SOFTWARE DEVELOPMENT AND ENGINEERING PRACTICES

Full-Stack Development with .NET and React

Level: Practitioner • 2 days (expandable to 3) • Virtual, In-person

Overview

A .NET API and a React front end is one of the most common architectures in enterprise software, and the hard part is rarely either half on its own. It is the seam: designing an API contract the front end can actually build against, moving data and errors across it cleanly, handling authentication on both sides, and getting the two halves developed, run, and deployed together without friction.

This is a hands-on, practitioner course. It builds one application from back to front in the order the dependencies run: first the .NET Web API and its data layer, then the React front end, then the seam between them, and finally authentication and deployment across the whole stack. Rather than attempt full coverage of two large ecosystems, it goes deep on the end-to-end path and the integration decisions that make or break these systems. Every module ends with a lab and builds on the one before, so you leave with a complete working application.

Who Should Attend

- C# and .NET developers adding a modern front end to their skill set
- Front-end developers who want to understand and build the .NET back end they consume
- Full-stack developers who have assembled these pieces before but never been taught the seam properly

  Developers new to modern JavaScript should take *Modern JavaScript (ES6+)* first.

Prerequisites

- Working proficiency in C#
- Solid modern JavaScript; prior React experience is helpful but not required
- Basic understanding of HTTP and JSON

What You Will Learn

- Design a full-stack architecture and an API contract both halves can build against
- Build a .NET Web API back end with validated endpoints and real persistence
- Build a React front end with components, hooks, and client-side routing
- Connect the two: data fetching, loading and error states, and CORS done correctly
- Implement authentication end to end, from login UI to protected API endpoints
- Run, debug, and deploy the front end and back end as one application

Course Outline

Day one: the back end, and the contract it offers

- Full-Stack Architecture and the Contract
  - How the pieces fit: SPA front end, JSON API, database
  - Designing the API contract first, so both halves have a target

  - Setting up the solution: projects, tooling, and running both halves locally
  - Lab: scaffold the API and React projects and define the contract for the application you will build
- Building the .NET API
  - Controllers, routing, and model binding for the application's resources
  - DTOs, validation, and consistent error responses the front end can rely on
  - Persistence with Entity Framework Core behind a service layer
  - Lab: implement and test the API endpoints against the agreed contract
- React Foundations
  - Components, props, and JSX: thinking in a component tree
  - State and events with the useState hook
  - Structuring a React project so it scales past the demo stage
  - Lab: build the application's core UI as components with local state and mock data

Day two: the seam, and shipping the whole thing

- Wiring Front End to Back End
  - Fetching data with fetch and useEffect, and where client-side data libraries fit
  - Loading, error, and empty states as first-class UI
  - CORS: what it is actually for and how to configure it correctly
  - Lab: replace the mock data with live API calls, handling every failure state deliberately
- Forms, Routing, and Full CRUD
  - Controlled forms in React and validating on both sides of the wire
  - Client-side routing with React Router
  - Keeping client state and server state honestly in sync
  - Lab: complete the full create, edit, and delete workflow through the UI
- Authentication and Deployment
  - JWT authentication: issuing tokens in .NET, storing and sending them from React
  - Protecting API endpoints and hiding UI the user cannot use
  - Building for production and deployment options for the combined application
  - Lab: add login end to end and deploy the finished application

Extended Version

The three-day version keeps the same gradient and adds depth across the stack:

- TypeScript on the front end, and generating typed API clients from OpenAPI
- Testing the stack: API integration tests and React component tests
- Production hardening: configuration, logging, and containerizing both halves
- A capstone: add a complete new feature to the application, from database column to React screen