

UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE INGINERIE MECANICĂ ȘI MECATRONICĂ
Programul de studii: Mecatronică si Robotică

**Interfața grafica Matlab-GUIDE transpusa într-o interfața
grafica WEB cu HTML, CSS si JavaScript**

Mocanu Sebastian

București

2021

Cuprins

1. Tema de proiectare	4
2. Proiectarea Interfeței Grafice	5
2.1. Inserarea codului HTML (HyperText Markup Language)	5
2.1.1. Adăugarea Logo-urilor	6
2.1.2. Adăugarea unor date de tip text	6
2.1.3. Adăugarea câmpurilor pentru introducerea datelor de intrare	7
2.1.4. Adăugarea Imaginii Ansamblului	9
2.1.5. Adăugarea rezultatelor si butonului dropdown pentru selectarea piesei	10
2.1.6. Adăugarea butoanelor pentru funcționalitatea scriptului	11
2.2. Inserarea codului CSS (Cascading Style Sheets)	12
2.2.1. Stilizarea anumitor elemente pentru întreg documentul	12
2.2.2. Stilizarea claselor din documentul HTML	13
2.2.3. Stilizarea id-urilor din documentul HTML	15
2.3. Inserarea codului JavaScript	17
2.3.1. Încărcarea DOM-ului	17
2.3.2. Valorile de Intrare si constantele folosite	18
2.3.3. Funcția de calcul pentru Masa, Volum si Momentul de Inerție	18
2.3.4. Funcția de calcul pentru Arbore	19
2.3.5. Funcția de calcul pentru Disc	20
2.3.6. Funcția de calcul pentru întreg Ansamblul	21
2.3.7. Funcțiile pentru butoanele de Reset si Exit	21
2.3.8. Funcționalitatea de Local Storage	22
2.3.9. Funcția pentru actualizare a datelor	24
2.3.10. Funcții de ascultare de evenimente	25

3. Soluție editor de text	28
4. Adăugarea programului pe GitHub.....	30
4.1. Instalarea Git	30
4.2. Crearea unui repertoriu si urcarea programului in acesta.....	30
4.3. Adăugarea proiectului la Git Pages	34
4.4. Link-uri pentru site si program	35

1. Tema de proiectare

Se dorește să se implementeze o interfață grafică care să conțină:

- Logo-ul Facultății de Inginerie Mecanică și Mecatronica
- Logo-ul Universității Politehnica București
- Un buton pentru calcularea volumului, masei și momentului de inerție pentru figura de mai jos.
- Un buton dropdown pentru a putea selecta ce parte din piesă se dorește să se calculeze, în cazul nostru se dorește calcularea arborelui, discului dar și întreg ansamblului
- Toate input-urile necesare pentru a putea efectua calculul.
- Rezultatele calculului
- Un buton pentru a putea reseta valorile introduse.
- Un buton pentru a putea ieși din aplicație.

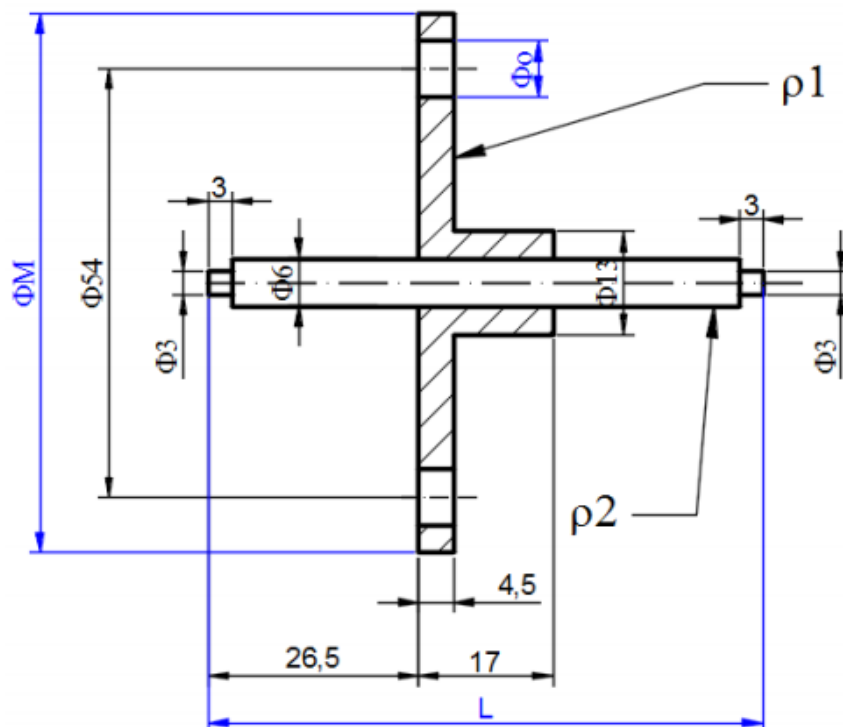


Fig. 1 Desenul ansamblului de pentru calculat

2. Proiectarea Interfeței Grafice

2.1. Inserarea codului HTML (HyperText Markup Language)

HTML este un limbaj de marcare care este utilizat pentru a putea crea pagini web si sa fie afișate într-un browser. HTML este utilizat pentru afișarea informațiilor cum ar fi titluri, paragrafe, butoane, imagini etc.

Următoarea secvență de cod reprezintă necesarul de HTML pentru a putea crea o pagina goala:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Interfata Grafica SSM</title>
</head>
<body>
</body>
</html>
```

Prima linie de cod: `<!DOCTYPE html>` reprezintă o informație pentru browser despre ce tip de document se așteaptă sa fie, este necesar sa fie prima line din cod.

Linia de cod `<html lang="en">` reprezintă un tag, in acesta se vor scrie toate celelalte elemente HTML, atributul „lang” reprezintă limba paginii web, adăugarea acestui atribut ajuta pentru căutarea pe motoarele de căutare web, in cazul nostru are valoarea „en” adică standardul ISO pentru limba engleza.

Linia de cod `<head>` împreuna cu lina `</head>` reprezintă un container pentru metadata si este plasat intre eticheta `<html>` si `<body>` bc. Metadatele sunt date despre conținutul HTML, acestea nu sunt afișate. Metadatele definesc de obicei titlul documentului, seturi de caractere, stilurile, scripturile si alte metainformatii.

Linia `<meta charset="UTF-8">` reprezinta are un atribut „charset” care specifica codificarea datelor pentru documentul HTML, specificatia HTML5 incurajeaza dezvoltatorii web sa utilizeze setul de date UTF-8, care acopera aproape toate caracterele si simbolurile din lume, aproximativ 1,112,064 de caractere.

Linia `<meta name="viewport" content="width=device-width, initial-scale=1.0">` oferă browser-ului instrucțiuni despre cum să controleze dimensiunile paginii și cum să o scaleze în funcție de rezoluția aparatului de pe care este accesat site-ul.

Linia `<meta http-equiv="X-UA-Compatible" content="ie=edge">` este un tag pentru a putea alege cu ce versiune de Internet Explorer este compatibilă.

Linia `<title>Interfata Grafica SSM</title>` reprezintă titlul paginii mai exact cum se vede în tab-urile browser-ului, se poate observa în figura 2

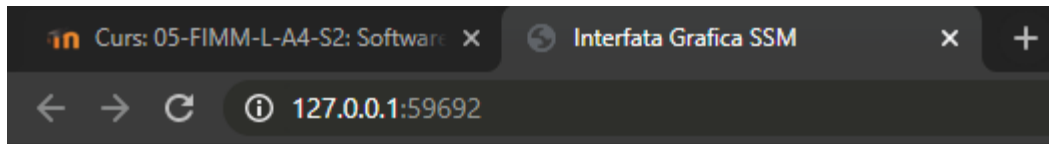


Fig. 2 Utilitatea tag-ului HTML `<title>`

Liniile `<body>` și `</body>` reprezintă locul unde se va scrie conținutul afișat pe site.

2.1.1. Adăugarea Logo-urilor

Pentru a adăuga imagini este necesar codul de mai jos, pentru moment se va omite utilizarea atributului `“class”` până la capitolul 3.

```
<div class="container">
  <div class="container div-imgs">
    
    
  </div>
</div>
```

Elementul `<div></div>` reprezintă un container pentru a putea aranja conținutul, nu are niciun efect asupra conținutului decât dacă este stilizat cu CSS.

Elementul `` este utilizat pentru a incorpora o imagine într-o pagină HTML, imaginile nu sunt inserate într-o pagină web, tag-ul `` este folosit pentru a crea un spațiu de reținere pentru imaginea de referință. Pentru a putea insera imaginea este nevoie de atributul `„src”` căruia se poate atribui o cale către imaginea din calculator sau un link către o imagine de pe un site web.

2.1.2. Adăugarea unor date de tip text

```
<div class="container text-container">
  <br>
  <h2 class="text-col">Laboratorul 10</h2>
  <h2 class="text-col">Interfață grafică utilizator</h2>
  <h2 class="text-col">GUIDE</h2>
</div>
```

Pentru adaugarea datelor de tip text s-a utilizat tag-ul `<h2></h2>` , in acesta se poate scrie text care are dimensiunea de 27.2px, valorile tuturor tagurilor „h” se pot observa in figura 3.

h1: 30px

h2: 27.2px

h3: 24.4px

h4: 21.6px

h5: 18.8px

h6: 16px

Fig. 3 Valori pentru tag-ul HTML "h"

2.1.3. Adăugarea câmpurilor pentru introducerea datelor de intrare

Pentru a putea adaugă câmpuri pentru introducerea datelor a fost necesara următoarea secvența de cod:

```
<div class="container col col-md-4">
  <div class="panel-date">
    <h4 class="text-col">Date de Intrare</h4>
    <div class="left">
      <div class="box">
        <label for="phiM">Φ <span class="small_letter">M </span>[mm] = </label>
        <input type="number" id="phiM" name="phiM" autocomplete="on" placeholder="...">
      </div>
      <div class="box">
        <label for="phiO">Φ <span class="small_letter">o </span>[mm] = </label>
        <input type="number" id="phiO" name="phiO" autocomplete="on" placeholder="...">
      </div>
      <div class="box">
        <label for="phi1">Φ <span class="small_letter">1 </span>[mm] = </label>
        <input type="number" id="phi1" name="phi1" autocomplete="on" placeholder="...">
      </div>
      <div class="box">
        <label for="phi2">Φ <span class="small_letter">2 </span>[mm] = </label>
        <input type="number" id="phi2" name="phi2" autocomplete="on" placeholder="...">
      </div>
      <div class="box">
        <label for="phi3">Φ <span class="small_letter">3 </span>[mm] = </label>
        <input type="number" id="phi3" name="phi3" autocomplete="on" placeholder="...">
      </div>
      <div class="box">
        <label for="phi4">Φ <span class="small_letter">4 </span>[mm] = </label>
        <input type="number" id="phi4" name="phi4" autocomplete="on" placeholder="...">
      </div>
    </div>
  </div>
</div>
```

```

    <label for="n">n <span class="small_letter">g </span>[ - ] = </label>
    <input type="number" id="n" name="n" autocomplete="on" placeholder="...">
  </div>
</div>
<div class="right">
  <div class="box">
    <label for="IT">L <span class="small_letter">T </span>[mm] = </label>
    <input type="number" id="IT" name="IT" autocomplete="on" placeholder="...">
  </div>
  <div class="box">
    <label for="I1">L <span class="small_letter">1 </span>[mm] = </label>
    <input type="number" id="I1" name="I1" autocomplete="on" placeholder="...">
  </div>
  <div class="box">
    <label for="I2">L <span class="small_letter">2 </span>[mm] = </label>
    <input type="number" id="I2" name="I2" autocomplete="on" placeholder="...">
  </div>
  <div class="box">
    <label for="I3">L <span class="small_letter">3 </span>[mm] = </label>
    <input type="number" id="I3" name="I3" autocomplete="on" placeholder="...">
  </div>
  <div class="box">
    <label for="I4">L <span class="small_letter">4 </span>[mm] = </label>
    <input type="number" id="I4" name="I4" autocomplete="on" placeholder="...">
  </div>
  <div class="box">
    <label for="rho1">p <span class="small_letter">1 </span>[kg/m<sup>3</sup>] = </label>
    <input type="number" id="rho1" name="rho1" autocomplete="on" placeholder="...">
  </div>
  <div class="box">
    <label for="rho2">p <span class="small_letter">2 </span>[kg/m<sup>3</sup>] = </label>
    <input type="number" id="rho2" name="rho2" autocomplete="on" placeholder="...">
  </div>
</div>
</div>
</div>

```

S-a utilizat cate un div pentru fiecare câmp, apoi s-a adăugat `<label for="phiM">Φ M [mm] = </label>` acest tag ajuta pentru adăugarea text-ului aferent câmpului pentru introducerea datelor de intrare, pentru a defini al cui este aferent sa utilizat atributul „for” care trebuie sa corespunda cu atributul „id” al input-ului. S-a utilizat si tag-ul „span” care este un container folosit pentru a marca o parte a unui text, în cazul nostru am atribuit o clasa acestui tag si am stilizat-o astfel încât sa devina indice.

Tag-ul `<input type="number" id="phiM" name="phiM" autocomplete="on" placeholder="...">` reprezintă câmpul pentru introducerea datelor propriu-zis, după cum se poate observa acesta are valoarea atributului „id” la fel ca si valoarea atributului „for” din tag-ul label, acest fapt reprezintă apartenenta bilaterala a acestora. Atributul „type” reprezintă ce fel de text va fi introdus în câmpul

respectiv, astfel constrânge tipurile de input adăugate de către utilizator, în cazul de față este un număr, „number”. Atributul „placeholder” afișează valoarea acestuia în momentul în care nu există niciun input oferit de către utilizator.

S-a repetat adăugarea „input” și „label” pentru fiecare valoare necesară calculului final.

2.1.4. Adăugarea Imaginii Ansamblului

S-a adăugat și imaginea ansamblului care urmează să fie schimbată, cu ajutorul JavaScript, în funcție de ce este selectat în butonul dropdown.

```
<div class="container col col-md-4">
  <div class="mecanism-centric">
    
  </div>
</div>
```

Pentru a putea observa mai ușor ce date trebuie introduse s-a modificat imaginea inițială

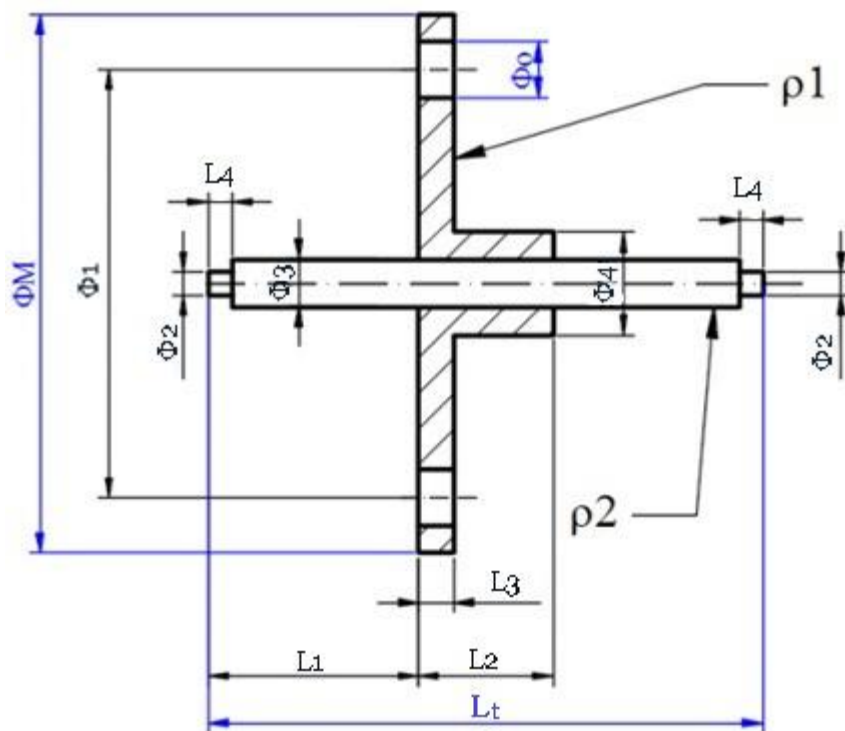


Fig. 4 Ansamblu modificat

S-a mai modificat imaginea și pentru a putea afișa ce este selectat în momentul respectiv

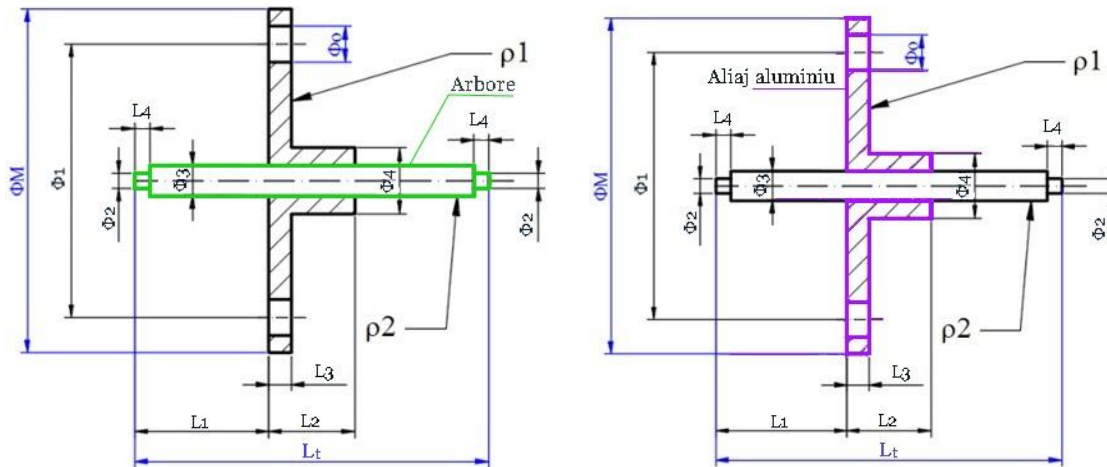


Fig. 5 Marcarea elementelor selectate

2.1.5. Adăugarea rezultatelor si butonului dropdown pentru selectarea piesei

Pentru adăugarea rezultatelor si butonului dropdown s-a adăugat următoarea secvență de cod:

```
<div class="container col col-md-4">
  <div class="row">
    <div class="panel-rezultate">
      <h4 class="text-col" style="text-align:center;">Rezultate</h4>
      <br>
      <h4 class="text-col" id="piesa_id">Ansamblu</h4>
      <br>
      <div class="">
        <h3 class="text-col">Volum = <span id="volum_piesa">...</span> <span class="unitate-de-masura">[m<sup>3</sup>]</span> </h3>
        <h3 class="text-col">Masa = <span id="masa_piesa">...</span> <span class="unitate-de-masura">[kg]</span></h3>
        <h3 class="text-col">Moment = <span id="moment_piesa">...</span> <span class="unitate-de-masura">[kg&sdot;m<sup>2</sup>]</span></h3>
      </div>
    </div>
  </div>

  <div class="row">
    <div class="toggle-button">
      <label for="toggle-id" class="text-col">Selecteaza piesa:</label>
      <select class="toggle-sty" name="toggle-stuff" id="toggle-id">
        <option value="ansamblu">Ansamblu</option>
        <option value="aliaj">Disc</option>
        <option value="arbore">Arbore</option>
      </select>
    </div>
  </div>
</div>
```

În cazul fiecărei valori s-a adăugat un tag „span” și îi s-a atribuit un „id” astfel se poate modifica conținutul tag-ului „span” cu JavaScript.

Pentru butonul dropdown s-a procedat similar cu adăugarea input-urilor pentru datele de intrare. S-a adăugat în label în care s-a scris descrierea butonului iar tag-ul „select” a ajutat la crearea containerului, „option” reprezintă ce valoare se dorește să fie selectată.

2.1.6. Adăugarea butoanelor pentru funcționalitatea scriptului

Secvența de cod necesară:

```
<button id="button-calcul" class="centric" type="button" name="button">Calculeaza Volumul, Masa, Momentul de Inertie</button>  
<button id="button-reset" class="centric" type="button" name="button">Reset</button>  
<button id="button-exit" class="centric" type="button" name="button">Exit</button>
```

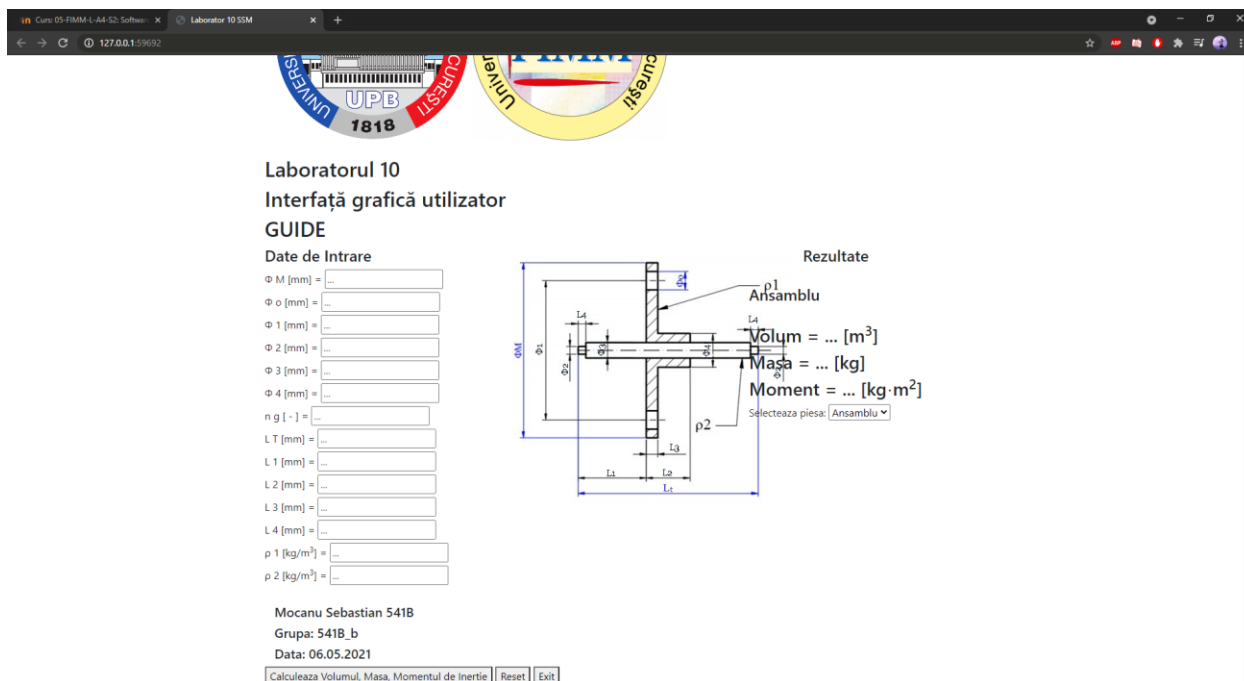


Fig. 6 Pagina HTML fără CSS

În figura 6 se poate observa cum arată o pagină web fără design, o asemenea pagină are un aspect destul de neplăcut, designul se poate modifica cu ajutorul CSS, există și alte distribuții prin care se poate stiliza o pagină HTML dar în cazul de față s-a folosit CSS.

2.2. Inserarea codului CSS (Cascading Style Sheets)

CSS reprezintă un standard pentru formatarea elementelor dintr-un document HTML. Stilurilor se pot atașa elementelor HTML prin intermediul unor fișiere externe sau în cadrul documentului HTML prin tagului `<style></style>`, acestea trebuie neapărat să se afle în `<head></head>`. În cazul de față s-a adăugat prin intermediul unui fișier separat. Pentru a putea stiliza un element este nevoie de adăugarea la elementul respectiv atributul „class” sau atributul „id”.

```
<link rel="stylesheet" href="/style.css">
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-Vkoo8x4CGsO3+Hhvx8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">
```

În cazul de față s-au folosit două fișiere CSS, prin intermediu tag-ului „link”, pentru a specifica ce fișier se dorește să se folosească în HTML este nevoie de atributul „href”. Primul fișier CSS se afla în directorul HTML-ului și s-a oferit calea către acesta, al doilea fișier reprezintă un fișier extern către Bootstrap, un framework de CSS folosit pentru a aranja mai ușor anumite elemente.

2.2.1. Stilizarea anumitor elemente pentru întreg documentul

Se dorește modificarea unor elemente pentru întreg documentul, mai există modificarea designului pentru clase și pentru id-uri.

```
body {
  background-color: #000 !important;
}
h2 {
  text-align: center;
}
h4 {
  font-size: 19px !important;
  margin: 10px 10px 10px 10px !important;
}
h3 {
  font-size: 17px !important;
  margin: 10px 10px 10px 10px !important;
}
```

Mai sus s-a modificat culoarea de background a tag-ului body în „#000” adică culoarea cu codul HEX 000000, aceasta înseamnă negru. Pentru toate tag-urile de tip „h2” s-a aliniat textul la centru, pentru toate tag-urile „h4” s-a suprascris mărimea fontului la 19 pixeli și s-au setat marginile la 10 pixeli sus, dreapta, jos și stânga pentru a oferi mai mult spațiu textului și a fi mai

lizibil, același lucru s-a repetat și pentru tag-ul „h3” doar ca mărimea fontului a fost setată 17 pixeli.

2.2.2. Stilizarea claselor din documentul HTML

Pentru designul claselor se scrie numele clasei definite în HTML și se adaugă un punct „.” în fața clasei, acest aspect se poate observa mai jos:

```
.text-col {
  color: #F8F8F8;
}
.img-right {
  height: 150px;
  width: 150px;
  float: right;
}
.img-left {
  height: 150px;
  width: 150px;
  float: left;
}
.div-imgs {
  margin: 50px 0px 50px 0px;
}
.panel-date {
  border: 3px groove rgb(14,99,42);
  border-radius: 20px;
  margin-top: 35px;
  width: 370px;
  padding: 20px;
  display: block;
  align-items: center;
  justify-content: center;
  height: 300px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.6);
}
.panel-rezultate {
  border: 3px groove rgb(14,99,42);
  border-radius: 20px;
  margin-top: 35px;
  margin-left: 20px;
  width: 350px;
  padding: 20px;
  display: block;
  align-items: center;
  justify-content: center;
  height: 300px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.6);
}
.box input{
  width: 60px;
  float: right;
}
.small_letter{
  font-size: 10px;
}
```

```

}
.box{
  display: block;
  align-items: center;
  justify-content: center;
  margin:0 auto;
  color: #F8F8F8;
}

.right{
  float: right;
  width: 160px;
  margin-bottom: 20px;
}

.left{
  float:left;
  width: 160px;
  border-right: 1.5px solid #16a596;
  padding-right: 10px;
  margin-bottom: 20px;
}

}
.centric {
  background-color: rgb(14,99,42);
  border: none;
  color: white;
  padding: 15px 40px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  font-weight: bold;
  border-radius: 10px;
  margin: 10px 10px 10px 10px;
}

.stuff {
  margin-top: 80px;
  height: 200px;
}

}

.img-grafic {
  border: 5px groove #222;
  border-radius: 20px;
}

}

.text-eu {
  margin-top: 100px;
}

}

.eu {
  margin-top: 200px;
}

}

.unitate-de-masura {
  float: right;
}

}

.toggle-button {
  margin-top: 10px;
  margin-left: 55px;
}

}

```

Clasa „text-col” setează culoarea textului in „#F8F8F8” adică un gri foarte deschis, aceasta culoare a fost aleasă deoarece este mai ușoară pentru ochi deschi un alb pur.

Clasa „img-right” setează înălțimea și lățimea la 150px și o orientează către dreapta, similar clasa „img-left” setează dimensiunile imaginii tot la 150 de pixeli și o orientează către stânga.

Clasa „div-imgs” setează marginile de sus și de jos la 50 de pixeli distanță.

Clasa „panel-date” adaugă o bordură cu dimensiunea de 3 pixeli care are culoarea verde `rgb(14,99,42)` reprezentând cât de multă intensitate este atribuită pe fiecare culoare RGB, „border-radius” reprezentant ce rază are teșitura bordurii, în cazul de față 20 de pixeli pentru a da un aspect mai estetic, s-a setat marginea de sus la o distanță de 35 de pixeli, s-au ales mărimile 370 de pixeli și 300 de pixeli pentru întreg block-ul, s-au centrat elementele tag-ului atribuit acestei clase și s-a adăugat o umbră pentru a putea oferi un efect mai tridimensional și un aspect mai plăcut elementului. Similar s-a modificat design-ul și pentru clasa „panel-rezultate”.

Clasa „box” a fost folosită pentru a aranja inputul și label-ul, astfel aliniindu-le.

Clasa „small_letter” a fost utilizată pentru a crea un indice.

Clasa „right” aliniază elementele către dreapta, și setează grosimea elementului la 160 de pixeli și o margine de 20 de pixeli, similar clasa „left” face același lucru dar are și o bordură care este afișată doar către dreapta pentru a putea distinge coloanele elementelor de intrare.

Clasa „centric” a fost folosită pentru designul butoanelor.

2.2.3. Stilizarea id-urilor din documentul HTML

Pentru designul id-urilor se scrie numele id-ului definit în HTML și se adaugă un hashtag „#” în fața id-ului, acest aspect se poate observa mai jos:

```
#img-figura {
  height: 350px;
  width: 370px;
}
#button-exit {
  float: right;
  margin: -50px 100px 0px 0px;
  color: red;
  background-color: #000;
  font-size: 20px;
  border: 3px groove white;
  border-radius: 0px;
  padding: 5px 25px;
```

```

}
#button-reset {
  float: right;
  margin: -50px 200px 0px 0px;
  color: #0047AB;
  background-color: #000;
  font-size: 20px;
  border: 3px groove white;
  border-radius: 0px;
  padding: 5px 25px;
}
#button-calcul {
  float: right;
  margin: -50px 400px 0px 0px;
}
}

```

Id-ul „#button-exit” a fost folosit pentru a modifica design-ului butonului de exit si pentru a-l alinia. Similar si pentru restul butoanelor.

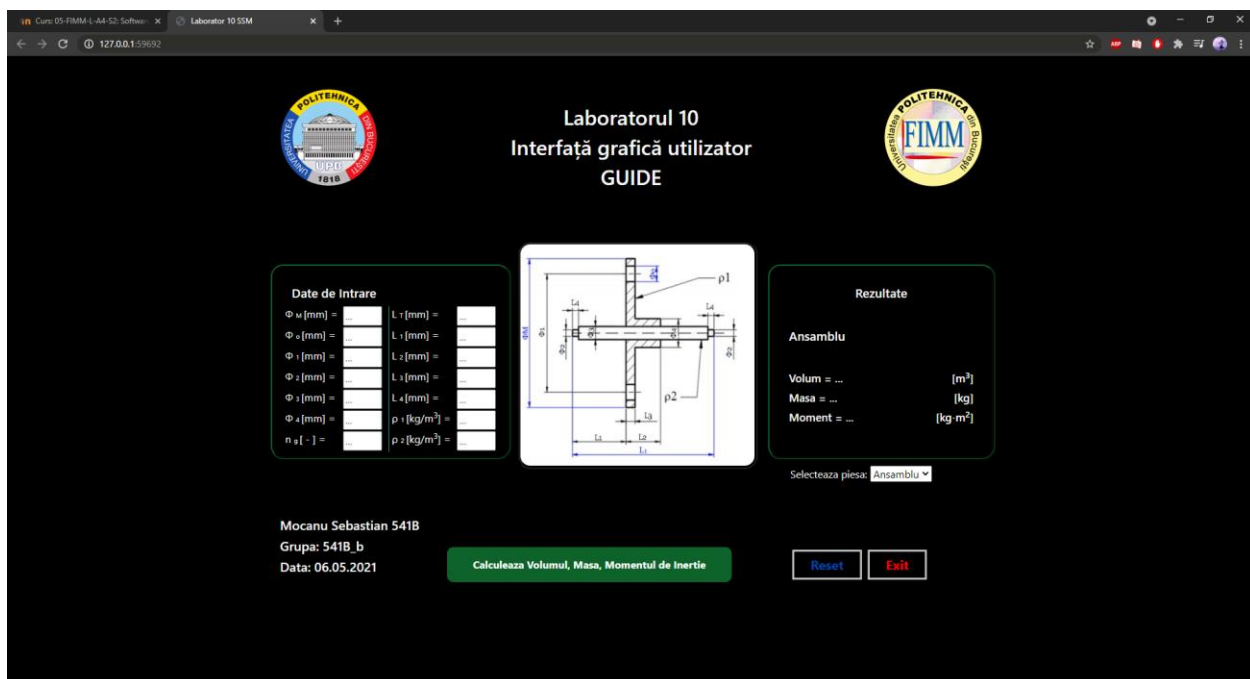


Fig. 7 Pagina Web cu design CSS

2.3. Inserarea codului JavaScript

JavaScript este un limbaj de programare orientat pe obiecte bazat pe conceptul prototipurilor. Acesta este folosit pentru introducerea unor funcționalități în paginile web, codul JavaScript fiind rulat de către browser. Între JavaScript și limbajul de programare Java nu există nicio legătură. Pentru a putea asocia JavaScript în documentul HTML se poate scrie în tagul `<head></head>` folosit, de obicei se pun link-uri către scripturi de pe internet sau se poate scrie în tagul `<body></body>` la sfârșit, pentru a optimiza viteza de încărcare a site-ului s-a pus în tagul „body”.

```
<script type="text/javascript" src="app.js"></script>
```

2.3.1. Încărcarea DOM-ului

Pentru o optimizare de viteza de indexare s-a încărcat DOM-ul (Document Object Model) la începutul script-ului, acest lucru înseamnă că script-ul parcurge fișierul HTML și caută elementele dorite iar după se poate ocupa de funcționalitatea lui, aceasta este o practică bună în dezvoltarea aplicațiilor web.

```
// Butoane functionalitate
const buttonCalcul = document.getElementById("button-calcul");
const buttonReset = document.getElementById("button-reset");
const buttonExit = document.getElementById("button-exit");
// Date de intrare
const phiM = document.getElementById("phiM");
const phiO = document.getElementById("phiO");
const phi1 = document.getElementById("phi1");
const phi2 = document.getElementById("phi2");
const phi3 = document.getElementById("phi3");
const phi4 = document.getElementById("phi4");
const IM = document.getElementById("IM");
const I1 = document.getElementById("I1");
const I2 = document.getElementById("I2");
const I3 = document.getElementById("I3");
const I4 = document.getElementById("I4");
const rho1 = document.getElementById("rho1");
const rho2 = document.getElementById("rho2");
const n = document.getElementById("n");
// Rezultate
const piesa_text = document.getElementById("piesa_id");
const volum_piesa = document.getElementById("volum_piesa");
const masa_piesa = document.getElementById("masa_piesa");
const moment_piesa = document.getElementById("moment_piesa");
// Imagine + toggle
const piesa_imagine = document.getElementById("img-figura");
const piesa_select = document.getElementById("toggle-id");
```

Mai sus atribuit o variabila fiecărui element de care este nevoie, s-a găsit fiecare element necesar cu ajutorul funcției „getElementById” și s-a oferit ca parametru id-ul stabilit în HTML.

În fața fiecărei variabile există keyword-ul „const”, acesta înseamnă că variabila nu se poate schimba sau reatribui, pe scurt, rămâne constantă.

2.3.2. Valorile de intrare și constantele folosite

Pentru a putea crea funcționalitatea scriptului a fost necesară atribuirea unor variabile pentru efectuarea calculelor.

```
// Valori de intrare și constante
const PI = Math.PI;
const e_3 = Math.pow(10, -3);
let phiM_num;
let phiO_num;
let phi1_num;
let phi2_num;
let phi3_num;
let phi4_num;
let n_num;
let IT_num;
let I1_num;
let I2_num;
let I3_num;
let I4_num;
let rho1_num;
let rho2_num;
```

Pentru a declara o variabilă se folosește keyword-ul „let”

2.3.3. Funcția de calcul pentru Masă, Volum și Momentul de Inerție

Pentru a putea calcula s-a folosit o funcție care returnează un vector cu trei elemente:

```
function calculVMI(raza, inaltime, densitate, distanta) {
    volum = PI * Math.pow(raza, 2) * inaltime;
    masa = volum * densitate;
    moment_inertie = masa * Math.pow(raza, 2) / 2 + masa * Math.pow(distanta, 2);

    return [volum, masa, moment_inertie];
}
```

În această funcție s-au folosit 4 parametri: raza, înălțimea, densitatea și distanța. Funcția Math.Pow() reprezintă ridicarea la putere, primul parametru reprezintă numărul care se dorește să fie ridicat la puterea valorii celui de-al doilea parametru. Spre exemplu Math.pow(10, 3) ridică numărul 10 la puterea a 3-a, e.g. 10^3 mai există o alternativă cu „**” adică $10^{**}3$ rezultatul fiind același.

2.3.4. Funcția de calcul pentru Arbore

Pentru a obține rezultatul calculului doar pentru arbore s-a scris o funcție calculArbore();

```
function calculArbore() {
  let densitate = rho2_num;
  let distanta = 0;

  // --- Arbore fara fus --- //
  let lungime_arbore = (IT_num - 2 * l4_num) * e_3;
  let raza_arbore = (phi3_num / 2) * e_3;

  // --- Fus --- //
  let lungime_fus = l4_num * e_3;
  let raza_fus = (phi2_num / 2) * e_3;

  let arbore = calculVMI(raza_arbore, lungime_arbore, densitate, distanta);
  let fus = calculVMI(raza_fus, lungime_fus, densitate, distanta);
  let result = arbore.map(function (item, index) {
    return item + (2*fus[index]);
  });

  return result;
}
```

În această funcție s-au calculat parametrii necesari pentru funcția *calculVMI()*; și s-a apelat această funcție pentru arborele fără fus și pentru fus, aceste rezultate fiind stocate într-o variabilă care reprezintă un vector cu trei elemente. Pentru rezultatul final trebuia să fie adunate valorile arborelui fără fus cu valorile fusului de două ori, pentru a nu utiliza un iterator „for” generic s-a folosit o metodă de ordin superior pentru matrici „map”. S-a declarat o variabilă și îi s-a atribuit metoda map rezultatului arborelui, apoi s-a atribuit ca parametru o funcție care are ca parametrii elementul matricei și index-ul, adică un iterator. Funcția rulând până la terminarea indexilor matricei astfel adunând fiecare element din matricea arborelui cu de două ori elementul de pe poziția index al matricei „fus”. În final funcția de calcul returnează rezultatul tot într-un vector de trei elemente.

2.3.5. Funcția de calcul pentru Disc

Pentru a obține rezultatul calculului doar pentru disc s-a scris o funcție calculAliaj();

```
function calculAliaj() {
  let densitate = rho1_num;
  let distanta = 0;
  let numar_gauri = n_num;

  // --- Gauri de scazut --- //
  let raza_gauri = (phiO_num / 2) * e_3;
  let lungime_gauri = l3_num * e_3;
  let distanta_gauri = (phi1_num/2) * e_3;

  // --- Aliajul mare fara bucsa --- //
  let raza_aliaj = (phiM_num / 2) * e_3;
  let lungime_aliaj = l3_num * e_3;

  // --- De scazut partea arborelui din aliajul fara bucsa --- //
  let raza_fara_arbore = (phi3_num / 2) * e_3;
  let lungime_fara_arbore = l3_num * e_3;

  // --- Bucsă aliajului --- //
  let raza_bucsă = (phi4_num / 2) * e_3;
  let lungime_bucsă = (l2_num - l3_num) * e_3;

  // --- De scazut partea arborelui din bucsa aliajului --- //
  let raza_fara_bucsă = (phi3_num / 2) * e_3;
  let lungime_fara_bucsă = (l2_num - l3_num) * e_3;

  let gauri = calculVMI(raza_gauri, lungime_gauri, densitate, distanta_gauri);
  let aliaj = calculVMI(raza_aliaj, lungime_aliaj, densitate, distanta);
  let fara_arbore = calculVMI(raza_fara_arbore, lungime_fara_arbore, densitate, distanta);
  let bucsa = calculVMI(raza_bucsă, lungime_bucsă, densitate, distanta);
  let fara_bucsă = calculVMI(raza_fara_bucsă, lungime_fara_bucsă, densitate, distanta);

  let result = aliaj.map(function(item, index) {
    return item - (numar_gauri * gauri[index]) + bucsa[index] - fara_arbore[index] - fara_bucsă[index];
  });

  return result;
}
```

Similar cu funcția pentru calculul arborelui s-a folosit același concept doar ca de data aceasta s-a apelat funcția *calculVMI()*; de 5 ori, pentru găuri, pentru aliajul fără bucsă, pentru bucsă și de scăzut partea arborelui din aliaj și bucsă. După stocarea acestor apelări în variabile s-a aplicat metoda de ordin superior pentru matrici map doar funcționalitatea a fost realizată în felul următor: din element s-au scăzut câte găuri existau, s-a adăugat bucsă și s-a scăzut partea arborelui din bucsă și din aliaj.

2.3.6. Funcția de calcul pentru întreg Ansamblul

În cazul acestei funcții se pleacă de la un principiu simplu, cu datele de la funcția de calcul pentru arbore și cea pentru aliaj s-a calculat și pentru întreg ansamblul

```
function calculAnsamblu() {  
  let arbore = calculArbore();  
  let aliaj = calculAliaj();  
  
  let result = arbore.map(function(item, index) {  
    return item + aliaj[index];  
  });  
  
  return result;  
}
```

S-a folosit metoda de ordin superior pentru matrici map pentru a adăuga fiecare element din prima matrice cu fiecare element din cea de a doua matrice.

2.3.7. Funcțiile pentru butoanele de Reset și Exit

Aceste funcții au fost implementate pentru butoanele aferente denumirii lor.

```
function resetValues() {  
  phiM.value = 0;  
  phiO.value = 0;  
  phi1.value = 0;  
  phi2.value = 0;  
  phi3.value = 0;  
  phi4.value = 0;  
  n.value = 0;  
  IT.value = 0;  
  I1.value = 0;  
  I2.value = 0;  
  I3.value = 0;  
  I4.value = 0;  
  rho1.value = 0;  
  rho2.value = 0;  
  phiM_num = 0;  
  phiO_num = 0;  
  phi1_num = 0;  
  phi2_num = 0;  
  phi3_num = 0;  
  phi4_num = 0;  
  n_num = 0;  
  IT_num = 0;  
  I1_num = 0;  
  I2_num = 0;  
  I3_num = 0;  
  I4_num = 0;
```

```

    rho1_num = 0;
    rho2_num = 0;
}
function closePage() {
    open(location, '_self').close();
    // window.open("", '_self', "");
    window.open(window.location, '_self').close();
    window.close()
}

```

Funcția resetValues() după cum spune și numele ei setează toate input-urile la 0 iar funcția closePage() reprezintă un script care închide pagina, din păcate nu merge pe toate browserele existente datorită problemelor de securitate.

2.3.8. Funcționalitatea de Local Storage

Pentru ca utilizatorul să nu fie nevoit să reintroducă de fiecare dată când intră pe pagina valorile de input, s-a adăugat o funcționalitate de Local Storage care permite salvarea perechilor cheie – valoare în browser. Acest obiect stochează date fără o dată de expirare. Datele pot fi șterse doar dacă se va șterge istoricul sau dacă se folosește „Clear browsing data”.

```

phiM.addEventListener('input', () => {
    phiM_num = parseFloat(phiM.value);
    localStorage.setItem("phiM_num", phiM_num);
});
phiO.addEventListener('input', () => {
    phiO_num = parseFloat(phiO.value);
    localStorage.setItem("phiO_num", phiO_num);
});
phi1.addEventListener('input', () => {
    phi1_num = parseFloat(phi1.value);
    localStorage.setItem("phi1_num", phi1_num);
});
phi2.addEventListener('input', () => {
    phi2_num = parseFloat(phi2.value);
    localStorage.setItem("phi2_num", phi2_num);
});
phi3.addEventListener('input', () => {
    phi3_num = parseFloat(phi3.value);
    localStorage.setItem("phi3_num", phi3_num);
});
phi4.addEventListener('input', () => {
    phi4_num = parseFloat(phi4.value);
    localStorage.setItem("phi4_num", phi4_num);
});
IT.addEventListener('input', () => {
    IT_num = parseFloat(IT.value);
    localStorage.setItem("IT_num", IT_num);
});
I1.addEventListener('input', () => {

```

```

    l1_num = parseFloat(l1.value);
    localStorage.setItem("l1_num", l1_num);
  });
  l2.addEventListener('input', () => {
    l2_num = parseFloat(l2.value);
    localStorage.setItem("l2_num", l2_num);
  });
  l3.addEventListener('input', () => {
    l3_num = parseFloat(l3.value);
    localStorage.setItem("l3_num", l3_num);
  });
  l4.addEventListener('input', () => {
    l4_num = parseFloat(l4.value);
    localStorage.setItem("l4_num", l4_num);
  });
  rho1.addEventListener('input', () => {
    rho1_num = parseFloat(rho1.value);
    localStorage.setItem("rho1_num", rho1_num);
  });
  rho2.addEventListener('input', () => {
    rho2_num = parseFloat(rho2.value);
    localStorage.setItem("rho2_num", rho2_num);
  });
  n.addEventListener('input', () => {
    n_num = parseFloat(n.value);
    localStorage.setItem("n_num", n_num);
  });

```

// ===== Local Storage setup ===== //

```

phiM.value = localStorage.getItem('phiM_num');
phiO.value = localStorage.getItem("phiO_num");
phi1.value = localStorage.getItem("phi1_num");
phi2.value = localStorage.getItem("phi2_num");
phi3.value = localStorage.getItem("phi3_num");
phi4.value = localStorage.getItem("phi4_num");
IT.value = localStorage.getItem("IT_num");
l1.value = localStorage.getItem("l1_num");
l2.value = localStorage.getItem("l2_num");
l3.value = localStorage.getItem("l3_num");
l4.value = localStorage.getItem("l4_num");
rho1.value = localStorage.getItem("rho1_num");
rho2.value = localStorage.getItem("rho2_num");
n.value = localStorage.getItem("n_num");

```

// ===== Local storage valori pentru calcul ===== //

```

phiM_num = parseFloat(phiM.value);
phiO_num = parseFloat(phiO.value);
phi1_num = parseFloat(phi1.value);
phi2_num = parseFloat(phi2.value);
phi3_num = parseFloat(phi3.value);
phi4_num = parseFloat(phi4.value);
IT_num = parseFloat(IT.value);
l1_num = parseFloat(l1.value);
l2_num = parseFloat(l2.value);
l3_num = parseFloat(l3.value);
l4_num = parseFloat(l4.value);

```

```
rho1_num = parseFloat(rho1.value);  
rho2_num = parseFloat(rho2.value);  
n_num = parseFloat(n.value);
```

În cadrul acestei secvențe de cod s-a adăugat câte un event listener pe fiecare input pentru a putea reține valoarea, totodată se setează aceasta valoare în localStorage având cheia primul parametru al funcției `setItem` iar valoarea cel de-al doilea parametru. Se folosește `parseFloat()` pentru fiecare valoare deoarece ceea ce introduce utilizatorul este un String și trebuie transformat într-un număr care poate avea zecimale. Pentru a seta valorile pe input-uri s-a luat elementul din local storage cu funcția `getItem()` și s-a setat valoarea fiecărui input cu ajutorul funcției `value`. În final pentru a putea calcula cu valorile din local storage s-au atribuit variabilelor de calcul valorile de pe inputuri.

2.3.9. Funcția pentru actualizare a datelor

Pentru butonul de calcul s-a creat această funcție

```
// --- Funcție pentru a actualiza în mod constant rezultatele --- //  
function updateValues() {  
  if (piesa_text.innerHTML === "Ansamblu") {  
    let result = calculAnsamblu();  
    volum_piesa.innerHTML = result[0].toExponential(8);  
    masa_piesa.innerHTML = result[1].toExponential(8);  
    moment_piesa.innerHTML = result[2].toExponential(8);  
  }  
  else if (piesa_text.innerHTML === "Arbore") {  
    let result = calculArbore();  
    volum_piesa.innerHTML = result[0].toExponential(8);  
    masa_piesa.innerHTML = result[1].toExponential(8);  
    moment_piesa.innerHTML = result[2].toExponential(8);  
  }  
  else {  
    let result = calculAliaj();  
    volum_piesa.innerHTML = result[0].toExponential(8);  
    masa_piesa.innerHTML = result[1].toExponential(8);  
    moment_piesa.innerHTML = result[2].toExponential(8);  
  }  
}
```

Această funcție verifică starea text-ului de la rezultate, în funcție de text calculează valorile aferente elementului acestuia și setează valoarea text-ului fiecărui „span” din rezultate cu ajutorul `innerHTML` cu rezultatul și indicii aferent volumului, masei și respectiv momentului, apoi pentru o afișare a rezultatelor ca în Matlab s-a folosit funcția `toExponential(8)`.

2.3.10. Funcții de ascultare de evenimente

În final pentru a putea adăuga funcționalitatea tuturor butoanelor s-au adăugat următoarei ascultători de evenimente (eventListener):

```
// Event Listeners
buttonReset.addEventListener("click", () => {
  if (confirm("Esti sigur ca doresti sa resetezi valorile?")) {
    resetValues();
  }
});
buttonExit.addEventListener("click", () => {
  if (confirm("Esti sigur ca doresti sa inchizi pagina?")) {
    closePage();
  }
});
// --- Event listener pe dropdown --- //
piesa_select.addEventListener("change", (event) => {
  if (event.target.value === "ansamblu") {
    piesa_text.innerHTML = "Ansamblu";
    piesa_imagine.src = "./img/ansamblu.jpeg";
    updateValues();
  }
  else if (event.target.value === "arbore") {
    piesa_text.innerHTML = "Arbore";
    piesa_imagine.src = "./img/arbore.jpeg";
    updateValues();
  }
  else {
    piesa_text.innerHTML = "Disc";
    piesa_imagine.src = "./img/aliaj_aluminiu.jpeg";
    updateValues();
  }
});
// --- Calcul Event Listener --- //
buttonCalcul.addEventListener("click", updateValues);
```

Primele doua ascultătoare sunt pe click, când se apasă cu click pe butonul respectiv o funcție este declanșată și astfel apare un PopUp care pune o întrebare, dacă răspunsul este afirmativ declanșează funcția aferentă fiecărui buton.

Pentru butonul de dropdown s-a adăugat un ascultător bazat pe evenimentul de schimbare a acestuia, s-a scris o funcționalitate pentru fiecare valoare a dropdown-ului și astfel dacă valoare evenimentului este valoarea de verificat se va schimba textul de la rezultate cu valoarea acestuia, va schimba sursa imaginii astfel fiind sursa aferentă piesei respective și se va apela funcția *updateValues()*; pentru a nu fi necesar să se apese butonul de calculează de fiecare dată.

În final pentru butonul funcționalitatea de calcul s-a adăugat un ascultător de evenimente bazat pe click care declanșează funcția *updateValues()*;

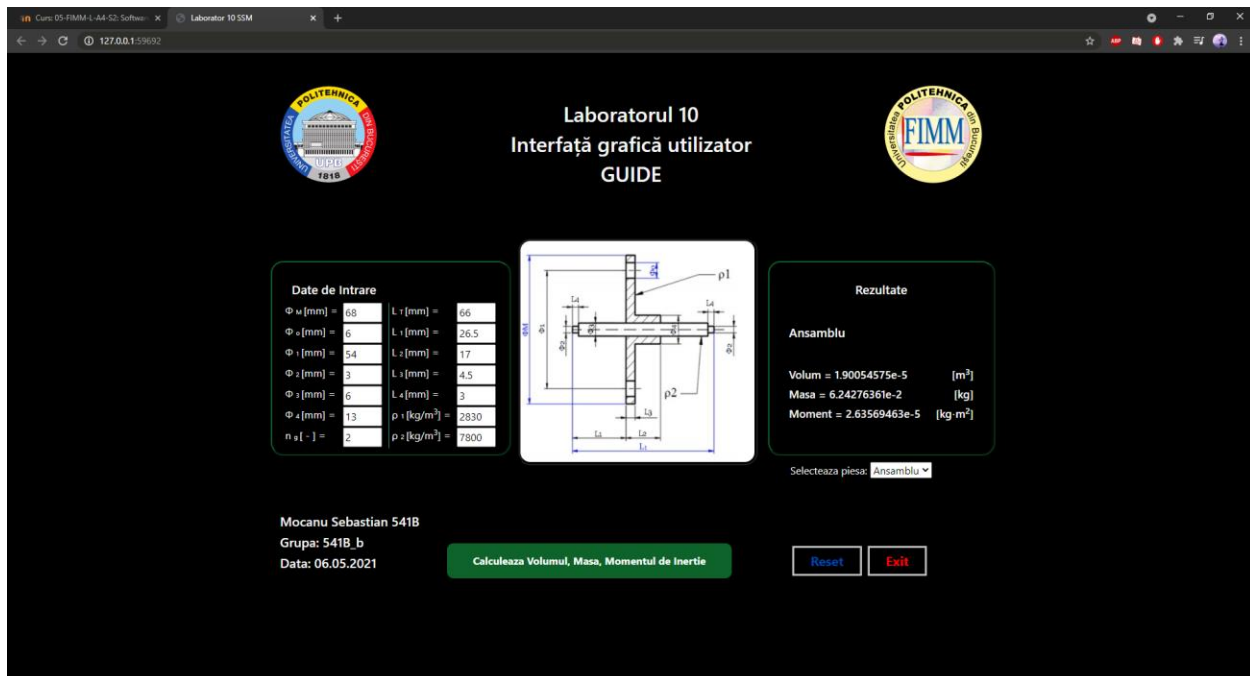


Fig. 8 Rezultate Ansamblu

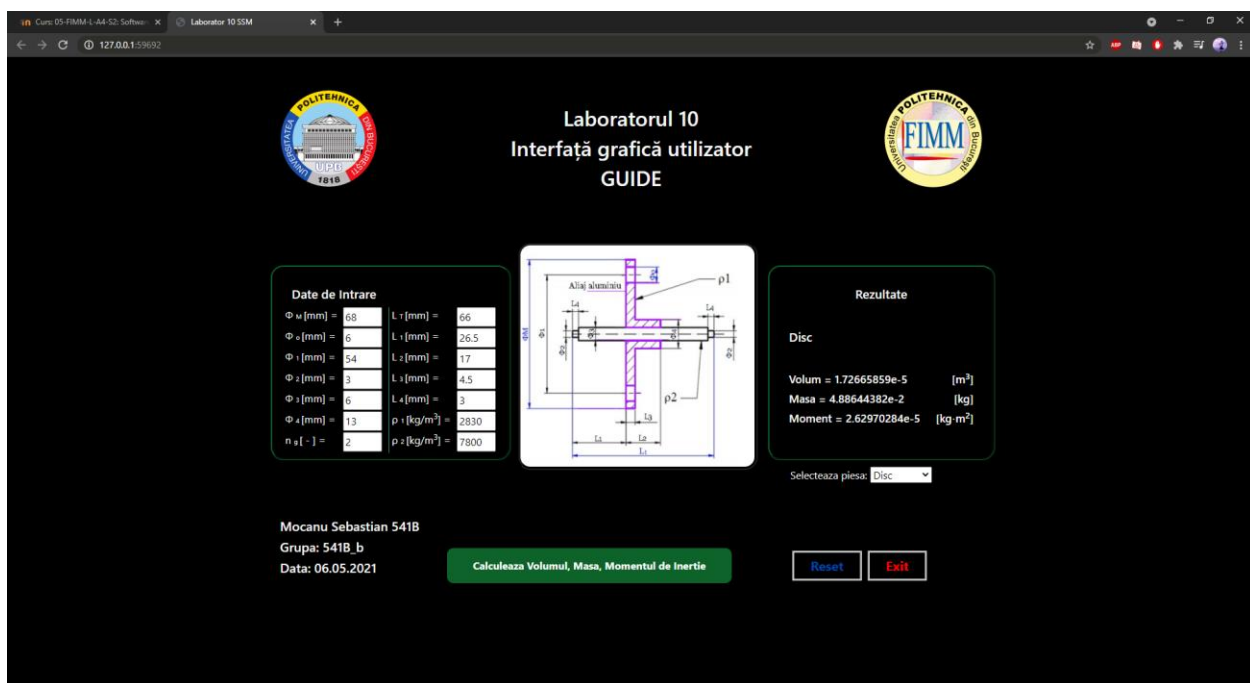


Fig. 9 Rezultate Disc

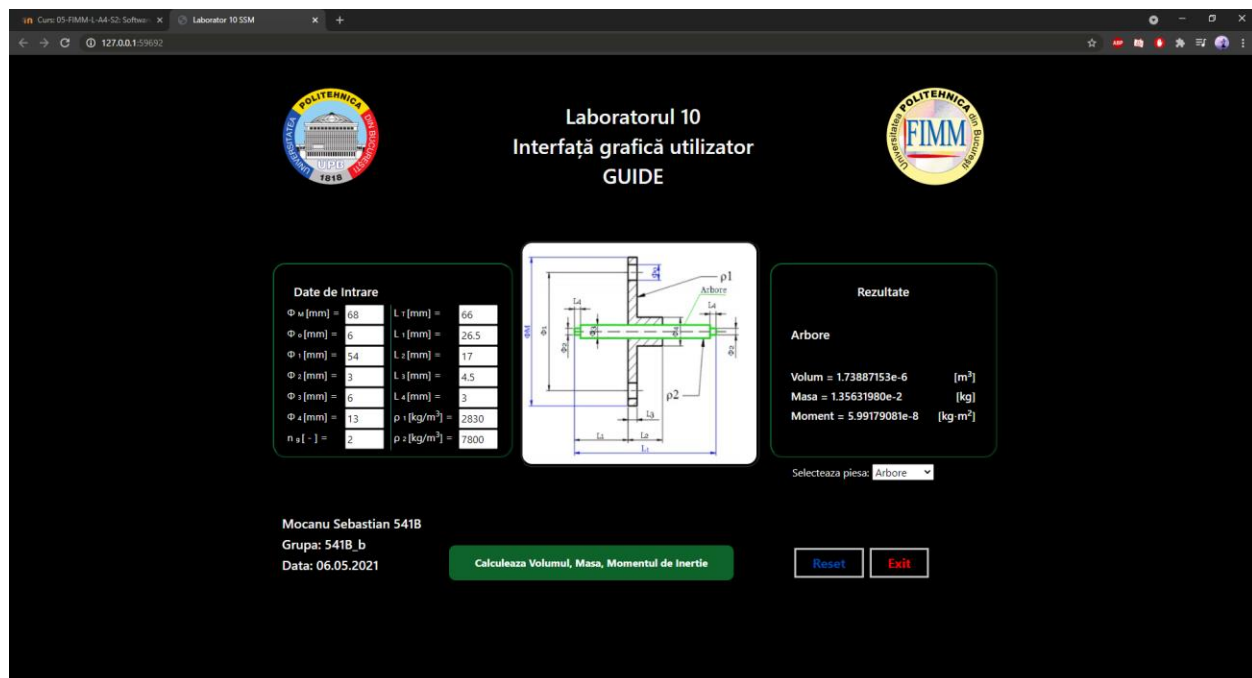


Fig. 10 Rezultate Arbore

3. Soluție editor de text

Un editor de text potrivit pentru aceasta activitate poate fi Atom, acesta este OpenSource, gratuit, si oferă o gama larga de pachete pentru a ușura scrisul si munca depusa.

Link către pagina principală:

<https://atom.io/>

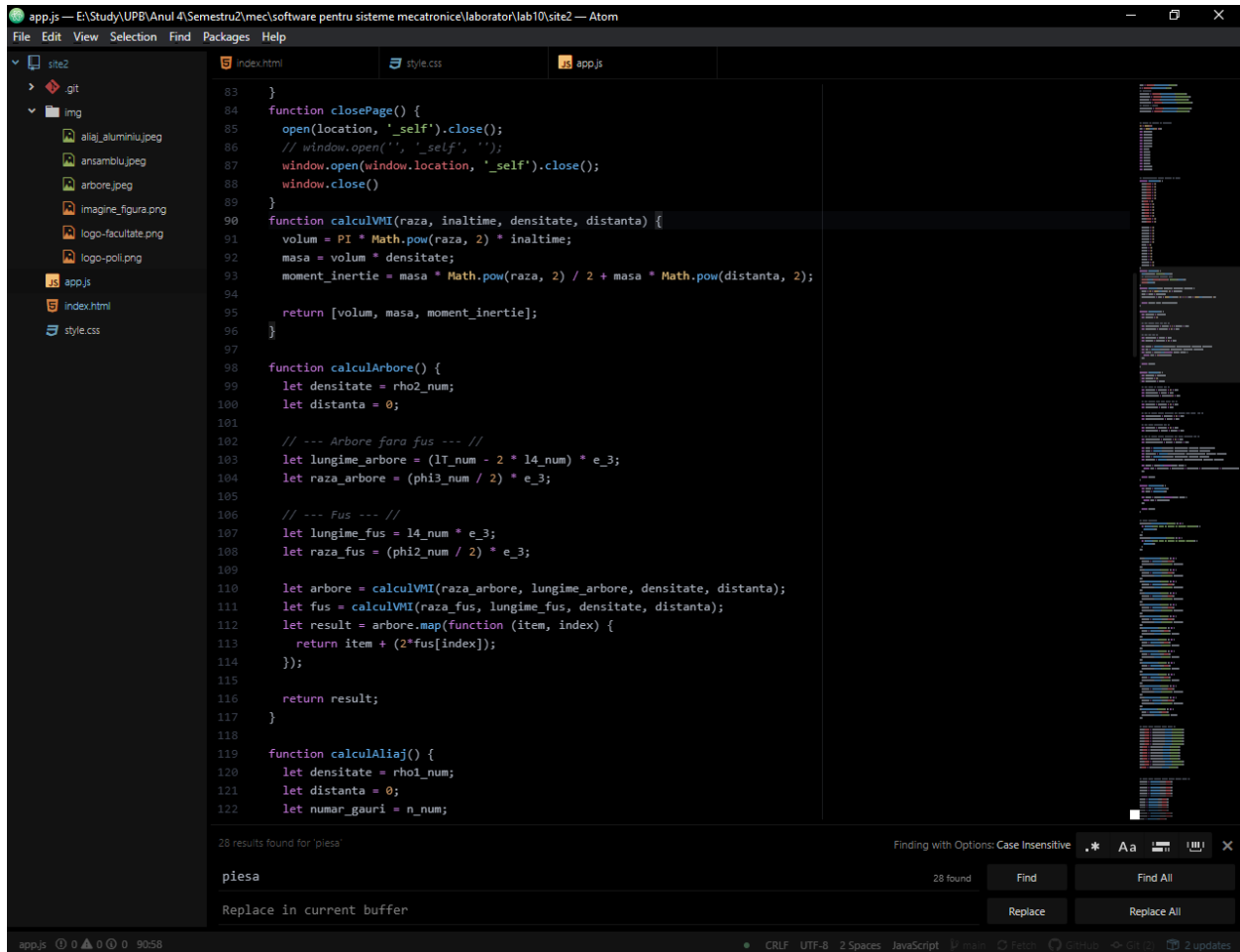
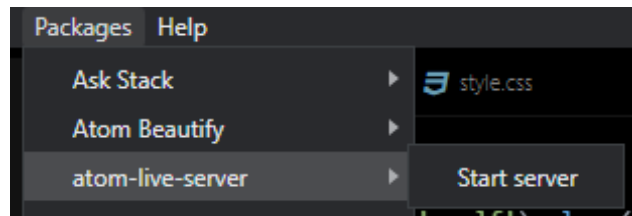


Fig. 11 Editor de text Atom

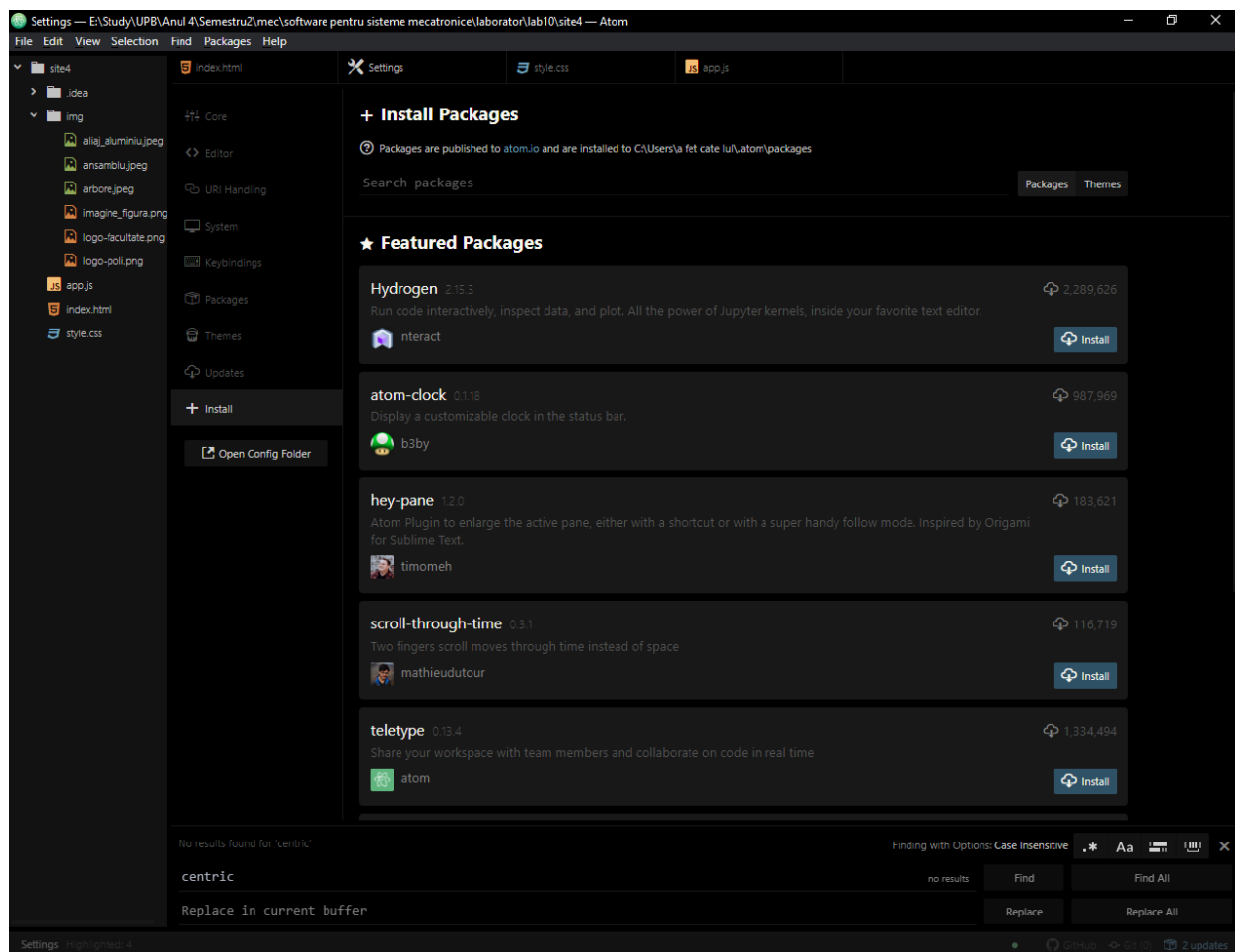
Acest editor are si diverse pachete pentru a putea ajuta in dezvoltarea aplicațiilor, spre exemplu in figura 12 se pot observa trei pachete, Ask Stack care poate ajuta la punerea întrebărilor pe StackOverflow, un website unde programatorii pun întrebări despre dificultățile lor iar alți programatori pot răspunde, apoi Atom Beautify care aranjeaza codul dupa niste standarde STAS

cum ar fi indentare, space-uri etc. si atom-live-server care deschide un server unde se pot observa modificările in timp real pe site.



Figură 12 Pachete utilizabile

Aceste pachete se pot instala din editor, trebuie sa se intre la settings si apoi la install, după se poate cauta pachet-ul dorit



Figură 13 Căutare de pachete

Bineînțeles exista si alte editoare de text similare cum ar fi VisualStudio Code, Notepad++, VIM, Sublime Text, etc. Alegerea editorului tine de preferinta.

4. Adăugarea programului pe GitHub

GitHub reprezintă un serviciu de găzduire web pentru proiecte de dezvoltare a software-ului care utilizează sistemul de control al versionarii Git.

Pentru urcarea proiectelor pe Github este necesar un cont, se poate face un cont pe site-ul oficial Github:

<https://github.com/>

4.1. Instalarea Git

Git este un sistem de version control care rulează pe majoritatea platformelor. Acesta este folosit de echipe de dezvoltare mari, în general nu există companie de programare care să nu folosească Git. Pentru a putea să utilizăm Github este necesară instalarea Git. Pe unele versiuni de linux acesta este deja instalat. Se poate descărca și instala de pe acest link:

<https://git-scm.com/downloads>

4.2. Crearea unui repertoriu și urcarea programului în acesta

Pentru a putea urca proiectul realizat pe Github este necesară crearea unui nou Repository <https://github.com/new>, apoi se va redirecționa la repository-ul nou creat unde instrucțiunile de adăugare vor fi clare.

Se deschide Git Bash și se schimbă directorul în directorul proiectului, pentru ușurință se poate da click dreapta în director și va exista și opțiunea de Git Bash

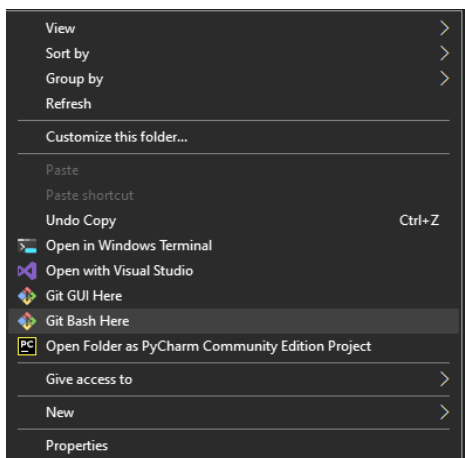
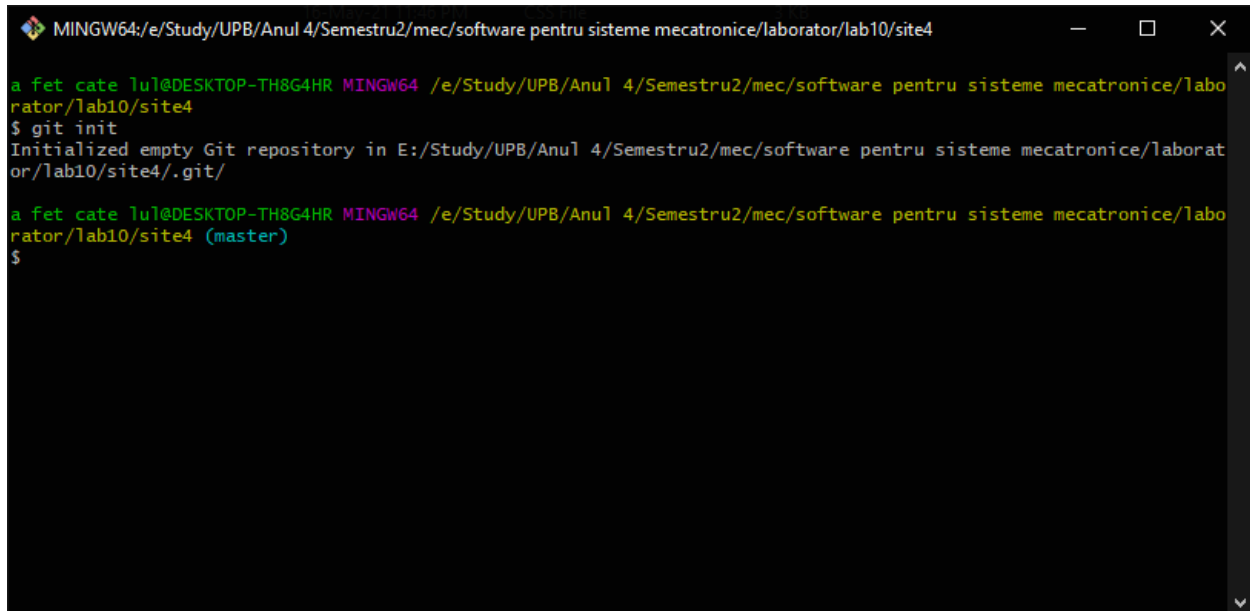


Fig. 14 Deschiderea Git Bash

Se da comanda „git init”

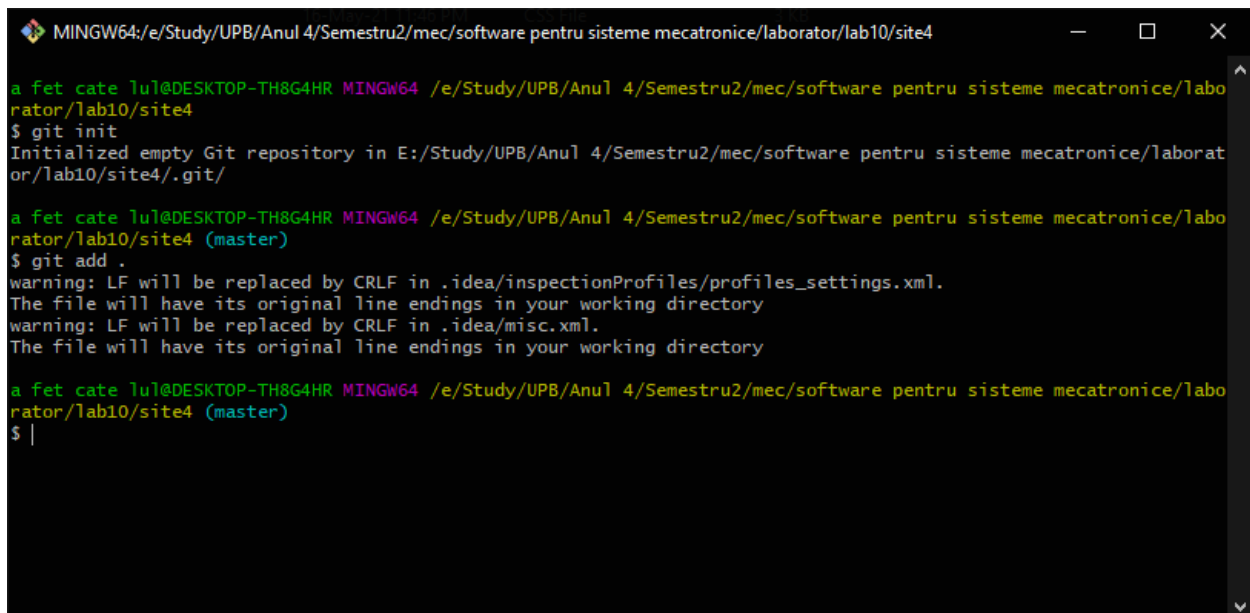


```
MINGW64:/e:/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/laborator/lab10/site4

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e:/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4
$ git init
Initialized empty Git repository in E:/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/laborat
or/lab10/site4/.git/

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e:/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4 (master)
$
```

Comanda „git add .”



```
MINGW64:/e:/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/laborator/lab10/site4

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e:/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4
$ git init
Initialized empty Git repository in E:/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/laborat
or/lab10/site4/.git/

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e:/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4 (master)
$ git add .
warning: LF will be replaced by CRLF in .idea/inspectionProfiles/profiles_settings.xml.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in .idea/misc.xml.
The file will have its original line endings in your working directory

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e:/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4 (master)
$ |
```

Comanda „git commit -m "program versiunea1" “

```
MINGW64:/e/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/laborator/lab10/site4
warning: LF will be replaced by CRLF in .idea/misc.xml.
The file will have its original line endings in your working directory

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4 (master)
$ git commit -m "program versiunea1"
[master (root-commit) dfd7176] program versiunea1
14 files changed, 672 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/site4.iml
create mode 100644 app.js
create mode 100644 img/aliaj_aluminiu.jpeg
create mode 100644 img/ansamblu.jpeg
create mode 100644 img/arbore.jpeg
create mode 100644 img/imagina_figura.png
create mode 100644 img/logo-facultate.png
create mode 100644 img/logo-poli.png
create mode 100644 index.html
create mode 100644 style.css

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4 (master)
$
```

Schimbarea branch-ului pe main „git branch -M main”

```
MINGW64:/e/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/laborator/lab10/site4
rator/lab10/site4 (master)
$ git commit -m "program versiunea1"
[master (root-commit) dfd7176] program versiunea1
14 files changed, 672 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/site4.iml
create mode 100644 app.js
create mode 100644 img/aliaj_aluminiu.jpeg
create mode 100644 img/ansamblu.jpeg
create mode 100644 img/arbore.jpeg
create mode 100644 img/imagina_figura.png
create mode 100644 img/logo-facultate.png
create mode 100644 img/logo-poli.png
create mode 100644 index.html
create mode 100644 style.css

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4 (master)
$ git branch -M main

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4 (main)
$
```

Adăugarea in repo - „git remote add origin https://github.com/brittleru/laboratorul10.git”


```
MINGW64:/e/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/laborator/lab10/site4
create mode 100644 .idea/.gitignore
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/site4.iml
create mode 100644 app.js
create mode 100644 img/aliaj_aluminiu.jpeg
create mode 100644 img/ansamblu.jpeg
create mode 100644 img/arbore.jpeg
create mode 100644 img/imagina_figura.png
create mode 100644 img/logo-facultate.png
create mode 100644 img/logo-poli.png
create mode 100644 index.html
create mode 100644 style.css

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4 (master)
$ git branch -M main

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4 (main)
$ git remote add origin https://github.com/brittleru/laboratoru10.git

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4 (main)
$
```

Urcarea propriu-zisa a programului „git push -u origin main”

```
MINGW64:/e/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/laborator/lab10/site4
create mode 100644 style.css

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4 (master)
$ git branch -M main

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4 (main)
$ git remote add origin https://github.com/brittleru/laboratoru10.git

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4 (main)
$ git push -u origin main
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 12 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (19/19), 242.72 KiB | 18.67 MiB/s, done.
Total 19 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/brittleru/laboratoru10.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

a fet cate lul@DESKTOP-TH8G4HR MINGW64 /e/Study/UPB/Anul 4/Semestru2/mec/software pentru sisteme mecatronice/labo
rator/lab10/site4 (main)
$
```

După urcare repository-ul ar trebui sa arate așa:

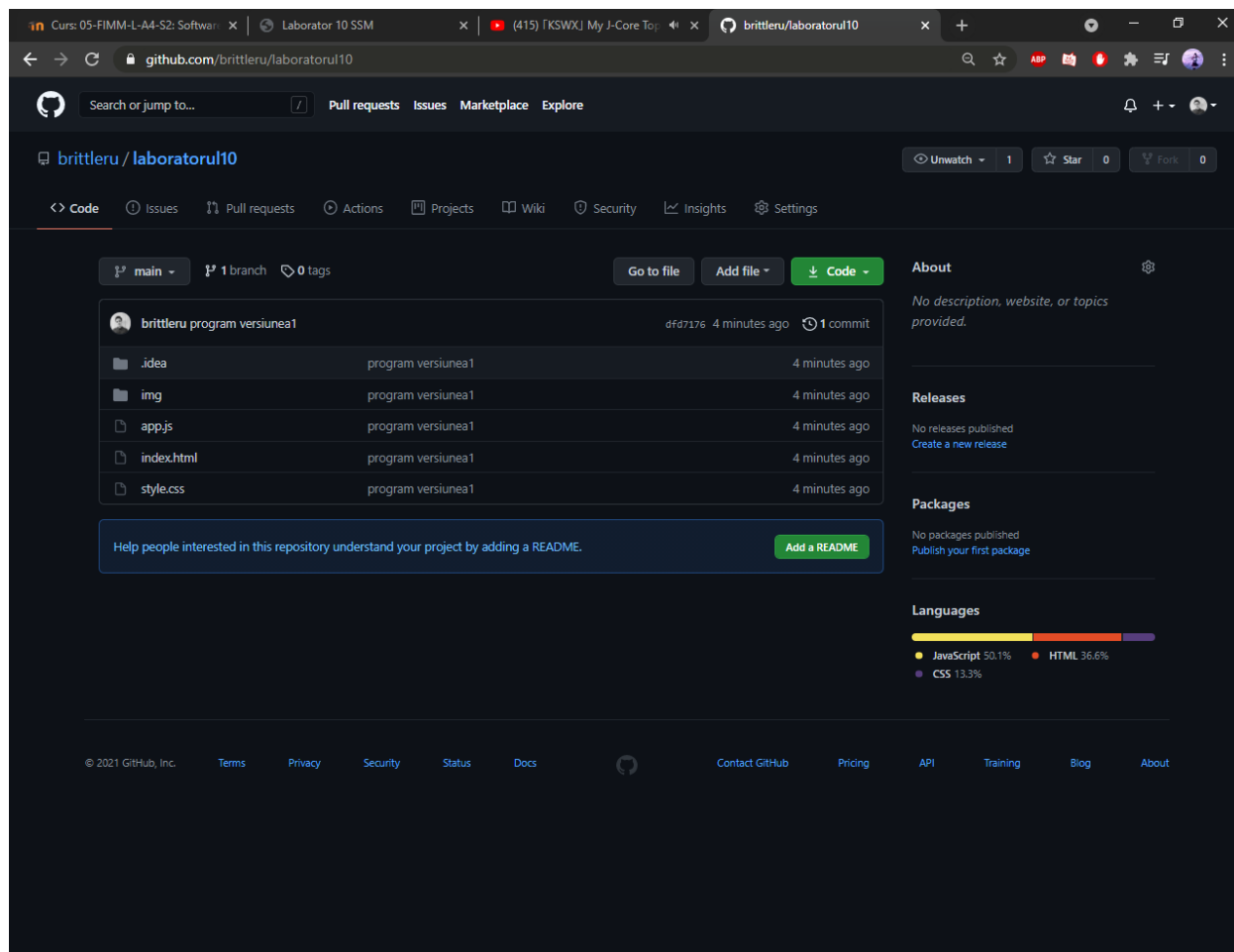


Fig. 15 Repository final

4.3. Adăugarea proiectului la Git Pages

Pentru adăugarea hostarea paginii realizate se in intermediu repository-ului creat se da la Settings > Pages apoi se selecteaza butonul dropdown None si se alege branch-ul „main” si se da save, link-ul fiind disponibil la adresa:

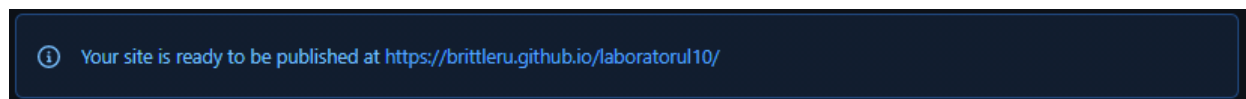


Fig. 16 Link-ul după salvarea pe branch-ul main

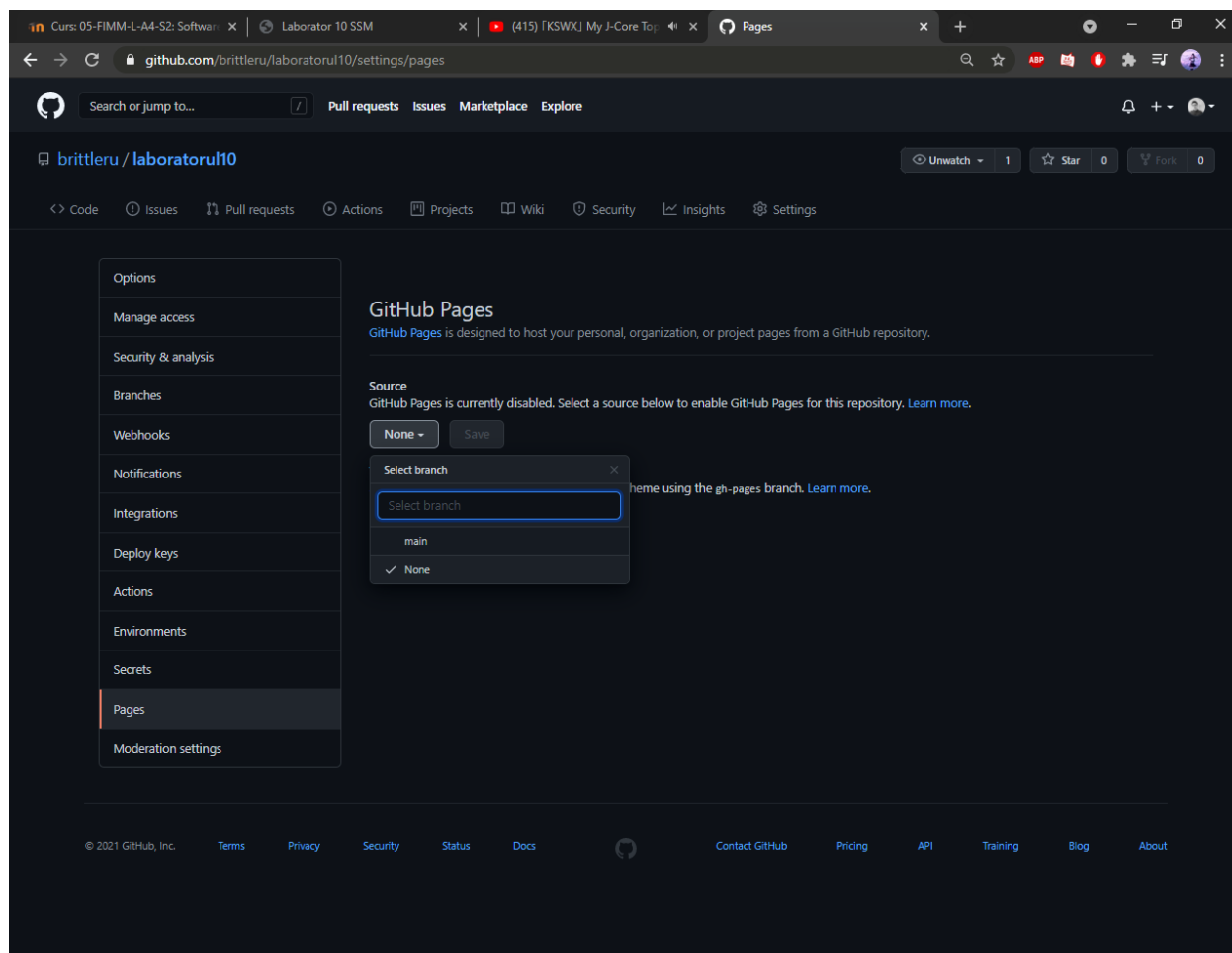


Fig. 17 Adăugare pagina pe branch-ul main

4.4. Link-uri pentru site și program

Codul este disponibil la link-ul:

<https://github.com/brittleru/Laborator10-SSM>

Iar site-ul este disponibil la link-ul:

<https://brittleru.github.io/Laborator10-SSM/>