Dezvoltarea Aplicațiilor Web utilizând ASP.NET Core MVC
Curs 1

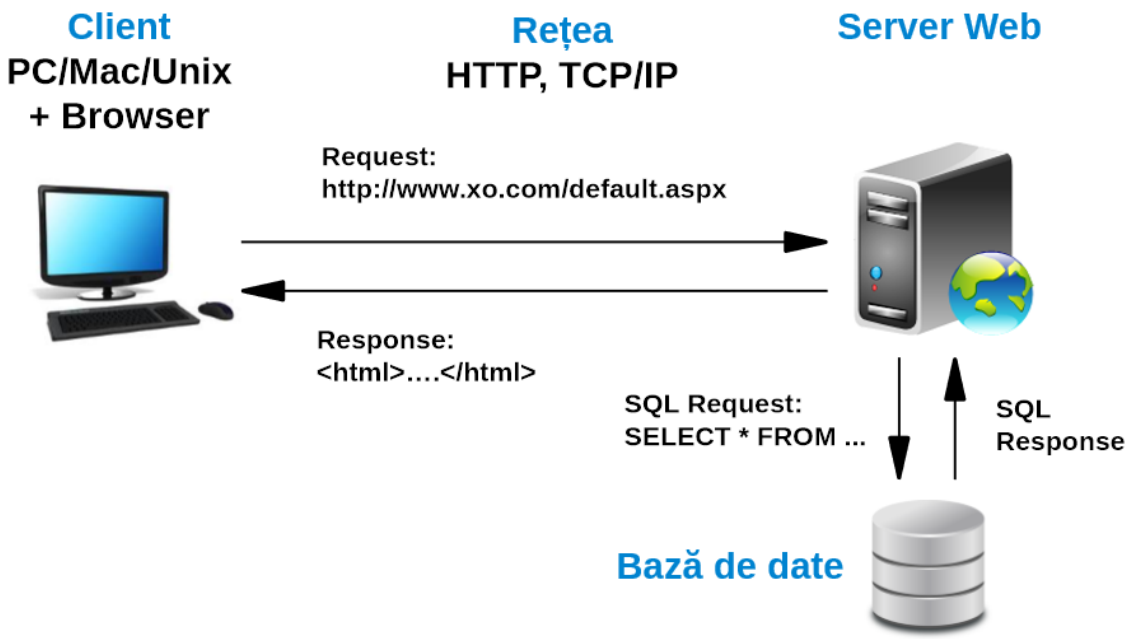
Cuprins

Ce este o aplicatie Web

O **aplicatie web** este o aplicatie care ruleaza intr-o arhitectura Client-Server bazata pe: **protocolul HTTP** (HyperText Transfer Protocol)**, TCP/IP** (Transmission Control Protocol/Internet Protocol); **browser Web**; **Server Web**.

Aplicatiile web sunt executate intr-un browser web si implementate folosind tehnologii precum: PHP, ASP, PYTHON, HTML, CSS, JAVASCRIPT, etc.

Arhitectura Web



Avantajele aplicatiilor Web

- Sunt independente de sistemul de operare
- Nu necesita instalare
- Actualizari foarte usor de facut deoarece modificarile se fac intr-un singur loc – pe server, ele propagandu-se pentru toti utilizatorii (in cazul aplicatiilor client-server clasice, interfata cu utilizatorul este asigurata prin intermediul unui program client instalat pe calculatorul fiecarui utilizator, orice modificare necesitand reinstalarea aplicatiei pentru fiecare utilizator in parte)

Introducere in ASP.NET

- **ASP.NET** este un framework Web, open source, conceput si dezvoltat de Microsoft, care face parte din framework-ul .NET
- Este utilizat pentru a dezvolta aplicatii si servicii web
- Ofera o integrare foarte buna a codului HTML, CSS, JavaScript
- Ofera posibilitatea crearii paginilor dinamice, prin intermediul sintaxei Razor, utilizand C#, HTML, CSS si JavaScript
- ASP.NET furnizeaza librarii specifice dezvoltarii aplicatiilor web, cum sunt cele pentru sistemul de autentificare (autentificare in mai multi pasi, autentificare utilizand componente 3^{-rd} party, etc), cele pentru crearea si prelucrarea bazei de date, cele pentru lucrul cu fisiere, etc
- Este construit pe baza **CLR (Common Language Runtime) – ruleaza cod compilat** si permite utilizatorilor sa scrie cod folosind orice limbaj acceptat de framework-ul .NET

Ce este CLR – Common Language Runtime?

Se ocupa de executia programelor C#. Atunci cand este **compilat** un program C# rezultatul compilarii **nu este un cod executabil.** In locul acestuia se produce un fisier care contine un tip de cod apropiat de codul masinii, numit limbaj intermediar sau pe scurt **IL – Intermediate Language**. Acest proces de compilare este realizat de un compilator C# (de exemplu, compilatorul din **Visual Studio**). Rezultatul acestui proces este un fisier cu extensia **.exe** sau **.dll**, care contine acest cod intermediar.

Prin intermediul unui compilator denumit **JIT – Just in Time**, **CLR** (Common Language Runtime) transforma codul intermediar in cod executabil.

Framework-ul .NET

- Este compatibil cu peste 20 de limbaje diferite, cele mai populare fiind C#, C++, Visual Basic, F#. In prezent, limbajul cel mai des folosit ramane C# deoarece restul limbajelor de programare au dezvoltat librarii similare cu .NET framework

- Pune la dispozitie o colectie impresionanta de clase, organizate in biblioteci

- Este construit din doua entitati importante:

1. **Common Language Runtime (CLR)**

   - mediul de executie al programelor fiind cel care se ocupa cu managementul si executia codului scris in limbaje specifice .NET

2. **Base Class Library**

   - Este biblioteca de clase .NET
   - Acopera o arie larga a necesitatilor de programare, incluzand **interfata cu utilizatorul**, **protocoale de conectare cu baza de date**, **accesarea datelor**

ASP.NET Core

- Este noul framework creat de Microsoft
- A fost conceput pentru a functiona indiferent de sistemul de operare (Windows, MAC, Linux), fiind astfel mult mai flexibil si rapid
- ASP.NET Core este un framework nou, nefiind o continuare a framework-ului ASP.NET 4.6
- ASP.NET Core aduce in plus securitate, scade costurile si imbunatateste performanta
- Se pot dezvolta foarte multe tipuri de aplicatii: desktop, web, mobile, machine learning, micro-servicii, jocuri
- Hostare mult mai rapida in cloud

Introducere in C#

- Este un limbaj compilat
- Este un limbaj orientat pe obiecte
- Permite dezvoltarea de aplicatii industriale, durabile
- A fost conceput ca un concurent pentru limbajul Java
- Este derivat al limbajului C++

Limbaj compilat vs limbaj interpretat

Limbaj compilat – codul scris, numit cod sursa, este translatat de catre compilator intr-un cod apropiat de nivelul masinii, numit **cod executabil**. Atunci cand aplicatia trece de compilare fara erori de sintaxa se va produce codul executabil, iar aplicatia va putea fi rulata. (exemplu limbaje compilate: C, C++, Rust, Go, etc).

Limbaj interpretat (la rulare) – cu ajutorul unui interpretor specific limbajului, fiecare linie de cod este interpretata chiar in momentul rularii, fiind preschimbata imediat in cod masina si executata (exemplu limbaje interpretate: PHP, Ruby, Python).

Limbajele compilate ofera performanta mai mare si optimizare, dar necesita un pas suplimentar de compilare.

Limbajele interpretate sunt mai usor de utilizat pentru dezvoltarea rapida si feedback al erorilor, dar sunt mai lente la execuție.

C# este considerat un **limbaj compilat** deoarece codul sursa este transformat intr-un cod intermediar printr-un proces de compilare, dar C# implică si un proces de **interpretare si compilare JIT** (**VEZI** definitia), unde codul intermediar este executat și transformat în cod mașină de către CLR la momentul execuției.

In acest mod, C# este considerat hibrid, ducand la cresterea portabilitatii si performantei, deoarece codul intermediar poate rula pe orice sistem care are instalată platforma .NET.

In acelasi timp, copilatorul JIT optimizeaza executia pe sistemul hardware al masinii pe care ruleaza.

Ciclul de viata al unei pagini Web

Paginile ASP.NET ruleaza pe server-ul Web Microsoft IIS (Internet Information Server). In urma prelucrarii pe server rezulta o pagina web HTML, care este trimisa catre browser.

Ciclul de viata al unei pagini Web ASP.NET are urmatorii pasi:

- **Page request (accesarea paginii)** – acest pas se intampla inaintea ciclului de viata, atunci cand o pagina este ceruta serverului
- **Start** – in acest stadiu se incarca proprietatile paginii, cum ar fi request-ul si raspunsul, dupa care se identifica tipul acestora (**GET – cerere resurse, POST – trimiterea de informatii catre server**)
- **Initialization (initializare)** – in acest pas se initializeaza directivele si controalele si se aplica codul din Master Page
- **Load (incarcarea)** – in aceasta faza daca cererea este de tip **postback**, controalele sunt incarcate cu informatii
- **Evenimentele Postback** – daca cererea este de tip postback se executa codul aferent. Dupa executia codului se aplica sistemele de validare
- **Rendering (ex: afisarea paginii)** – in acest pas se construieste pagina finala pe server, care va fi afisata in browser
- **Unload (eliberarea memoriei)** – dupa ce pagina a fost trimisa utilizatorului, resursele alocate pentru aceasta sunt eliberate