IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710


# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

## IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 12, Issue 5, May 2023

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.423

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 5, May 2023 ||

| DOI:10.15680/IJIRSET.2023.1205349 |

# A Study of Latest Trends in .NET Technology

**Kartik Dipak Shegaonkar [1], Netraja Mulay [2]**

P.G. Student, Department of Computer Application, PES's Modern College of Engineering, Pune, Maharashtra, India[1]

Assistant Professor, Department of Computer Application, PES's Modern College of Engineering, Pune, Maharashtra, India [2]

**ABSTRACT**: The .NET framework is one of the most extensively used modes of development for building advanced applications, both on the desktop and the web. With its enriched set of libraries and tools, .NET has allowed developers to build high-performance, scalable and secure applications that meet the demands of the modern-day business. In recent years, there have been several advancements in the .NET ecosystem that have further improved the capabilities of the platform. These advancements include new tools, frameworks, and languages that make .NET development faster, more efficient, and more enjoyable. Here, we will explore some of the latest trends in .NET technology. We will discuss the benefits of such technologies and how they can assist developers to build competent applications with increased speed and efficiency. By the end of this paper, we will have a better knowledge base of the latest trends in .NET technology and how they can leverage these advancements to improve their development process.

**KEYWORDS**: Dot Net Technology, Xamarin, Blazor, ML.NET, SignalR.

## I. INTRODUCTION

.NET is a software development platform provided by Microsoft to simplify software, application and web oriented development. .NET provides a variety of options for developing a variety of applications including console applications, desktop applications, web applications, websites and many more. .NET also provides a way to manage the backend processes of applications and software platforms that do not support backend programming. .NET has a wide and efficient functionality and it provides a platform for development in majority of programming languages ranging from frontend languages like HTML, CSS, JavaScript, Angular, React, etc. to programming languages like C++, C#, etc. .NET is majorly known and for its efficient and sturdy architecture and framework which is popularly known as the .NET framework which enables the user to implement various approaches of programming including Event-driven programming, procedure oriented programming, object oriented programming, etc.

## II. ARCHITECTURE OF .NET

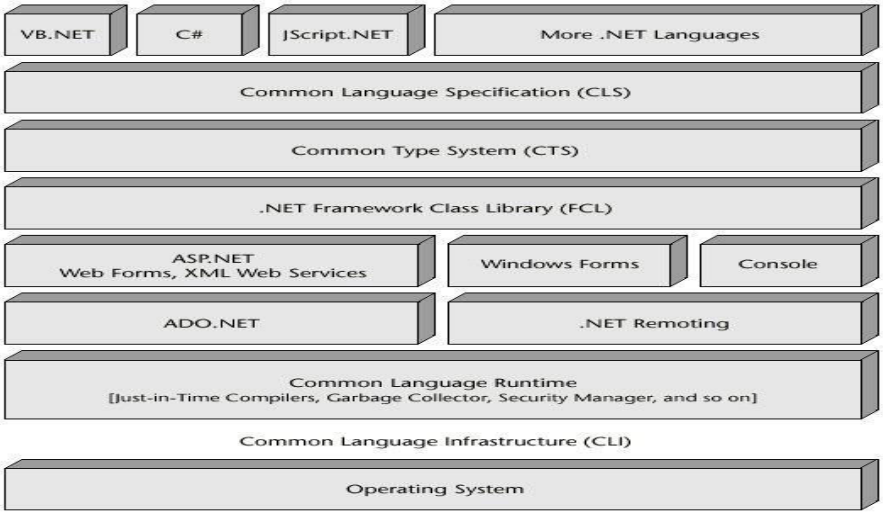


Fig 1: .NET Architecture

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 5, May 2023 ||

| DOI:10.15680/IJIRSET.2023.1205349 |

The .NET architecture consists of components which are interdependent on each other and which together form the .NET framework. These components can be described as below:

**a) Operating System:** The operating system in the above diagram represents the operating system of the system which provides a stage for other application softwares to perform their tasks and acts as a mediator between the user and the system:

**b) Common Language Infrastructure (CLI):** The CLI in the .NET Framework is a runtime environment that enables execution of code from different programming languages and facilitates interoperation and resource sharing.

**c) Common Language Runtime (CLR):** Offers a runtime environment for executing .NET applications.

**d) ASP.NET:** a web application framework for building dynamic web pages and web based services using .NET.

**e) Windows Forms:** a feature for building desktop applications with a graphical user interface.

**f) ADO.NET:** a set of classes for working with databases in .NET applications.

**g) .NET Framework Class Library:** a collection of pre-built classes, interfaces, and value types that provide functionality for .NET applications.

**h) Common Type System (CTS):** The CTS, an integral part of the .NET architecture, establishes guidelines for defining and utilizing data types in various .NET Framework languages, enabling seamless data sharing and interoperation between different languages.

**i) Common Language Specification (CLS):** The CLS, a set of guidelines within the .NET Framework, specifies a subset of the CTS, ensuring that code written in different .NET languages can interoperate and coexist within the same program.

**j)** Above these layers are all the .NET languages like VB.NET, C#, F#, etc. through which the user can create programs to perform certain tasks in a specific way as per requirement.

## III. LATEST TRENDS IN .NET TECHNOLOGY

### A) Xamarin:

Xamarin is a technology that will overshadow the mobile app development market since it has the capability to build advanced apps for multiple operating systems such as windows, Android, and iOS. Also, developers working with Xamarin can build native .NET apps. As a result, the apps will work and look the same way across diverse platforms and offer users a native interface. Moreover, working with Xamarin takes less effort. If you are already familiar with .NET OR C# development, you will find it hard to adopt the tech and build apps on it. Xamarin was acquired by Microsoft in the year 2016 and is now a part of Visual Studio family of tools. With Xamarin, coders can share code among multiple platforms, resulting in quicker development and easier application maintenance. Xamarin provides a rich set of development tools, including Xamarin-Forms, a one of a kind feature which allows developers to create a solo, shared user interface for their apps, as well as enables access to native APIs and device-specific features. In short, Xamarin is a powerful tool for building high-quality mobile apps using familiar technologies and tools.

Some of the salient features of Xamarin are as follows:

a) **Native Performance**: Xamarin permits developers to build native mobile oriented applications using C#, which is compiled to native code, resulting in performance that is comparable to apps built with native languages like Java and Objective-C.

b) **Shared Codebase:** Xamarin facilitates code sharing across various platforms, such as iOS, Android, and Windows, leading to reduced development time and cost for developers.

c) **Xamarin-Forms:** Xamarin.Forms is a UI toolkit that empowers developers to generate a unified and reusable user interface for their applications, compatible with multiple platforms.

d) **Access to Native APIs:** Xamarin provides access to native APIs and device-specific features, which allows developers to build apps that can take advantage of the full capabilities of the underlying platform.

e) **Integration with Visual Studio:** Xamarin integrates seamlessly with Visual Studio, allowing developers to use familiar tools and workflows to build, test, and deploy their apps.

f) **Community and Support:** Xamarin has a large and active community of developers, as well as support from Microsoft, which provides resources, training, and support for developers using the platform.

### B) Blazor:

Many developers believe Blazor will be the next big thing in web development. It is a relatively new technology for building web apps. Blazor enables the development of full-stack web applications using .NET and C#, eliminating the need to write any JavaScript code. Also, you can easily create an amazing user experience fast with

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 5, May 2023 ||

| DOI:10.15680/IJIRSET.2023.1205349 |

Blazor's flexible and reusable component model, which is simple and efficient to use. Blazor enables developers to create web applications that run entirely on the server-side, entirely on the client-side, or a combination of both, using Web Assembly technology to execute .NET code directly in the browser. Blazor also provides a rich set of UI components and tooling for building modern web applications. It was first introduced by Microsoft in 2018 as an experimental project and was later released as a stable product in 2020. Blazor allows developers to create web applications using familiar C# language constructs and .NET APIs, which can increase productivity and simplify the development process. Developers can use Visual Studio or Visual Studio Code to create Blazor applications, and can choose from several hosting models, including client-side, server-side, or hybrid.

Here are some salient features of Blazor:

a) **Server-side and client-side hosting models:** Blazor supports both server-side and client-side hosting models, allowing developers to choose the best approach for their specific use case.
b) **WebAssembly technology:** Blazor uses WebAssembly technology to execute .NET code directly in the browser, providing a fast and responsive user experience.
c) **.NET Core and C#:** Blazor is built on top of .NET Core and uses C# as the primary programming language, allowing developers to leverage their existing .NET skills and tools.
d) **Reusable UI components:** Blazor provides a powerful component model that allows developers to create reusable UI components, improving code organization and reducing duplication.
e) **Data binding and validation:** Blazor supports data binding and validation, making it easy to create rich and interactive user interfaces.
f) **Routing:** Blazor supports client-side routing, allowing developers to create single-page applications with multiple views and URLs.
g) **Open-source and cross-platform:** Blazor is an open-source project and can run on a variety of platforms, including Windows, Linux, and macOS.
h) **Integration with other .NET tools:** Blazor integrates with other .NET tools, including Visual Studio, Visual Studio Code, and the .NET CLI, making it easy to get started and build applications quickly.

**C) ML.NET:**

Machine learning has to be the hottest topic right now. And if you want to build apps using machine learning, ML.NET is there to help you out. A cross-platform, open-source machine learning framework has been developed specifically for .NET developers, enabling the creation of machine learning applications using existing ML.NET skills. Also, it lets you use other popular ML libraries like Infer.NET, TensorFlow, ONNX, and others in your app development journey. ML.NET encompasses diverse machine learning algorithms (classification, regression, clustering, recommendation) and tools for data preparation and feature engineering, enabling developers to build various ML applications like fraud detection, sentiment analysis, image recognition. It integrates smoothly with popular .NET tools (Visual Studio, Visual Studio Code, Azure Machine Learning) for easy adoption and creation of robust ML apps.

Some of the interesting features of ML.NET are as follows:

a) **Open-source and cross-platform:** ML.NET is an open-source machine learning framework that is cross-platform and can run on Windows, macOS, and Linux.
b) **No prior machine learning expertise required:** ML.NET is designed for .NET developers who may not have expertise in machine learning. The framework provides high-level APIs and tools for training and using machine learning models, allowing developers to focus on their applications rather than the underlying machine learning algorithms.
c) **Integration with .NET ecosystem:** Utilizing the foundation of .NET Core, ML.NET seamlessly integrates with prominent .NET tools like Visual Studio, Visual Studio Code, and Azure Machine Learning, streamlining the initiation and development of robust machine learning applications.
d) **Range of machine learning algorithms:** ML.NET incorporates a variety of machine learning algorithms like classification, regression, clustering, and recommendation, alongside data preparation and feature engineering tools.
e) **Pre-built models:** ML.NET includes pre-built models that can be used out of the box, saving developers time and effort.
f) **Easy to use:** ML.NET is designed to be easy to use, with a simple and intuitive API that allows developers to quickly build and train machine learning models.
g) **Scalable:** ML.NET is scalable and can be used to build machine learning applications for a wide range of use cases, from small-scale projects to large-scale enterprise applications.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 5, May 2023 ||

| DOI:10.15680/IJIRSET.2023.1205349 |

h) **Interoperability with other machine learning frameworks:** ML.NET supports interoperability with other machine learning frameworks such as TensorFlow and ONNX, allowing developers to import pre-trained models into their ML.NET applications.

**D) SignalR:**

SignalR is a real-time web application framework developed by Microsoft for the .NET platform. It enables server-side code to push content to connected clients in real-time, eliminating the need for the client to continuously poll the server for updates. SignalR uses WebSocket technology, which provides a full-duplex, bidirectional communication channel between the client and server, allowing both sending and receiving messages at any time. However, SignalR can also use fallback transports such as Server-Sent Events (SSE) and Long Polling for older browsers that do not support WebSockets. SignalR can be used for a variety of real-time applications, such as chat applications, online gaming, and real-time monitoring systems. It provides a simple programming model with a few lines of code, making it easy to get started with real-time web applications. SignalR is available as a NuGet package and can be used with various .NET platforms, including ASP.NET Core, ASP.NET MVC, and Windows Forms. It also supports cross-platform development, enabling developers to build real-time web applications for Windows, macOS, and Linux.

Some exceptional features of SignalR are as follows:

a) **Real-time Communication:** SignalR allows for real-time bi-directional communication between the server and client, enabling the server to push updates to the client instantly.
b) **Automatic Reconnection:** SignalR handles automatic reconnection to the server in case of network disruption, providing a seamless experience for the user.
c) **Cross-Platform Support:** SignalR is available for multiple .NET platforms, including ASP.NET Core, ASP.NET MVC, and Windows Forms, and supports cross-platform development.
d) **Scalability:** SignalR can handle a large number of connected clients with minimal impact on server resources, making it highly scalable.
e) **Security:** SignalR provides built-in support for authentication and authorization, enabling developers to secure their real-time applications easily.
f) **Flexibility:** SignalR is highly flexible and can be used for a wide range of real-time applications, including chat applications, online gaming, and real-time monitoring systems.
g) **Simple Programming Model:** SignalR provides a simple programming model, making it easy for developers to get started with real-time web applications quickly.

**E) Server less Computing:**

Serverless computing, also referred to as Function-as-a-Service (FaaS), is a cloud computing paradigm where the infrastructure management and resource allocation are automated by the cloud provider based on application requirements. Developers simply write and upload their code as functions, while the cloud provider handles scaling, monitoring, and infrastructure management. Azure Functions, within the .NET ecosystem, is a widely adopted serverless computing platform offered by Microsoft. It enables developers to write and deploy small code segments, known as functions, which execute in response to events like HTTP requests or messages in a queue. Multiple programming languages, including C#, JavaScript, and Python, are supported for developing these functions.

Some of the benefits of Azure functions can be listed as below:

a) **Automatic scaling:** Azure Functions automatically scales the infrastructure to meet demand, without the need for developers to manage servers.
b) **Pay-per-use pricing:** Serverless computing allows developers to solely pay for the precise utilization of computing resources by their application, eliminating any upfront expenses.
c) **Easy integration with other Azure services:** Azure Functions effortlessly integrates with a plethora of Azure services like Azure Blob Storage and Azure Cosmos DB, facilitating the seamless construction of intricate applications.

## IV. CONCLUSION

Many flagship technologies are yet to arrive in the .NET market which will be mostly available in the near future. These technologies will be capable of revolutionizing the development approach of businesses leading to very high demand in jobs related to the .NET domain for upcoming as well as existing .NET professionals.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 5, May 2023 ||

| DOI:10.15680/IJIRSET.2023.1205349 |

Throughout the study conducted, we have examined various aspects of .NET, including the transition from .NET Framework to .NET Core, the shift towards cross-platform compatibility, modular architecture, and the expanding ecosystem of tools and frameworks.

The advantages of adopting .NET Core have been highlighted, such as enhanced performance, improved deployment and management, and the ability to build applications that can run seamlessly on multiple operating systems. The increasing support for cloud-native development, containerization, and microservices architecture within the .NET ecosystem has been discussed.

Furthermore, the growing community and industry support for .NET, as well as the integration with Microsoft Azure and its impact on cloud-based solutions, have been emphasized as significant trends. The availability of machine learning and AI libraries through ML.NET has also been highlighted as a notable trend in leveraging the power of data and intelligence within .NET applications.

## REFERENCES

[1] Programming ASP.NET Core – Dino Esposito
[2] Professional C# and .NET 2021st Edition – Christian Nagel
[3] Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals - Daniel Hermes
[4] .NET Documentation (https://dotnet.microsoft.com/en-us/learn/dotnet/)
[5] C# Documentation (https://learn.microsoft.com/en-us/dotnet/csharp/)
[6] Azure Documentation (https://learn.microsoft.com/en-us/azure/?product=popular)
[7] Asp.net and its framework – Komal Chauhan
[8] .Net Technology a Realistic Programming Approach using Framework – Jayesh Kumar Patel
[9] C# and the .NET framework: ready for real time? – M .H .Lutz

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.423

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

## IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details