app development framework, would be *deprecated* in November 2021 in favour of a new .NET based product called MAUI - Multiform App User Interface.
- .NET MAUI Essentials arrived in fall 2021. Alongside more details of MAUI, Microsoft says it will end updates to the Xamarin mobile app development platform in November 2022. Xamarin has been the Microsoft technology for developers to use if they want to develop apps for iOS and Android using C#.

Reference

- TecInfo Systems: Xamarin Tutorial: Beginner to Advanced
  - 01 Xamarin Tutorial – What is Xamarin?: link in here.
  - 02 Installing Xamarin on a PC: link in here.
  - 03 Installing Xamarin on a MAC: link in here.
  - 04 Xamarin Solution Architecture: link in here.
  - 05 Xamarin Forms UI with XAML: link in here.
  - 06 Xamarin Forms MVVM with XAML: link in here.
  - 07 Xamarin Forms Navigation with XAML: link in here.
  - 08 Create a single page UI in Xamarin Forms application: link in here.
  - 09 Xamarin Forms MVVM in C#: link in here.
  - 10 Xamarin Forms Navigation Using C#: link in here.
  - 11 Xamarin Full Tutorial: link in here.
  - … and so on can be seen in here.
- Xamarin Online Tutorial Full Code: link in here.
- Xamarin learning materials @Microsoft: link in here.



Reference (continued)

- AngelSix: Xamarin Android Tutorial 01 – Getting Started: link in here.
- AngelSix: Xamarin Android Tutorial 02 – Lifecycle: link in here.
- AngelSix: GitHub – link in here.