UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA

FACULTATEA DE MATEMATICǍ ŞI INFORMATICǍ

SPECIALIZAREA INFORMATICĂ ROMÂNĂ

LUCRARE DE LICENŢĂ/ DISERTAŢIE/ABSOLVIRE

DEZVOLTAREA APLICATIILOR WEB FOLOSIND REST

Conducător ştiinţific

Dr. Grigoreta Sofia Cojocar

Absolvent

Vlasie Daniel - Sandrino

**2021**

# Contents

[Contents 2](#_Toc68282569)

[1 Introducere 3](#_Toc68282570)

[2 Aplicatiile web 4](#_Toc68282571)

[2.1 Ce este o aplicatie web? 4](#_Toc68282572)

[2.2 Istoria aplicatiilor web 4](#_Toc68282573)

[2.3 Structura unui browser web 5](#_Toc68282574)

[2.4 Concepte de baza 6](#_Toc68282575)

[2.4.1 HTML 6](#_Toc68282576)

[2.4.2 Cookies 7](#_Toc68282577)

[2.4.3 Arhitectura client-server 7](#_Toc68282578)

[2.4.4 Securitate 8](#_Toc68282579)

[3 REST 10](#_Toc68282580)

[3.1 Ce este REST mai exact ? 10](#_Toc68282581)

[3.2 Client Server 11](#_Toc68282582)

[3.3 Lipsa starii 11](#_Toc68282583)

[4 Tehnologii Web 12](#_Toc68282584)

[4.1 React 12](#_Toc68282585)

[4.2 Angular 12](#_Toc68282586)

[5 Despre Aplicatie 13](#_Toc68282587)

[5.1 Cerinte 13](#_Toc68282588)

[5.2 Proiectare 13](#_Toc68282589)

[5.2.1 Diagrame de clase 13](#_Toc68282590)

[5.2.2 Diagrame de secventa 13](#_Toc68282591)

[5.2.3 Structura baza de date 13](#_Toc68282592)

[5.3 Implementare 13](#_Toc68282593)

[5.4 Manual de utilizare 13](#_Toc68282594)

[6 Concluzii 14](#_Toc68282595)

[7 Referinte 15](#_Toc68282596)

# Introducere

Aplicația pe care urmează să o prezint este o aplicație cu tematică gastronomică, care însumează o serie de atribute facil de uzitat. O primă condiție pentru a utiliza aplicația web creată de mine este autentificarea, fără de care nu este permis accesul. Autentificarea se poate realiza fie accesând contul deja existent, fie creând unul nou prin accessarea butonului „signup”, care va redirecționa utilizatorul spre paginea de creare a contului. În urma acestei acțiuni va fi afișată pagina principală unde găsim rețete culinare împărtășite de alți utilizatori. În mod prestabilit, aceste rețete se află sub următorul format: nume acompaniat de o ilustrație vizuală. Printr-un simplu click pe butonul aflat sub rețetă se accesează detaliile necesare pentru realizarea acesteia, precum: ingredientele, cantitățile și pașii de urmat.

În partea superioară a paginii, este situată bara de navigare, unde se regăsesc ,atât butonul pentru accesarea paginii principale, cât și numele de utilizator, alături de poza de profil și butonul de logout. Odată cu crearea contului personal, se obține posibilitatea de a împărtăși o rețetă proprie .Sunt necesari următorii câțiva pași: acționarea butonului din partea inferioară a paginii principale care permite completarea numelui rețetei, adăugarea imaginii descriptive, specificarea ingredientelor, a cantităților acestora și pașii de preparare. De asemenea, postările pot fi modificate sau șterse doar de către creator.

Această aplicație web îți permite conturarea unui profil culinar care conține postările personale, postările apreciate și rețetele marcate ca “gătite”. Profilul se poate personaliza prin adăugarea unei fotografii ce poate fi schimbată în mod frecvent printr-un simplu click al butonului specific. Aplicația contorizează, atât postările proprii ,cât și postările apreciate de utilizator și cele marcate ca fiind “gătite”. Posibilitatea de a aprecia sau de a marca „gătită„ o rețetă se datorează existenței celor două butoane, primul sub forma unei inimi, respectiv sub forma unor tacâmuri, care se colorează prin intermediul unui click.

# Aplicatiile web

## Ce este o aplicatie web?

O aplicatie web este o aplicatie software executata de un server web care raspunde unor cereri facute de anumite pagini web dinamice, realizate prin intermediul protocolului HTTP. O aplicatie web este compusa dintr-o colectie de mai multe scripturi, aflate pe un server web, ce interactioneaza cu baze de date sau cu alte surse de continut dinamic. Folosind interfata Internetului, aplicatiile web permit furnizorilor de servicii si clientilor sa partajeze si sa manipuleze informatii intr-o maniera independenta de platforma pe care ruleaza, ceea ce inseamna ca aplicatia poate fi mutata de pe o platforma pe alta cu modificari minime sau chiar deloc. Aplicatiile web folosesc o arhitectura distribuita pe mai multe niveluri. In principal exista un client, reprezentat de browserul web, un server web, unul sau mai multe servere ale aplicatiei si o baza de date. (1) Aplicatiile web sunt mult mai complexe decat par si totodata si decat alte tipuri de aplicatii software, datorita serverelor care pot rula de la distanta si a clientului, reprezentat de browserul web, ce poate fi accesat de pe diverse platforme, fara a necesita o preinstalare. Complexitatea acestora este data si de amestecul de limbaje de programare, precum PHP si JavaScript, dar si limbaje de formatare si continut, cum ar fi HTML si CSS, pe care aplicatiile il pot cuprinde.

## Istoria aplicatiilor web

Pentru o mai buna intelegere a ceea ce reprezinta aplicatiile web contemporane, sa analizam evolutia aplicatiilor web in ultimele decenii. Daca revenim la inceputul anilor 1990, se poate observa ca inca nu exista conceptul de aplicatii web si tot ceea ce ne putea oferi Internetul, referitor la acest subiect, erau pagini HTML statice, formate din simple documente text. Aceasta arhitectura purta numele de Web 1.0, fiind formata, in prima parte, de un server, ale carui singure atributii erau sa transmita date text catre client, cea de-a doua parte a acestei arhitecturi. Mai tarziu a devenit posibila adaugarea de imagini, fisiere video si audio, insa paginile web au ramas statice, neavand posibilitatea de interactiona in vreun fel cu acestea. In acest punct al dezvoltarii aplicatiilor web existau in jur de 100,000 de site-uri si 50 milioane de utilizatori. Dorinta de a dinamiza paginile HTML a dus la dezvoltarea limbajului de programare JavaScript, pe partea de client, in 1995. Cu ajutorul acestei noi tehnologii s-a putut realiza executia codului direct in browser, prezentarea diverselor elemente interactive, incluzand animatiile vectoriale, oferind astfel utilizatorilor posibilitatea de a interactiona cu noile pagini web create. In 1996 a fost introdus Macromedia Flash, un program care creeaza continut media interactiv si animat precum grafica video, imbunatatind considerabil animatiile. Acesta a fost momentul care a dus la dezvoltarea jocurilor video. Anul 1999 a adus la aparitia conceptului de aplicatie web in limbajul Java. Tot in aceasta perioada s-a facut si trecerea de la Web 1.0 la Web 2.0, ajungandu-se la peste 100 milioane de site-uri web si aproximativ un miliard de utilizatori. In 2005 a fost introdusa Ajax, o noua abordare a dezvoltarii de aplicatii web si aplicatii web asincrone, ce au permis utilizarea conceptului de “responsive web design”(RWD), ceea ce inseamna ca paginilie web pot fi randate bine pe diferite dispozitive si pe o varietate mare de dimensiuni de ferestre. Datorita noii tehnologii Ajax, a fost posibila trecerea de la paginile web statice la cele dinamice. Evolutia tehnologiei a dus ca, peste un deceniu, in jurul anului 2013 sa isi faca aparitia Web 3.0, cu un miliard de site-uri si 2,5 miliarde de utilizatori, iar mai apoi, in 2014, sa fie creata noua tehnologie HTML5, o imbunatatire a celei deja existente, respectiv HTML. HTML5 a permis introducerea unor noi tipuri de multimedia si crearea de aplicatii web independente de browsere si platforme. La scurt timp dupa aparitia HTML5, in 2015, si-au facut aparitia aplicatiile web progresive (PWA), combinand caracteristicile majoritatii browserelor moderne cu cele ale aplicatiilor mobile, facand utilizatorul sa nu poata observa nicio diferenta intre o aplicatie mobila si una de tip PWA. (2)

## Structura unui browser web

In zilele noastre aplicatiile web au ajuns sa le depaseasca pe cele de tip desktop, fiind utilizate din ce in ce mai mult, datorita similitudinii functionalitatilor oferite si, mai mult decat atat, a lejeritatii cu care pot fi utilizate, doar prin simpla accesare a unui browser web. Printre componentele esentiale ale unui browser web modern se numara:

1. DOM (Document Object Model) este un mod, independent de platforma pe care ruleaza, de a reprezenta o multitudine de obiecte ce alcatuiesc o pagina dintr-un browser web. Acesta permite paginii sa fie analizata si modificata de catre JavaScript, limbajul de programare prezent pe partea de frontend a unei aplicatii web. Practic, DOM este o interfata intre limbajul de programare si browser, ce le permite acestora doua sa comunice intre ele cu usurinta.

2. HTML este limbajul de marcaj predominant folosit la crearea paginilor web. Reprezinta un mijloc de a descrie structura informatiilor bazate pe text, marcandu-le ca fiind paragrafe, liste, titluri sau alte tipuri de structuri. Totodata ajuta si la imbinarea textului cu diverse imagini, formulare interactive sau altele.

3. CSS este un limbaj de stilizare, folosit pentru a descrie aspectul documentelor scrise intr-un limbaj de marcaj precum HTML. Un limbaj de acest gen permite ca diferite proprietati ale documentului stilizat, precum culoarea, fontul sau efectele, sa fie stabilite si definite indepentent de continutul paginii web.

4. Pe langa HTML si CSS, documentul necesar unui browser web poate contine si cod executabil, cel mai frecvent utilizat limbaj in acest scop din zilele noastre fiind JavaScript. Din punct de vedere sintactic, JavaScript este foarte asemanator cu limbajele Java si C. Este totusi mult mai dinamic decat acestea, poate si datorita particularitatilor imprumutate de la alte limbaje precum Self, Smalltalk sau Lisp. (3)

## Concepte de baza

### HTML

Hypertext Transfer Protocol (HTTP) a fost creat pe la mijlocul anilor 1990 ca modalitate de comunicare intre servere si browsere web. De atunci este protocolul din spatele fiecarei tranzactii web. Fiecare cerere pentru un document, trimitere a unui formular sau click al unui link, are la baza acest protocol de tip cerere/raspuns. Utilitatea oferita de acesta este capacitatea de a transfera diferite tipuri de date: text, grafica, audio si video, specificand modul in care cererile sunt facute de catre client, dar si modul in care serverul raspunde unor astfel de cereri.(4)

Protocolul HTTP se foloseste de Universal Resource Identifier (URI) ca si nume (URN) sau ca si locatie (URL), pentru identificarea unei resurse. La crearea unui hiperlink, URL e de foma “http://host:port/ruta/numele-fisierului.extensie”. Portul folosit de obicei este 80, insa si alte porturi se pot folosi fara probleme. Pentru ca un server sa poata comunica cu un client, intre acestia doi se realizeaza o conexiune de tip TCP/IP. Clientul face o cerere corespunzatoare catre server, in format URI, adaugand versiunea protocolului si un mesaj format din modificatorii necesari cererii, informatiile despre client si eventualul continut. Ca raspuns din partea serverului, clientul primeste linia de stare a raspunsului, formata din versiunea protocolului si codul corespunzator (succes sau eroare), urmata de un mesaj cu informatii despre server, metainformatii ale entitatii transmise si continutul aferent. Atat cererea, cat si raspunsul includ cate un antet de tip MIME (Multipart Internet Mail Extension), ce contine informatii despre datele transmise, cum ar fi tipul datelor sau alte informatii specifice. Incepand cu versiunea HTTP/1.1, se realizeaza o conexiune persistenta intre client si server, ceea ce insemna ca nu se creeaza o noua conexiune la fiecare cerere.(5)

Tranzactiile aferente acestui protocol permit un set de metode folosite pentru a le putea fi precizat scopul. Cele mai folosite metode sunt:

GET – Aceasta metoda este folosita pentru a cere o anumita informatie (sub forma unei entitati). Este recomandat sa fie folosita doar in cazul in care cererea nu afecteaza starea bazei de date.

POST – O metoda utilizata cel mai des (si recomandat) pentru a transmite date de la client catre server, cu scopul de a fi prelucrate sau anexate, printr-o metoda de adaugare, la o baza de date. In cazul prelucrarii datelor, codul returnat cel mai potrivit este 200 (ok) sau 204 (fara continut), in functie de existenta sau lipsa unei entitati care sa descrie rezultatul. Daca pe server a fost creata o entitate noua, codul potrivit ar fi 201.

PUT - Cu ajutorul acestei metode se transmit date catre server, pentru a se realiza actualizarea unei resurse deja existente, caz in care codurile returnate ar trebui sa fie 200 sau 204. Este posibil ca datele transferate sa nu faca referinta la o resursa deja existenta. Daca serverul poate defini o noua resursa cu datele primite, atunci codul returnat va fi 201 (creat).

DELETE – Aceasta metoda cere serverului sa stearga resursa identificata de datele trimise. Succesul trebuie garantat de server doar dupa ce resursa a fost stearsa sau mutata intr-un loc inaccesibil, codurile corespunzatoare fiind 202 (acceptat), daca actiunea urmeaza sa fie efectuata sau 200, respectiv 204, daca actiunea a fost efectuata cu succes si raspunsul include sau nu o entitate.(6)

### Cookies

Cookie-urile au fost create pentru a pastra informatii legate de interactiunea cu un anumit mediu lipsit de aceste date (de ex. HTML). Aceasta tehnologie a acaparat marea majoritate a site-urilor web, tinand evidenta interactiunii dintre acestea si utilizatorii lor. In mare parte, existenta unor cookie-uri pe computerele clientilor aduce beneficii, insa poate fi si un motiv de ingrijorare.

Cookie-urile sunt structuri mici de date, pe care un server le trimite catre browserul pe care navigheaza clientul, fiind apoi salvate local pe computerul acestuia intr-un fisier text. Nu sunt altceva decat informatii legate de utilizator in contextul unei anumite aplicatii web, cum ar fi parole, produse adaugate in cosul de cumparaturi, preferintele sau identitatea acestuia. La prima accesare a unui site web care se foloseste de aceasta tehnologie, un nou cookie este creat si salvat, iar la urmatoarele accesari in aceleasi conditii (acelasi browser, acelasi computer si aceeasi pagina web), clientul este recunoscut si ii este afisat un continut adaptat pentru el. Un cookie este un antet al protocolului HTTP care are nevoie de doi parametri obligatorii ca sa functioneze, si anume denumirea lui si valoarea stocata. Pe langa acestia, mai pot fi adaugati alti parametri optionali precum data de expirare, domeniul, calea si daca trebuie folosita o conexiune securizata sau nu.

Utilitatea cookie-urilor este data de faptul ca protocolul HTTP nu pastreaza niciun fel de informatie legata de cererile efectuate, lucru care ingreuneaza unele tranzactii. Folosindu-le, apar beneficii atat pe partea de client, cat si pe partea de server. Aplicatiile web au putut dintotdeauna sa pastreze informatii legate de utilizatorii lor, salvandu-le intr-o baza de date, cu costul memoriei necesare. Aparitia cookie-urilor a facut posibila salvarea acestor date pe partea de client, salvand serverul de acest cost. Un beneficiu adus clientilor este completarea automata a formularelor cu datele salvate (nume de utilizator, parole). Un dezavantaj vazut de catre utilizatori este salvarea datelor personale ale acestora, temandu-se ca ar putea fi urmariti prin intermediul cookie-urilor. Totusi, nu este o amenintare la intimitatea clientilor, datele salvate fiind doar cele introduse pe pagina web cu buna stiinta.

### Arhitectura client-server

In zilele noastre, avansarea tehnologiei a facut ca folosirea internetului sa fie aproape inevitabila in tot ceea ce facem. Utilizarea internetului este, de fapt, utilizarea unei arhitecturi de tipul client-server. Clientul este cel care cere, serverul fiind cel care ofera serviciul cerut, in majoritatea aplicatiilor de acest tip, serverul proceseaza datele, care mai apoi sunt trimise catre client.

Modelul client-server a devenit foarte popular in randul aplicatiilor, fiind folosit din ce in ce mai des. Acesta poate fi definit ca o arhitectura formata atat din clienti, cat si din servere, ce imparte procesarea in cel putin doua procese, cel mai adesea pe cel putin doua masini. Unul dintre protocoalele standardizate folosite este HTTP, cel despre care am discutat mai sus. Orice aplicatie care gestioneaza datele unei baze de date si le afiseaza in alta parte este o aplicatie ce se foloseste de modelul client-server. Ideea de baza din spatele acestui model a fost posibilitatea mai multor utilizatori de a avea acces la aceleasi date.

De obicei, o arhitectura de acest tip este formata dintr-un client (de ex un browser web), un server si o baza de date, cele mai mari doua categorii fiind:

* Arhitectura pe 2 nivele, formata doar din baza de date si client. In acest caz, clientul este cel care detine intreaga logica, trimitand cereri catre baza de date prin intermediul unei retele si tot el afisand raspunsurile utilizatorului. Acest client mai este numit si “client gros” (thick client). Arhitectura pe 2 nivele se folosea mai ales la inceputurile acestei noi structuri si era perfecta pentru pagini web statice sau cu formulare ale caror date nu necesitau procesare inainte de salvarea lor in baza de date.
* Arhitectura pe 3 nivele, compusa din client, baza de date si un server, ca parte intermediara intre celelalte doua. Poate fi extinsa cu usurinta la o arhitectura pe n nivele, prin adaugarea mai multor servere. In aplicatiile ce beneficiaza de aceasta structura, clientul are rolul doar de a prezenta utilizatorului informatiile din baza de date, furnizate de catre nivelul intermediar (serverul), supranumit si “middleware”. Middleware este o denumire oferita nivelului interpus intre client si baza de date, reprezentand un software, care ruleaza de obicei pe alt computer, ce are ca scop furnizarea logicii aplicatiei.

### Securitate

Aplicatiile web interactioneaza cu diverse medii din exterior, facand presupuneri (de multe ori gresite), despre acestea. Supozitiile nevalidate corespunzator pot duce la scaderea securitatii unei aplicatii. Principala problema care poate duce la aparitia unor vulnerabilitati de securitate este validarea neconforma a datelor primite de la utilizatori.

Scopul pentru care au fost create aplicatiile web este de a permite unei game cat mai largi de utilizatori sa le poata folosi doar cu ajutorul unui browser web si a unei conexiuni la internet. Programatorii le pot crea cu presupunerea ca utilizatorii vor introduce doar date valide, asa cum au gandit ei aplicatiile, fara sa tina cont de faptul ca atat valoarea datelor de intrare cat si modul in care acestea sunt introduse, pot fi gresite. Acelasi lucru se poate intampla si cu bazele de date unde s-a mers pe ideea ca toate interogarile vor fi plauzibile. Toate aceste ipoteze trebuie verificate pentru a nu permite crearea de vulnerabilitati ce vor putea fi valorificate de catre utilizatorii rau intentionati. Deci, principala metoda de a evita bresele de securitate este identificarea si prevenirea din timp a acestora.(11)

Principalele vulnerabilitati de securitate ale aplicatiilor web sunt “command injection” si “Cross-Site Scripting” (XSS). Intr-un atac de tipul “Command Injection”, atacatorul incearca sa acceseze informatii la care nu are acces, introducand cod si schimband firul de executie al aplicatiei. Cea mai utilizata metoda de atac din aceasta categoria este “SQL Injection”, care presupune introducerea de comenzi SQL ca si date de intrare pentru a modifica logica interogarii initiale. Un exemplu pertinent ar fi:

String interogare = “SELECT \* FROM Users WHERE username = ‘” + username + “’ AND password = ‘” + password + “’;”;

Cele doua date de intrare “username” si “password” sunt preluate de la utilizator si fara o verificare corespunzatoare, acesta ar putea insera “orice’ OR 1=1;--” ca si valoare pentru parametrul “username”, valoarea corespunzatoare variabilei “password” fiind irelevanta. Interogarea rezultata va fi: “SELECT \* FROM Users WHERE username = ‘ceva’ OR 1=1;--‘ AND password = ‘orice’;”. “--“ este suportat de cele mai multe baze de date ca fiind operatorul ce marcheaza un comentariu pe o singura linie. Astfel, tot ce se afla dupa acest operator va fi marcat ca si comentariu, ducand ca interogarea sa returneze toate datele din tabelul “Users”. Cateva solutii pentru impiedicarea acestui tip de atac sunt: refuzarea datelor de intrare ce contin anumite siruri de caractere, cosiderate suspecte de catre programator, limitarea lungimii acestora sau folosirea de “prepared statements”.

“Cross-Site Scripting” este o metoda prin care atacatorii trimit coduri (sub forma unor scripturi) altor utilizatori prin intermediul aplicatiei. Browserele clientilor la care a ajuns codul rau intentionat il vor rula, neavand de unde sa stie ca nu este o sursa de incredere. La executia acestor scripturi pot fi furate cookie-urile, sau chiar datele introduse de utilizatori in aplicatie, fara ca acestia sa aiba vreo banuiala. “Cross-Site Scripting” este o metoda foarte eficienta in atacurile de tip “phishing”.(12)

# REST

Internetul este o retea de mare succes, creata initial cu scopul de a conecta intre ele laboratoarele de cercetare. La scurt timp de la aparitie, numarul de site-uri web a inceput sa creasca fulgerator de repede, fapt care i-a ingrijorat pe dezvoltatorii acestei retele, ea nefiind gandita pentru un numar asa mare de utilizatori. Unul dintre acesti primi pionieri a fost Roy Fielding, cel care a si creat un set de sase reguli, denumite colectiv “Representational state transfer” sau “REST”. Menirea acestor constrangeri este de a oferi simplicitate, scalabilitate, fiabilitate si alte caracteristici dorite, oricarei arhitecturi ce le respecta, din cadrul unei retele de dimensiunea Internetului.

## Ce este REST mai exact ?

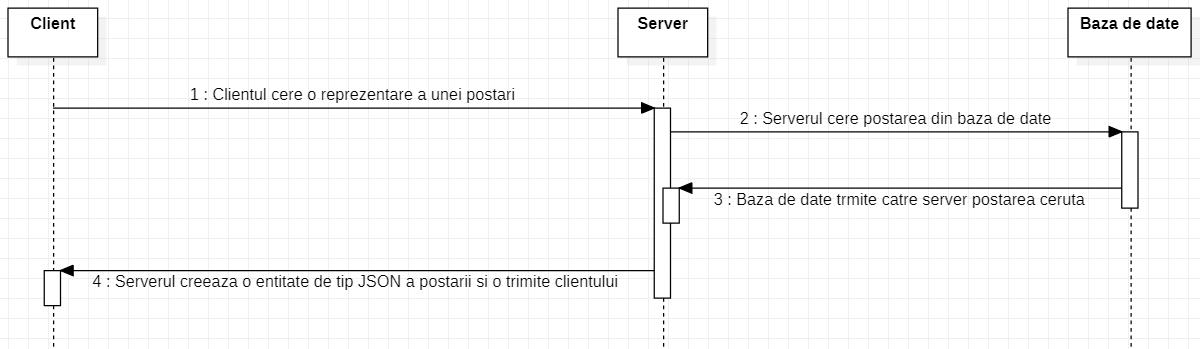
Dupa cum am precizat si anterior REST este o prescurtare a termenului “Representational state transfer”, al carui concept este unul destul de simplu. Cea vrea sa zica mai exact, este ca atunci cand un server are o resursa dorita de client, acestuia din urma nu ii pasa de reprezentarea interna a resursei oferita de catre server, care o poate stoca sub orice forma (de ex. orice baza de date, fisier text sau chiar generata pe moment). Mai jos, in **Figura 1**, este un exemplu in care clientul cere o postare de la server, singurul lucru relevant fiind structura resursei trimisa de la server catre client. Formatul JSON este unul des utilizat in cadrul REST, constand intr-o simpla notatie de tipul denumire/valoare. Un exemplu de reprezentare in format JSON a entitatii “postare” poate fi:

Figura 1

{

“nume”: “Lasagna”,

“ingrediente”: “lapte-carne-branza”

}

Deci, conceptul REST se refera doar la reprezentarea datelor transferate intre client si server, iar orice API care il foloseste (respecta cele sase conditii explicate mai amanuntit in cele ce urmeaza) poarta denumirea de “RESTful”.

## Client Server

Prima conditie si poate cea mai usor de indeplinit este ca arhitectura sistemului sa fie de tipul client/server. Acest stil de arhitectura este atat de raspandit in zilele noastre, incat, in majoritatea cazurilor, nici nu mai sunt luate in considerare altele.

Serverele sunt cele care gestioneaza datele si logica aplicatiei, iar clientii sunt cei care se ocupa de experienta utilizatorilor si de modul in care acestia vizualizeaza informatiile primite de pe partea de backend, fapt care duce la o delimitare clara intre cele doua parti. Separarea evidenta a acestora permite existenta simultana a mai multor clienti si reduce semnificativ complexitatea serverului, imbunatatind scalabilitatea aplicatiei, ceea ce permite componentelor sa evolueze independent unele fata de celelalte.

O chestiune importanta de precizat este faptul ca utilizarea protocolului HTTP nu este impusa din niciun punct de vedere, chiar daca mai mereu dezvoltatorii serviciilor web “RESTful” il prefera, si oricum, nu prea se gasesc motive pertinente pentru a nu o face.

## Lipsa starii

O alta caracteristica ce simplifica si mai mult interactiunea dintre client si server este lipsa starii. Asta inseamna orice informatie legata de sesiunilor clientilor este stocata de catre ei, ceea ce aduce un plus de scalabilitate, serverul nemaifiind nevoit sa gestioneze nicio resursa legata de acestia.

O consecinta a acestei conditii o reprezinta transmiterea de fiecare data a tuturor informatiilor necesare unei cereri, datorita lipsei oricarui fel de context. Roy Fielding chiar a si scris la un moment dat ca un client trece mai usor peste niste erori atunci cand nu este necesara sincronizarea contextului sesiunii intre cele doua componente, aducand fiabilitate aplicatiei. Principala cauza a scalabilitatii scazute si a insuccesului aplicatiilor web este data de dependenta sesiunilor prezente pe server.

# Tehnologii Web

## React

## Angular

# Despre Aplicatie

## Cerinte

## Proiectare

### Diagrame de clase

### Diagrame de secventa

### Structura baza de date

## Implementare

## Manual de utilizare

# Concluzii

# Referinte

1. Fong, Elizabeth, and Vadim Okun. "Web application scanners: definitions and functions." 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07). IEEE, 2007.

(https://sci-hub.st/https://ieeexplore.ieee.org/abstract/document/4076950)

1. Oleg Uryutin, A brief history of web app, [https://aplextor.medium.com/a-brief-history-of web-app-50d188f30](https://aplextor.medium.com/a-brief-history-of%20web-app-50d188f30)d
2. Mikkonen, Tommi, and Antero Taivalsaari. "Web applications–spaghetti code for the 21st century." 2008 Sixth international conference on software engineering research, management and applications. IEEE, 2008.

(https://sci-hub.st/https://ieeexplore.ieee.org/abstract/document/4609440)

1. Wong, Clinton. Http pocket reference: Hypertext transfer protocol. " O'Reilly Media, Inc.", 2000.

(<https://books.google.ro/books?hl=ro&lr=&id=dOIlEeG1v4UC&oi=fnd&pg=PA1&dq=hypertext+transfer+protocol&ots=C8wJKHKuxI&sig=rAMrqPQe9GY03CdrNf2T7akJwks&redir_esc=y#v=onepage&q&f=false>)

1. Yannakopoulos, John. "HyperText Transfer Protocol: A Short Course." University of Crete. August (2003).

(<https://d1wqtxts1xzle7.cloudfront.net/33767641/http.pdf?1400789276=&response-content-disposition=inline%3B+filename%3DHyperText_Transfer_Protocol_A_Short_Cour.pdf&Expires=1616593139&Signature=ZZ9MoTsuAZqpuVF45w27MvK5CSf7Xs9F50ifzyD2GXSfQy~4HbmDIddMIUTTN5anZcbeN9OGqhvyTPZrq4VuTWuNiyrQvS225mlSu6BYzZkZksyCOJV30zcGyXnlE53~GCBFRkleRBwfj7b8GbPI-GzT-avIP3WENkYXgFcUyANEvE6O7Hx71V9HyeCBhVPZB699JxaE4lg8io2NB~nbULt2yxYpWj~~uPaOle3US2y69S1qJ0Gt5bfiAFciDznllOlhsvPV62fAkv5JtY4slDEarRIDr20KndpZ334tokX-JE9Uf~LmRVARs-LAfsRfZkumYP55k9ACDhjH4~xOag__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>)

1. Fielding, Roy, et al. "RFC2616: Hypertext Transfer Protocol--HTTP/1.1." (1999).

(https://dl.acm.org/doi/pdf/10.17487/RFC2616)

1. Peng, Weihong, and Jennifer Cisna. "HTTP cookies–a promising technology." Online Information Review (2000).

(https://sci-hub.st/https://doi.org/10.1108/14684520010330346)

1. Kristol, David M. "HTTP Cookies: Standards, privacy, and politics." ACM Transactions on Internet Technology (TOIT) 1.2 (2001): 151-198.

(https://sci-hub.st/https://doi.org/10.1145/502152.502153)

1. Oluwatosin, Haroon Shakirat. "Client-server model." IOSRJ Comput. Eng 16.1 (2014): 2278-8727.

(http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1083.8741&rep=rep1&type=pdf)

1. "Distributed Application Architecture". Sun Microsystem. Archived from the original on 6 April 2011. Retrieved 2009-06-16

(https://web.archive.org/web/20110406121920/http://java.sun.com/developer/Books/jdbc/ch07.pdf)

1. Wassermann, Gary, and Zhendong Su. "An analysis framework for security in web applications." Proceedings of the FSE Workshop on Specification and Verification of component-Based Systems (SAVCBS 2004). 2004.

(http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.72.2255&rep=rep1&type=pdf#page=82)

1. Nguyen-Tuong, Anh, et al. "Automatically hardening web applications using precise tainting." IFIP International Information Security Conference. Springer, Boston, MA, 2005.

(https://sci-hub.st/10.1007/0-387-25660-1\_20)