



## Programming in Blazor

Tecnologias de Informação - Desenvolvimento / Programação

Promoção: **Curso com 15% de desconto**Aproveite já!

Live Training ( também disponível em presencial )

- **Localidade:** Imprimir Curso
- **Data:** 11 Nov 2024
- **Preço:** 2390 € ( Os valores apresentados não incluem IVA. Oferta de IVA a particulares e estudantes. )
- **Horário:** Laboral das 9h00 - 17h00
- **Nível:**
- **Duração:** 35h

---

### Sobre o curso

Este curso tem como objetivo dotar os participantes das competências, conhecimentos e técnicas necessárias à utilização do framework ASP .Net Core 5.0 Blazor.

---

### Destinatários

Este curso destina-se a todos os Developers que desejam utilizar o BLAZOR para facilitar o seu processo de criação de Web Apps.

---

### Objetivos

No final da ação de formação os participantes deverão estar aptos a:

- Desenvolver uma Web App em WebAssembly(Server and Cliente)
  - WebPage, Components, Local Storage, Offline, Events
- Compreender os fundamentos e conceitos do Blazor, PWA, SPA
- Aceder a recursos remotos por WebApi, gRPC, SignalR
- Criar Layouts para as paginas e componestes
- Efectuar o deployment de uma Blazor APP

- Compreender e implementar segurança OAUTH
  - Health Checks and Swagger Open Api Documentation
- 

## Pré-requisitos

Conhecimentos de C#

---

## Programa

- Introduction
- Creating a Blazor layout
- Components
- Render trees
- Templating components with RenderFragments
- Routing
- Forms
- Component libraries
- JavaScript interop
- Dependency injection
- Injecting dependencies into Blazor components
- Dependency lifetimes and scopes
- Globalization and Localization
- WebApi, SignalR and gRPC
- Deployment Blazor App – Server and Local

### Introduction

- What is Blazor?
- What is WebAssembly?
- Blazor hosting models
- Installing Blazor
- Creating a new project
- Creating a page
- Layouts

### Creating a Blazor layout

- Using a layout
- Nested layouts

## Components

- Creating a component
- One-way binding
- Literals, expressions, and directives
- Directives
- Component events
- Browser DOM events
- Two-way binding
- Binding directives
- Cascading values
- Cascading values by name
- Cascading values by type
- Overriding cascaded values
- Code generated HTML attributes
- Capturing unexpected parameters
- Replacing attributes on child components
- Component lifecycles
- Multi-threaded rendering
- Thread safety using InvokeAsync

## Render trees

- Incremental RenderTree proof
- Optimising using @key

## Templating components with RenderFragments

- Creating a Control
- Passing data to a RenderFragment
- Using @typeparam to create generic components
- Passing placeholders to RenderFragments

## Routing

- Defining routes
- Route parameters
- Constraining route parameters
- Optional route parameters
- 404 – Not found
- Navigating our app via HTML
- Navigating our app via code

- Detecting navigation events

## **Forms**

- Editing form data
- Descending from InputBase
- Validation
- Handling form submission
- EditContext, FieldIdentifiers, and FieldState
- Accessing form state
- Writing custom validation

## **Component libraries**

### **JavaScript interop**

- JavaScript boot process
- Calling JavaScript from .NET
- Updating the document title
- Passing HTML element references
- Calling .NET From JavaScript
- Lifetimes and memory leaks
- Type safety
- Calling static .NET methods

### **Dependency injection**

#### **Injecting dependencies into Blazor components**

#### **Dependency lifetimes and scopes**

- Transient dependencies
- Singleton dependencies
- Scoped dependencies
- Comparing dependency scopes
- Component scoped dependencies

#### **Globalization and Localization**

#### **WebApi, SignalR and gRPC**

#### **Deployment Blazor App – Server and Local**