**Cap. 1. TEHNOLOGII WEB**

* 1. **Scurt istoric**

Lucrarea de față își propune să facă o sinteză și o prezentare pe scurt a diverselor tehnologii web, deoarece la ora actuală sunt foarte multe medii de dezvoltare a paginilor web.

Arhitectura unei aplicații informatice folosită în interiorul unei organizații a trecut prin mai multe etape:

1. Etapa Mainframe

2. Etapa Client-Server

3. Etapa Web Based (aplicații web sau web-enabled)

1. Etapa Mainframe

Caracteristicile acestei etape erau:

* era un calculator de tip Mainframe;
* erau mai multe terminale neinteligente (de cele mai multe ori doar niște tastaturi) conectate la acesta;
* aplicațiile erau centralizate (rulau pe sistemul mare);
* de cele mai multe ori nu existau conexiuni în/din afara organizației;
* securitatea nu era o problema majora.

1. Etapa client-server

Caracteristicile acestei etape sunt:

* era un calculator sau mai multe de tip minicalculator pe care rulează serverul;
* erau mai multe calculatoare pe care rulau programele client (programe dedicate furnizate o data cu aplicația);
* exista o rețea locală prin care comunică mașinile respective;
* în această etapă apare și posibilitatea calculului distribuit (distributed computing) favorizat de vitezele din ce în ce mai mari de comunicație.
* softul de baza (sistemele de operare, sisteme de gestionare a bazelor de date) se adaptează și el pentru lucrul în rețea și calcul distribuit.
* în prima perioada a etapei client – server aplicațiile erau dezvoltate individual pentru fiecare departament, iar ele nu erau interconectate, dar puteau fi accesate prin rețea de pe fiecare calculator individual. Din aceasta cauza problemele de conectivitate si securitate nu erau critice, cele mai importante fiind problemele de performanta.
* spre finalul acestei etape a apărut sistemul informatic integrat de întreprindere (Enterprise Resource Planning), ceea ce dus la necesitatea interconectării departamentelor și integrării aplicațiilor departamentelor într-un tot unitar. În acest moment s-a pus problema securității și a vitezei de conectare.

În această etapă au apărut problemele noi legate de:

* Conectivitate
* Performanta
* Securitate

1. Etapa web-based

În această etapă accentul s-a pus pe modul de accesare a aplicației. Din acest punct de vedere avem:

* accesarea aplicației printr-un client standard (browserul, de ex: Internet Explore, Chrome, Firefox, etc);
* accesarea aplicație se face de cele mai multe ori din afara locației serverului pe care se află acea aplicație;
* de cele mai multe ori comunicația se face nu prin canale private/dedicate ci prin conexiunea publică la Internet.

În figura 1.1. se prezintă o arhitectură tipică. Nu este necesar ca in toate locațiile să existe servere pentru conexiune deoarece acestea costă foarte mult si de cele mai multe ori nu se justifică investiția. La existența / absența serverului din interiorul unei locații se va avea in vedere

* numărul de utilizatori dintr-o anumită locație;
* lățimea de bandă disponibilă între locația respectivă și locația principală;
* timpul de răspuns necesar de asigurat utilizatorilor.

*Fig. 1.1. Structura generală a unei arhitecturi tipice.*

Utilizatori la distanță



Server aplicații

Server baze de date

Locația 1

Locația 2

Locația 3



Deoarece în această etapă datele pot ajunge în afara rețelei pentru a crește securitatea lor, este indicată o metodă de codare a lor. De asemenea trebuie ținut cont că fiind mai mulți utilizatori pot folosi browser-e diferite de aceea aplicația trebuie dezvoltată astfel încât toate funcționalitățile să poată rula pe toate aceste programe de navigare pe Internet. Totodată conexiunea fiind din locații diferite timpul de răspuns trebuie să fie cât mai mic, totodată timpul de accesare trebuie să fie minim.

* 1. **Noțiuni introductive**

Pentru afișarea unei pagini web utilizați diverse browser-e precum Chrome, Internet Explorer, Firefox, etc. Browser-ul acesta va aranja informația pe baza unei descrieri găsite într-un fișier scris în format Hypertext Markup Language (HTML) sau alt limbaj de programare.

Tehnologiile utilizate în aplicațiile web sunt limbajele de programare utilizate la crearea de site-uri web complexe, interactive, al căror concept presupune stăpânirea cât mai detaliată a unor limbaje de scripting sau de programare web și baze de date.

În categoria tehnologii Web se pot include:

1. o multitudine de limbaje de programare web, cum ar fi **PHP, JavaScript, ASP**, etc.

2. iar ca baze de date pot fi utilizate **MySQL, MsSQL, Access, ORACLE**, etc.

Ce aplicații putem dezvolta cu tehnologiile web:

* portal-uri
* forum-uri
* magazine virtuale
* formulare de înscriere

Sunt două tipuri de tehnologii web ce pot fi utilizate:

* tehnologii web **Open Source** ce pot fi utilizate de oricine, fără a fi nevoit sa cumpere licență;
* tehnologii web la care trebuie cumpărată licență pentru a putea fi utilizată.

Dintre cele mai utilizate tehnologii pentru dezvoltarea aplicațiilor web putem aminti:

1. **HTML** – a apărut în anii 1990. Pentru a putea publica unitar informația a apărut necesitatea unui limbaj în care să se poată „așeza” această informație în pagină. Acest limbaj de programare (**html**) a determinat dezvoltarea spectaculoasă a Internetului.

În cadrul acestui limbaj de programare sunt folosite marcaje în cadrul documentului care indică browser-ului cum să afișeze conținutul util al fișierului

De exemplu cea mai simpla pagină web poate fi scrisă in limbaj HTML, utilizând programul “Notepad” integrat în sistemul de operare Windows. În acest program se va descrie doar structura și semantica unui document și nu vor fi descrise aspectele elementelor din pagină. Acestui fișier i se va atașa un alt fișier **cascading style sheets** având extensia CSS. În acest fișier vor fi descrise elementele de formatare a diverselor obiecte incluse în pagina web dezvoltată.

Spre deosebire de HTML, abrevierea HTTP provine de la **Hypertext Transfer Protocol,** și este un protocol de comunicare, construit peste suita de protocoale TCP/IP. HTTP definește lucruri precum adresele URL. Adresele URL definesc calea către o resursă de pe internet. Cu aceste adrese URL, protocolul HTTP asociază un set de operații, cum ar fi: ***GET, POST, UPDATE sau DELETE***. În momentul în care se scrie o adresă în câmpul de adresă al browser-ului utilizat, autorizați broser-ul să facă o solicitare de tip ***GET*** către adresa introdusă. Aceasta este cea mai folosită metodă, fiind utilizată atunci când serverului i se cere o resursă.

Când completați un formular pe o pagină web și apăsați butonul „Trimite”, de obicei se va face o solicitare de tip ***POST***. Aceasta colectează intrarea dumneavoastră și o va trimite către server. Metoda ***PUT*** este folosită pentru a depune documente pe server, fiind inversul metodei GET.

Hypertext Transfer Protocol (HTTP) este metoda cea mai des utilizată pentru accesarea informațiilor în Internet care sunt păstrate pe servere World Wide Web (WWW). Protocolul HTTP este un protocol de tip text, fiind protocolul "implicit" al WWW. Adică, dacă un URL nu conține partea de protocol, aceasta se consideră ca fiind HTTP, aceasta presupune că pe calculatorul destinație rulează un program care înțelege protocolul. Fișierul trimis la destinație poate fi un document HTML, un fișier grafic, fișier tip: sunet, animație sau video, de asemenea un program executabil pe server-ul respectiv sau și un editor de text.

Un alt tip de operație este ***REST*** înseamnă „conversia” unui serviciu aflat pe un server în lumea exterioară în limbajul protocolului HTTP. Rezultatul unei procesări ***REST*** conduce la obținerea unei reprezentări a unei resurse. Rezultatul unei procesări conduce la obținerea unei reprezentări a unei resurse *resursă Web* utilizator având cont în cadrul unui sistem, mesaj al unei persoane, fotografie, flux de știri, componentă software, set de date (dataset), model 3D. Sau rezultatul unei procesări conduce la obținerea unei reprezentări a unei resurse *reprezentare pe baza unui format de date* de tip text sau binar, de exemplu formate deschise: HTML, CSV, PNG, SVG, PDF etc. Însă din cauza procedeului complicat metoda este mai puțin utilizată.

Metoda ***DELETE*** șterge resursa specificată. Metoda ***UPDATE*** actualizează resursa specificată.

Comenzi http (metode)

Începând cu versiunea HTTP 1.1, există 8 comenzi (metode) HTTP care sunt acceptate pe scară largă. Mai jos sunt date cele opt comenzi:

1. GET

2. HEAD

3. POST

4. CONNECT

5. DELETE

6. OPTIONS

7. PUT

8. TRACE

Alte trei comenzi de asemenea, acceptate în specificația HTTP 1.1sunt date mai jos, dar lipsa de suport le face mai puțin utilizate.

1. LINK

2. UNLINK

3. PATCH

Comanda HEAD este identică cu comanda GET din toate punctele de vedere, cu o singură diferență: răspunsul nu trebuie să aibă un secțiunea **body**. Toate informațiile solicitate sunt returnate în secțiunea **antet** a răspunsului.

Comparație între metodele GET și POST

Metoda **GET** înseamnă preluarea oricărei informații (sub forma unei entități) este identificată de Request-URI (Cerere-URI-ului). Dacă Request-URI se referă la un proces de producere a datelor, este vorba despre datele produse care sunt returnate ca entitate în răspuns și nu textul sursă al procesului, cu excepția cazului în care textul respectiv este rezultatul procesului.

Metoda **POST** este utilizată pentru a solicita ca serverul de origine să accepte entitatea inclusă în cerere ca o nouă subordonată a resursei identificate de Request-URI în linia de solicitare.

Metoda **POST** este conceput pentru a permite o metodă uniformă să acopere următoarele funcții:

- Adnotarea resurselor existente;

- Postarea unui mesaj pe un forum, grup de știri, listă de corespondență, sau un grup similar de articole (informații similare);

- Furnizarea unui bloc de date, cum ar fi rezultatul trimiterii unui formular, la un proces de gestionare a datelor;

- Extinderea unei baze de date printr-o operațiune anexată.

Funcția reală îndeplinită prin metoda **POST** este determinată de server și depinde, de obicei, de Request-URI. Entitatea postată este subordonată acelui URI în același mod în care un fișier este subordonat unui director care îl conține, un articol de știri este subordonat unui grup de știri la care este postat sau o înregistrare este subordonată unei baze de date.

Acțiunea efectuată prin metoda **POST** poate să nu conducă la o resursă care poate fi identificată de un URI. În acest caz, pot apărea diverse coduri de eroare, (de exemplu: 200 (OK) sau 204 (Fără conținut)) reprezintă starea de răspuns adecvată, și pot fi două cazuri:

1. răspunsul include o entitate care descrie rezultatul
2. nu o entitate care descrie rezultatul.

Diferențele dintre GET și POST

1. Metoda **GET** este destinată obținerii (preluării) datelor, în timp ce **POST** poate implica orice, cum ar fi stocarea sau actualizarea datelor, sau comandarea unui produs sau trimiterea unui e-mail

2. Când sunt utilizate pentru trimiterea datelor dintr-un formular, **GET** atașează aceste date la adresa URL a cererii, după caracterul „**?**”, ca o secvență de perechi „nume = valoare”, separate prin caracterul „&” sau „**;**”. Pe de altă parte, datele din formular trimise de **POST** pot fi codate fie ca mai sus (utilizând *application/x-www-form-urlencoded*), sau în corpul mesajului, (codat ca *multipart/form-data*).

3. O solicitare **POST** necesită o transmisie suplimentară pentru a prelua corpul mesajului, în timp ce un **GET** cererea permite ca datele trimise prin URL să fie procesate imediat.

Codul de stare

Un număr întreg de trei cifre, în care prima cifră identifică categoria generală de răspuns:

• **1xx** - indică doar un mesaj informativ

• **2xx** - indică succesul de un fel

• **3xx** - redirecționează clientul către o altă adresă URL

• **4xx** - indică o eroare din partea clientului

• **5xx** - indică o eroare din partea serverului

Cele mai frecvente coduri de stare sunt:

• **200 OK** - cererea a reușit, iar resursa rezultată (de exemplu, fișierul sau ieșirea scriptului) este returnat în corpul (body) mesajului.

• **404** - Necunoscut - resursa solicitată nu există.

• **301** - mutat permanent

• **302** - mutat temporar

• **303** -resursa s-a mutat la o altă adresă URL (dată de Locație: antet de răspuns) și ar trebui să fie preluat automat de către client. Aceasta este adesea folosit de un script CGI pentru redirecționarea browser-ului către un fișier existent.

• **500** -Server Error - o eroare de server neașteptată. Cea mai frecventă cauză este o parte a serverului scriptul care are sintaxă greșită, eșuează sau altfel nu poate rula corect.

1. **XHTML** - reprezintă prescurtarea de la eXtensible HyperText Markup Language (denumirea oficiala a standardului). Este o actualizare, deci o versiune mai nouă a limbajului de programare HTML
2. **CSS - Fişierul CSS (Cascading Style Sheet - foi de stil în cascadă) –** a fost dezvoltat pentrua ușura lucru cu fișierele html, în sensul de a nu formata fiecare element din acest fișier, ceea ce ar duce la un număr mare de linii de comandă și timp îndelungat de particularizare a aplicației web. Un fișier CSS este un fișier text cu extensia ".css", definind stiluri pentru paginile html. Fişierele CSS permit stilizarea în bloc a documentelor html cu un efort semnificativ mai mic decât în cazul stilizării elementelor de pagină în cadrul fiecărui document html în parte, prin intermediul atributelor tag-urilor.
3. **CGI** - Common Gateway Interface - sunt acele script-uri scrise în orice limbaj de programare indiferent dacă sunt compilate sau interpretate, script-uri care sunt executate pe server.
4. **ISAPI** - Internet Server API – reprezintă o alternativa celor de la Microsoft la CGI (Common Gateway Interface), rulabilă din păcate doar pe platforme Windows. Acesta poate fi scris în orice limbaj cu suport pentru fișierele cu extensia “.dll”.
5. **JAVASCRIPT** - Un alt limbaj relativ simplu de dezvoltare a paginilor web este **JavaScript** care oferă și posibilitatea dezvoltării de pagini web interactive spre deosebire de codul simplu HTML. Deoarece limbajul **JavaScript** a fost utilizat pe scară largă din el a derivat limbajul **JSON.** Acesta are sintaxa pentru șiruri, tablouri, numere și dicționare. În acest limbaj au fost excluse sintaxele pentru bucle, dacă instrucțiunile și funcțiile sunt excluse. Limbajul **Javascript** este un limbaj dezvoltare a scripturilor dezvoltat inițial de Netscape, permițând scrierea de secvențe de program care se execută la apariția unui eveniment utilizator.
6. **FLASH** - este un alt mediu de lucru dezvoltat de compania Macromedia; cu ajutorul FLASH-ului se poate realiza un design de calitate, conținut interactiv, animații profesionale.
7. **MYSQL** - este o componentă cheie a limbajului PHP, fiind componenta pentru gestionarea bazelor de date relaționale.
8. **PHP** a apărut în anul 1994 iar **ASP** a apărut în anul 1996. Acestea sunt două limbaje puternice care au adus o schimbare în modul de realizarea a design-ul aplicațiilor web.

Cele două tehnologii au o serie de asemănări:

• ambele sunt interpretate,

• ambele generează scripturi care pot fi inserate în limbajul html, date de tip text, etc.

• Ambele limbaje oferă suport pentru lucrul cu diverse baze de date (de ex: MySQL, MsSQL, PostgreSQL, Oracle).

Scripturile PHP sunt legate de un server web și necesită scrierea interogării bazei de date explicite pentru a genera un conținut dinamic. Chiar și aceste interogări sunt scrise într-un mod specific unui anumit cod specific bazei de date utilizate. În PHP, programatorul de aplicații scrie interogările SQL și le încorporează direct în script, amestecând prezentarea și logica în același proces (cod). Nu se acordă asistență directă pentru gestionarea acumulării de componente și a ciclului de viață, gestionarea sesiunilor clientului, colectarea conexiunilor de baze de date, persistența, gestionarea tranzacțiilor, autentificarea și controlul accesului. Pe de altă parte, toate acestea pot fi realizate printr-un server de aplicații.

Sunt foarte puține cazurile în care tehnologiile **PHP** și **ASP** sunt utilizate în varianta simplificată, preponderent sunt în combinație cu bazele de date. Principalul dezavantaj al celor doua limbaje este viteza, acestea fiind lente deoarece fiecare accesare presupune procesarea şi interpretarea lor la fiecare accesare. La o accesare și interpretare nu poate fi salvat rezultatul pentru ca la următoarea accesare să fie aduc rezultatul și astfel să se reducă timpul.

1. **JSP - Java Server Pages** – este o altă tehnologie de pentru designul al aplicațiilor web ce permite crearea acestor aplicații independente de platformă. Principalul avantaj îl reprezintă administrarea facilă, independentă de platformă, separarea logicii aplicației de partea de interfață utilizator.
2. **ASP.NET** – este cea mai nouă tehnologie dezvoltată de firma Microsoft, păstrează unele caracteristici de compatibilitate cu tehnologia **ASP**, însăse încearcă o îmbunătățire a tehnologiei **ASP.**

Cele mai multe tehnologii pentru dezvoltarea paginilor web necesită scrierea liniilor de cod. Însă sunt și aplicații cu cu interfață grafică în care se pot dezvolta pagini web (de ex: FrontPage, Dreamwever,etc). Aceste aplicații cu interfață grafică lucrează în spate cu diverse tehnologii amintite mai sus.

De exemplu FrontPage lucrează cu tehnologia html, Dreamwever poate lucra în diverse tehnologii precum: html, asp, php.

Fiecare din aceste tehnologii prezintă diverse avantaje şi dezavantaje, dintre care amintim mai jos câteva, dar au un singur scop crearea unor aplicații web:

* interactive;
* securizate;
* cu timp de răspuns cât mai redus;
* să ofere o interfață grafică plăcută și intuitivă.

Ultimele tendințe

**SOA** sau **Arhitectura orientată către servicii** este noua tendință în **Enterprise Application Integration (EAI)**. SOA oferă metode pentru dezvoltarea și integrarea aplicațiilor centrate în jurul unui set de servicii interoperabile. Aceste servicii constau din unități funcționale neasociate care nu au apeluri reciproce încorporate în ele. Sincronizarea se face printr-un limbaj executabil, **BPEL (Business Process Execution language)**.

În termeni practici, codul BPEL este generat printr-un front-end grafic, care mapează diagramele **BPMN (Business Process Modeling Notation)** în codul **BPEL** real. Codul **BPEL** este executat în cadrul unui motor BPEL. Implementările motoarelor **BPEL** au fost furnizate de diverse companii, începând cu **ActiveBPEL** a **Active Endpoints** și terminând cu **WebSphere Process Server al IBM**.

O notă specială despre interacțiunea umană.

Specificațiile anterioare **BPEL** aveau probleme grave în încorporarea actelor umane în procese. În cele din urmă, în iunie 2007, **Active Endpoints, Adobe, BEA, IBM, Oracle și SAP** au publicat în comun specificațiile **BPEL4People și WSHumanTask**, care descriu modul în care interacțiunea umană poate fi implementată în procesele **BPEL.**