Arhitectura Client Server

Arhitectura Client Server este un model de calcul în care serverul găzduiește, furnizează și gestionează majoritatea resurselor și serviciilor care urmează să fie consumate de client. Acest tip de arhitectură are unul sau mai multe computere client conectate la un server central printr-o conexiune la rețea sau la internet. Acest sistem partajează resurse de calcul. Arhitectura client/server este cunoscută și ca model de calcul în rețea sau rețea client/server, deoarece toate cererile și serviciile sunt livrate printr-o rețea.

Caracteristici:

* Mașinile client și server au nevoie de cantități diferite de resurse hardware și software.
* Mașinile client și server pot aparține unor furnizori diferiți.
* Scalabilitate orizontală (creșterea mașinilor client) și scalabilitate verticală (migrare la un server mai puternic sau la o soluție multiserver)
* O aplicație client sau server interacționează direct cu un protocol de nivel de transport pentru a stabili comunicarea și pentru a trimite sau primi informații.
* Protocolul de transport utilizează apoi protocoale de nivel inferior pentru a trimite sau a primi mesaje individuale. Astfel, un computer are nevoie de un teanc complet de protocoale pentru a rula fie un client, fie un server.
* Un singur computer de tip server poate oferi mai multe servicii în același timp; este necesar un program server separat pentru fiecare serviciu.

Avantaje:

* Organizațiile caută adesea oportunități de a menține concurența în servicii și calitate pentru a-și susține poziția pe piață cu ajutorul tehnologiei în care modelul client/server are un impact eficient. Implementarea computerelor client/server într-o organizație va crește pozitiv productivitatea prin utilizarea de interfețe de utilizator rentabile, stocare îmbunătățită a datelor, conectivitate vastă și servicii de aplicații fiabile. Dacă este implementat corespunzător, este capabil să îmbunătățească comportamentul organizațional cu ajutorul lucrătorului cu cunoștințe - care poate manipula datele și poate răspunde la erori în mod corespunzător.
* Îmbunătățirea schimbului de date: Datele sunt reținute de procesele obișnuite de afaceri și manipulate pe un server sunt disponibile pentru utilizatorii desemnați (clienți) printr-un acces autorizat. Utilizarea Structured Query Language (SQL) acceptă accesul deschis din toate aspectele clientului și, de asemenea, transparența în serviciile de rețea arată că date similare sunt partajate între utilizatori.
* Integrarea Serviciilor: Fiecărui client i se oferă posibilitatea de a accesa informațiile corporative prin interfața desktop, eliminând necesitatea de a se autentifica într-un mod terminal sau într-un alt procesor. Instrumentele desktop, cum ar fi foile de calcul, prezentările Power Point etc. pot fi folosite pentru a trata datele corporative cu ajutorul serverelor de baze de date și de aplicații rezidente în rețea pentru a produce informații semnificative.
* Resurse partajate între diferite platforme: Aplicațiile utilizate pentru modelul client/server sunt construite indiferent de platforma hardware sau de fundalul tehnic al software-ului îndreptățit (S/W Sistem de operare) oferind un mediu de calcul deschis, forțând utilizatorii să obțină serviciile clienților și serverelor (bază de date, aplicație, servere de comunicații).
* Inter-operarea datelor: Toate instrumentele de dezvoltare utilizate pentru aplicațiile client/server accesează serverul de bază de date back-end prin SQL, un limbaj de acces și definire a datelor standard din industrie, util pentru gestionarea consecventă a datelor corporative. Produsele avansate de baze de date permit utilizatorului/aplicației să obțină o imagine combinată a datelor corporative dispersate pe mai multe platforme. Mai degrabă decât o singură platformă țintă, aceasta asigură integritatea bazei de date cu capacitatea de a efectua actualizări în mai multe locații, impunând considerații de calitate și recuperare.
* Capacitatea de procesare a datelor în ciuda locației: ne aflăm într-o eră care trece printr-o transformare a sistemelor centrate pe mașină în sisteme centrate pe utilizator. Sistemele centrate pe mașină, cum ar fi mainframe, aplicațiile mini-micro au platforme unice de acces și chei de funcționalitate, opțiuni de navigare, performanță și securitate erau toate vizibile. Prin intermediul client/server, utilizatorii se pot conecta direct la un sistem în ciuda locației sau tehnologiei procesoarelor.
* Întreținere ușoară: Deoarece arhitectura client/server este un model distribuit care reprezintă responsabilități dispersate între computere independente integrate într-o rețea, este un avantaj în ceea ce privește întreținerea. Este ușor să înlocuiți, să reparați, să faceți upgrade și să mutați un server în timp ce clienții rămân neafectați. Această neconștientizare a schimbării este numită încapsulare.
* Securitate: Serverele au un control mai bun de acces și resurse pentru a se asigura că numai clienții autorizați pot accesa sau manipula datele, iar actualizările serverului sunt administrate eficient.

Dezavantaje (comparativ cu rețelele peer-peer):

* Servere supraîncărcate: atunci când există solicitări frecvente simultane ale clienților, serverele se supraîncărcă grav, formând congestie de trafic. Dar într-o rețea P2P adăugarea mai multor noduri va crește lățimea de bandă, deoarece este calculată ca suma lățimilor de bandă ale fiecărui nod din rețea.
* Impactul arhitecturii centralizate: deoarece este centralizată, dacă un server critic eșuează, solicitările clientului nu sunt îndeplinite. Prin urmare, clientul/serverul nu are robustețea unei rețele P2P bune (resursele sunt distribuite între multe noduri).

Single Page Applications (SPAs) vs Server Sided Rendering (SSR)

**Ce este o aplicație cu o singură pagină (SPA)?**

O aplicație cu o singură pagină (SPA) este un termen general larg pentru aplicațiile care sunt redate atunci când clientul le solicită. SPA-urile sunt structurate ca o singură pagină HTML care nu are conținut preîncărcat. Conținutul este încărcat prin fișiere Javascript pentru întreaga aplicație și găzduit într-o singură pagină HTML. Fișierele Javascript găzduiesc toate datele referitoare la logica aplicației, interfața de utilizare și comunicarea cu serverul. Framework-urile și bibliotecile Javascript populare pentru construirea de SPA-uri includ React, AngularJS, Vue.js, Ember.JS și Svelte, printre alții.

Când utilizatorii navighează prin diferitele părți ale SPA, nu va exista niciun timp suplimentar de încărcare între diferitele elemente ale aplicației. Deoarece totul este încărcat pe partea clientului, echipele trebuie să țină cont de o gamă largă de clienți, asigurând în același timp o experiență de utilizator rapidă și fără întreruperi. Cu cadrele moderne, divizarea codului permite încărcarea unor elemente la cerere, ceea ce poate ajuta la eliminarea acestei probleme.

Avantaje:

* În timp ce încărcarea inițială poate fi mai lungă, odată ce aplicația s-a încărcat complet, nu va fi necesară încărcare suplimentară.
* Alegere bună pentru experiențele dinamice în care echipele au nevoie de un sentiment personalizat pentru experiența lor de utilizator.
* Echipele au foarte mult control asupra arhitecturilor lor și pot folosi cadre web modern.
* Poate fi folosit în tandem cu alte tehnologii.

Dezavantaje:

* Pe măsură ce aplicația crește în dimensiune și complexitate, aceasta poate afecta grav timpul inițial de încărcare, ceea ce poate duce la o deteriorare a experienței utilizatorului.
* Menținerea unui SEO bun este aproape imposibilă din cauza timpilor de încărcare și a lipsei conținutului inițial pe HTML.
* Fișierele mari pentru aplicații web complexe pot deveni dificil de întreținut și organizat.
* Provocările legate de abordarea SPA necesită soluții care pot fi costisitoare și consumatoare de timp.

**Ce este Server-Side Rendering (SSR)?**

Redarea pe server permite echipelor să ofere experiențe de conținut dinamice care pot fi personalizate. Sunt ideale pentru experiențe personalizate în care pot fi vizualizate modificările în direct ale datelor. Cu SSR, clienții primesc o pagină complet randată la cerere, în loc să fie nevoiți să aștepte câteva secunde pentru a se încărca anumite elemente. Redarea are loc pe server înainte de a le transmite browserului. Când conținutul este solicitat pe client, datele sunt preluate dintr-o bază de date sau CMS pe măsură ce utilizatorul navighează pe pagină. Încărcarea totul la cerere face ca procesul să fie mai lent, dar asigură că conținutul este actualizat și că orice modificări sunt disponibile live. Unele elemente pot fi stocate în cache, cum ar fi activele și fișierele CSS și chiar unele pagini randate de server, dar, de obicei, datele sunt extrase direct din baza de date la cerere.

Site-urile randate pe partea de server (sau aplicațiile randate pe partea serverului (SRA)) sunt alegeri excelente pentru conținutul care este sensibil la timp și aplicațiile care se bazează pe cantități mari de interacțiune a utilizatorului. Cu SSR, personalizarea este mult mai ușoară și poate fi o opțiune bună pentru comerțul electronic. Cu SSR-urile, este important să vă asigurați că infrastructura dumneavoastră poate gestiona solicitările către servere și că serverele se pot scala cu ușurință pe măsură ce traficul continuă să crească.

Avantaje:

* SSR permite echipelor să creeze experiențe de conținut dinamice, personalizate, fără soluții care necesită multă muncă.
* Modificările la conținut sunt afișate instantaneu, spre deosebire de SSG-urile (Static Site Generator) în care echipele trebuie să reconstruiască site-ul pentru a vedea modificările la conținut.
* Site-urile SSR sunt agnostice de client, diferite de SPA-urile în care clienții pot determina timpul sau calitatea de încărcare a paginii.
* Este mai ușor să vă clasați bine pentru SEO cu SSR decât SPA-uri, oferind totuși experiențe personalizate.

Dezavantaje:

* SSR-urile necesită de obicei mai multe apeluri API către server.
* SSR-urile în mod implicit sunt adesea mai lente decât SPA-urile.