**MINISTERUL EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

FACULTATEA CALCULATOARE, INFORMATICǍ

ȘI MICROELECTRONICǍ

Departamentul Informatică și Ingineria Sistemelor

**Raport**

la practica în producție

Tema: Prezentarea unei companii ce prestează servicii software în industria automotive

Conducătorii practicii: \_\_\_\_\_\_\_ / Țurcan Ana /

Executant: \_\_\_\_\_\_ / Serba Cristina /

Chişinǎu 2025

Cuprins

[INTRODUCERE 3](#_Toc197694488)

[1. DESCRIEREA UNITĂȚII ECONOMICE UNDE S-A DESFĂȘURAT STAGIUL DE PRACTICĂ 4](#_Toc197694489)

[**1.1** **Structura și domeniul de activitate a organizației** 4](#_Toc197694490)

[**1.2** **Formularea sarcinii** 4](#_Toc197694491)

[2. DESCRIEREA INSTRUMENTELOR ȘI TEHNOLOGIILOR UTILIZATE 6](#_Toc197694492)

[**2.1 Etapele de proiectare și realizare a unei pagini web** 6](#_Toc197694493)

[**2.2 Descrierea limbajelor utilizate la realizarea unei pagini web** 6](#_Toc197694494)

[**2.2.1 Limbajul HTML** 6](#_Toc197694495)

[**2.2.2 Limbajul CSS** 8](#_Toc197694496)

[**2.2.3 Limbajul JavaScript** 9](#_Toc197694497)

[**2.2.2 Limbajul PHP** 10](#_Toc197694498)

[3. ELABORAREA PAGINII WEB 16](#_Toc197694499)

[**3.1** **Arhitectura generală a paginii web** 16](#_Toc197694500)

[**3.2** **Implementarea funcționalităților** 18](#_Toc197694501)

[**3.2.1** **Conexiunea cu baza de date** 18](#_Toc197694502)

[**3.2.2** **Navigare și layout** 18](#_Toc197694503)

[**3.2.3** **Pagini dinamice** 18](#_Toc197694504)

[**3.2.4** **Responsivitate și stil** 19](#_Toc197694505)

[CONCLUZII 20](#_Toc197694506)

[BIBLIOGRAFIE 21](#_Toc197694507)

[ANEXE 22](#_Toc197694508)

[**ANEXA 1. CODUL SURSĂ** 22](#_Toc197694509)

[**ANEXA 2. ASPECTUL PAGINII WEB** 27](#_Toc197694510)

# INTRODUCERE

Stagiul de practică în producție reprezintă o etapă esențială în formarea profesională a unui student, oferindu-i ocazia de a aplica în mod practic cunoștințele teoretice acumulate și de a se familiariza cu cerințele și dinamica unui mediu real de lucru.

Stagiile de practică sunt parte integrantă obligatorie a procesului educaţional şi se realizează în scopul aprofundării cunoştinţelor teoretice acumulate de către studenţi pe parcursul anului/anilor de studii şi formării competenţelor stabilite de Cadrul Naţional al Calificărilor pe domenii de formare profesională [1].

Practica de specialitate are rolul de a familiariza studenții cu experiența reală din mediul economico-social și a dobândi abilităţi practice printr-o implicare directă alături de specialiștii din domeniul specific pregătirii profesionale.

Sarcinile puse în faţa practicanţilor, în cadrul acestor stagii de practică, prezintă un grad înalt de complexitate şi pot viza, pe lângă însuşirea şi analiza tehnologiilor aplicate la întreprinderea dată, şi aspecte legate de activitatea nivelului inferior de management al întreprinderii.

Prezentul raport își propune să evidențieze structura, activitatea și specificul companiei, precum și cunoștințele și abilitățile dobândite pe parcursul perioadei de practică.

# DESCRIEREA UNITĂȚII ECONOMICE UNDE S-A DESFĂȘURAT STAGIUL DE PRACTICĂ

## **Structura și domeniul de activitate a organizației**

Quasar Development este o companie inovatoare, fondată în august 2022, cu sediul în Chișinău, Republica Moldova. Activitatea principală a organizației este orientată către dezvoltarea de soluții software și hardware pentru industria auto, cu un accent deosebit pe sisteme embedded și arhitectura AUTOSAR.

Structura organizațională este una flexibilă și orientată spre eficiență, fiind formată dintr-o echipă tânără, dinamică și experimentată de 10 persoane, specializată în:

* dezvoltarea de software pentru unități de control electronice (ECU);
* implementarea de tool-uri pentru testare și diagnosticare;
* migrarea firmware-ului legacy pe platforme moderne;
* soluții IoT și dezvoltare de aplicații embedded [2]

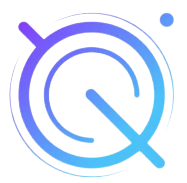


Figura 1. Logo Quasar Development

## **Formularea sarcinii**

Sarcina de bază a practicii în producție este ca studentul la finele practicii să cunoască:

* elaborarea unei aplicații software cu utilizarea limbajelor de programare și tehnologiilor web: HTML, PHP, Java, CSS, C#, JavaScript, JQuery, AJAX etc.;
* verificarea, testarea, validarea şi promovarea aplicațiilor web;
* identificarea etapelor de proiectare şi promovare a aplicațiilor și serviciilor web.

Tema aleasă a fost „Prezentarea unei companii ce prestează servicii software în industria automotive”. Pagina web dezvoltată conține o bară de navigare orizontală cu patru secțiuni principale:

* Acasă – include imaginea principală a paginii, denumirea companiei și logo-ul acesteia, precum și o scurtă prezentare a activităților companiei în domeniul software pentru industria automotive;
* Servicii – prezintă serviciile oferite de companie, cum ar fi dezvoltarea de software pentru sisteme de bord, managementul flotelor auto, soluții de conectivitate și alte tehnologii inovatoare pentru automotive;
* Proiecte – afișează informații despre proiecte terminate și descrierea lor în raport cu domeniul software automotive;
* Contacte – informații de contact și amplasarea companiei pe mapă.

Prin această temă, am aplicat cunoștințele dobândite în practică pentru a crea o aplicație web funcțională și intuitivă, care să prezinte în mod eficient activitatea și serviciile unei companii specializate în software pentru industria automotive.

# DESCRIEREA INSTRUMENTELOR ȘI TEHNOLOGIILOR UTILIZATE

## **2.1 Etapele de proiectare și realizare a unei pagini web**

* Studierea literaturii și surselor informaţionale în domeniul tematicii alese;
* Analiza paginilor web/proiectelor deja existente în domeniul tematicii alese;
* Studierea limbajelor de programare, mediilor de dezvoltare necesare pentru implementarea paginii web.
* Definirea cerinţelor paginii web și criteriilor pe care trebuie să le îndeplinească pagina;
* Stabilirea arhitecturii sistemului: componentele sistemului, interfeţele şi modul lor de comportare.
* Implementarea paginii web;
* Testarea paginii web și eliminarea erorilor.

## **2.2 Descrierea limbajelor utilizate la realizarea unei pagini web**

### **2.2.1 Limbajul HTML**

Unul din primele elemente fundamentale ale WWW (World Wide Web) este ***HTML (Hypertext Markup Language),*** care descrie formatul primar în care documentele sunt distribuite și văzute pe Web. [2]

Orice document HTML începe cu notația **<html>** și se termină cu notația **</html>**. Aceste "chestii" se numesc în literatura de specialitate „TAG-uri”. Prin convenție, toate informațiile HTML încep cu o paranteză unghiulară deschisă **„< ”** și se termină cu o paranteză unghiulară închisă **„>”**. Între cele două marcaje **<html>** și **</html>** vom introduce două secțiuni:

* secțiunea de antet **<head>...</head>**
* corpul documentului **<body>...</body>**. Blocul **<body>...</body>** cuprinde conținutul propriu-zis al paginii HTML, adică ceea ce va fi afișat în fereastra browser-ului.

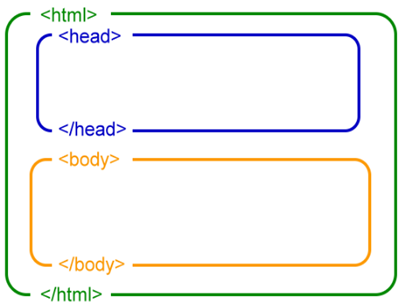


Figura 2 Structura unei pagini HTML

În primul rând, titlul unei pagini se obține inserând în secțiunea **<head>...</head>** a următoarei linii: **<title>Aceasta este prima mea pagina de Web</title>**. În plus, în secțiunea **<body>...</body>** putem scrie texte cât dorim. Dacă nu întâlnim nici un marcaj **<** sau **>** atunci interpretorul HTML le va lua drept texte simple și le va afișa pe ecran.

Pentru ca browser-ul să interpreteze corect caracterele „spațiu”, „tab” și „CR/LF” ce apar în cadrul unui text, acest text trebuie inclus într-un bloc **<pre>...</pre>**.

O culoare poate fi precizată în două moduri:

* Culoarea unei pagini se precizează prin intermediul unui atribut al etichetei **<body>**. Culoarea fondului paginii Web se stabilește cu atributul bgcolor al etichetei **<body>**, de exemplu: **<body bgcolor = culoare>.**
* Culoarea textului se face prin intermediul atributului text al etichetei **<body>** după sintaxa **<body text=culoare>.** O etichetă poate avea mai multe atribute. De exemplu, o etichetă cu trei atribute arată astfel:**<eticheta atribut1 = valoare1 atribut2 = valoare2 atribut3 = valoare3>.**

Pentru ca un bloc de text să apară în pagina evidențiat (cu caractere aldine), trebuie inclus între delimitatorii **<b>...</b>** (b vine de la „bold” = îndrăzneț). Pentru ca un text să fie scris cu caractere mai mari cu o unitate decât cele curente acesta trebuie inclus într-un bloc delimitat de etichetele **<big>...</big>**. Pentru ca un text să fie scris cu caractere mai mici cu o unitate decât cele curente acesta trebuie inclus într-un bloc delimitat de etichetele **<small>...</small>**.

Pentru ca un text să fie scris cu caractere cursive acesta trebuie inclus intr-un bloc delimitat de etichetele **<i>...</i>** (i vine de la „italic”).

Pentru a insera secvențe de text aliniate ca indice (sub-script) sau ca exponent (super-script), aceste fragmente trebuie delimitate de etichetele **<sub>...</sub>**, respectiv **<sup>...</sup>**.

Pentru a insera un bloc de caractere subliniate se utilizează etichetele **<u>...</u>** (u vine de la „underline”). Pentru a insera un bloc de caractere tăiate se utilizează etichetele **<strike>...</strike>** sau **<s>...</s>**.

Formatarea caracterelor, organizarea textului

Un font este caracterizat de următoarele atribute:

* culoare (stabilită prin atributul *color*);
* tipul sau stilul (stabilit prin atributul *face*);
* mărimea (definită prin atributul *size*);
* mărimea în puncte tipografice (stabilită prin atributul *point-size*);
* grosime (definită prin atributul *weight*).

Toate aceste atribute aparțin etichetei **<font>**, care permite inserarea de blocuri de texte personalizate.

Dacă într-o pagină web trebuie inclusă o adresă, atunci putem utiliza facilitățile oferite de o etichetă dedicată: **<address>...</address>**.

Cu ajutorul etichetei paragraf **<p>** este posibil trecerea la o linie nouă și permite:

* inserarea unui spațiu suplimentar înainte de blocul paragraf;
* inserarea unui spațiu suplimentar după blocul paragraf, dacă se folosește delimitatorul **</p>** (acesta fiind opțional);
* alinierea textului cu ajutorul atributului align, având valorile posibile „left”, „center” sau „right”.

Într-un text titlurile (headers) de capitole pot fi introduse cu ajutorul etichetelor **<h1>, <h2>, <h3>, <h4>, <h5> și <h6>**.

Într-o pagină Web pot fi inserate linii orizontale. Acest lucru se face cu ajutorul etichetei **<hr>**. Pentru a configura o linie orizontală se utilizează următorele atribute ale etichetei **<hr>**:

* align - permite alinierea liniei pe orizontală. Valorile posibile sunt " left " ," center " și " right ";
* width - permite alegerea lungimii liniei;
* size - permite alegerea grosimii liniei;
* noshade - când este prezent definește o linie fără umbră;
* color - permite definirea culorii liniei.

Modalitatea cea mai eficientă de delimitare și de formatare a unui bloc de text este folosirea

delimitatorilor **<div>...</div>**. Un parametru foarte foarte util pentru stabilirea caracteristicilor unui bloc **<div>** (diviziune) este align (aliniere). Valorile posibile ale acestui parametru sunt:

* left (aliniere la stânga);
* center (aliniere centrală);
* right (aliniere la dreapta).

Un bloc **<div>...</div>** poate include alte sub-blocuri. În acest caz, alinierea precizată de atributul align al blocului are efect asupra tuturor subblocurilor incluse în blocul **<div>**. Un bloc **<div>...</div>** admite atributul „nowrap” care interzice întreruperea rândurilor de către browser.

Pentru a insera un tabel se folosesc etichetele corespondente **<table>...</table>**. Un tabel este format din rânduri. Pentru a insera un rând într-un tabel se folosesc etichetele **<tr>...</tr>** (de la „table row” = rând de tabel). Folosirea etichetei de sfârșit **</tr>** este opțională.

Adresele absolute identifică locația unui fișier imagine de pe Internet sau alte calculatoare conectate la Internet și au sintaxa: **protocol/nume\_calculator/cale/nume\_fișier**. Un exemplu de adresă URL absolută: <http://web.ceiti.md/images/eu.jpg>.

O adresă relativă identifică locația unei imagini, descriind o cale relativă la directorul curent unde se află fișierul imagine. Adresele relative se referă la localizarea fișierului imagine de pe calculatorul personal și are sintaxa: **cale/nume\_fișier**. Un exemplu de adresă URL relativă: **/../../../friend.jpg** care se mai poate scrie **c:/windows/web/wallpaper/friend.jpg**

### **2.2.2 Limbajul CSS**

CSS (Cascading Style Sheets) sau foide stil în cascadă, reprezintă un mecanism simplu de a adăuga elemente de stil (tipuri de caractere, culori, aliniere, spațiere etc.) unui document web. [4]

O construcție CSS are două părți: un selector, și una sau mai multe declarații:

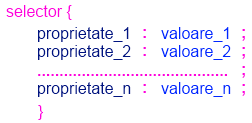
[](https://web.ceiti.md/images/selector-mare-color.png)

Figura 3 Structura construcției CSS

unde:

* selector - identifică elementul HTML, căreia i se aplică stilul dat;
* acoladele - delimitează, care proprietăți și cu ce valori se aplică elementului HTML;
* proprietăți - sunt elementele de aspect care vrem să le modificăm;
* valori ale proprietăților - sunt noile valori pe care vrem să le atribuim proprietății respective.

Fiecare proprietate suportă, anumite tipuri de valori, de exemplu culoarea poate fi dată în trei moduri:

* hexazecimal -**#ffaa99**;
* prin denumire - **red**;
* cu valori rgb - **rgb(123,255,17)**.

O foaie de stil poate fi inserată în trei moduri:

* Foaie externă de stil - este ideală atunci când stilul respectiv se aplică mai multor pagini. Cu o foaie CSS externă, puteți modifica aspectul întregului site modificând un singur fișier.
* Foaie internă de stil*-* trebuie folosită când un singur document are un anumit stil. Foaia internă de stil se definește în secțiunea **head** a paginii HTML folosind tagul **<style>.**
* Stilul inline -pierde multe din avantajele foilor de stiluri amestecând conținutul cu prezentarea. Pentru a utiliza un stil inline, se folosește atributul **style** în tagul respectiv. Atributul **style** poate conține orice proprietăți CSS.

CSS-ul are mai multe tipuri de selectori. Selectorii cei mai des folosiți, prin urmare și cei mai importanți sunt:

* Selectorul element -aplică proprietățile conținute, tuturor elementelor de acest tip.
* Selectorul id - este utilizat pentru a specifica stilul unui singur element. Selectorul ***id*** folosește atributul **id** al unui element HTML, și este definit cu **„#”**.
* Selectorul class -este utilizat pentru a specifica stilul unui grup de elemente. Astfel, poate fi setat un stil particular pentru toate elementele HTML care face parte din aceeași clasă. Selectorul ***class*** folosește atributul HTML **class** și este definit cu „**.**”.

### **2.2.3 Limbajul JavaScript**

**JavaScript** a fost dezvoltat prima dată de către firma **Netscape**, cu numele de **Live Script**, un limbaj de script care extinde capacitățile HTML, oferă o alternativă parțială la utilizarea unui număr mare de scripturi CGI pentru prelucrarea informaţiilor din formulare și care adaugă dinamism în paginile web.

Pentru a insera cod JavaScript într-un document HTML, folosim tagul **<script>** şi respectiv **</script>** [5]. Deci tagurile **<script>** şi **</script>** marchează locul în care începe, respectiv se sfârşeşte scriptul.

În exemplele vechi se utiliza atributul **type** pentru a defini limbajul în care este scris scriptul.

Atributul **type** nu este necesar, JavaScript este limbajul de scripting implicit în HTML. Pentru inserarea script-urilor în documente HTML pot fi folosite cele 4 metode prezentate mai jos:

* plasarea script-ului în antet-ul paginii (între **<head>** şi **</head>**);
* plasarea script-ului în corpul paginii (între **<body>** şi **</body>**);
* utilizarea fişierelor sursă externe;
* crearea unui gestionar de evenimente.

### **2.2.2 Limbajul PHP**

PHP (Hypertext Preprocessor) este un limbaj de scripting de uz general, cu cod-sursă deschis (Open Source), utilizat pe scară largă, și care este potrivit în special pentru dezvoltarea aplicațiilor Web și poate fi integrat în HTML. Sintaxa sa provine din C, Java și Perl și este ușor de învățat. Scopul principal al limbajului este acela de a permite programatorilor web să creeze rapid pagini web generate dinamic, însă cu PHP pot fi realizate mult mai multe. [6]

Pentru a putea rula scenarii php (scripturi), este nevoie de a fi programat calculatorul în calitate de server Web, server de aplicații PHP și nu în ultimul rând server de BD MySQL. Aceste componente software de bază necesare (MySQL, Apache și PHP) sunt gratuite (din categoria „open source”) și pot fi descărcate de pe site-ul oficial al producătorului. După instalarea fiecăruia din aceste aplicații apare necesitatea configurării lor, care uneori este dificilă pentru începători. Pentru a evita problemele configurării, poate fi instalat un pachet complet ce conține Apache + PHP + MySQL + PHPMyAdmin care se numește **XAMPP (PHPTriad, EasyPhp, Base, WampServer, Denver, ...).**Aplicația PHPMyAdmin inclusă în pachetul**XAMPP** permite administrarea comodă a BD MySQL.

Codul PHP trebuie inclus în fișierele ce vor fi deservite utilizatorilor finali și trebuie delimitat de restul conținutului prin tagul de început **<?php** și tagul de sfârșit **?>**. Practic tot ce se află în interiorul acestor etichete va fi interpretat ca fiind cod-sursă PHP (cod ce va fi executat de interpretorul PHP). Textul din afara celor 2 taguri este lăsat neschimbat, fiind ulterior interpretat de browser. Se pot folosi și alte etichete în funcție de configurația serverului web, dar acestea nu sunt recomandate.

Pentru a obține acest text, secvențele de cod trebuie să conțină instrucțiuni explicite care să „spună” intrepretorului ce anume trebuie afișat. Instrucțiunile de afișare sunt **print**, **echo** și **printf**. Există și alte modalități dar aceste 3 instrucțiuni sunt cele mai uzuale și ușor de folosit. Diferențe între ele există, dar ca idee generală **print** și **echo** fac același lucru și poate fi folosită oricare dintre ele. **printf** este folosita mai rar datorită sintaxei oarecum greoaie.

Combinarea codului HTML cu cod PHP se întâmplă foarte des. În interiorul unui script HTML poate exista script PHP delimitat de etichetele **<?php**și **?>**însă fișierul final trebuie să fie PHP.

Variabilele sunt mărimi care pot lua diferite valori. Numele dat unei variabile începe întotdeauna cu simbolul $ urmat de o literă sau caracterul „\_”. Nu poate fi precedat niciodată de o cifră. Restul caracterelor care formează numele variabilei pot fi litere, cifre sau liniuțe de subliniere. În PHP, sunt considerate litere toate caracterele cuprinse între '**a**' și '**z**', cele cuprinse între '**A**' și '**Z**', precum și cele care au codul ASCII cuprins între 127 și 255.

O variabilă se definește astfel: $<variabila>=<valoare>;

Atribuirea de valori se realizează la fel ca și declararea prin intermediul instrucțiunii de atribuire.

De obicei nu e nevoie ca variabilele să fie șterse după ce au fost folosite. Totuși, lucrul acesta se poate face în următoarele moduri:

|  |  |
| --- | --- |
|  | unset( $variabila );  $variabila = null; |

Afișarea tipului de date a variabilei se realizează utilizînd funcția **gettype($var).**

Variabile globale [7]:

* **$GLOBALS** - pot fi accesate toate variabilele globale care sunt accesibile script-ului PHP curent; acest vector este indexat chiar prin numele variabilelor globale. Această variabilă superglobală este accesibilă începînd cu versiunea 3.0.0
* **$\_SERVER** - conține o serie de variabile ale căror valori sunt setate de server-ul web; majoritatea valorilor variabilelor din acest vector depind de mediul de execuție al script-ului curent. A fost introdusă începînd de la versiunea 4.1.0. În versiunile anterioare se folosea **$HTTP\_SERVER\_VAR** .

Elementele acestor variabile sunt cele de mai jos (atenție: în funcție de setările serverului și browserului unele elemente pot lipsi)

* **PHP\_SELF**- conține numele fișierului unde se execută scriptul curent, relativ la rădacină (document root) De exemplu, $\_SERVER['PHP\_SELF'] într-un script la adresa http://example.com/test.php/foo.bar va fi /test.php/foo.bar . Constanta **\_\_FILE\_\_** conține toată calea și numele fișierului curent.
* **argv** - Un array al argumentului trimis către script. Când scriptul rulează în linie de comandă, va da acces C-style la parametrii liniei de comanda. Când este accesat prin metoda GET, acesta va conține un șir de interogare.
* **argc** - Conține un număr al parametrilor liniei de comandă trimise către script (dacă este rulat în linie de comanda).
* **GATEWAY\_INTERFACE** - Returnează versiunea CGI utilizată pe server (exemplu CGI/1.1).
* **SERVER\_NAME** - Numele server-ului de hosting unde se execută scriptul curent. Dacă scriptul rulează pe un host virtual, aceasta va conține valoarea definită pentru acel host virtual.
* **SERVER\_SOFTWARE** - Returnează un șir de identificare al server-ului dat în header ca raspuns la cerere.
* **SERVER\_PROTOCOL** - Numele și versiunea protocolului prin care este cerută pagina (exemplu HTTP/1.0)
* **REQUEST\_METHOD** - returnează metoda utilizată pentru accesarea paginii (exemplu GET, HEAD, POST, PUT)
* **REQUEST\_TIME** - Valoarea timestamp dată la pornirea cererii. Este valabilă începând de la versiunea PHP 5.1.0.
* **QUERY\_STRING** - Șirul de interogare, dacă este, de la pagina care a fost accesată.
* **DOCUMENT\_ROOT** - Directorul radăcină unde este rulat scriptul curent, așa cum este definit în fișierul de configurație al serverului.
* **HTTP\_ACCEPT** - Conținutul header-ului acceptat de la cererea curentă, dacă este unul.
* **HTTP\_ACCEPT\_CHARSET** - Conține setul de caractere acceptate și care este trimis de header-ul cererii respective, dacă există. Exemplu: iso-8859-1,\*,utf-8.
* **HTTP\_ACCEPT\_ENCODING** - Conține tipul de encodare trimis de cerera curentă prin header, dacă există. Exemplu: ' gzip '.
* **HTTP\_ACCEPT\_LANGUAGE** - Conține limbajul trimis de cererea curentă prin header, dacă exista. Exemplu: ' en '.
* **HTTP\_CONNECTION** - Conține informații despre conecsiiunea trimisă de cererea curentă prin header, dacă există. Exemplu: ' Keep-Alive '.
* **HTTP\_HOST** - Conține informații despre host, header-ul de la cererea curentă, dacă există.
* **HTTP\_REFERER** - Adresa paginii (dacă este) de unde a venit userul în pagina curentă. Aceasta este setată de către browser. Nu toate browserele setează această valoare, și chiar unele au posibilitatea de a modifica HTTP\_REFERER așa cum vor. Pe scurt, nu este de încredere.
* **HTTP\_USER\_AGENT** - Conține semnătura browser-ului trimisă prin header la cererea curentă, dacă există. Un exemplu tipic este: Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586) . În afară de asta, poate fi utilizat **get\_browser()** pentru a vedea ce capabilitați are browserul folosit de user și astfel să returnați pagina specifică pentru acel browser.
* **HTTPS** - Setează o valoare dacă scriptul a fost cerut printr-un protocol securizat HTTPS
* **REMOTE\_ADDR** - Adresa de IP a userului care accesează pagina curentă.
* **REMOTE\_HOST** - Numele hostului userului care vede pagina curentă. Este inversul DNS-ului userului bazat pe REMOTE\_ADDR. Atenție: Serverul web trebuie configurat pentru a crea această variabila. De exemplu în Apache va trebui HostnameLookups On în fișierul de configurare httpd.conf. Pentru alte detalii este folosită funcția **gethostbyaddr()**.
* **REMOTE\_PORT** - Portul prin care userul comunică cu serverul web.
* **SCRIPT\_FILENAME** - Calea absolută unde se execută scriptul curent. Atenție: Dacă scriptul este rulat cu CLI, ca și cale relativă, cum ar fi file.php sau ../file.php , $\_SERVER['SCRIPT\_FILENAME'] va conține calea relativă specificată de către user.
* **SERVER\_ADMIN** - Valoarea dată de directiva SERVER\_ADMIN (pentru Apache) în fișierul de configurare al web serverului. Dacă scriptul rulează pe un host virtual, acesta va conține valoarea definită pentru acel host virtual.
* **SERVER\_PORT** - Portul de pe server care este folosit pentru comunicare pe web. Pentru setările standard, acesta este ' 80 '; pentru SSL, de exemplu, se va schimba cu oricare port definit în transmisia securizată HTTP.
* **SERVER\_SIGNATURE** - Conține un șir care cuprinde versiunea serverului și numele hostului virtual dacă este.
* **PATH\_TRANSLATED** - Calea de bază a sistemul de fișiere (nu rădacina documentului) de la scriptul curent, după ce serverul a terminat maparea oricăror cai ( virtual-to-real mapping) Atenție: Începînd de la PHP 4.3.2, **PATH\_TRANSLATED** nu mai este setat implicit în Apache 2 SAPI în contrast cu situația din Apache 1, unde este setată aceeași valoare ca și la **SCRIPT\_FILENAME**. Această schimbare a fost făcută pentru a fi compatibilă cu specificațiile CGI în care **PATH\_TRANSLATED** trebuie să existe doar dacă **PATH\_INFO** este definit. Utilizatorii Apache 2 pot utiliza AcceptPathInfo = On în fișierul de configurație httpd.conf pentru a defini **PATH\_INFO** .
* **SCRIPT\_NAME** - Conține calea către scriptul curent. Acesta este util pentru pagini care trebuiesc definite către ele însuși. Constanta \_\_FILE\_\_ conține calea completă și numele de fișier al scriptului curent.
* **REQUEST\_URI** - URI care este dat pentru accesarea paginii curente, de exemplu /index.html.
* **PHP\_AUTH\_DIGEST** - Când este rulată sub Apache ca modul acțiunea Digest HTTP, valoarea acestei variabile este setată în 'Authorization' header trimis către client (care ar trebui utilizat pentru a face o validare corespunzătoare).
* **PHP\_AUTH\_USER** - Când este rulat sub Apache sau IIS (ISAPI pe PHP 5) ca modul, actiunea HTTP authentication, această variabilă conține numele userului trimis de către user.
* **PHP\_AUTH\_PW** - Când este rulat sub Apache sau IIS (ISAPI pe PHP 5) ca modul, actiunea HTTP authentication, această variabilă conține parola userului trimisă de către user.
* **AUTH\_TYPE** - Când este rulat sub Apache ca modul acțiunea HTTP authenticated , această variabilă seteaza tipul de autentificare.
* Variabilele pentru trimiterea datelor
* **$\_GET** este o variabilă array globală. A fost introdusă în versiunea 4.1.0. Pentru versiunile anterioare se folosește **$HTTP\_GET\_VARS** care nu este variabilă globală. Poate fi folosită **$\_GET** pentru a trimite variabile cu valori prin intermediul linkurilor.
* **$\_POST** este de asemenea o variabilă array globală. A fost introdusă în versiunea 4.1.0. Pentru versiunile anterioare se folosește **$HTTP\_POST\_VARS** care nu este variabilă globală. Poate fi folosită **$\_POST** pentru a trimite variabile cu valori prin intermediul formularelor.
* **$\_GET**și **$\_POST** conțin variabile primite de script prin intermediul unor transferuri care folosesc metodele HTTP get, respectiv post. De exemplu, prin intermediul acestor vectori pot fi accesate valorile câmpurilor dintr-un formular care a fost completat și transmis folosind una dintre cele două metode.
* **$\_COOKIE** conține valorile variabilelor care conțin informații referitoare la cookie-urile păstrate pe calculatorul utilizatorului care accesează pagina web. A fost introdus în versiunea 4.1.0. În versiunile anterioare, este utilizat **$HTTP\_COOKIE\_VARS** .
* **$\_FILES** conține variabile primite de script prin intermediul încărcărilor de fișiere prin metoda post. A fost introdus în versiunea 4.1.0. În versiunile anterioare se folosește **$HTTP\_POST\_FILES** care nu este variabilă globală.
* **$\_ENV** conține variabile disponibile prin intermediul mediului în care este executat. A fost introdus în versiunea 4.1.0. În versiunile anterioare, este utilizat **$HTTP\_ENV\_VARS** .
* **$\_REQUEST** conține variabile disponibile prin intermediul oricărui tip de mecanism cu ajutorul căruia utilizatorul poate introduce date. Este de tip array și conține valorile variabilelor **$\_GET** , **$\_POST** , și **$\_COOKIE**. A fost introdusă în versiunea 4.1.0. Începând de la versiunea 4.3.0, informațiile din **$\_FILES** au fost de asemenea incluse în **$\_REQUEST**
* **$\_SESSION**conține variabile care corespund sesiunii curente a script-ului, sub formă de array. A fost introdusă în versiunea 4.1.0 și este de asemenea globală. În versiunile anterioare se folosește **$HTTP\_SESSION\_VARS**.

Structura **if-** una dintre cele mai importante structuri în orice limbaj este cea alternativă. În PHP sintaxa acestei structuri este următoarea: if (conditie) {instructiuni;}. Construcția elsese execută dacă valoarea expresiei condiție este **false**. În PHP poate fi utilizată o construcție de tipul else în acest scop. Sintaxa este:

if (conditie) {instructiuni #1;} else {instructiuni #2;}

Uneori, este nevoie de a folosi structuri alternative imbricate în diferite scopuri. Folosind construcții de tipul **if - else** se va obține o secvență de tipul:

|  |  |
| --- | --- |
|  | if (conditie #1) {instructiuni #1;}  else if (conditie #2) {instructiuni #2;}  ...  else if (conditie #n) {instructiuni #n;}  else {instructiuni #n+1;} |

În anumite cazuri trebuie efectuate anumite operații în funcție de anumite valori ale unei expresii. O soluție ar fi folosirea unor structuri if imbricate sau a uneia singure în care apar mai multe construcții elseif. [8]

Majoritatea script-urilor PHP vor conține cicluri. Pentru a folosi o buclă anterior condiționată este utilizată structura while. Sintaxa acestei structuri este:

|  |  |
| --- | --- |
|  | while (conditie) {instructiuni;} |

Uneori, este nevoie să fie folosite bucle posterior condiționate. În acest scop, este structura do - while a cărei sintaxă este:

|  |  |
| --- | --- |
|  | do {instructiuni;} while (conditie); |

Structura for este foarte asemănătoare cu cea din limbajele C/C++ și Java [2] și anume:

|  |  |
| --- | --- |
|  | for (expresie #1; conditie; expresie #2) {    instructiuni;   } |

# ELABORAREA PAGINII WEB

Pentru a reflecta identitatea și activitatea companiei Quasar Development [3], s-a realizat o pagină web modernă și adaptabilă, menită să prezinte într-un mod clar serviciile oferite, membrii echipei, proiectele desfășurate și datele de contact. Site-ul este dezvoltat utilizând tehnologii web standard precum HTML, CSS, PHP, MySQL și Bootstrap pentru un design responsive.

## **Arhitectura generală a paginii web**

Arhitectura paginii web urmează o structură modulară, separând logic conținutul dinamic de interfața grafică. Aceasta este organizată pe baza principiului separației între conținut, stil și funcționalitate, având următoarele componente principale:

1. Frontend (interfața utilizatorului):

Realizat în HTML5 și stilizat cu CSS și Bootstrap, permite o navigare intuitivă și o afișare atractivă a informațiilor. Include componente comune reutilizabile, cum ar fi header.php și footer.php, care sunt incluse în toate paginile. Este structurat în mai multe pagini:

* index.php – Pagina principală (prezentare generală)
* despre.php – Informații despre companie (misiune, viziune, echipă)
* servicii.php – Lista serviciilor oferite (ex: dezvoltare bootloader, firmware)
* proiecte.php – Portofoliu de proiecte realizate
* contact.php – Date de contact și hartă integrată

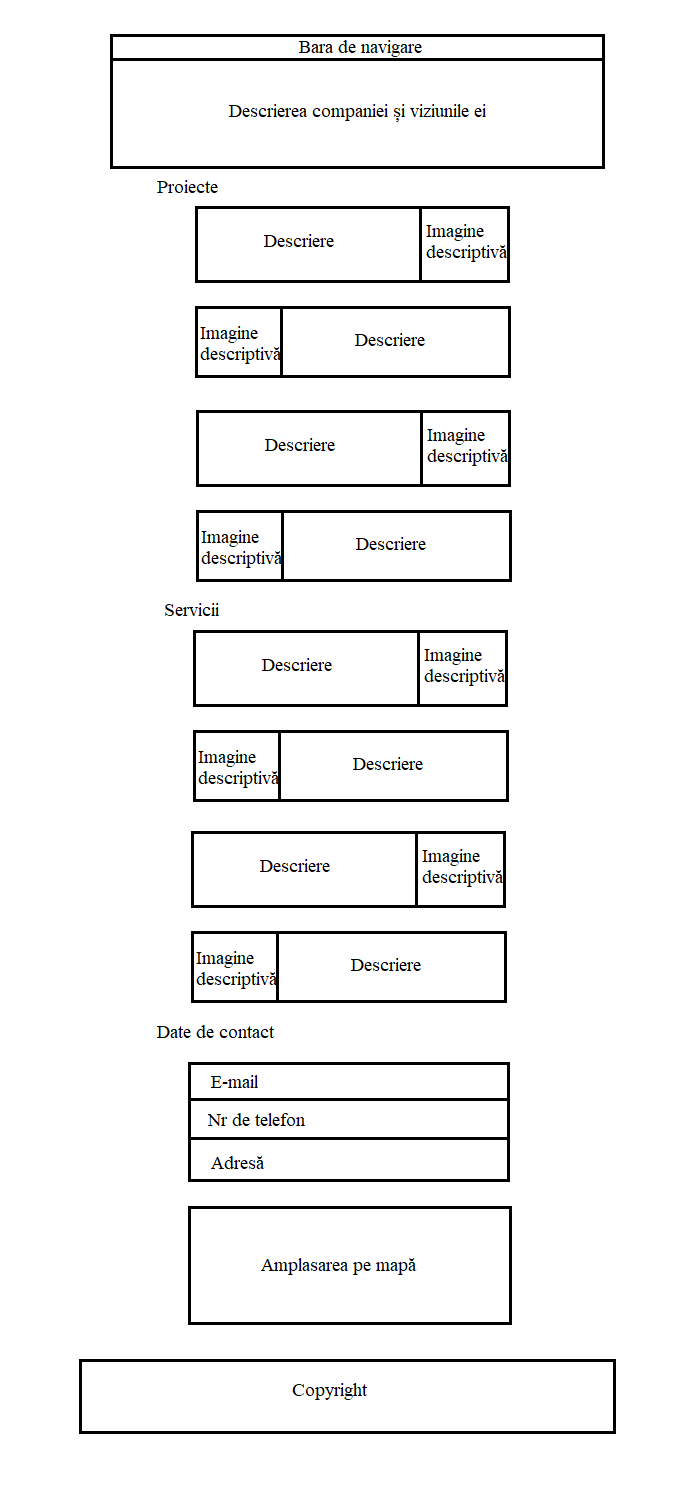


Figura 4.1 Structura arhitecturală a paginii web

1. Backend (funcționalitatea dinamică):

Scris în PHP, gestionează conexiunea la baza de date și extrage informațiile necesare din tabelele MySQL.

Codul este organizat pentru a separa logica de prezentare (ex: interogările SQL sunt gestionate într-un fișier de tip db\_connect.php).

1. Baza de date:

Realizată în MySQL și conține tabele precum:

* companie (pentru misiune, viziune, imagine reprezentativă)
* servicii (titlu, descriere, imagine)
* proiecte (titlu, descriere, imagine)
* contacte (email, telefon, adresă, link Google Maps)

Aceasta permite actualizarea conținutului dinamic fără modificarea codului sursă.

1. Design responsive:

Utilizarea Bootstrap asigură compatibilitatea cu diferite dispozitive (desktop, tabletă, mobil). Imaginile sunt afișate în carduri cu dimensiuni fixe, pentru uniformitate vizuală.

## **Implementarea funcționalităților**

Pentru realizarea funcțională a paginii web, s-au implementat mai multe componente dinamice care preiau și afișează date din baza de date, contribuind la o administrare eficientă și o prezentare clară a informațiilor despre companie. Toate paginile sunt interconectate și au un design unitar, cu elemente reutilizabile (header și footer), pentru coerență vizuală.

### **Conexiunea cu baza de date**

Toate fișierele PHP folosesc un fișier comun, db\_connect.php, pentru realizarea conexiunii la baza de date MySQL. Astfel, codul este mai ușor de întreținut, iar eventualele modificări în parametrii de conexiune nu trebuie repetate în fiecare pagină.

### **Navigare și layout**

Navigarea între pagini este realizată printr-un meniu plasat în antet (header.php), care este inclus în fiecare fișier. Footer-ul (footer.php) conține informații generale și este fixat în partea de jos a ecranului indiferent de dimensiunea conținutului.

### **Pagini dinamice**

* Pagina „Despre Noi” (despre.php)

Această pagină preia din baza de date informații despre misiunea și viziunea companiei, precum și o imagine reprezentativă. Conținutul este afișat în mod structurat, cu titluri și paragrafe pentru lizibilitate crescută.

* Pagina „Servicii” (servicii.php)

Afișează toate serviciile oferite de companie (ex: bootloader software, firmware, încălzitoare pentru automobile), fiecare într-un card Bootstrap cu imagine, titlu și descriere. Imaginile au dimensiuni uniforme și sunt tăiate automat (object-fit: cover) pentru un aspect profesional.

* Pagina „Proiecte” (proiecte.php)

Similară cu pagina serviciilor, această secțiune oferă o prezentare vizuală a proiectelor realizate, folosind carduri dinamice. Fiecare card conține imagine, titlu și descriere, toate extrase din baza de date.

* Pagina „Contact” (contact.php)

Prezintă informațiile de contact (adresă, e-mail, telefon), precum și o hartă Google Maps integrată pentru localizare. Datele provin dintr-un singur rând al tabelei contacte.

### **Responsivitate și stil**

Website-ul utilizează framework-ul Bootstrap pentru a se adapta automat la diferite dimensiuni de ecran. Totodată, fișierul styles.css personalizat extinde stilurile Bootstrap cu culori și poziționări specifice identității vizuale a companiei.

Figura 4.2 Structura generală

# CONCLUZII

În urma desfășurării practicii în producție, am reușit să îndeplinesc sarcinile propuse, dobândind cunoștințe și abilități practice în dezvoltarea aplicațiilor software și a tehnologiilor web. Tema aleasă, **„Prezentarea unei companii ce prestează servicii software în industria automotive”**, mi-a oferit oportunitatea de a aplica limbaje de programare și tehnologii precum HTML, CSS, SQL și PHP pentru a crea o pagină web funcțională și informativă.

Pagina web dezvoltată include o bară de navigare orizontală cu patru secțiuni principale: **Acasă**, **Proiecte**, **Servicii** și **Contact**.

* Secțiunea **Acasă** conține imaginea principală a paginii, denumirea companiei și o scurtă prezentare a activităților companiei în domeniul software pentru industria automotive.
* Secțiunea **Proiecte** prezintă imagini și descrieri ale proiectelor realizate de companie, demonstrând soluții inovatoare în domeniul automotive.
* Secțiunea **Servicii** detaliază serviciile oferite de companie, evidențiind expertiza în dezvoltarea de software specializat pentru industria automotive.
* Secțiunea **Contact** oferă utilizatorilor datele de contact ale companiei, precum și o hartă cu locația sediului principal.

Am învățat să verific, să testez și să validez aplicația web, asigurându-mă că aceasta funcționează corect pe diferite dispozitive și browsere. Prin această temă, am explorat modul în care tehnologiile web pot fi utilizate pentru a promova servicii și proiecte specifice industriei automotive, oferind o experiență interactivă și ușor de navigat pentru utilizatori.

Această experiență mi-a consolidat abilitățile tehnice și mi-a oferit o perspectivă practică asupra procesului de dezvoltare și promovare a aplicațiilor web.

# BIBLIOGRAFIE

[1] SUDACEVSCHI V., ABABII V., ȚURCAN A. Ghid metodic pentru organizarea și susținerea stagiilor de practică de specialitate. Ch: UTM, 2020

[2] Services - QUASAR Development [Sursă electronică <https://quasar-development.com/our-services/>]

[3] HTML: HyperText Markup Language [Sursă electronică <https://developer.mozilla.org/en-US/docs/Web/HTML>]

[4] CSS Syntax - W3Schools [Sursă electronică <https://www.w3schools.com/css/css_syntax.asp>]

[5] Script Tag in JavaScript - Insert JavaScript Into HTML [Sursă electronică <https://hackernoon.com/insert-javascript-into-html-using-the-script-tag-u91k355d?>]

[6] PHP Introduction [Sursă electronică <https://www.w3schools.com/php/php_intro.asp>]

[7] PHP: Hypertext Preprocessor [Sursă electronică <https://www.php.net/>]

[8] Control Structures - Manual - PHP [Sursă electronică <https://www.php.net/manual/en/language.control-structures.php>[PHP+4](https://www.php.net/manual/en/language.control-structures.php?utm_source=chatgpt.com)]

# ANEXE

## **ANEXA 1. CODUL SURSĂ**

Tabelul 1.1 Codul sursă al index.php

|  |
| --- |
| <?php include 'header.php'; ?>  <section class="hero">      <div class="hero-content">          <h1>Bun venit la <?php echo $companie['nume']; ?></h1>          <p>Noi transformăm ideile în realitate.</p>      </div>  </section>  <section class="who-we-are">      <div class="container">          <div class="who-we-are-content">              <div class="who-we-are-text">                  <h2>Cine Suntem</h2>                  <p>                      Suntem o echipă de profesioniști dedicați, pasionați de tehnologie și inovație.                      Cu o experiență vastă în domeniu, ne concentrăm pe oferirea de soluții personalizate                      care să răspundă nevoilor clienților noștri. Misiunea noastră este să aducem valoare                      prin inovație și să contribuim la succesul partenerilor noștri.                  </p>                  <p>                      Prin abordarea noastră colaborativă și orientată pe rezultate, ne asigurăm că fiecare                      proiect este realizat la cele mai înalte standarde de calitate.                  </p>              </div>              <div class="who-we-are-image">                  <img src="images/embedded.jpeg" alt="Cine Suntem" loading="lazy">              </div>          </div>      </div>  </section>  <section class="about">      <div class="container">          <h2>Despre Noi</h2>          <div class="about-content">              <div class="about-text">                  <p><?php echo $companie['misiune']; ?></p>                  <p><?php echo $companie['viziune']; ?></p>              </div>              <div class="about-image">                  <img src="images/embedded2.jpeg" alt="Despre Noi" loading="lazy">              </div>          </div>      </div>  </section>  <?php include 'footer.php'; ?> |

Tabelul 1.2 Codul sursă al proiecte.php

|  |
| --- |
| <?php include 'header.php'; ?>  <section>      <div class="container">          <h1>Proiectele Noastre</h1>          <?php          $sql = "SELECT \* FROM proiecte";          $result = $conn->query($sql);          $count = 0; // Contor pentru alternare          while ($proiect = $result->fetch\_assoc()) {              $class = ($count % 2 == 0) ? 'left' : 'right'; // Alternare stânga/dreapta              $count++;          ?>              <div class="item <?php echo $class; ?>">                  <div class="item-imagine">                      <img src="<?php echo $proiect['imagine']; ?>" alt="<?php echo $proiect['titlu']; ?>" loading="lazy">                  </div>                  <div class="item-content">                      <h2><?php echo $proiect['titlu']; ?></h2>                      <p><?php echo $proiect['descriere']; ?></p>                  </div>              </div>          <?php } ?>      </div>  </section>  <?php include 'footer.php'; ?> |

Tabelul 1.3 Codul sursă al servicii.php

|  |
| --- |
| <?php include 'header.php'; ?>  <section>      <div class="container">          <h1>Serviciile Noastre</h1>          <?php          $sql = "SELECT \* FROM servicii";          $result = $conn->query($sql);          $count = 0; // Contor pentru alternare          while ($serviciu = $result->fetch\_assoc()) {              $class = ($count % 2 == 0) ? 'left' : 'right'; // Alternare stânga/dreapta              $count++;          ?>              <div class="item <?php echo $class; ?>">                  <div class="item-imagine">                      <img src="<?php echo $serviciu['imagine']; ?>" alt="<?php echo $serviciu['titlu']; ?>" loading="lazy">                  </div>                  <div class="item-content">                      <h2><?php echo $serviciu['titlu']; ?></h2>                      <p><?php echo $serviciu['descriere']; ?></p>                  </div>              </div>          <?php } ?>      </div>  </section>  <?php include 'footer.php'; ?> |

Tabelul 1.4 Codul sursă al header.php

|  |
| --- |
| <?php  // Include the database connection  include 'db\_connect.php';  // Fetch company data  $sql = "SELECT \* FROM companie LIMIT 1";  $result = $conn->query($sql);  $companie = $result->fetch\_assoc();  ?>  <!DOCTYPE html>  <html lang="ro">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title><?php echo $companie['nume']; ?></title>      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">      <link rel="stylesheet" href="styles.css">  </head>  <body>  <header>      <h1><?php echo $companie['nume']; ?></h1>      <nav>          <ul>              <li><a href="index.php" <?php echo basename($\_SERVER['PHP\_SELF']) == 'index.php' ? 'class="active"' : ''; ?>>Acasă</a></li>              <li><a href="proiecte.php" <?php echo basename($\_SERVER['PHP\_SELF']) == 'proiecte.php' ? 'class="active"' : ''; ?>>Proiecte</a></li>              <li><a href="servicii.php" <?php echo basename($\_SERVER['PHP\_SELF']) == 'servicii.php' ? 'class="active"' : ''; ?>>Servicii</a></li>              <li><a href="contact.php" <?php echo basename($\_SERVER['PHP\_SELF']) == 'contact.php' ? 'class="active"' : ''; ?>>Contact</a></li>          </ul>      </nav>  </header> |

Tabelul 1.1 Codul sursă al contact.php

|  |
| --- |
| <?php include 'header.php'; ?>  <section>      <div class="container">          <h1>Contact</h1>          <?php          $sql = "SELECT \* FROM contacte LIMIT 1";          $result = $conn->query($sql);          $contact = $result->fetch\_assoc();          ?>          <div class="contact-info">              <p><strong>Email:</strong> <?php echo $contact['email']; ?></p>              <p><strong>Telefon:</strong> <?php echo $contact['telefon']; ?></p>              <p><strong>Adresă:</strong> <?php echo $contact['adresa']; ?></p>          </div>          <div class="map-wrapper">              <iframe src="<?php echo $contact['google\_maps\_link']; ?>" width="100%" height="400" style="border:0;" allowfullscreen="" loading="lazy"></iframe>          </div>      </div>  </section>  <?php include 'footer.php'; ?> |

## **ANEXA 2. ASPECTUL PAGINII WEB**

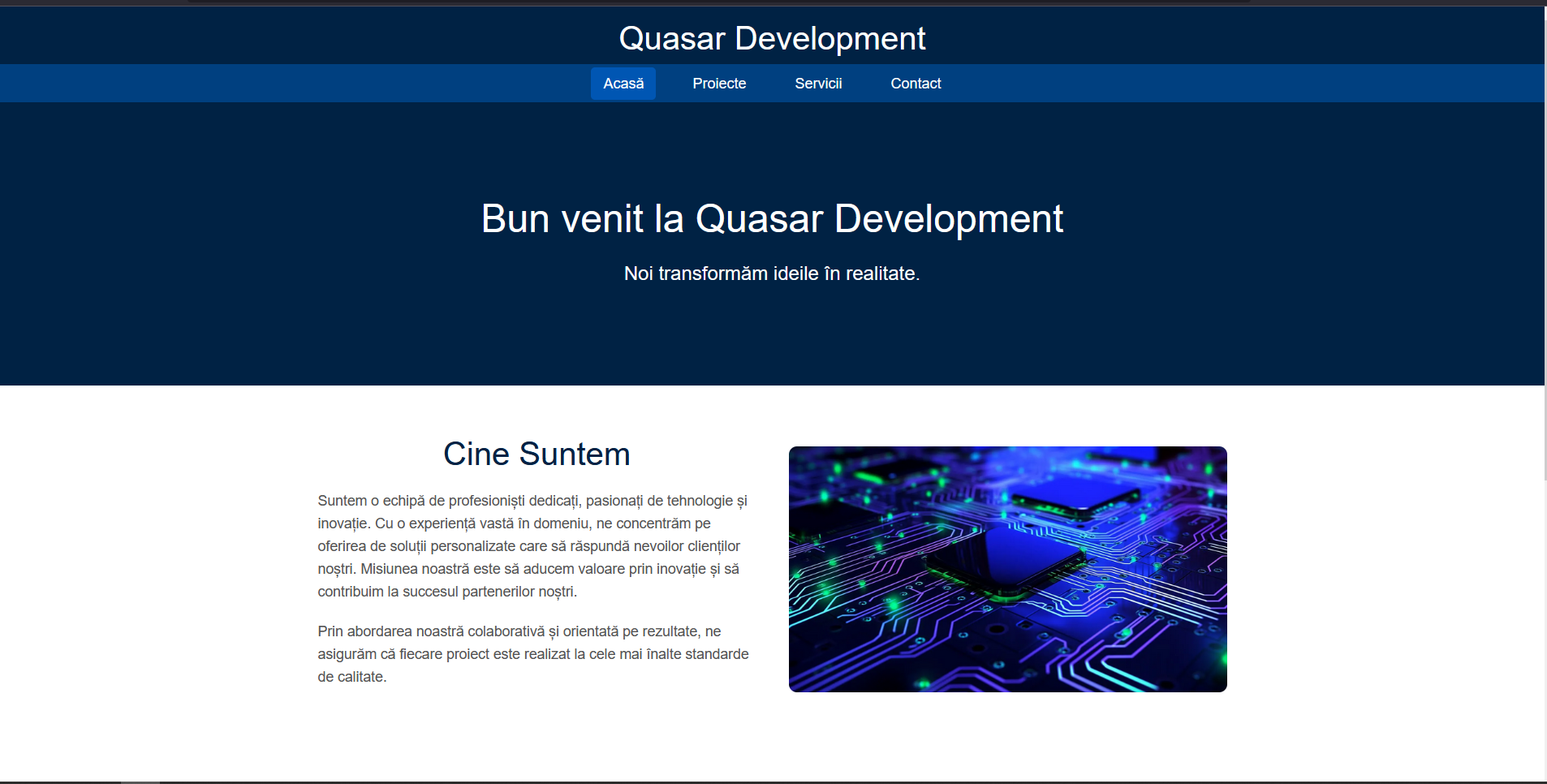


Figura 2.1 Pagina principală

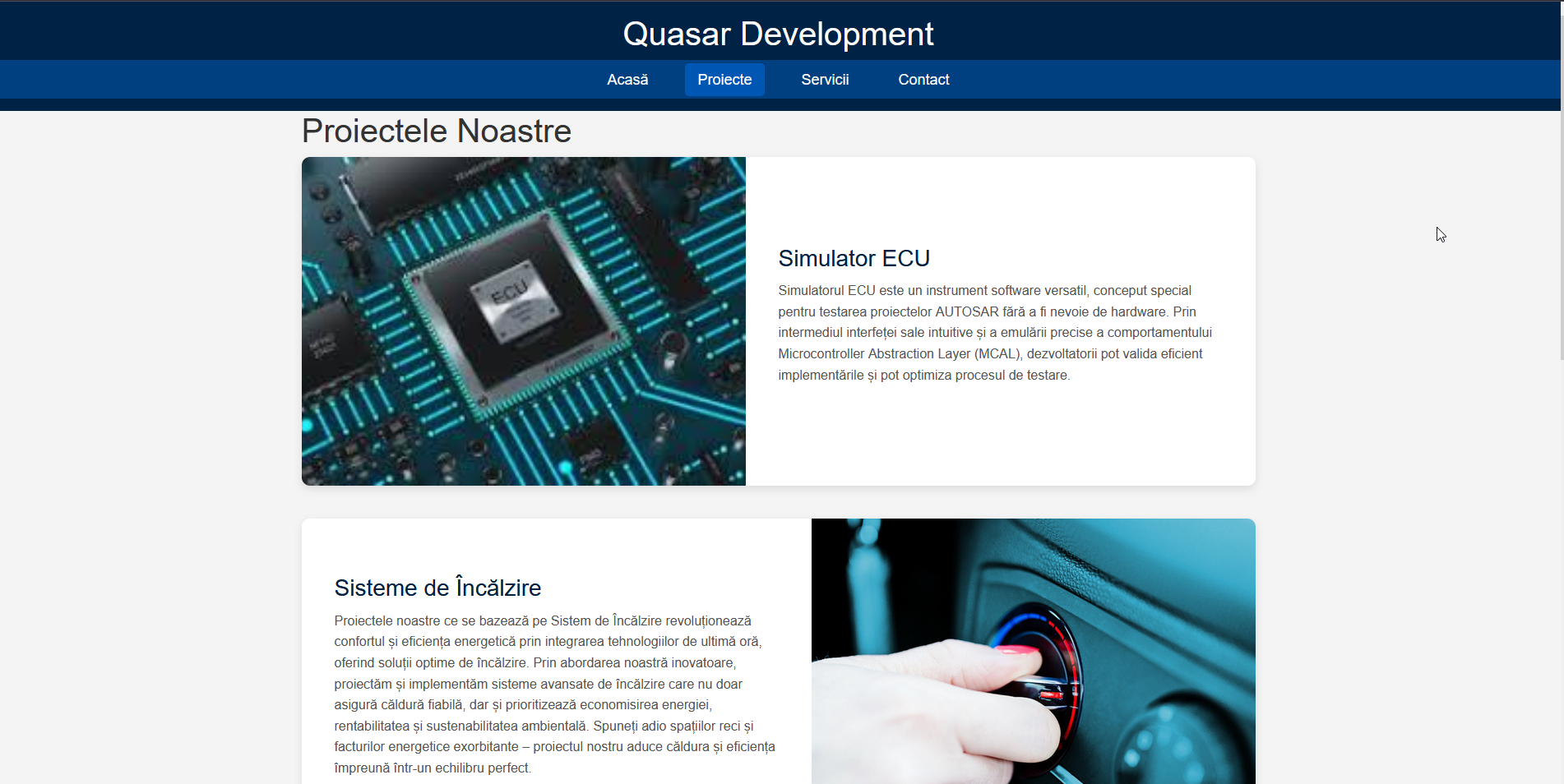


Figura 2.2 Pagina proiectelor



Figura 2.3 Pagina serviciilor

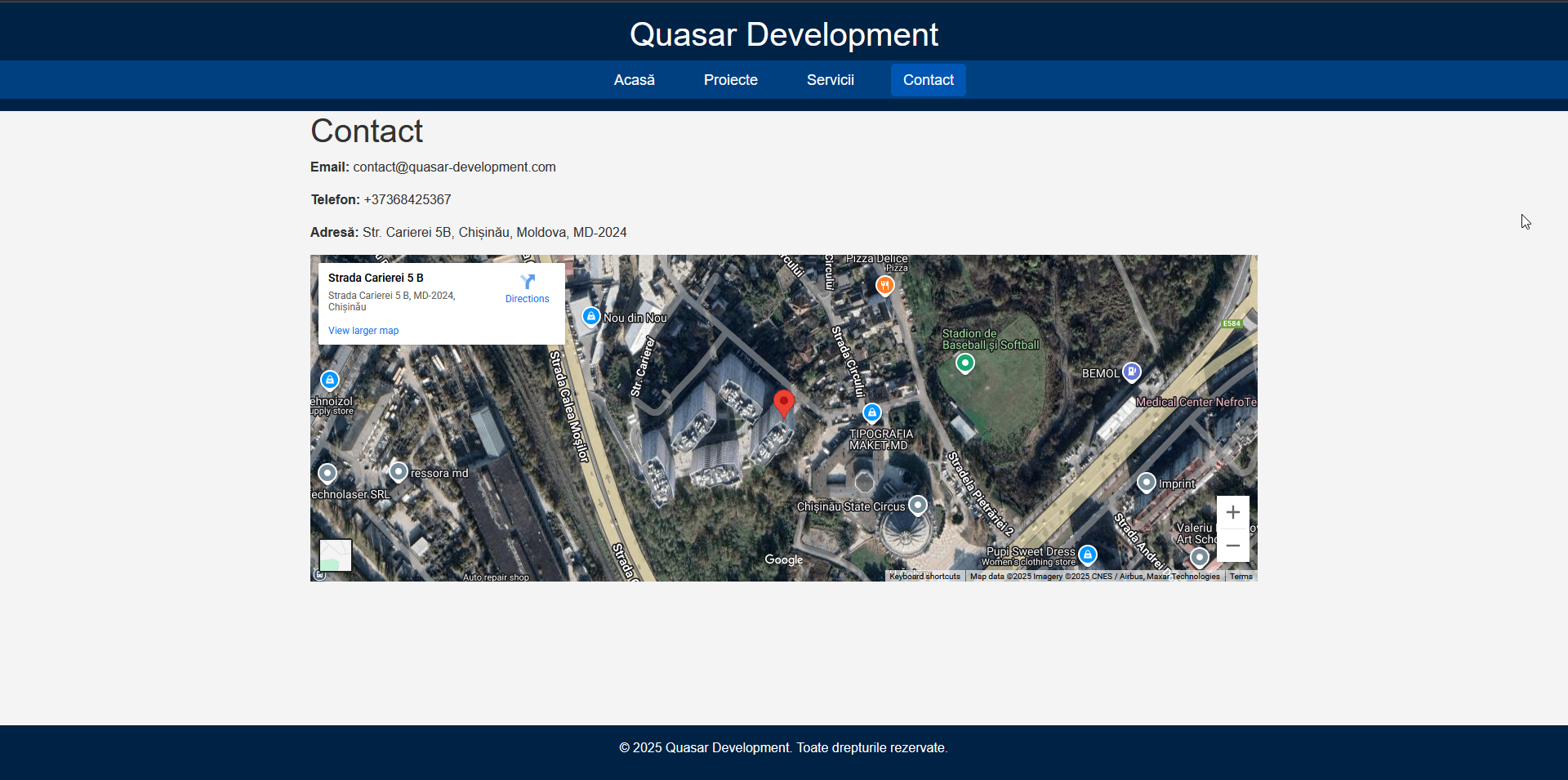


Figura 2.4 Pagina de contact