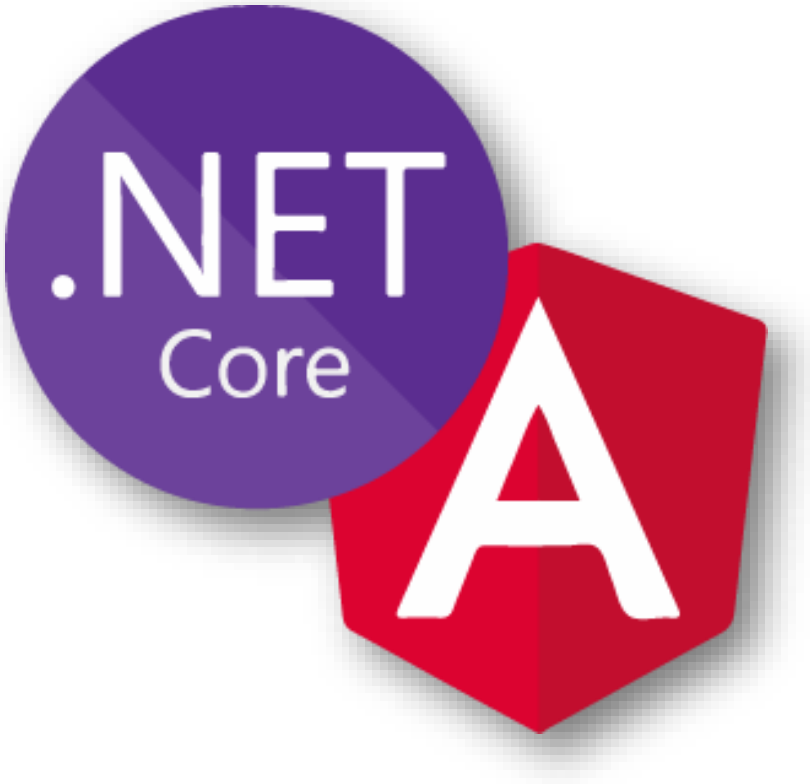
ASP.NET Core 8.0 and Angular 18.0 with Material



اهداف دوره : فناوری ASP.NET Core پلتفرمی یکپارچه و دارای کدهای باز Open-Source بوده که قابلیت اجرا روی سیستم عامل های Windows, Linux, Mac را دارد و بر اساس آخرین ارزیابی های انجام شده TechEmpower Framework Benchmarks در لینک https://bit.ly/3SGPI9R این فریم ورک توانسته است بالاترین سطح کارایی را نسبت به سایر فریم ورک های موجود از قبیل Node.js, Java Servlet کسب نماید.

همچنین توسعه روز افزون فناوری های فرانت، منجر به ظهور فریم ورک قدرتمند با عنوان Angular شده که آنرا شرکت گوگل ارایه نموده است و می تواند در بالاترین سطح کیفیت نیازمندی های توسعه نرم افزار را در بخش فرانت تامین نماید.

از طرفی طی ارزیابی منتشر شده توسط Stack Overflow در لینک https://bit.ly/3QHBEKZ زبان های برنامه نویسی C#, TypeScript نیز توانسته اند بیشترین میزان رشد محبویت را در سطح برنامه نویس ها کسب نمایند. بدیهی است تلفیق تکنولوژی های قدرتمند و محبوب فوق می تواند تجربه بسیار جذاب و هیجان انگیزی را در توسعه نرم افزار ایجاد نماید.

مخاطبین دوره : تمامی افرادی که علاقه مند به یادگیری و تولید برنامه های Web-Based می باشند، می توانند در این دوره شرکت نمایند. بازار کار امروز بدلیل رقابتی شدن، انتظار داشتن مهارت های بسیار زیادی را حتی از مبتدیان این رشته دارد که یادگیری آنها نیازمند زمان و هزینه قابل توجهی می باشد! پس چه باید کرد ؟ ما بر اساس تجربه و شناخت دقیق بازار کار و محدودیت های اقتصادی در حال حاضر، تلاش کردیم دوره ای را طراحی نمائیم که با رویکرد پروژه محور و اجتناب از تئوری پردازی محض، در کوتاه ترین زمان مهارت های مذکور را آموزش دهد.

پیشنیاز دوره : گذراندن دوره های Programming In C# (1) و Web Fundamental یا تسلط به مطالب دوره های مذکور

مدت دوره : مدت زمان آموزشی این دوره ۵۱ ساعت می باشد.

دستاوردها: در انتهای دوره فراگیران توانایی طراحی فرم های اطلاعاتی با استفاده از انگولار متریال، کار با کنترل های پیشرفته نظیر جدول و دیالوگ، اعتبارسنجی دادها، پیاده سازی سرویس های ارسال و دریافت اطلاعات، ایجاد مدل های اطلاعاتی سمت فرانت اند/بک اند، ساخت بانک اطلاعاتی در محیط SQL Server با رویکرد Code First، آشنایی با بعضی از اصول مهندسی نرم افزار نظیر SOLD, DRY, PI, SoC, IoC، الگوهای طراحی نظیر Factory, Builder, Chain Of Responsibility, Repository, Result، آشنایی با معماری های

Monolith Modular, Clean Architecture و پیاده سازی بخشی از نیازمندیهای امنیتی داشته باشند و در نهایت همه موارد فوق را از طریق ساخت یک پروژه عملی تجربه نمایند.

سرفصل مطالب آموزشی:

Introduction

- ASP.NET Core
- TypeScript
- Angular

Development IDE

- Visual Studio Code and Required Plugins
- .NET 8.0 SDK
- Node.js

Part 1 – Backend

ASP.NET Core Fundamentals

- Dotnet-CLI Command
- Create new project
- Overview project structure
- Top-level Statement
- File Scoped Namespace
- Global Namespace
    - Implicit
    - Explicit
    - Custom
- Program

Middleware

- Middleware
- Request delegate

- Async/await
- Register Middleware
- Register Priority
- Create Custom Middleware
- Middleware Pipelining
- Middleware Extension
- Built-In Middleware's

Service Provider

- Service Provider
- Create Service Instance
- Dependency
    - Structural
    - Behavioral
- SOLID Principals
- Dependency Inversion Principal
- Inversion of control
- Dependency Injection
    - Constructor Injection
    - Method Injection
    - Property Injection
- Dependency Graph
- Object life cycle
- Service Extension
- Application Configuration
    - Launch
    - Setting
    - Command line arguments
- Option Pattern
    - Option Setting

        - Option Setup
    - Application Environment
        - Development
        - Production
        - Staging

RESTfull Service

- Rest Architecture
- Design aspect
- RESTful
- Controller, Action, Routing
- Specific Type, IActionResult, ActionResult<T>
- GET, POST, PUT, DELETE, PATCH
- HTTP Status Code
    - Informational
    - Success
    - Redirection
    - Client Error
    - Server Error
- Minimal API, Endpoint
- Carter Framework
- URI best practice
- API Test Tools
    - Swagger
    - Postman
    - Thunder Client

Architectural Concepts

- Monolithic application
- All-in-one Project
- Layers

- Traditional "N-Layer"
- Onion Architecture
- DRY Principal
- Domain-Driven Design (DDD)
- Clean Architecture

Setup Catalog Service Project

- Overview eShop Application
- Creating Catalog Service
- Add Presentation Layer
- Add Application Layer
- Add Domain Layer
- Add Infrastructure Layer

Domain Layer

- DDD Overview
- Domain Primitives
    - Contracts
        - IEntity
        - IRepository
    - Abstraction
        - Entity
- Domain Features
    - Product Features
        - Product Entity
        - Product Repository

Application Layer

- Application Overview
- Contracts
    - IProductManager
    - IUserManager

        - IPasswordService
        - IAuthenticationManager
    - Application Features
        - Result Pattern
        - Product Features
            - Implement Manager
            - Implement Mapper by AutoMapper
            - Implement DTOs with C# Record
            - Implement Validators with FluentValidation
            - Implement Data Services
                - ✓ Paging
                - ✓ Sorting
                - ✓ Filtering
                - ✓ Searching
        - User Features

Infrastructure Layer

- Persistence
    - Faking with Bogus
    - Implement Product Fake Repository
    - Add Entity Framework Core
    - Implement Product Configuration
    - Implement Catalog Context
    - Implement SQL Repository
    - Implement Product Repository
    - Add Migrations
    - Update Database
- Security
    - Add Microsoft Identity Manager
    - Authentication

- User Registration
- User Login
- JSON Web Token
    - ✓ Implement Identity Token
    - ✓ User Claims
    - ✓ Token Handler

Presentation Layer

- Add Carter Framework
- Implement Product End Points
    - Add, Update, GetById, Delete
- Implement User End Points
    - Register, Login

Part 2 - Frontend

TypeScript

- Basic Syntax
- Object Oriented Programming
    - Class
    - Interface
    - Inheritance
    - Generic
- Modular Programming
    - Class, Function, Variable, Constant Named or Default Export
    - Import Module Item
    - Namespace
- Functional Programming
    - Pure Function
    - Higher Order
- Aspect Oriented Programming
    - Cross Cutting Concern
    - Decorator Pattern
- Asynchronous Programming
    - Promise
    - async/await
- Reactive Programming

        - RxJs
        - Observable
        - Subject
        - Operators
    - Webpack
        - Configuration
        - Building TypeScript Library
        - Publish Package
        - Creating NPM Repository
        - Publish Package to NPM

Angular

- Project structure
- Basic syntax
- Building blocks overview
    - Module
    - Component
    - Directive
- Template-Driven form
- Reactive-Driven form
    - FormControl
    - FormGroup
    - ArrayControl
- Service
- Dependency Injection
- HttpClient
- Material
    - Field
    - Validator
    - Table
    - Dialog
- Toastr
- Routing

Setup Catalog UI Project

Components

- Install Material Components
- Design Layouts

- Configuring routing
- Product Components
    - Product List
    - Public Search
    - Add/Edit Dialog Service
        - Reactive Forms
    - Delete and Confirmation Message
    - Data Validations
        - Required
        - Length
        - Pattern
- User Components

Models

- Product
- User
- Token

Services

- Product Service
- User Service
- Client-Side Storage