Gmail

Yashasvi Shukla <yashasvishukla123@gmail.com>

**.NET Developer Roadmap 2023.**
1 message

**Dr. Milan Milanovic from Tech World With Milan** <techworldwithmilan@substack.com> Thu, Nov 2, 2023 at 9:30 PM
Reply-To: "Dr. Milan Milanovic from Tech World With Milan" <reply+2ae05m&1mam49&&72873ea0caba442934e85853a3fd824af3067cf9389a513af09cb0bbe82b9ed6@mg1.substack.com>
To: yashasvishukla123@gmail.com



TechWorld WithMilan
simplifying complex topics

# .NET Developer Roadmap 2023.

DR. MILAN MILANOVIĆ
NOV 2

READ IN APP

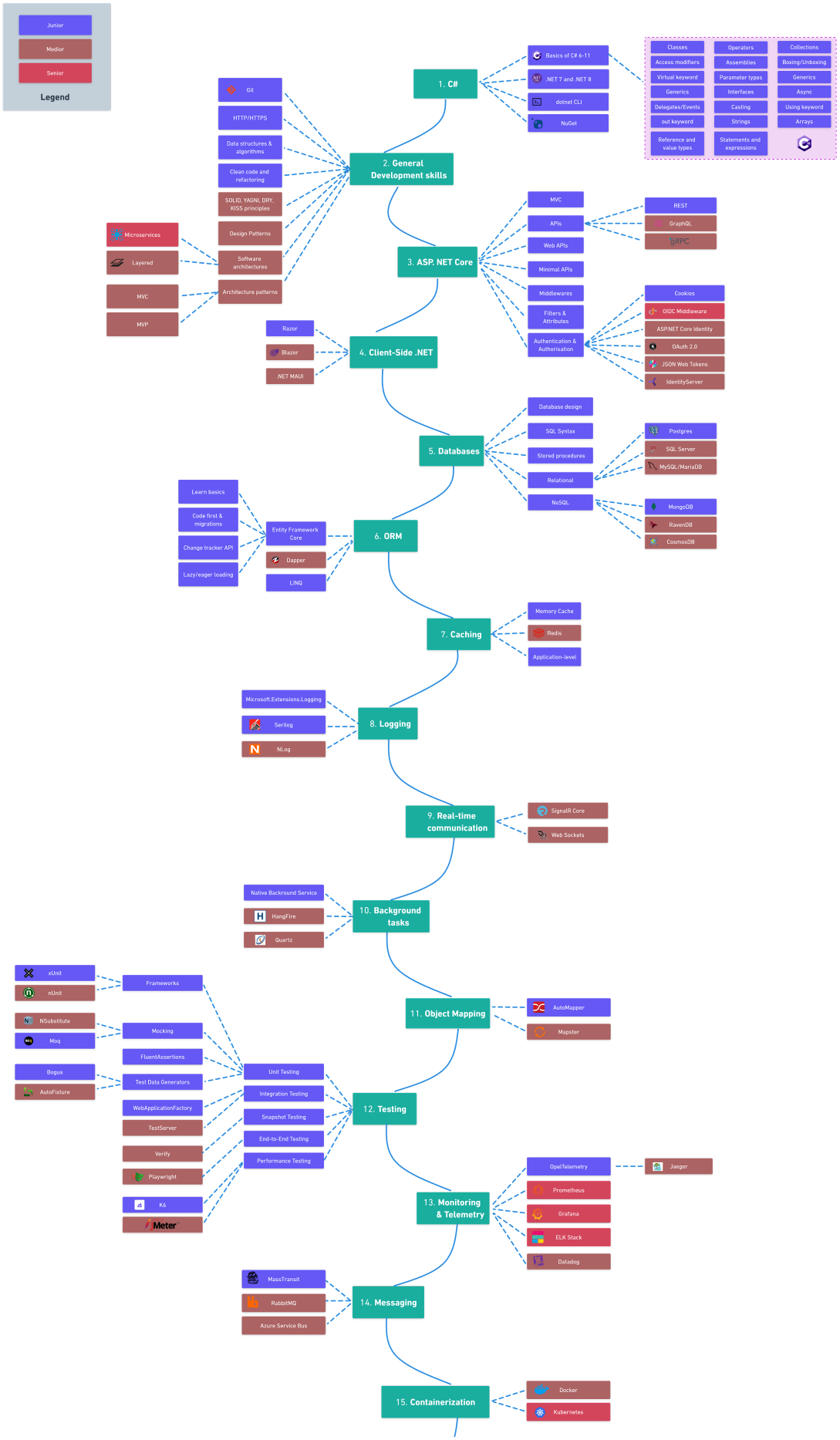
This is a step-by-step guide to becoming a .NET Developer, with links to relevant learning resources and by seniority level.

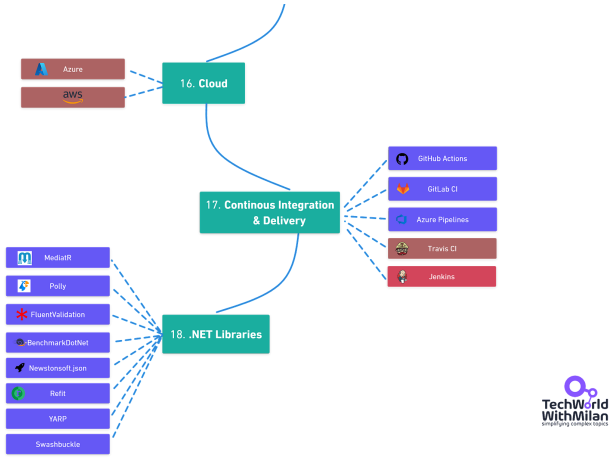
## Postman's VS Code Extension (Sponsored)

*Postbot is now available across Postman with enhanced capabilities! The latest refresh of Postbot now offers a consistent, conversational interface available to you across your workspace. Learn how you can leverage Postbot throughout Postman to get contextual assistance.*


Check it out!

# .NET Developer Roadmap 2023.






# Learning resources

## 1. C#

C# is a programming language developed by Microsoft. It's a go-to choice for building anything from desktop applications and games (using Unity) to cloud-based solutions and web services. With strong support for object-oriented programming and a rich library, it's designed to be easy and efficient.

You need to understand different **C# language features**, such as:

- Object-oriented programming (classes, objects, interfaces, inheritance, polymorphism)
- Variables, data types, and operators
- Reference and value types
- Control flow (conditionals, loops)
- Generics
- Exception handling
- Delegates and events
- Assemblies
- Collections
- Async and await for asynchronous programming

But also **.NET libraries and APIs** for:

- File I/O and serialization

- Collections and data structures
- Networking
- Multithreading and task parallelism
- Security and cryptography

**Resources**:

- Microsoft Learn C#.
- Microsoft C# Fundamentals for Absolute Beginners.
- Microsoft C# 101
- Udemy C# for Beginners - Coding From Scratch (.NET Core)
- C# Basics for Beginners: Learn C# Fundamentals by Coding
- Learn dotnet CLI
- Learn.NET official Microsoft tutorials
- Dot Net Perls - Many code examples in C#
- Become a Full-stack .NET Developer - Advanced Topics
- Advanced concepts:
  - Async/Await by Stephen Toub
  - Threading in C# by Joseph Albahari

## 2. General Development Skills

Mastering design patterns, clean code, and version control like Git enables you to write efficient, maintainable code that works and thrives in a team environment. It's the difference between being a coder and a skilled software engineer.

Here, you need to know different principles, such as:

**SOLID Principles**:

- Single Responsibility Principle (SRP)
- Open/Closed Principle (OCP)
- Liskov Substitution Principle (LSP)

- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DIP)

But also:

- DRY (Don't Repeat Yourself)
- KISS (Keep It Simple, Stupid)
- YAGNI (You Ain't Gonna Need It)
- Law of Demeter (LoD) or Principle of least knowledge
- Composition over Inheritance
- The Principle of least astonishment

**Resources**:

- Learn Git
- Learn HTTP(S) protocol
- Learn Data Structures & Algorithms
- Learn Clean Code
- Learn Refactoring fundamentals
- Learn Design Patterns from the book or video tutorials
- Learn Main software design principles
- Learn SOLID principles of OO Design in depth.
- Learn Fundamentals of Software Architectures
- Learn Microservices and DAPR

## 3. ASP.NET Core

It is a cross-platform, high-performance framework developed by Microsoft for building web apps, APIs, and microservices. You can also run your apps on Windows, Linux, or macOS. It's engineered for flexibility and scalability with features like built-in dependency injection and a robust configuration system.

Here, you also need to know **web application fundamentals**, such as:

- HTML, CSS, and JavaScript for front-end development

- HTTP protocols, request/response model, and RESTful APIs
- Routing, middleware, authentication, and authorization
- Model-View-Controller (MVC) and Razor Pages patterns

**Resources**:

- ASP.NET Core Fundamentals by Scott Alen course
- ASP.NET MVC 5 Fundamentals by Scott Alen course
- Advanced ASP.NET Core 3.1 MVC Udemy course
- Pro ASP.NET Core 6 book
- APIs
  - REST
  - GraphQL
  - gRPC
- Dependency Injection
  - Life Cycles
  - Microsoft Extensions Dependency Injection
  - Autofac
  - Scrutor
- Application Settings & Configurations
- Middlewares
- Filters & Attributes
- Authentication or this Reddit thread
- Authorization
- IdentityServer
- Auth0
- OIDC

## 4. Client-Side .NET

If you want to build UIs in .NET, you will need these frameworks. Razor is a template engine for creating dynamic HTML, while Blazor takes it up a notch, letting you build interactive web UIs using C# instead of JavaScript. MAUI is a Xamarin successor made for building cross-platform mobile apps.

**Resources**:

- Razor
- Blazor
- .NET MAUI

## 5. Databases

Good database design ensures efficient data storage and quick retrieval, making your app run smoother and scale easier. SQL, the go-to language for database interaction, gives you the power to query, update, and manage the data you've so carefully designed to store.

Here, you need to know:

- SQL Syntax
- Basics of Database design (normal forms, keys, relationships)
- The Difference Between Inner, Left, Right, and Full Join
- SQL Queries Execution Order
- What is Query Optimizer

**Resources**:

- Database design
- Learn SQL
- Relational
    - SQL Server
    - PostgreSQL
    - MariaDB
    - MySQL

- NoSQL
  - MongoDB
  - RavenDB
  - CosmosDB
- Tools:
  - SQLFlow - a great tool to visualize SQL queries.

## 6. ORM

Object-relational mapping (ORM) is like a translator between your object-oriented C# code and the relational database, eliminating the tedious task of writing SQL queries for basic CRUD operations. Using ORM frameworks like Entity Framework, you can manipulate data as objects in your code, making it more readable and maintainable. This speeds up development, minimizes errors, and lets you focus on complex business logic rather than wrestling with database syntax.

For **Entity Framework**, you need to know the following:

- DbContext and DbSet for managing database connections and querying data
- Code-First and Database-First approaches for defining data models
- Migrations for managing database schema changes
- Querying data using LINQ and raw SQL
- Tracking changes and saving data

**Resources**:

- Entity Framework Core
  - Code First Migrations
  - Change Tracker API
  - Lazy Eager Explicit Loading
- Dapper
- LINQ

## 7. Caching

Caching is like your app's personal short-term memory, storing frequently accessed data so it can be quickly retrieved without taxing your database. By reducing database load and speeding up data access, caching gives your app the competitive edge it needs to meet user demands for responsiveness and availability.

**Resources**:

- Memory Cache
- Redis
- Application-Level
  - Built-in
  - Output Caching

## 8. Logging

Logging captures runtime information, errors, and other crucial data that can help you quickly identify and fix issues, making your application more reliable and secure. Logging frameworks like NLog or Serilog integrate seamlessly into .NET, giving you a real-time diagnostic tool indispensable for monitoring application health, troubleshooting problems, and even gathering insights for future development.

**Resources**:

- Serilog
- NLog
- Microsoft.Extensions.Logging

## 9. Real-time communication

Real-time communication technologies, like SignalR in the .NET ecosystem, enable these functionalities by maintaining a constant connection between server and client. They are used in interactive experiences, whether live chat, notifications, or real-time updates.

**Resources**:

- SignalR Core
- WebSockets
- Socket.IO

## 10. Background tasks

These services run tasks in the background, freeing up your application to focus on user interactions. Whether data processing, automated emails, or periodic clean-ups, background services ensure these tasks don't slow down or interrupt the user experience.

**Resources**:

- Native BackgroundService
- HangFire
- Quartz

## 11. Object Mapping

Their libraries automate the task of mapping between objects, eliminating the need for repetitive, error-prone manual mapping code. This boosts productivity and minimizes bugs, especially when dealing with complex models and DTOs (Data Transfer Objects).

**Resources**:

- AutoMapper
- Mapster

## 12. Testing

Unit tests focus on isolated pieces of your code, integration tests ensure different parts play well together, and end-to-end tests validate the entire user journey within your application. Together, they form a safety net, catching bugs early, simplifying debugging, and making your codebase robust and maintainable.

Here you need to know:

- Test frameworks (xUnit, NUnit, MSTest)
- Test runners and test explorers
- Asserts and test attributes
- Mocking libraries (Moq, NSubstitute, etc.)

**Resources**:

- Unit Testing
  - Frameworks
    - xUnit
    - NUnit
    - MSTest
  - Mocking
    - Moq
    - NSubstitute
  - Assertion
    - FluentAssertion
    - Shouldly
  - Test Data Generators
    - Bogus
    - AutoFixture
- Integration Testing
  - WebApplicationFactory
  - TestServer
  - Testcontainers
- Snapshot Testing
  - Verify
- Behavior Testing

    - SpecFlow
- End-to-End Testing
    - Playwright
- Performance Testing
    - K6
    - JMeter

## 13. Monitoring & Telemetry

These tools provide real-time insights into your application's performance, user behavior, and error rates, enabling you to address issues before they escalate into full-blown problems proactively.

- **Monitoring** focuses on the health and availability of services and systems, often triggering alerts for predefined conditions.
- **Telemetry** collects, processes, and transmits data from systems, enabling analysis of patterns, trends, and anomalies.

**Resources**:

- Prometheus
- Grafana
- Datadog
- ELK Stack
- OpenTelemetry
- Jaeger

## 14. Messaging

Messaging systems act as a middleman between different parts of your system, allowing them to communicate without being directly connected. This decouples your components, making scaling, maintaining, and adding new features easier. Plus, it improves fault tolerance—so if one part fails, it doesn't bring down the whole system.

**Resources:**

- RabbitMQ
- MassTransit
- Azure Service Bus
- NServiceBus

## 15. Containerization

Container solutions encapsulate your .NET application, libraries, and runtime into isolated containers. This enables consistency across multiple development and production environments, resolving dependency issues. With features like layered file systems, you can easily manage container images for ASP.NET, .NET Core, or other .NET services, optimizing build times and resource utilization.

**Resources:**

- Docker
- Kubernetes

## 16. Cloud

Cloud providers provide a layer of APIs to abstract infrastructure and provision it based on security and billing boundaries. The cloud runs on servers in data centers, but the abstractions cleverly give the appearance of interacting with a single "platform" or extensive application. The ability to quickly provision, configure, and secure resources with cloud providers has been critical to the tremendous success and complexity of modern DevOps.

The most popular cloud providers in the market are **AWS** and **Azure**, as well as **Google Cloud**.

Here, you must know how to manage users and administration, networks, virtual servers, etc.

**Resources**:

- AWS
- Azure

- Google Cloud

## 17. Continuous Integration & Delivery (CI/CD)

CI/CD automates the building, testing, and deployment stages into a streamlined, error-resistant pipeline. This means faster releases, bug fixes, and more time to focus on feature development.

Here you need to know how to:

- Build and deployment tools (MSBuild, dotnet CLI)
- Version control systems (Git, Azure DevOps)
- CI/CD platforms (GitHub Actions, Azure Pipelines, Jenkins, TeamCity)

**Resources**:

- GitHub Actions
- Gitlab CI
- Azure Pipelines
- Travis CI
- Jenkins
- TeamCity

## 18. NET Libraries

Some useful .NET libraries:

- **MediatR** - Mediator pattern implementation in .NET
- **Polly** - Fault-handling library that allows expressing policies such as Retry and Circuit Breaker.
- **Fluent Validation** - .NET validation library for building strongly typed validation rules.
- **Benchmark.NET** - .NET library for benchmarking.
- **Newstonsoft.json** - High-performance JSON framework for .NET.
- **Refit** - Turns your REST API into a live interface.
- **YARP** - Reverse proxy server.

- **Swashbuckle** - Swagger tools for documenting APIs built on ASP.NET Core.

> ℹ Along with the roadmap presented here, there is a **living GitHub repo** with more info. Check it out **here**.

## More ways I can help you

1. **1:1 Coaching:** Book a working session with me. 1:1 coaching is available for personal and organizational/team growth topics. I help you become a high-performing leader 🚀.
2. **Promote yourself to 18,000+ subscribers** by sponsoring this newsletter.

LIKE COMMENT RESTACK