(19) OFICIUL DE STAT
PENTRU INVENŢII ŞI MĂRCI
Bucureşti

ROMÂNIA

(11) **RO 133734 A2**
(51) Int.Cl.
G06F 17/30 (2006.01);
H04L 29/06 (2006.01)

(12) # CERERE DE BREVET DE INVENȚIE

(21) Nr. cerere: **a 2018 00133**

(22) Data de depozit: **28/02/2018**

(41) Data publicării cererii:
29/11/2019 BOPI nr. **11/2019**

(71) Solicitant:
• **SOFT STUDIO SOLUTIONS S.R.L.**,
STR.ROMAN CIOROGARIU NR.30, AP.1, ORADEA, BH, RO

(72) Inventatori:
• **DEMIAN HORIA**, *STR.CALEA ARADULUI NR.13, BL.P72, AP.6, ORADEA, BH, RO*

(74) Mandatar:
CABINET DE PROPRIETATE INDUSTRIALĂ CIUPAN CORNEL,
STR. MESTECENILOR NR. 6, BL. 9E, SC.1, AP. 2, CLUJ NAPOCA, CJ

(54) # METODĂ DE OPTIMIZARE A ÎNCĂRCĂRII MULTIPLE A UNEI SECȚIUNI DE PAGINĂ WEB, ÎNTR-O APLICAȚIE DE TIP MVC

(57) **Rezumat:**

Invenția se referă la o metodă de reducere a timpului de încărcare multiplă a unei secțiuni de pagină web care oferă posibilitatea realizării unor operații diverse, precum emiterea unei facturi sau înregistrarea unei plăți. Pentru implementarea acestor operații este necesară realizarea unor secțiuni distincte care au la bază fișiere HTML, CSS, Javascript, iar metoda conform invenției constă în primirea, de către un server, printr-un mecanism de rutare (**2**), a unei cereri (**1**) din partea unui client, transferarea acesteia la un controler (**3**) care apelează un model (**4**), încarcă (**8**) reprezentarea grafică printr-o metodă alternativă şi o livrează (**9**) clientului fără nicio altă prelucrare, datorită respectării convențiilor de construcție şi a utilizării unei metode alternative de transfer de conținut.

Revendicări: 3
Figuri: 11

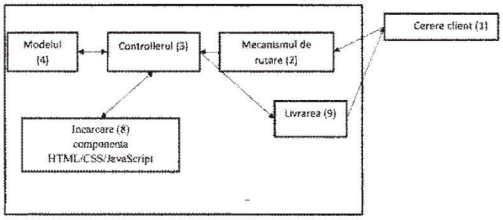

Fig. 2

Cu începere de la data publicării cererii de brevet, cererea asigură, în mod provizoriu, solicitantului, protecția conferită potrivit dispozițiilor art.32 din Legea nr.64/1991, cu excepția cazurilor în care cererea de brevet de invenție a fost respinsă, retrasă sau considerată ca fiind retrasă. Întinderea protecției conferite de cererea de brevet de invenție este determinată de revendicările conținute în cererea publicată în conformitate cu art.23 alin.(1) - (3).

RO 133734 A2

OFICIUL DE STAT PENTRU INVENTII SI MARCI
Cerere de brevet de invenţie
Nr. a 2018 00133
Data depozit 28-02-2018

Metoda de optimizare a încărcării multiple a unei sectiuni de pagina web, intr-o aplicatie de tip MVC

Invenţia se referă la o metodă de reducere a timpului de încărcare multipla a unei sectiuni de pagina web, în vederea optimizării muncii operatorului.

La inceput un site web era format din una sau mai multe pagini. Accesarea continutului site-ului presupunea completarea unei adrese web intr-un browser, iar pornind de la pagina principala se putea naviga catre celelalte pagini prin intermediul link-urilor. De fiecare data, accesarea unei pagini presupunea distrugerea continutului existent in browser si inlocuirea lui cu continutul noii pagini. Browser-ele au evoluat si au aparut cele care permit deschiderea de pagini in tab-uri diferite (zone de memorie diferite), astfel permitand utilizatorului accesarea de continut din pagini diferite in tab-uri diferite. Evolutiile ulterioare au permis ca in cazul unei pagini incarcate sa se poata injecta continut, primit in urma unor cereri fara a fi nevoie de incarcarea intregii pagini inca o data. Aceste facilitati noi au permis, modificarea continutului unei pagini prin actiuni programate sau la cererea utilizatorului.

US2009024982 "Apparatus, system, and method for archiving small objects to improve the loading time of a web page" prezinta o metoda de optimizare a incarcarii unei pagini web si nu a unei sectiuni a acesteia, printr-o metoda de arhivare a elementelor componente in anumite conditii bazate pe un prag de timp si dimensiune.

US2017103049 "System and method for loading a web page" descrie o metoda de executare a codurilor din interiorul scripturilor bazandu-se pe o prioritizare a acestora si executarea asincrona a lor in vederea incarcarii unor sectiuni dependente unele de altele si nu se concentreaza asupra optimizarii incarcarii aceleasi sectiuni.

Dezavantajele metodelor cunoscute sunt :

- Prelucrarea suplimentara pe server, in cazul view-urilor, acestea vor trece prin motorul Razor de prelucrare in cazul aplicatiilor MVC cu IIS.
- Gestionarea cache-ului de catre server necesita interventia administratorului de sistem
- Gestionarea cache-ului de catre browserul client se realizeaza diferit in functie de browserul utilizat (exista diferente constructive intre browserul Chrome si browserul Internet Explorer)

- Reincarcarea de catre browser a unor stiluri CSS sau biblioteci de cod javascript chiar daca ele exista deja.
- Chiar daca patentul US2009024982 are in vedere optimizarea incarcarii unei pagini web prin arhivarea locala a unor obiecte continute in pagina web, asemanator cu ce face browserul prin intermediul cache-ului nu se refera la conceptul de sectiune si optimizarea incarcarii sectiunii, in timp ce patentul US2017103049 se concentreaza asupra incarcarii asincrone a unei intregi pagini si nu a aceleasi sectiuni. Niciunul dintre cele doua patente nu fac referinta la lucruri concrete asociate aplicatiilor de tip MVC si nu au formulat conventii de denumire/notare.

Problema tehnică pe care o rezolvă invenţia este de a oferi o metoda de incarcare a paginilor web care permite deschiderea multipla a unei sectiuni de acelasi tip, oferind utilizatorului confort sporit de operare si reducerea timpului de incarcare.

Multitasking-ul sta la baza productivitatii. In sistemele de operare DOS se putea rula la un moment dat un singur program, iar intre timp daca se dorea accesul la o alta informatie, sau program, programul ce rula trebuia inchis si redeschis altul. Nu exista posibilitatea de comutare intre acestea si se pierdea timp. Astazi este perfect normal sa putem realiza mai multe operatii simultan, navigam pe internet in timp ce in background ascultam muzica etc.

Pronind de la ideea de multitasking, dorim ca utilizatorul sa poata realiza simultan mai multe operatii in cadrul aceleasi pagini web deschise. De exemplu consultarea catalogului de produse si editarea unei facturi. Dar nu ne oprim aici. Metoda pe care o propun se refera la editarea simultana a mai multor facturi, editarea simultana a mai multor receptii, si nu doar o singura operatie de acelasi tip in cadrul aceleiasi pagini web. Se vor putea realiza mai multe operatii de acelasi tip in aceeasi pagina web.

Pentru o mai buna intelegere vom rezenta un exemplu in care un utilizator deschide o sectiune de facturare si incepe facturarea propriu zisa. Intre timp apare necesitatea consultarii unei facturi emise anterior in vererea relistarii. In mod normal utilizatorul are doua optiuni si anume: reincarcarea paginii web cu noul continut, ceea ce are ca efect distrugerea continutului editat pana atunci sau deschiderea inca o data a aplicatiei intr-un nou browser si consultarea acolo. Metoda noastra permite deschiderea multipla a unei sectiuni de acelasi tip. Avantajul este acela de sporire a confortului de operare si cresterea vitezei de incarcare.

Elementele componente ale unei aplicatii web

Aplicatiile web au la baza limbajul HTML, CSS, Javascript, Html fiind limbajul descriptiv de definire a elementelor, CSS se utilizeaza pentru formatarea aspectului, iar javascript pentru asocierea de actiuni, tratarea de evenimente etc.

Arhitectura unei aplicatii Web MVC

O aplicatie orientata pe principiile MVC poate fi o colectie de triade model/view/controller, fiecare responsabila de un element diferit al interfetei cu utilizatorul. MVC este des intalnit in aplicatii web unde view-ul este codul HTML generat de aplicatie. Controller-ul primeste variabile GET si POST ca si input si decide ce sa faca cu acestea, trimitandu-le mai departe model-ului. Model-ul, care contine logica de business si regulile aferente, poate efectua operatiile necesare asupra datelor pentru a putea permite aplicatiei generarea codului HTML mai sus mentionat via engine-urile de template, pipeline-uri XML, cereri de tip Ajax, etc.

Model-ul nu este neaparat doar o baza de date, fiind adesea atat baza de date cat si logica de business necesara pentru a manipula datele in interiorul aplicatiei. Multe aplicatii folosesc un mecanism persistent de stocare a datelor. MVC nu specifica in mod explicit nivel de acces la date tocmai pentru ca este de la sine inteles ca acesta se afla incapsulat in model. In unele aplicatii simple care au putine reguli de business logic impuse, model-ul se poate limita doar la o baza de date si la functionalitatile oferite de aceasta.

View-ul, de asemenea nu este limitat doar la afisarea informatiei, el avand un rol important si in interactiunea cu utilizatorul. In cazul exemplului de mai sus al aplicatiilor web interfata generata via cod html este cea care se ocupa si de preluarea input-ului si de masurile luate pentru ca acesta sa fie corect.

Controller-ul este adesea confundat cu aplicatia insasi, pe cand rolul sau este de a dirija datele intre celelalte doua clase de obiecte. Intr-adevar, se pot executa multe operatii asupra datelor de catre model, insa aceste operatii tin de formatul in care se prezinta datele la un moment dat. Adesea se intalneste cazul in care datele afisate/culese de la utilizator difera semnificativ de cele stocate in baza de date. Aceste diferente se datoreaza conversiilor ce pot fi aplicate asupra datelor de catre controller pentru a facilita traficul de informatie intre componente. Fiecare clasa de obiect are anumite expectative definite in ceea ce priveste formatul datelor, ori aceste transformari de format trebuie realizate automat pentru a mentine un flux constant de date, degrevand celelalte clase de grija conversiilor si asigurand aplicatia ca fiecare modul

primeste ceea ce asteapta. Asta pe langa functia de baza de controla traficul de cereri intre module.

Functionare

Schema de functionare a unei aplicatii modelate dupa arhitectura MVC decurge, in linii mari, in felul urmator (a se consulta fig.1):

1. Utilizatorul interactioneaza cu interfata (exemplu: apasa un buton la tastatura)

2. Controller-ul primeste actiunea apasarii butonului si o converteste intr-o actiune pe intelesul model-ului

3. Controller-ul notifica model-ul de actiunea utilizatorului, urmand de obicei o schimbare a starii model-ului (exemplu: model-ul reimprospateaza starea campului de adresa)

4. Controllerul trimite datele din model unui view interogheaza model-ul pentru a genera o interfata corespunzatoare (exemplu: view-ul afiseaza noua adresa alaturi de cea veche alaturi de un buton de confirmare)

5. Procesarea view-ului de catre motorul de procesare Razor si livrarea rezultatului catre client

6. Interfata asteapta actiuni suplimentare din partea utilizatorului, ciclul reluandu-se.

In vederea atingerii obiectivului de reducere a timpului de încărcare multipla a unei sectiuni de pagina web s-au creat urmatoarele conventii si metode ce vor fi utilizate in procesul de creare si incarcare a unei sectiuni:

- Conventiile/ regulile ce se aplica la constructiile sectiunilor si a elementelor componente
- Metode utilizate pe Server in Controller-ul aferent unei sectiui care vor fi absolut necesare a fi definite in vederea implementarii conceptului.
- Metode utilizate pe Client pentru incarcarea unei sectiuni care vor fi absolut necesare a fi definite pentru realizarea obiectivului

Faza de constructie consta in structurarea datelor si realizarii programarii efective, etapa in care vor fi respectate conventiile de denumire si organizare a informatiei. Fara respectarea

acestor conventii de constructie nu putem atinge cele doua obiective: incarcarea multipla si optimizarea vitezei de incarcare a unei sectiuni.

In faza de constructie se vor crea urmatoarele doua componente:

- Crearea componentelor ce vor rula pe server si anume: componenta Controller si componenta Model
- Crearea componentelor ce vor rula pe client si anume: componenta HTML, componenta CSS si componenta Javascript.

Faza de functionare consta in utilizarea componentelor si parcurgerea unor pasi in vederea contructiei efective a unei sectiuni in pagina din browserul clientului (Fig.3) dupa cum urmeaza:

- Formularea cererii de incarcare a unei sectiuni
- Verificarea daca utilizatorul are drepturi de acces la aceasta sectiune
- Verificarea daca elementele componente ale sectiunii exista deja in localstorage
- Daca nu exista in localstorage se vor formula cereri catre server in vederea incarcarii acestora in localstorage
- Daca componenta Javascript exista incarcata in memorie aceasta nu va mai fi incarcata inca o data din localstorage.
- Daca componenta CSS exista incarcata in memorie aceasta nu va mai fi incarcata inca o data din localstorage.
- Citirea din localstorage a componentei HTML si constructia efectiva a sectiunii.

Metoda, conform inventiei, permite incarcarea rapida a unei sectiuni de pagini web (inlatura dezavantajele metodelor cunoscute) deoare presupune efectuarea urmatorilor pasi:

- Elimina faza Razor-prelucrare view de pe server (Fig.1. vs Fig.2), deoarece prin conventiile de constructie nu mai este necesara. De aceea s-a putut modificata actiunea controllerului de incarcare a view-ului astfel ca se evita faza Razor-prelucrare. Spre exemplificare actiunea de alocare dinamica a identificatorului de sectiune se realizeaza prin conventie pe client si nu pe server ceea ce inseamna ca o actiune de prelucrare pe server a fost eliminata. In momentul de fata controllerul este invatat sa raspunda la un apel client cu un sir de caractere (text), iar clientul este invatat sa prelucreze textul si sa il utilizeze dupa necesitati.

- Elimina reincarcarea componentei Javascript sau CSS aferente unei sectiuni daca aceasta a fost deja incarcata in memorie (Fig 3 vs Fig 4 respectiv Fig.5 vs Fig. 6) Am creat un controller care este invatat sa raspunda la un apel client cu un sir de caractere (text), iar clientul este invatat sa prelucreze textul si sa il utilizeze dupa necesitati.
- Reduce timpul de creare a unei noi sectiuni de acelasi tip (Fig 7 vs Fig 8) deoarece au fost eliminate etape de formulare de cereri catre server ceea ce duce la elminarea prelucrarilor de pe server si au fost inlocuite cu constructia pe baza datelor deja existente pe client, deoarece discutam de reincarcarea unei sectiuni ce a mai fost utilizata.

Se prezintă un exemplu de aplicare a invenţiei în legatura cu figurile 1-11, care reprezinta:

-figura 1, reprezinta succint procesarile ce au loc in componenta server atunci cand se primeste o cerere de la client(1) pentru livrarea componentei HTML aferenta unei sectiuni. Cererea clientului este primita de catre server care prin mecanismul de rutare(2) o preda mai departe controllerului(3). Controllerul(3) apeleaza dupa caz modelul (4), incarca view-ul (5) care sufera o procesare (6) si rezultatul procesarii este livrat clientului (7). Aceasta procesare(6) duce la cresterea timpului de livrare.

-figura 2, reprezinta succint procesarile ce au loc in componenta server **noua** atunci cand se primeste o cerere de la client (1). Cererea clientului(1) este primita de catre server care prin mecanismul de rutare (2) o preda mai departe controllerului (3). Controllerul(3) apeleaza dupa caz modelul (4), incarca view-ul (8) prin metoda alternativa si este livrat clientului (9) fara nicio alta prelucrare datorita respectarii conventiilor de constructie si a utilizarii unei metode alternative de transfer a unui continut catre client.

- figura 3, reprezinta modalitatea de functionare a componentei client atunci cand se creeaza o sectiune pentru prima data. Componenta client prezentata aici comunica cu componenta Server prezentata in fig. 2. Se porneste de la cererea de creare a sectiunii (10) care apeleaza metoda Incarcare (11). Metoda Incarcare(11) va verifica daca componenta HTML exista in localstorage(16). Daca nu exista va apela metoda LoadModel (12). Metoda LoadModel este responsabila de realizarea comunicarii cu serverul in vederea obtinerii componentei HTML prin formularea unei cereri client (1), receptia componentei si realizarea depozitarii acesteia in localstorage(16), controlul fiind predat apoi metodei Incarcare(11). Metoda Incarcare(11) va verifica daca componenta Javascript exista in localstorage(16), iar daca nu exista va apela

metoda LoadBibliotecaJS(13). Metoda LoadBibliotecaJS(13) este responsabila de realizarea comunicarii cu serverul in vederea obtinerii componentei Javascript prin formularea unei cereri client (1) , receptia componentei si realizarea depozitarii acesteia in localstorage(16), controlul fiind predat apoi metodei Incarcare(11). Metoda Incarcare(11) va verifica daca componenta CSS exista in localstorage(16), iar daca nu exista va apela metoda LoadCSS(14). Metoda LoadCSS(14) este responsabila de realizarea comunicarii cu serverul in vederea obtinerii componentei CSS prin formularea unei cereri client (1) , receptia componentei si realizarea depozitarii acesteia in localstorage(16), controlul fiind predat apoi metodei Incarcare(11). Metoda Incarcare(11) a primit controlul din nou si toate componentele exista in localstorage(16) si de aceea continua cu apelarea metodei ContructiePeBazaDeModel(15). Metoda ConstructiePeBazaDeModel(15) va citi componentele HTML, javascript si CSS din localstorage (16) si va continua cu crearea sectiunii(17).

- figura 4, reprezinta modalitatea de functionare a componentei client atunci cand se doreste recrearea unei sectiuni. Se porneste de la cererea de creare a sectiunii (10) care apeleaza metoda Incarcare (11). Metoda Incarcare(11) va verifica daca componenta HTML exista in localstorage(16). Deoarece exista (nefiind prima incarcare) va continua cu verificarea existentei componentei Javascript in localstorage(16). Deoarece exista va continua cu verificarea existentei componentei CSS in localstorage(16). Deoarece exista, continua cu apelarea metodei ContructiePeBazaDeModel(15). Metoda ConstructiePeBazaDeModel(15) va citi componentele HTML, javascript si CSS din localstorage (16) si va continua cu crearea sectiunii(17).

- figura 5 descrie elementele componente ale unei sectiuni ce s-a incarcat pentru prima data in memoria calculatorului. Astfel pentru sectiune se incarca componenta HTML (21), componenta Javascript (19) si componenta CSS(20).

- figura 6 descrie elementele componente ale doua sectiuni de acelasi tip create in aceeasi pagina web. Se observa ca vor exista doua componente HTML si anume (22) si (21), iar componentele javascript(19) si CSS(20) le deserveste pe amandoua.

- figura 7 reprezinta o captura de ecran ce descrie traficul care se genereaza intre client si server pentru constructia unei sectiuni de un anumit tip pentru prima data si evidentiaza exista apelului metodei LoadBibiotecaJS. De asemenea se poate observa timpul de 446 ms.

- figura 8 reprezinta o captura de ecran ce descrie traficul care se genereaza intre client si server pentru constructia unei sectiuni de un anumit tip ulterioara primei incarcari si evidentiaza lipsa apelului metodei LoadBibiotecaJS. Se poate observa timpul de 263 ms.

- figura 9 reprezinta o captura de ecran ce prezinta constructia componentei HTML ce nu respecta conventiile enuntate. Se pot observa utilizarea de variabile, link-uri sau cod javascript direct injectate in continut.

- figura 10 reprezinta o captura de ecran ce prezinta construcita componentei HTML ce respecta conventiile enuntate.

- figura 11 reprezinta o captura de ecran ce prezinta constructia componentei Javascript ce respecta conventiile enuntate.

Modelul conceptual si functionarea acestuia

Tehnologia de astazi permite incarcarea unei sectiuni de pagina web, incarcarea unor functii definite in javascript si incarcarea de stil-uri CSS. Pentru fiecare operatie pe care un utilizator dorim sa o poata realiza vom defini o sectiune ce va fi compusa din cele trei componente. Fiecare componenta este pastrata intr-un fisier distinct pe server. Pentru realizarea unei operatii de emitere facturi vom avea urmatoarele fisiere: Factura.cshtml pentru descrierea elementelor sectiunii, Factura.CSS pentru stylesheet, iar pentru codul asociat evenimetelor vom avea Factura.JS.

Incarcarea unei sectiuni in browser-ul clientului va insemna incarcarea celor trei componente.

Conventii ce stau la baza inventiei

Urmatoarele conventii au fost create pentru denumirea elementelor ce compun o sectiune:

- Conventia de denumire a unei sectiuni
- Conventia de identificare a elementelor unei sectiuni
- Conventia de izolare a elementelor unei sectiuni
- Conventia de definire a elementelor unei sectiuni
- Conventia de izolare a functiilor unei sectiuni
- Conventia de definire a functiilor unei sectiuni
- Conventia de initializare a unei sectiuni

- Conventia de distrugere a unei sectiuni
- Conventia de izolare a variabilelor unei sectiuni

Conventia de denumire a unei sectiuni

Fiecare sectiune va avea o denumire unica in interiorul proiectului si anume:
DenumireSectiune

Conventia de identificare a elementelor unei sectiuni

O sectiune reprezinta o lista de elemente incarcate intr-o pagina web. Chiar daca lista de elemente a unei sectiuni a fost creata cu un element radacina comun, acestea nu vor fi plasate neaparat intr-o zona continua de elemente. Ea poate suferi modificari si anumite subsectiuni (sublista de elemente) pot fi plasate in interiorul altor elemente, ce nu apartin neaparat de sectiunea noastra. Un exemplu este cel al dialogurilor.

O problema ce deriva din injectarea unei sectiuni intr-o pagina web, este aceea de a determina toate elementele ce apartin de o sectiune. Pentru rezolvarea acestei probleme putem apela la utilizarea identificatorilor pentru elementele unei sectiuni si vom crea urmatoarea **conventie de notare:** Fiecare element al unei sectiuni va avea un identificator al sectiunii si un identificator al elementului, de exemplu: **DenumireSectiune_DenumireElement** .

Izolarea unei sectiuni

Incarcarea multipla a unei sectiuni, va avea ca efect crearea de mai multe ori a acelorasi elemente. Elementele au identificatori, ceea ce duce la existenta mai multor elemente cu acelasi identificator. Vom stabili urmatoareel conventii de izolare:

Conventia de izolare a elementelor unei sectiuni:

Fiecare sectiune va fi identificabila de un identificator de sectiune unic, iar fiecare identificator al unui element al sectiunii va contine identificatorul de sectiune:

DenumireSectiune_DenumireElement_IdentificatorDeSectiune.

Pentru **IdentificatorDeSectiune** se va utiliza o valoare de tip text denumita #MODELID si care va fi inlocuita in momentul constructiei sectiunii pe client cu un identificator ce respecta conventia unui uniqueidentifier. Se asigura astfel unicitatea valorii obtinute. Faptul ca acest

identificator va fi alocat pe client in momentul constructiei efective a sectiunii ne ofera avantajul ca nu este necesara prelucrarea componentei HTML pe serverpt alocarea identificatorului.

*In urma respectarii **conventiei de izolare a elementelor unei sectiuni** vom atinge urmatorul obiectiv si anume acela ca fiecare element va fi unic identificat in interiorul paginii prin intermediul atributului id.*

Conventia de definire a elementelor unei sectiuni

In interiorul componentei HTML nu se vor utiliza variabile sau modele ce necesita prelucrari, ci doar cod HTML pur ce va fi prelucrat doar de catre browserul clientului fara a fi nevoie de nicio prelucrare pe server. Aceasta conventie permite eliminarea unei faze de executie pe server si anume faza Razor – prelucrare View (a se consulta Fig1 vs Fig 2.) In interiorul componentei HTML nu se vor insera link-uri catre componente javascript sau CSS si nici sectiunea <Script> </Script> .

Pentru exemplificarea diferentei de abordare se poate consulta Fig.9 vs Fig 10 din anexa.

Conventia de izolare a functiilor unei sectiuni

Fiecare functie definita pentru o sectiune va avea o denumire ce va respecta urmatoarea conventie de notare: **DenumireSectiune_DenumireFunctie**, unde **DenumireFunctie** trebuie sa fie o valoare unica in interiorul functiilor aferente sectiunii.

Respectand aceasta convemtie functiile vor fi unic definite in interiorul paginii web eliminand asstfel eventualele conflicte generate de suprascrierea acestora. Se poate consulta Fig. 11 ce exemplifica corpul unei functii din componenta Javascript ce respecta aceasta conventie.

Conventia de definire a functiilor unei sectiuni:

Fiecare functie definita in fisierul de functii aferent unei sectiuni va avea obligatoriu un argument **IdentificatorDeSectiune** ce se va utiliza pentru identificarea domeniului de aplicabilitate al functiei.

Respectarea acestei conventii de asigura, ca aceeeasi functie se va putea aplica daor elementelor unei sectiuni la un nmoemnt dat fara a genera efecte secundare in alte sectiuni. (a

se consulta Fig 11 pentru exemplificare unde IcidModel reprezinta indentificatorul de sectiune asupra careia se va aplica corpul functiei)

Conventia de initializare a unei sectiuni

Pentru fiecare sectiune va exista o functie cu denumirea **DenumireSectiune_Initializare** ce va fi utilizata pentru intializarea elementelor sectiunii. (Fig. 11 exemplifica existenta acestei functii pentru sectiunea Documentiesire)

Conventia de distrugere a unei sectiuni

Pentru fiecare sectiune va exista o functie cu denumirea **DenumireSectiune_Deinitializare** ce va fi utilizata pentru deintializarea elementelor sectiunii.

Conventia de izolare a variabilelor unei sectiuni

Exista variabile globale si variabile locale. Variabilele globale sunt accesibile in orice functie existenta, in timp ce variabilele locale sunt vizibile doar in interiorul functiei in care acestea au fost definite. Exista Valori pe care dorim sa le partajam doar intre functiile unei sectiuni. In vederea realizarii acestui obiectiv stabilim conventia de izolare a variabilelor si anume: Pentru fiecare variabila pe care o dorim paratajata intre mai multe functii vom avea un element de tip hidden ce va respecta **Conventia de izolare a elementelor unei sectiuni**, lucru care ne asigura atat unicitatea cat si izolarea.

In interiorul functiilor se vor utiliza variabile cu domeniul de vizibilitate local, ceea ce inseamna ca odata parasit corpul functiei variabila va fi eliberata din memorie.

Incarcarea unei sectiuni

Incarcarea unei sectiuni presupune incarcarea tuturor celor trei componente: **componenta HTML**, **componenta CSS** si **componenta JavaScript** in documentul existent.

Din pagina web existenta in broserul clientului ca rezultat al unei actiuni intreprinse de acesta se formuleaza o cerere catre server pentru sectiunea dorita. In interiorul sectiunii exista o legatura catre componenta CSS si o legatura catre componenta JS. Browserul in urma incarcarii sectiunii va incepe interpretarea continutului acesteia, identifica legaturile catre componenta CSS si componenta JS, formuleaza noi cereri de incarcare a acestora. In urma reincarcarii componentei CSS si a componentei JavaScript se vor executa scripturile din

sectiune si vor fi aplicate stilurile din CSS asupra tuturor elementelor existente in document. Aceste stiluri vor afecta si alte elemente existente in alte sectiuni identice ale documentului.

Faza de constructie

In aceasta etapa se va realiza structurarea datelor si realizarea programarii efective cu respectareaa conventiile de denumire si organizare a informatiei. Respectarea acestora stau la baza atingerii celor doua obiective: incarcarea multipla si optimizarea vitezei de incarcare a unei sectiuni. In aceasta etapa sunt definite componentele ce vor rula pe server si componentele ce vor ajunge si functiona la client.

Componentele care vor rula pe server sunt: componenta Controller si componenta Model.

Definirea Controller-ului. Pentru fiecare sectiune se va crea propriul controller cu numele DenumireSectiune. In aceasta etapa caracterul de noutate il reprezinta crearea unor metode speciale ce sunt utilizate in procesul de incarcare a unei sectiuni in pagina deschisa in browserul clientului si anume: DenumireSectiune_Model, DenumireSectiune_Version, DenumireSectiune_VerificareDrepturi, DenumireSectiune_LoadBibliotecaJS si DenumireSectiune_LoadCSS.

Definirea Modelului. Fiecare controller asociat unei sectiuni va avea un model al datelor denumit Model_DenumireSectiune.

Metode utilizate pe Server in Controller-ul aferent unei sectiui care vor fi absolut necesare a fi definite in vederea implementarii conceptului

Au fost create urmatoarele metode pe server in vederea optimizarii timpului de raspuns si anume:

- DenumireSectiune_Model
- DenumireSectiune_Version
- DenumireSectiune_VerificareDrepturi
- DenumireSectiune_LoadBibliotecaJS
- DenumireSectiune_LoadCSS

Componentele care vor rula pe client sunt: componenta HTML, componenta CSS si componenta Javascript.

Definirea view-urilor (**componenta HTML**), a elementelor componente ale unei sectiuni. Pentru fiecare sectiune se va crea un view denumite DenumireSectiuneModel ce va contine cod HTML pur utilizat pentru definirea elementelor ce compun sectiunea. Aici am avut in vederea eliminarea oricaror referinte catre variabile de sesiune, astfel incat sa nu fie nevoie de procesarea fisierului inainte de livrare. Continutul va putea sa fie livrat asa cum este fara nicio procesare. Caracterul de noutate il reprezinta conventiile utilizate ce va asigura unicitatea elementelor si izolarea acestora si eliminarea oricaror prelucrari necesare pe server.

Definirea CSS-urilor (**componenta CSS**) asociate elementelor unei sectiuni. Pentru fiecare sectiune se va defini un CSS denumit DenumireSectiune. Caracterul de noutate il reprezinta conventia de denumire ce va asigura unicitatea stilurilor si izolarea acestora.

Definrea actiunilor si evenimentelor (**componenta Javascript**) ce vor fi asociate elementelor din sectiune. Caracterul de noutate il reprezinta conventiile de denumire utilizate, conventii ce vor asigura unicitatea si izolarea. Tot aici vor fi incluse si metodele utilizate pe client

Metode utilizate pe Client pentru incarcarea unei sectiuni

Metodele utilizate pe client vor fi definite in componenta Javascript si sunt necesare pentru atingerea obiectivelor.

Localstorage reprezinta o zona de memorie a browserului ce poate fi accesata si gestionata de cod existent in pagina web. In aceasta zona de memorie se pot defini locatii de memorie in care putem pastra diverse informatii (variabile). O importanta caracteristica este aceea ca valorile memorate aici persista in urma inchiderii browserului.

Am identificat necesitatea existentei mai multor metode care vor gestiona incarcarea unei **sectiuni** si anume: metoda **LoadModel**, metoda **LoadBibliotecaJs**, metoda **LoadCSS,** metoda **Incarcare** si metoda **ConstructiePeBazaDeModel**.

Metoda **LoadModel** este utilizata pentru incarcarea componentei HTML de pe server in localstorage.

Metoda **LoadBibliotecaJs** este utilizata pentru incarcarea componentei JS de pe server in localstorage.

Metoda **LoadCSS** este utilizata pentru incarcarea componentei JS de pe server in localstorage.

Metoda **Incarcare** va fi apelata de fiecare data cand dorim crearea unei noi sectiuni, si este responsabila de verificarea existentei in localstorage a componentelor necesare. Metoda Incarcare va face apel la una din metodele **LoadModel, LoadBibliotecaJs sau LoadCSS** daca componenta corespunzatoare nu exista in localstorage. In urma certificarii faptului ca exista componentele in localstorage va face apelul medodei **ConstructiePeBazaDeModel.**

Metoda **ConstructiePeBazaDeModel** este responsabila pentru creare efectiva a sectiunii in pagina pe baza modelelor existente in localstorage.

Prin utilizarea acestei metode avem un control direct asupra intregului proces de creare a unei sectiuni si in acelasi timp beneficiem de performanta la incarcarea multipla a acesteia.

Schema de functionare a metodei

Incarcarea pentru prima data a unei sectiuni.

Schema de functionare a metodei in cazul incarcarii pentru prima data a unei sectiuni decurge dupa urmatorul algoritm :

1) Utilizatorul interactioneaza cu interfata (exemplu: apasa un buton de pe ecranul paginii web) si declanseaza o cerere de incarcare a unei sectiuni (Fig.3 - Cerere creare sectiune(10))
2) Se executa metoda **Incarcare** (Fig.3 - Incarcare (11)) care va verifica daca exista componentele necesare in **localstorage**(Fig.3 - localstorage(16)) si deoarece nu le gaseste va apela urmatoarele metode din figura 3: **LoadModel(12), LoadCSS(13), LoadBibliotecaJS(14)** si **ConstructiePebazaDeModel (15).**
3) Se executa metoda **LoadModel** (Fig.3 - **LoadModel (12)**) si se va realiza formularea unei cereri catre server asa cum se poate vizualiza in **fig.2** – Cerere client (1)
4) Controllerul aflat pe server (fig.2. – Controllerul(3)) primeste cererea din partea clientului pentru componenta HTML, realizeaza consultarea modelului daca este cazul (fig.2. –

Modelul (4)), incarca componenta (fig.2. – Incarcare componenta(8)) si o livreaza (fig.2. – Livrarea (9)) fara nicio procesare suplimentara a acesteia.

5) Clientul primeste componenta HTML (Fig.3 - **LoadModel (12)**) si o salveaza in **localstorage** (Fig.3 - localstorage(16)).
6) Se executa metoda **LoadBibliotecaJS** (Fig.3 – **LoadBibliotecaJS(13)**) si se va realiza formularea unei cereri catre server asa cum se poate vizualiza in **fig.2** – Cerere client(1)
7) Controllerul aflat pe server (fig.2. – Controllerul(3)) primeste cererea din partea clientului pentru componenta Javascript, realizeaza consultarea modelului daca este cazul (fig.2. – Modelul(4)), incarca componenta (fig.2. – Incarcare componenta(8)) si o livreaza (fig.2. – Livrarea(9)) fara nicio procesare suplimentara a acesteia.
8) Clientul primeste componenta Javascript (Fig.3 - **LoadBibliotecaJS (13)**) si o salveaza in **localstorage** (Fig.3 - localstorage(16)).
9) Se executa metoda **LoadCSS** (Fig.3 – **LoadCSS(14)**) si se va realiza formularea unei cereri catre server asa cum se poate vizualiza in **fig.2** – Cerere client (1)
10) Controllerul aflat pe server (fig.2. – Controllerul(3)) primeste cererea din partea clientului pentru componenta CSS, realizeaza consultarea modelului daca este cazul (fig.2. – Modelul(4)), incarca componenta (fig.2. – Incarcare componenta (8)) si o livreaza (fig.2. – Livrarea (9)) fara nicio procesare suplimentara a acesteia.
11) Clientul primeste componenta CSS (Fig.3 - **LoadCSS(14)**) si o salveaza in **localstorage** (16).
12) In momentul in care toate componentele sunt incarcate in localstorage(16) este predat controlul mai departe metodei **ConstructiePebazaDeModel** (Fig.3 - **ConstructiePebazaDeModel(15)**) responsabila pentru citirea componentei HTML, CSS si Javascript din localstorage(Fig.3 - localstorage(16)), incarcarea acestora in browser si crearea efectiva sectiunii(Fig.3 - **Sectiune(17)**).

In aceasta schema sunt implicate atat metodele de pe client cat si metodele de pe server.

Timpul de incarcare a unei sectiuni pentru prima data este ilustrat in **fig. 7** din anexa unde pt un anumit tip de sectiune s-au consumat 446 milisecunde.

Reincarcarea unei sectiuni

Schema de functionare a metodei in cazul reincarcarii unei sectiuni decurge dupa urmatorul algoritm (fig.4):

1) Utilizatorul interactioneaza cu interfata (exemplu: apasa un buton de pe ecranul paginii web) si declanseaza o cerere de incarcare a unei sectiuni (Fig.4 - Cerere creare sectiune(10))
2) Se executa metoda **Incarcare** (Fig.4 - Incarcare(11)) care va verifica daca exista componentele necesare in localstorage(Fig.4 – localstorage(16)) si deoarece le gaseste va apela metoda **ConstructiePebazaDeModel(Fig 4. – ConstructiePebazaDeModel(15)).**
13) Se executa metoda **ConstructiePebazaDeModel** (Fig.4 – **ConstructiePebazaDeModel(15)**) responsabila pentru citirea componentei HTML, CSS si Javascript din localstorage (Fig 4 – localstorage(16)), incarcarea acestora in browser daca ele nu exista deja (CSS-ul si Javascript ar trebui sa existe) si crearea efectiva sectiunii(Fig.4 - **Sectiune(17)**).

Se observa in faza de reincarcare a sectiunii, eliminarea unor comunicari cu serverul pentru livrarea componentelor. Optimizarea timpului este ilustrata in **fig 8** unde avem un timp de 263 ms in raport cu figura 7 unde avem 446 ms, pentru reincarcarea aceleasi sectiuni.

REVENDICARI

1. Metoda de constructie a unei sectiuni in cadrul unei pagini web bazata pe o cerere client (1) printr-un mecanism de rutare (2) catre un controler (3) care apeleaza un modul de incarcare (8) si livreaza printr-un modul (9) o sectiune (17), **caracterizata prin aceea ca**, pentru reducerea timpului de incarcare a sectiuni (17) se realizeaza urmatorii pasi:
   1.1. Utilizatorul interactioneaza cu interfata si declanseaza o **cerere de creare a** unei sectiuni (17)
   1.2. Procedura **Incarcare (11)** care va verifica daca exista componentele necesare in modulul **localstorage (16)** iar daca nu le gaseste va apela urmatoarele proceduri: **LoadModel (12), LoadBibliotecaJS (13), LoadCSS (14)** si **ConstructiePebazaDeModel (15)**
   1.3. Procedura **LoadModel (12)** va realiza formularea Cerere client (1) catre serverul (18)
   1.4. Controllerul (3) aflat pe server (18) primeste cererea (1) din partea clientului pentru componenta HTML (21), daca este cazul realizeaza consultarea modelului (4), prin modulul Incarcare (8), incarca componenta si o livreaza clientului, prin modulul Livrare (9), fara nicio procesare suplimentara a acesteia
   1.5. Clientul primeste componenta HTML(21) livrata de serverul(18) si o salveaza in localstorage (16)
   1.6. Procedura **LoadBibliotecaJS** (13) va realiza formularea unei Cerere client (1) catre serverul (18)
   1.7. Controllerul (3) aflat pe serverul (18) primeste cererea din partea clientului pentru componenta Javascript (19), realizeaza consultarea modelului (4) daca este cazul, incarca componenta (19) si o livreaza clientului, prin modulul Livrare (9), fara nicio procesare suplimentara a acesteia
   1.8. Clientul primeste componenta Javascript (19) si o salveaza in localstorage (16).
   1.9. Procedura **LoadCSS** (14) va realiza formularea unei Cerere client (1) catre server (18)
   1.10. Controllerul (3) aflat pe server (18) primeste cererea din partea clientului pentru componenta CSS, realizeaza consultarea modelului (4) daca este cazul, incarca componenta CSS (20) si o livreaza clientului, prin modulul Livrare (9), fara nicio procesare suplimentara a acesteia

1.11. Clientul primeste componenta CSS (20) si o salveaza in localstorage (16).

1.12. In momentul in care toate componentele sunt incarcate in localstorage este predat controlul mai departe procedurii **ConstructiePebazaDeModel** (15) responsabil pentru citirea componentei HTML (21), CSS (20) si Javascript (19) din localstorage (16), incarcarea acestora in browser si crearea efectiva sectiunii I (17).

2. Metoda de constructie a unei sectiuni in cadrul unei pagini web, conform revendicarii 1, **caracterizata prin aceea ca**, pentru constructia unei noi sectiuni (17) care a mai fost creata se parcurg urmatorii pasi:

   2.1. Utilizatorul interactioneaza cu interfata si declanseaza o cerere de incarcare a unei sectiuni, Cerere creare sectiune (10)

   2.2. Se executa incarcarea prin modulul **Incarcare** (11) care va verifica daca exista componentele HTML (22), Java Script (19) si CSS (20) necesare sunt in localstorage (16) si deoarece acestea au fost create anterior, le gaseste si va apela modulul **ConstructiePebazaDeModel (15)**

   2.3. Modulul **ConstructiePebazaDeModel** (15) va citi componentele HTML (22), din localstorage (16), si o incarca in browser. Daca componentele CSS si javascript exista in broser nu le va mai incarca din localstorage si continua cu initializarea sectiunii obtinindu-se Sectiunea II (17).

3. Metoda de constructie a unei sectiuni in cadrul unei pagini web, conform revendicarii 1, **caracterizata prin aceea ca**, pentru deschiderea simultana a mai multor sectiuni de acelasi tip, cu reducerea timpului de incarcare se utlizeaza urmatoarele conventii:

   - Conventia de denumire a unei sectiuni
   - Conventia de identificare a elementelor unei sectiuni
   - Conventia de izolare a elementelor unei sectiuni
   - Conventia de definire a elementelor unei sectiuni
   - Conventia de izolare a functiilor unei sectiuni
   - Conventia de definire a functiilor unei sectiuni
   - Conventia de initializare a unei sectiuni
   - Conventia de distrugere a unei sectiuni
   - Conventia de izolare a variabilelor unei sectiuni.

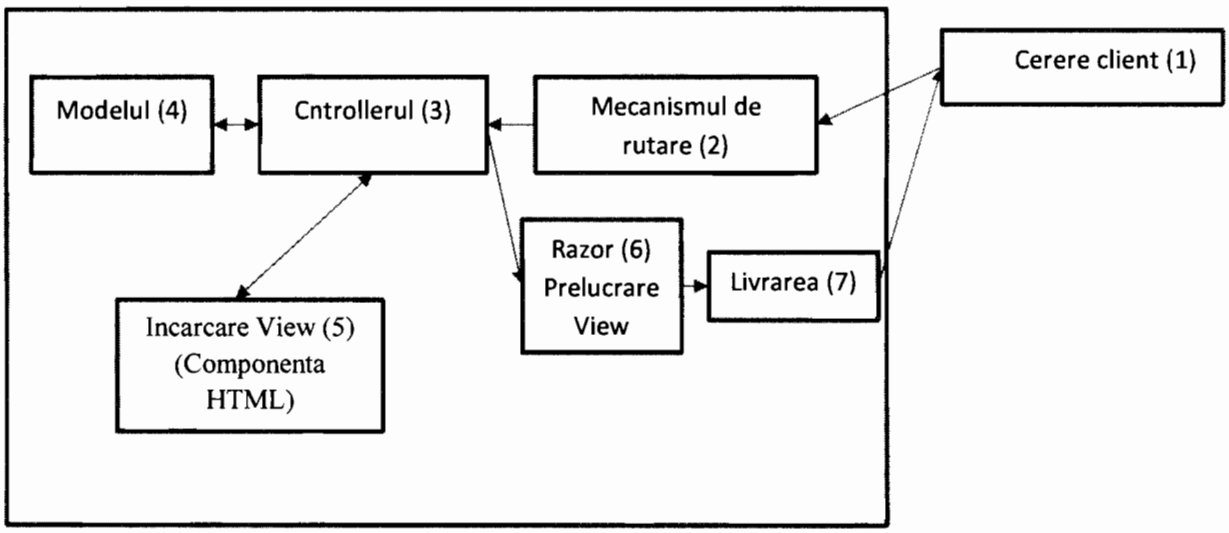
Modelul (4)
Cntrollerul (3)
Mecanismul de rutare (2)
Cerere client (1)
Razor (6) Prelucrare View
Livrarea (7)
Incarcare View (5) (Componenta HTML)

Fig.1.

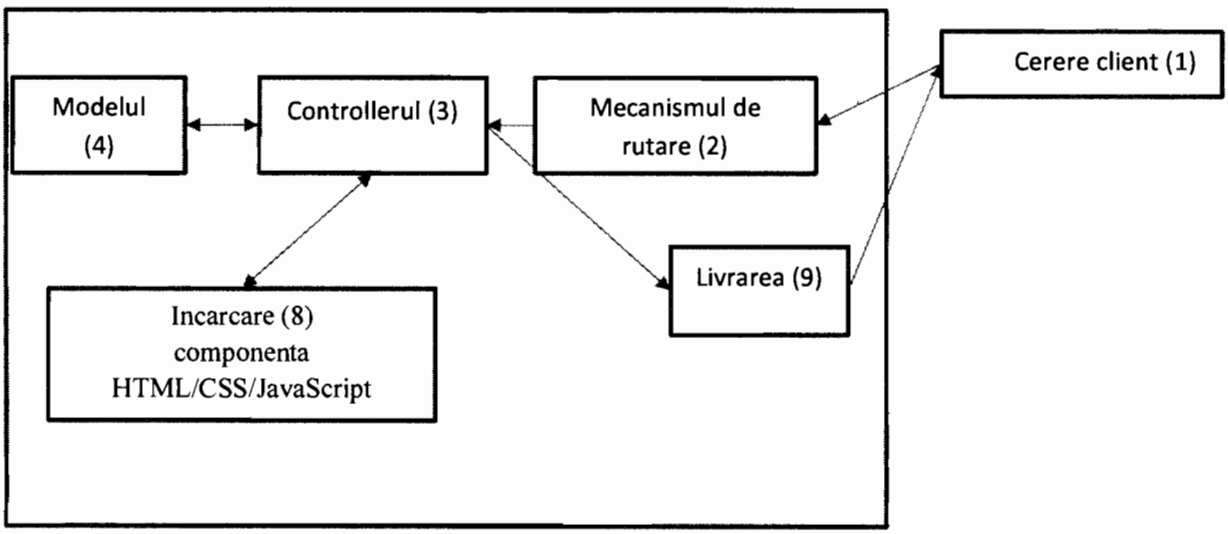
Modelul (4)
Controllerul (3)
Mecanismul de rutare (2)
Cerere client (1)
Livrarea (9)
Incarcare (8) componenta HTML/CSS/JavaScript

Fig. 2.

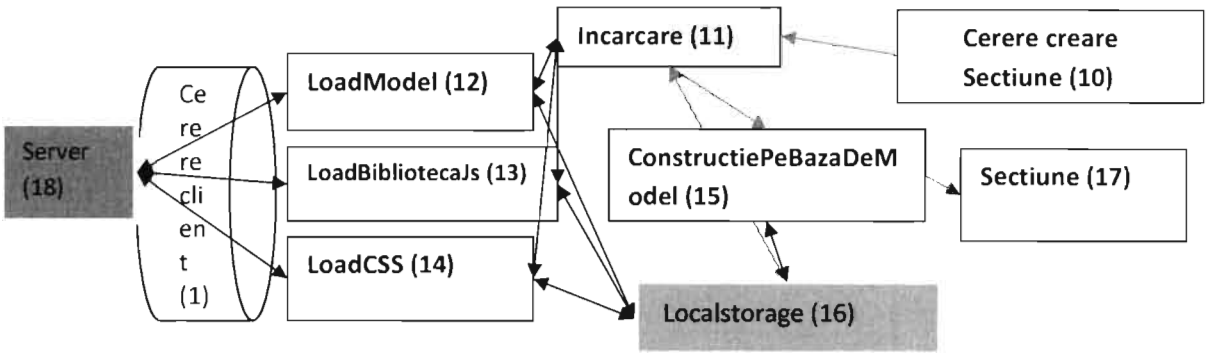

Fig. 3.

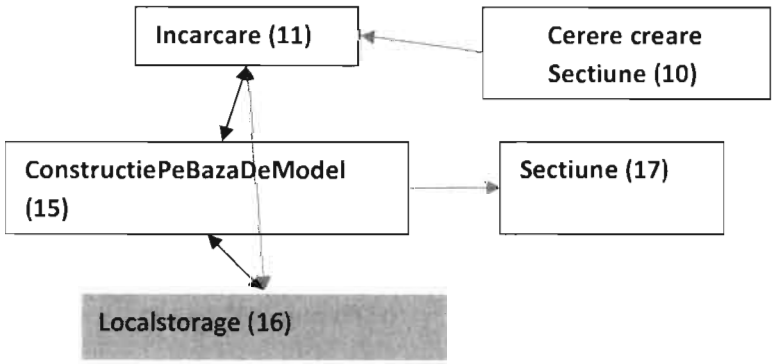

Fig. 4.

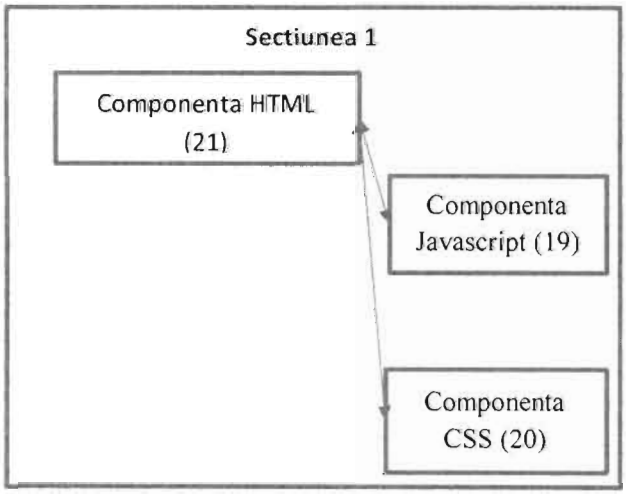

Fig. 5.

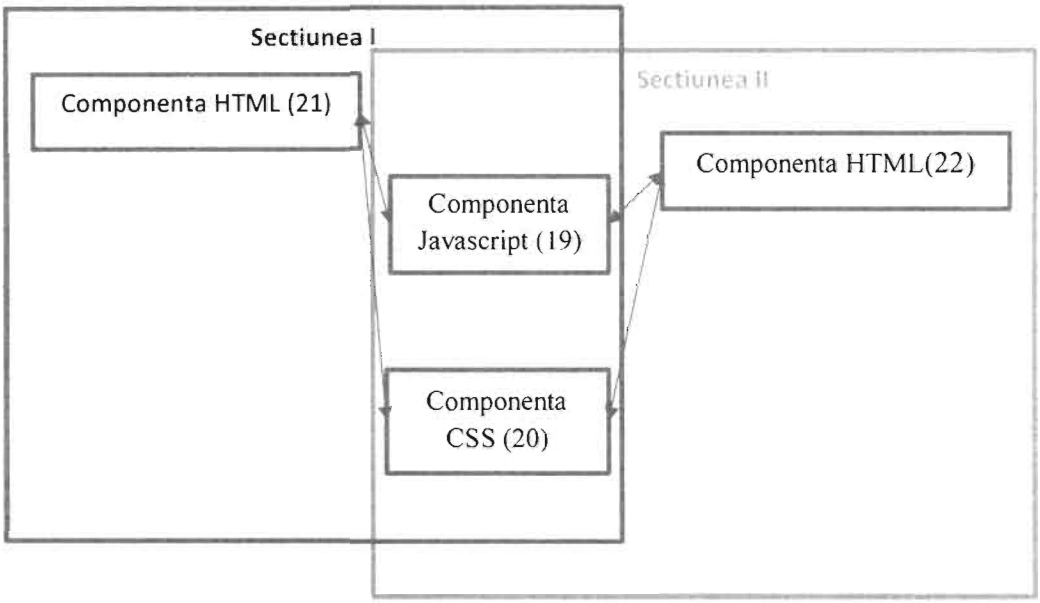
Sectiunea I
Componenta HTML (21)
Sectiunea II
Componenta HTML(22)
Componenta Javascript (19)
Componenta CSS (20)

Fig. 6.

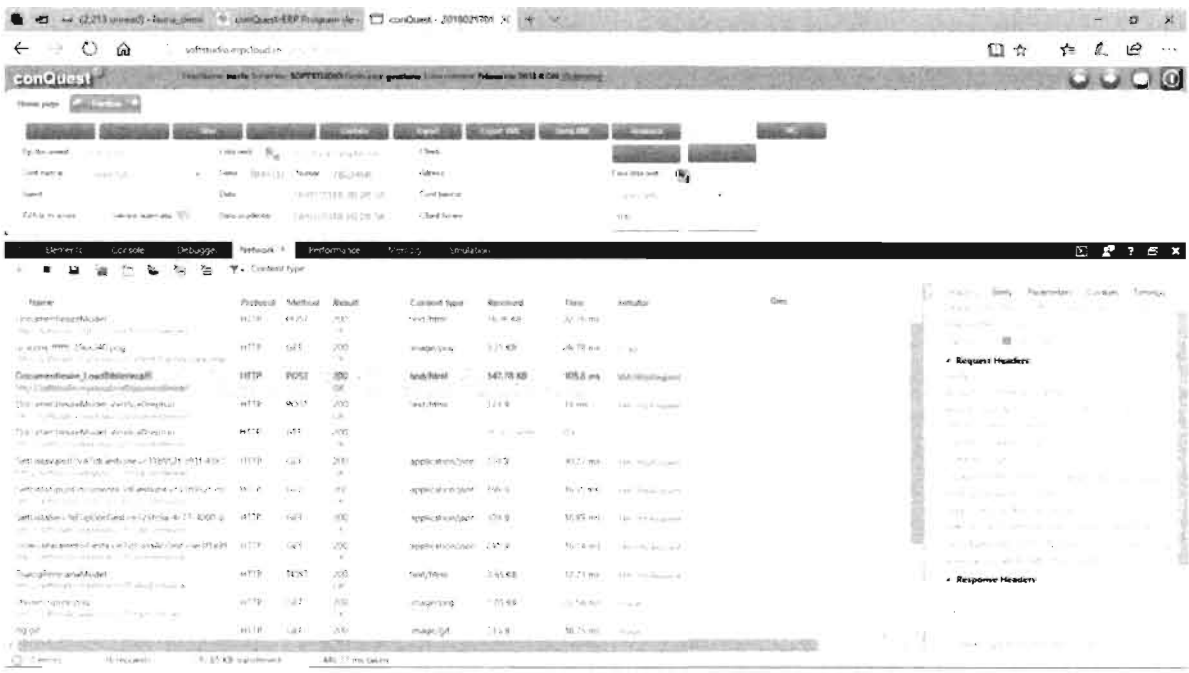
conQuest

Fig. 7.

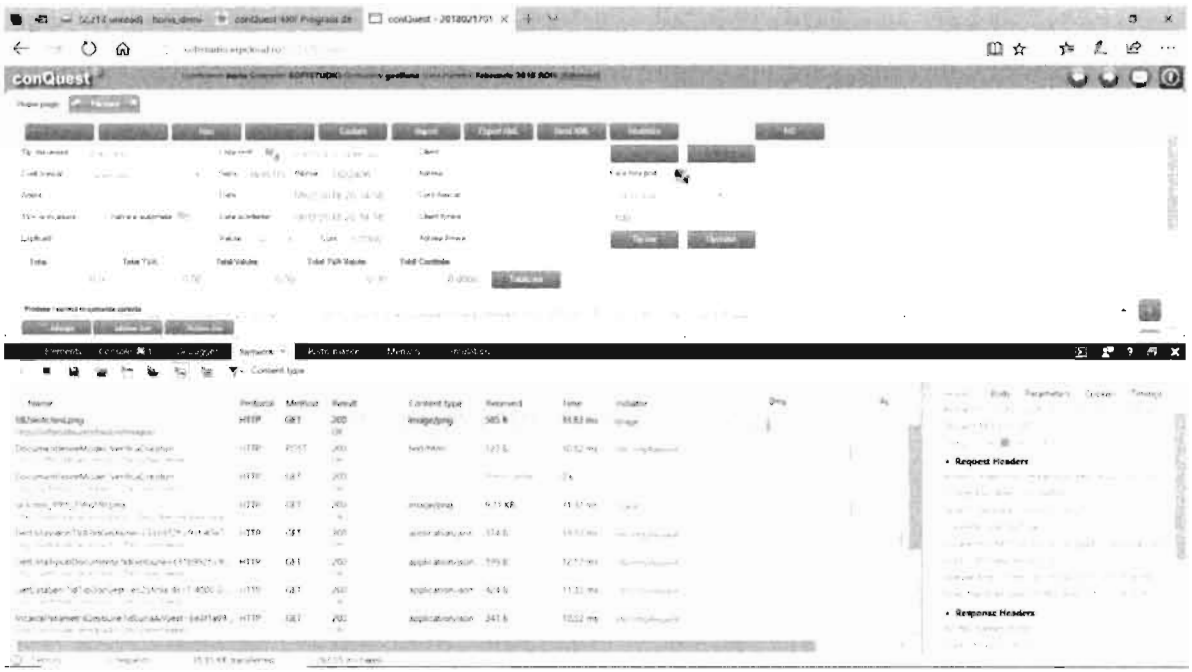

Fig. 8.

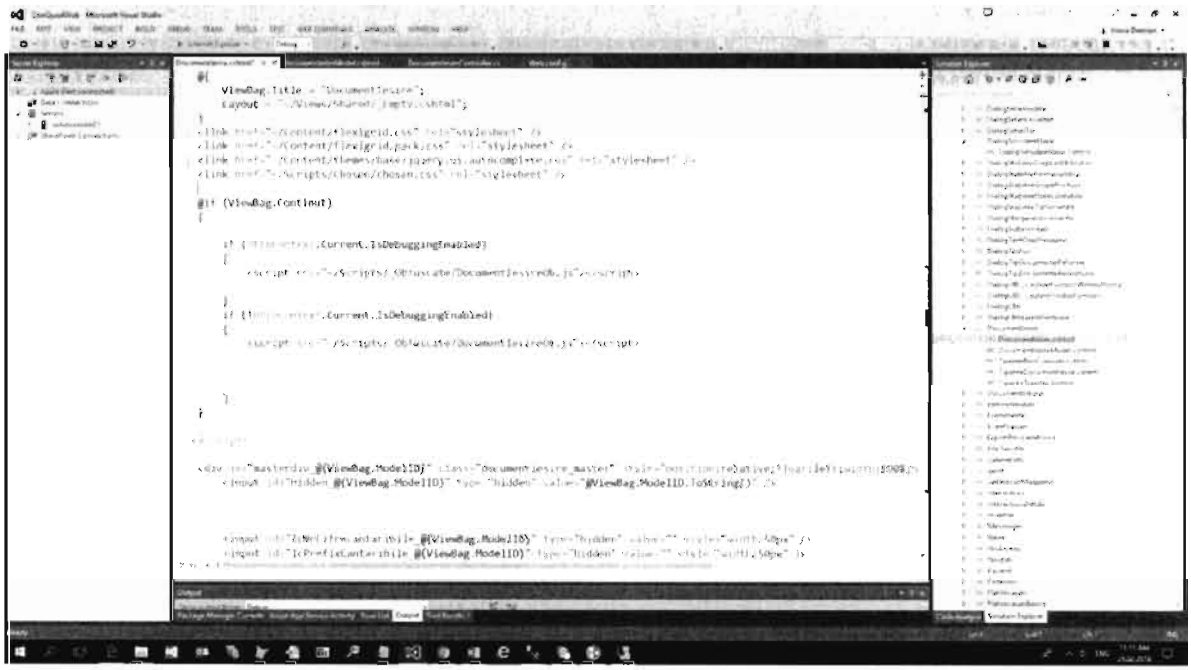

Fig. 9.

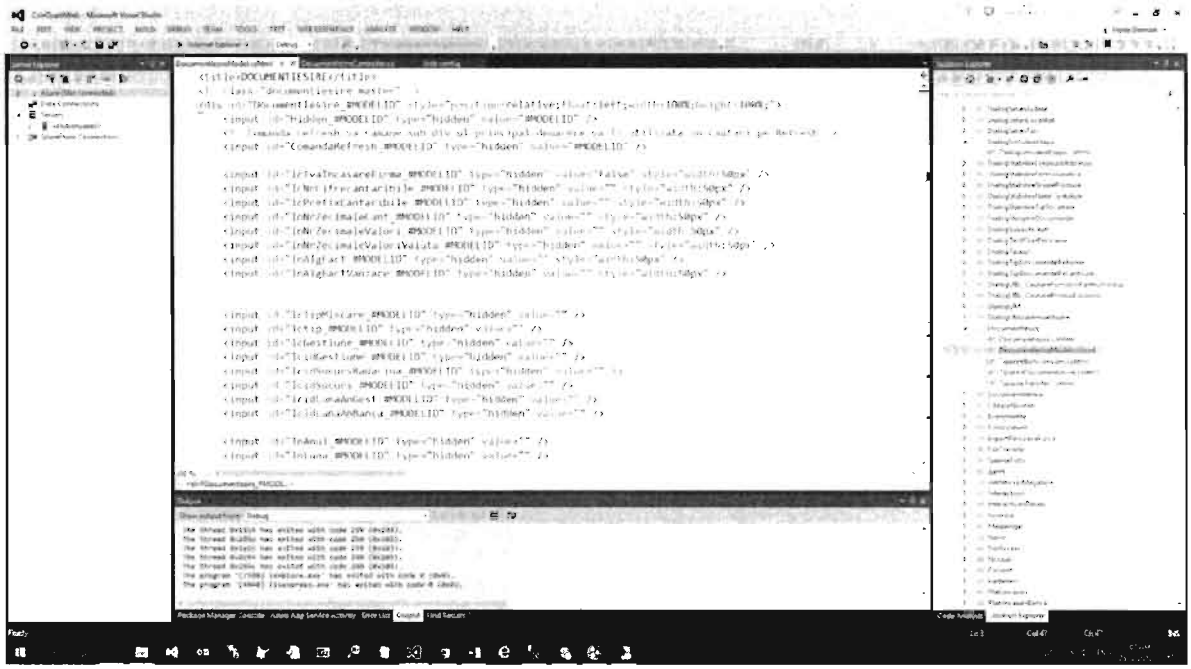

Fig. 10.

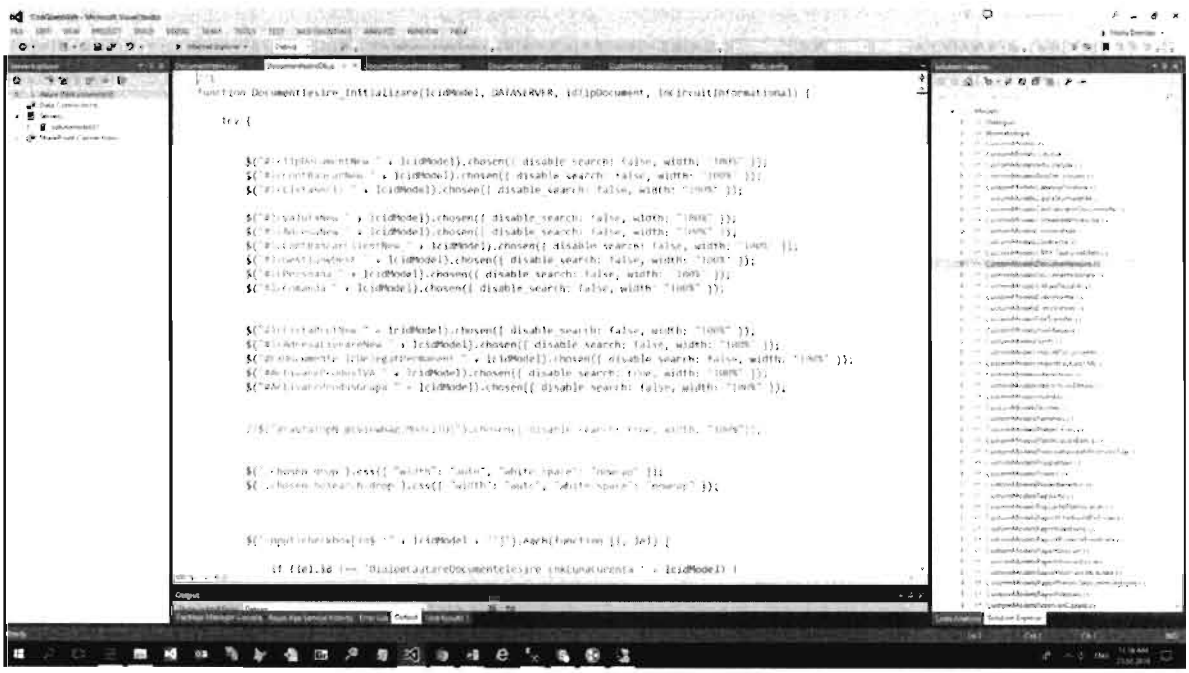

Fig. 11.