**Ministerul Educației al Republicii Moldova**

**Colegiul Universității Tehnice a Moldovei**

**Administrarea aplicațiilor WEB**

**RAPORT**

**Lucrarea de laborator nr.1**

**Disciplina:** Asistenta pentru programarea client-side a site-urilor WEB

**Tema:** Dezvoltarea unei aplicații web client-side interactive utilizând JavaScript – interacțiune cu utilizatorul, controlul fluxului și organizarea codului prin funcții

**Efectuat:** st.gr. AAW-231 Botezatu Marian

**Verificat: profesor discipline tehnice** Pistrui Eutalia

**Chișinău 2025**

**Scopul lucrării:**

Formarea și consolidarea competențelor de dezvoltare a aplicațiilor web client-side utilizând limbajul JavaScript, prin realizarea unei pagini web interactive care permite comunicarea cu utilizatorul, controlul fluxului de execuție, prelucrarea datelor și structurarea codului prin funcții proprii.

**Indicații metodice:**

Studentul va fi capabil să:

* A1. Creeze și integreze scripturi JavaScript în pagini web HTML;
* A2. Utilizeze funcțiile de interacțiune cu utilizatorul pentru citirea și afișarea datelor;
* A3. Aplice instrucțiuni condiționale (if, else, switch) pentru luarea deciziilor în aplicații web;
* A4. Utilizeze structuri repetitive și instrucțiuni de salt (for, while, break, continue);
* A5. Elaboreze algoritmi simpli de procesare a datelor (calcul, validare, comparare);
* A6. Declare și apeleze funcții JavaScript proprii, utilizând corect parametrii și domeniul de vizibilitate al variabilelor;
* A7. Organizeze codul JavaScript într-o structură clară și reutilizabilă, respectând bunele practici de programare client-side.

**Sarcinile lucrării:**

**Varianta II (nota 8–10)**

1) Structura proiectului (modulară)

1. Creează folder: Lab\_JS\_01\_NumePrenume\_AV.
2. Creează fișiere:
   * index.html
   * script.js
3. În index.html conectează fișierul JS: <script src="script.js"></script> pus înainte de </body>

2) Interfață minimă în HTML (fără CSS obligatoriu)

1. În body, adaugă:
   * un titlu
   * un paragraf cu instrucțiuni scurte (ce face aplicația)
   * un buton: „Pornește aplicația”
   * o zonă pentru rezultate (ex.: un <div id="rezultat"></div>)
2. În script.js, la click pe buton, pornește aplicația: folosește addEventListener("click", ...)

3) Meniu interactiv (control flux)

1. După pornire, afișează un alert de bun venit și cere numele.
2. Creează un meniu cu prompt, repetat într-o buclă:

„Alege opțiunea:  
1 – Verificare vârstă  
2 – Tabla înmulțirii  
3 – Calculator (sumă, medie, max)  
4 – Joc: ghicește numărul  
0 – Ieșire”

1. Utilizatorul introduce opțiunea.
2. Folosește switch(optiune) pentru a controla aplicația:
   * case "1": ...
   * case "2": ...
   * etc.
3. Dacă opțiunea e invalidă → mesaj și revii la meniu.

4) Funcții obligatorii (modularizare reală)

1. Creează funcțiile:

* citesteText(mesaj) – returnează textul din prompt, tratând null ca șir gol.
* citesteNumar(mesaj) – cere număr până e valid (repetă prompt-ul).
* afiseazaInPagina(text) – scrie rezultatul în div#rezultat (ex.: adaugă paragrafe).
* curataRezultate() – golește zona de rezultate.
* verificareVarsta() – rulează logica vârstei (cu validare).
* tablaInmultirii() – generează tabla și o afișează în pagină + în consolă.
* calculator3Numere() – sumă, medie, max.
* jocGhicit() – joc în buclă (vezi mai jos).

1. Fiecare funcție trebuie să aibă:

* parametri unde are sens;
* return unde are sens;
* comentariu scurt la început: ce primește, ce face, ce returnează.

5) Domeniul de vizibilitate al variabilelor (scope) – cerință obligatorie

1. Demonstrează corect let / const:

* definește o constantă globală: const APP\_NAME = "Asistent JS";
* definește variabile locale în funcții cu let

1. Creează intenționat o variabilă cu același nume în două scope-uri diferite (global vs funcție) și arată în console.log că sunt diferite.

6) Instrucțiuni repetitive + salt (nivel avansat)

1. În tablaInmultirii():

* folosește for
* adaugă continue pentru a sări peste un rând (ex.: când i este multiplu de 3)
* explică în comentariu de ce sari peste el.

1. În meniul principal: folosește buclă while(true) și ieși doar când opțiunea este 0 (folosește break).

7) Algoritm suplimentar: „Joc – ghicește numărul” (obligatoriu)

1. La opțiunea 4:

* generează un număr aleator între 1 și 20:
  + Math.floor(Math.random() \* 20) + 1
* utilizatorul are 5 încercări.
* pentru fiecare încercare:
  + cere numărul
  + dacă e prea mic → mesaj
  + dacă e prea mare → mesaj
  + dacă e corect → mesaj de succes și ieși din joc (break)
* dacă se termină încercările → afișează numărul corect.

1. Afișează în pagină:

* numărul de încercări folosite
* dacă a câștigat sau nu

8) Cerințe de calitate (nota 8–10)

1. Fără cod duplicat: orice validare repetată trebuie pusă în funcții.
2. Rezultatele să fie afișate:

* minim o dată cu alert
* minim o dată în console.log
* minim o dată în pagină (div#rezultat)

1. Cod curat:

* denumiri clare (camelCase)
* comentarii scurte și utile
* fără variabile globale inutile

**Produsul variantei II:**

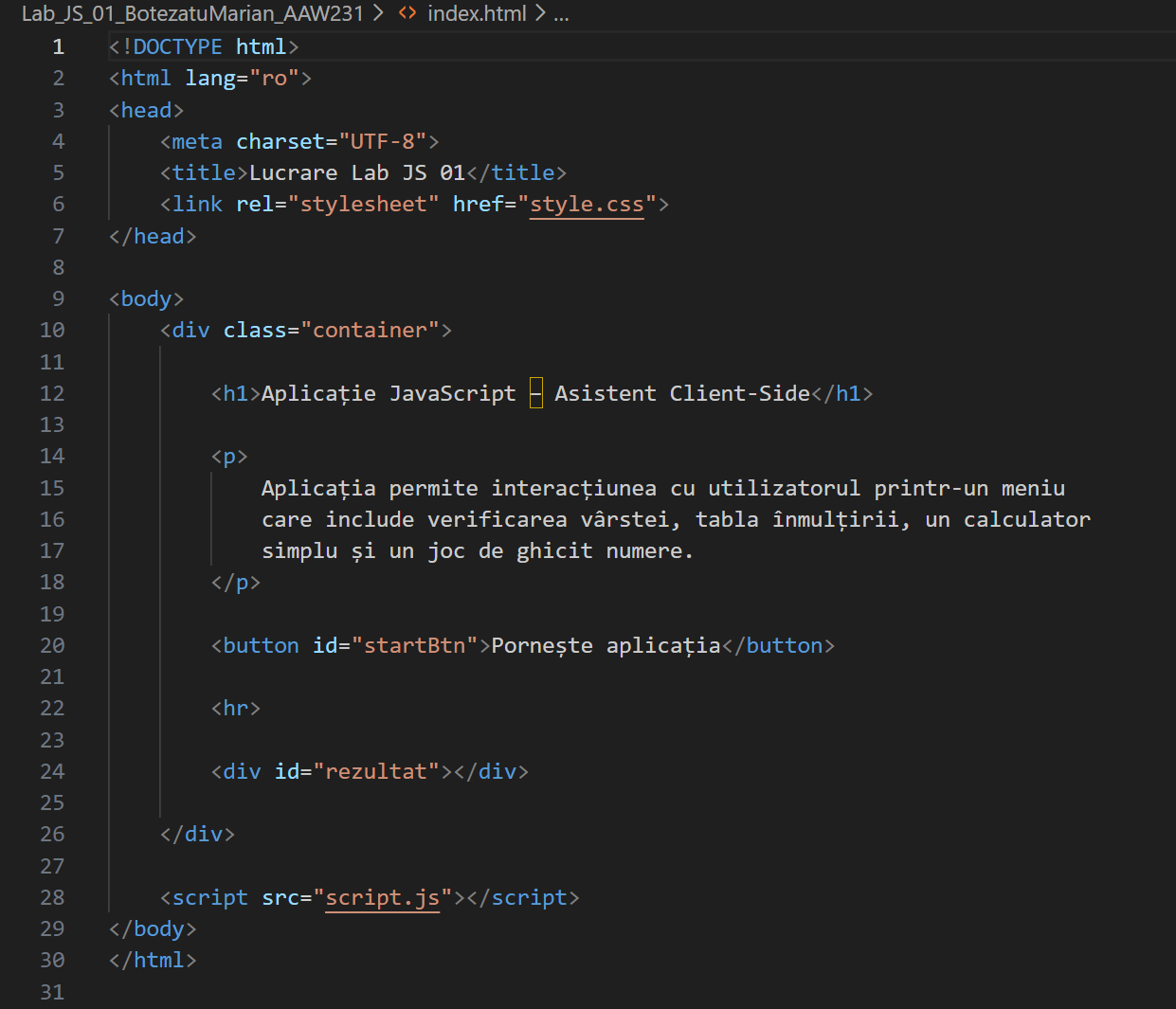
* index.html
* script.js
* raport

**Mersul lucării:**

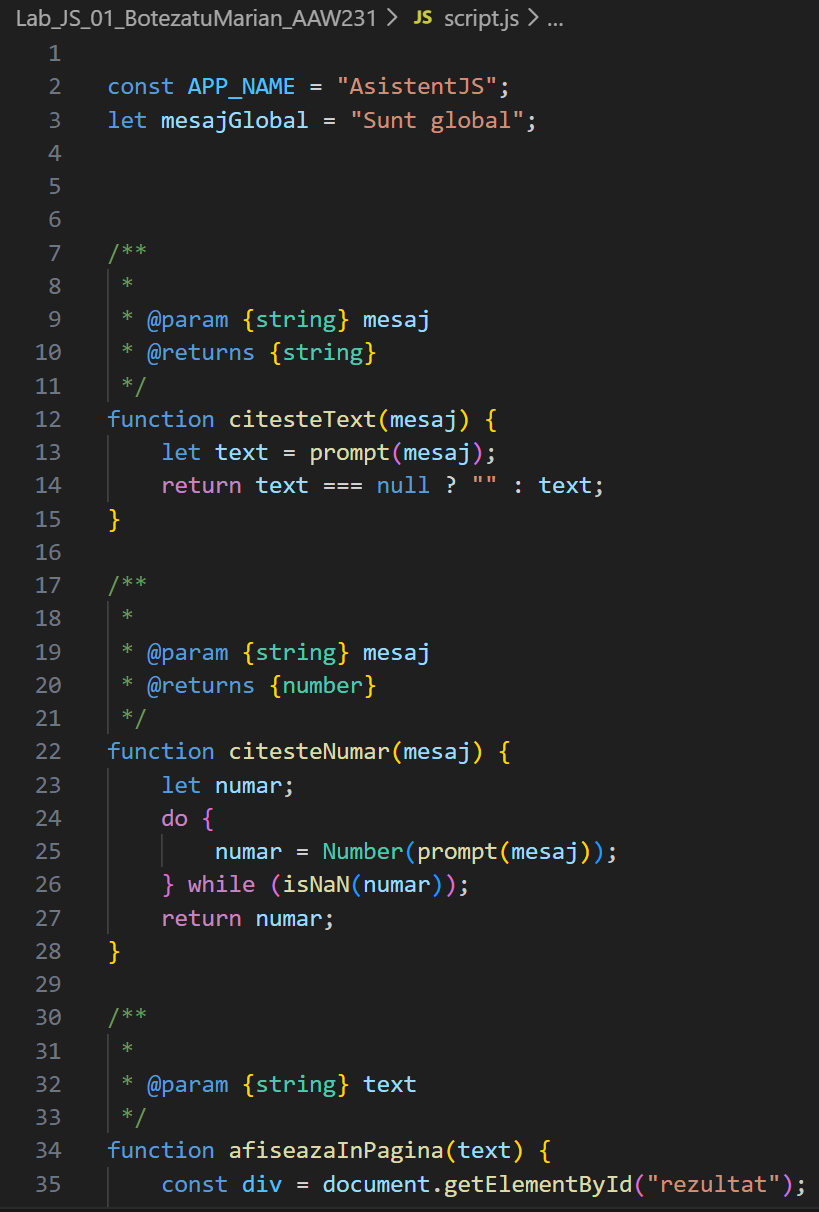
1. A fost creat folderul proiectului *Lab\_JS\_01\_NumePrenume\_AAW*, care conține fișierele principale ale aplicației.



1. A fost realizat fișierul index.html, în care s-a definit structura de bază a paginii web, incluzând titlul aplicației, descrierea, butonul de pornire și zona destinată afișării rezultatelor.

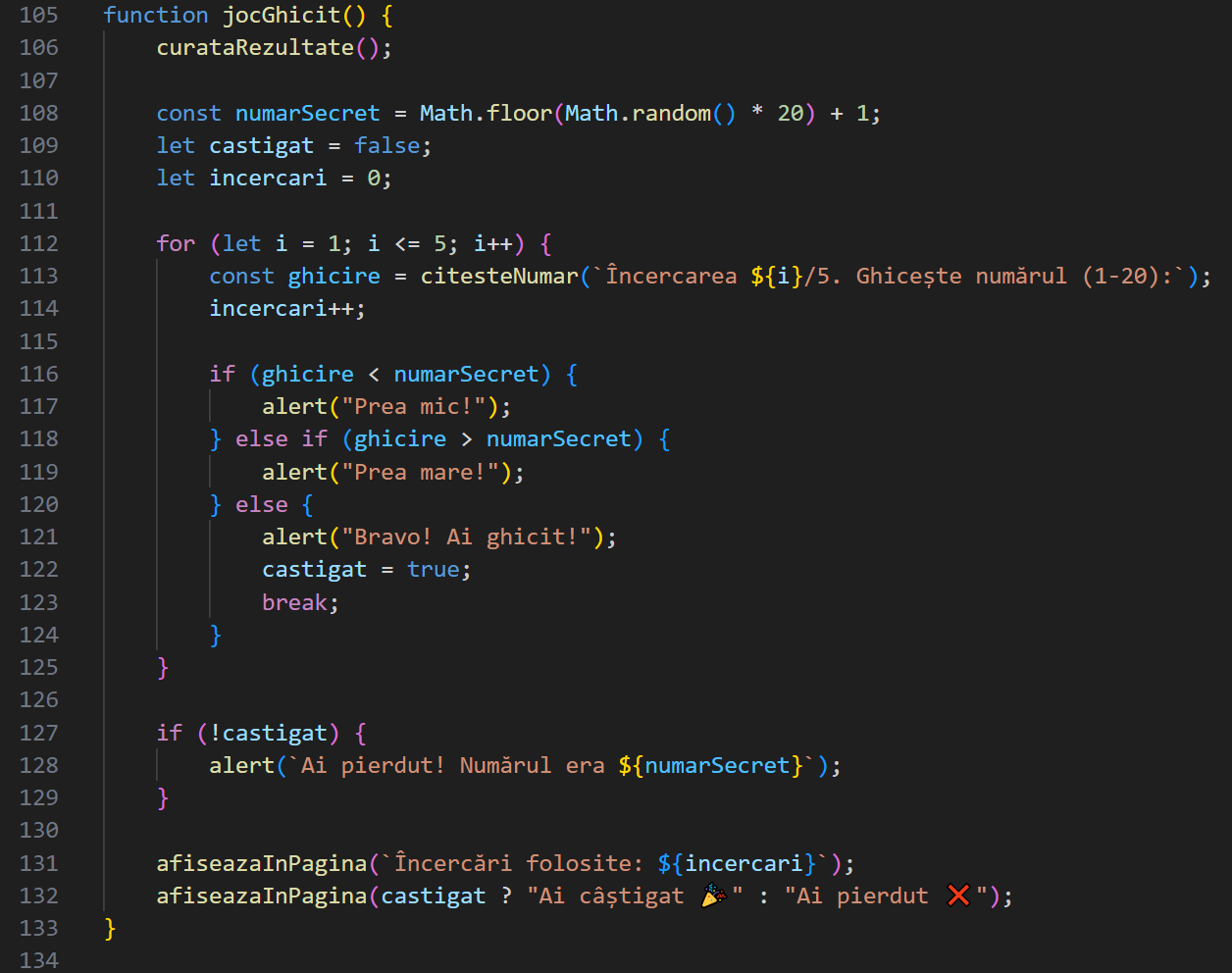


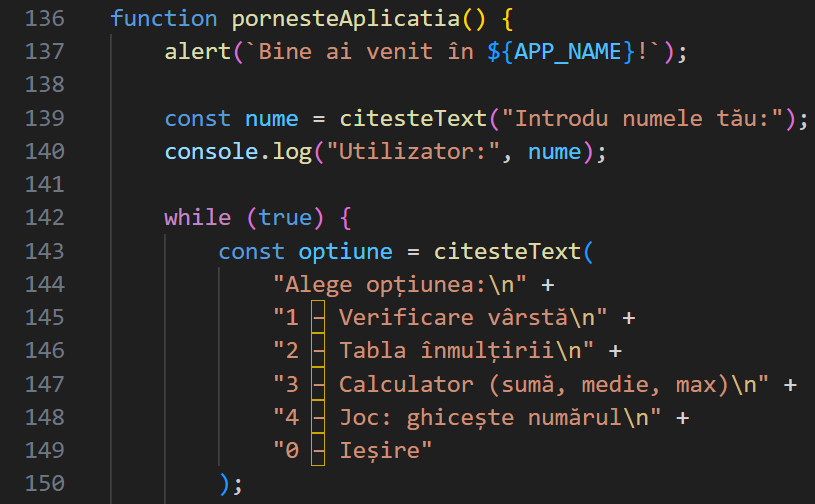
1. A fost creat fișierul script.js, care conține codul JavaScript responsabil de logica aplicației și de interacțiunea cu utilizatorul.









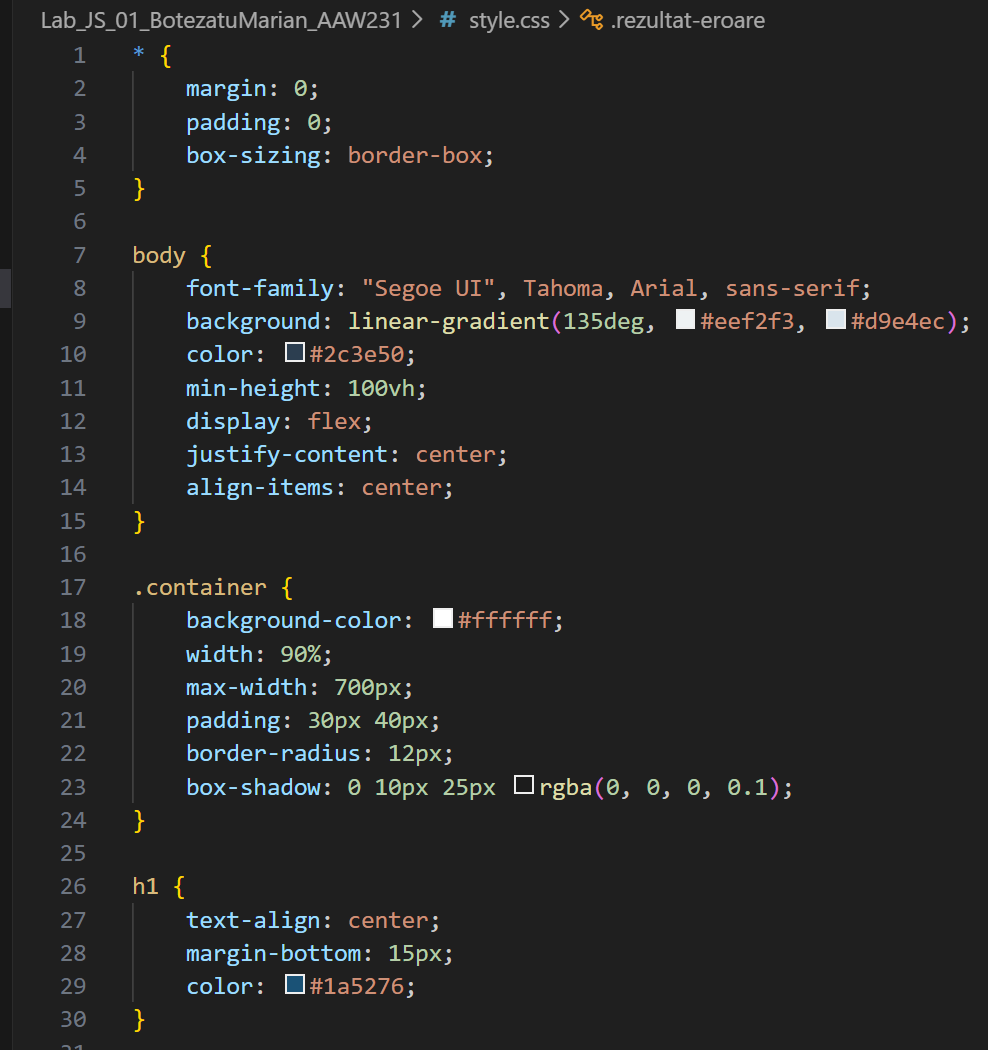




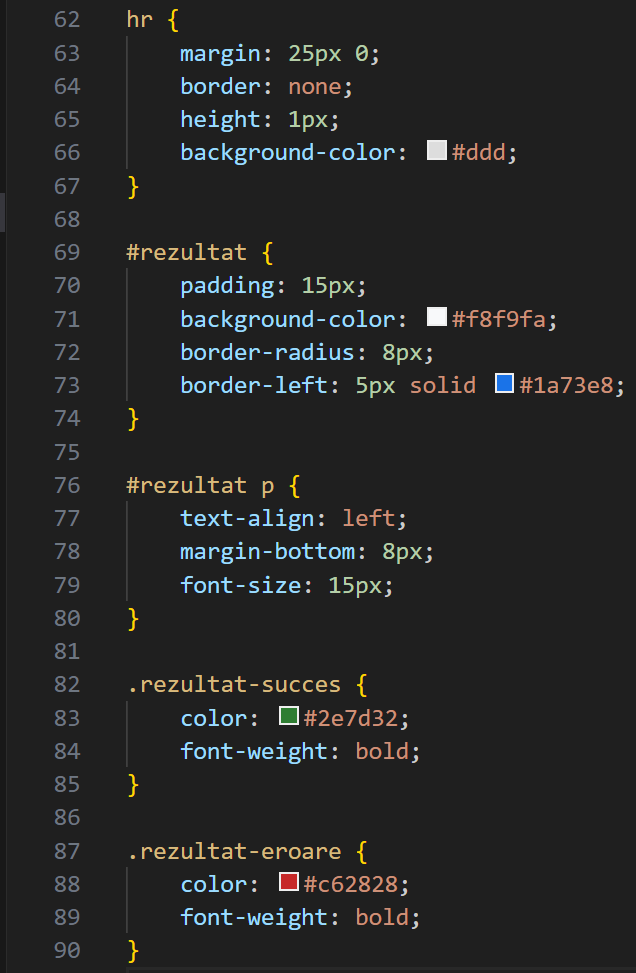
1. Fișierul JavaScript a fost conectat extern în pagina HTML utilizând elementul <script>, plasat înainte de închiderea etichetei </body>.



1. A fost creat fișierul style.css pentru stilizarea aplicației web, în scopul îmbunătățirii aspectului vizual și al lizibilității.



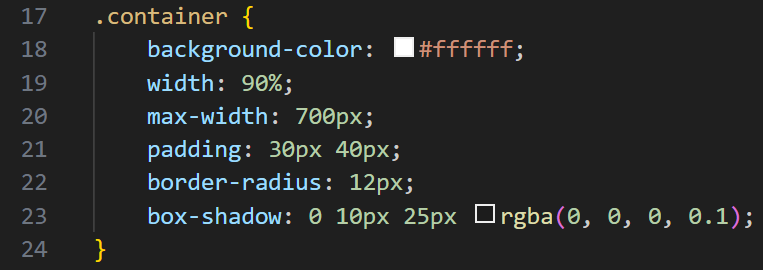




1. Fișierul CSS a fost conectat în index.html prin intermediul etichetei <link>, fiind aplicate stiluri pentru corpul paginii, containerul principal, titlu, buton și zona de rezultate.



1. A fost implementat un container principal pentru organizarea conținutului paginii și centrare acestuia în fereastra browserului.



1. În fișierul script.js au fost definite funcții JavaScript pentru citirea și validarea datelor introduse de utilizator, afișarea rezultatelor și curățarea conținutului afișat.

/\*\*

 \*

 \* @param {string} mesaj

 \* @returns {string}

 \*/

function citesteText(mesaj) {

    let text = prompt(mesaj);

    return text === null ? "" : text;

}

/\*\*

 \*

 \* @param {string} mesaj

 \* @returns {number}

 \*/

function citesteNumar(mesaj) {

    let numar;

    do {

        numar = Number(prompt(mesaj));

    } while (isNaN(numar));

    return numar;

}

/\*\*

 \*

 \* @param {string} text

 \*/

function afiseazaInPagina(text) {

    const div = document.getElementById("rezultat");

    const p = document.createElement("p");

    p.textContent = text;

    div.appendChild(p);

}

function curataRezultate() {

    document.getElementById("rezultat").innerHTML = "";

}

1. A fost realizat un meniu interactiv utilizând instrucțiunea switch, care permite accesarea funcționalităților aplicației.

function pornesteAplicatia() {

    alert(`Bine ai venit în ${APP\_NAME}!`);

    const nume = citesteText("Introdu numele tău:");

    console.log("Utilizator:", nume);

    while (true) {

        const optiune = citesteText(

            "Alege opțiunea:\n" +

            "1 – Verificare vârstă\n" +

            "2 – Tabla înmulțirii\n" +

            "3 – Calculator (sumă, medie, max)\n" +

            "4 – Joc: ghicește numărul\n" +

            "0 – Ieșire"

        );

        switch (optiune) {

            case "1":

                verificareVarsta();

                break;

            case "2":

                tablaInmultirii();

                break;

            case "3":

                calculator3Numere();

                break;

            case "4":

                jocGhicit();

                break;

            case "0":

                alert("La revedere!");

                return;

            default:

                alert("Opțiune invalidă!");

        }

    }

}

1. Au fost utilizate structuri repetitive (for, while) și instrucțiuni de salt (break, continue) pentru controlul fluxului de execuție.

function verificareVarsta() {

    let mesajGlobal = "Sunt local";

    console.log("Global:", window.mesajGlobal);

    console.log("Local:", mesajGlobal);

    const varsta = citesteNumar("Introdu vârsta:");

    let mesaj;

    if (varsta < 18) {

        mesaj = "Ești minor.";

    } else {

        mesaj = "Ești major.";

    }

    alert(mesaj);

    console.log(mesaj);

    afiseazaInPagina(mesaj);

}

function tablaInmultirii() {

    curataRezultate();

    const n = citesteNumar("Introdu un număr pentru tabla înmulțirii:");

    for (let i = 1; i <= 10; i++) {

        if (i % 3 === 0) {

            continue;

        }

        const rezultat = `${n} x ${i} = ${n \* i}`;

        console.log(rezultat);

        afiseazaInPagina(rezultat);

    }

}

function calculator3Numere() {

    curataRezultate();

    const a = citesteNumar("Introdu primul număr:");

    const b = citesteNumar("Introdu al doilea număr:");

    const c = citesteNumar("Introdu al treilea număr:");

    const suma = a + b + c;

    const medie = suma / 3;

    const max = Math.max(a, b, c);

    alert(`Suma: ${suma}`);

    console.log("Rezultate calculator:", suma, medie, max);

    afiseazaInPagina(`Suma: ${suma}`);

    afiseazaInPagina(`Media: ${medie}`);

    afiseazaInPagina(`Maximul: ${max}`);

}

function jocGhicit() {

    curataRezultate();

    const numarSecret = Math.floor(Math.random() \* 20) + 1;

    let castigat = false;

    let incercari = 0;

    for (let i = 1; i <= 5; i++) {

        const ghicire = citesteNumar(`Încercarea ${i}/5. Ghicește numărul (1-20):`);

        incercari++;

        if (ghicire < numarSecret) {

            alert("Prea mic!");

        } else if (ghicire > numarSecret) {

            alert("Prea mare!");

        } else {

            alert("Bravo! Ai ghicit!");

            castigat = true;

            break;

        }

    }

    if (!castigat) {

        alert(`Ai pierdut! Numărul era ${numarSecret}`);

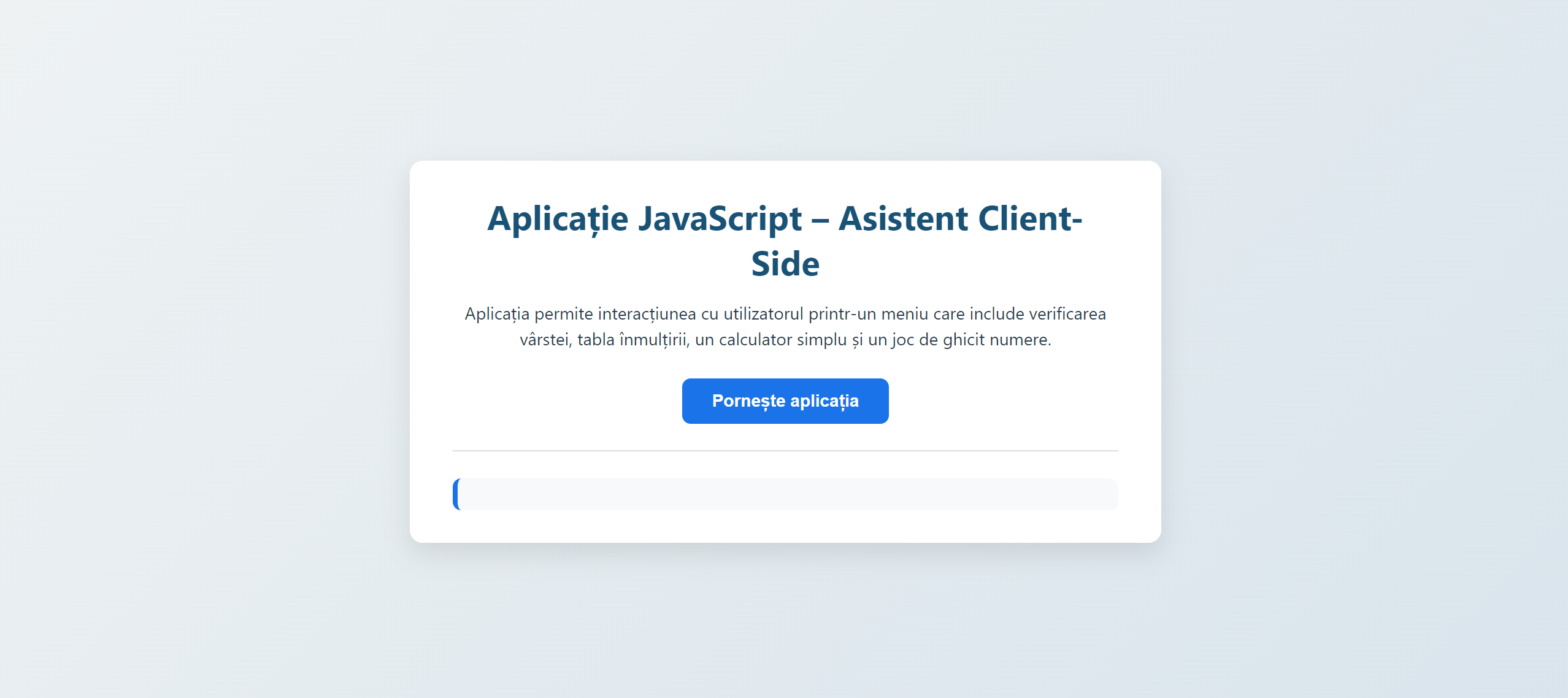
    }

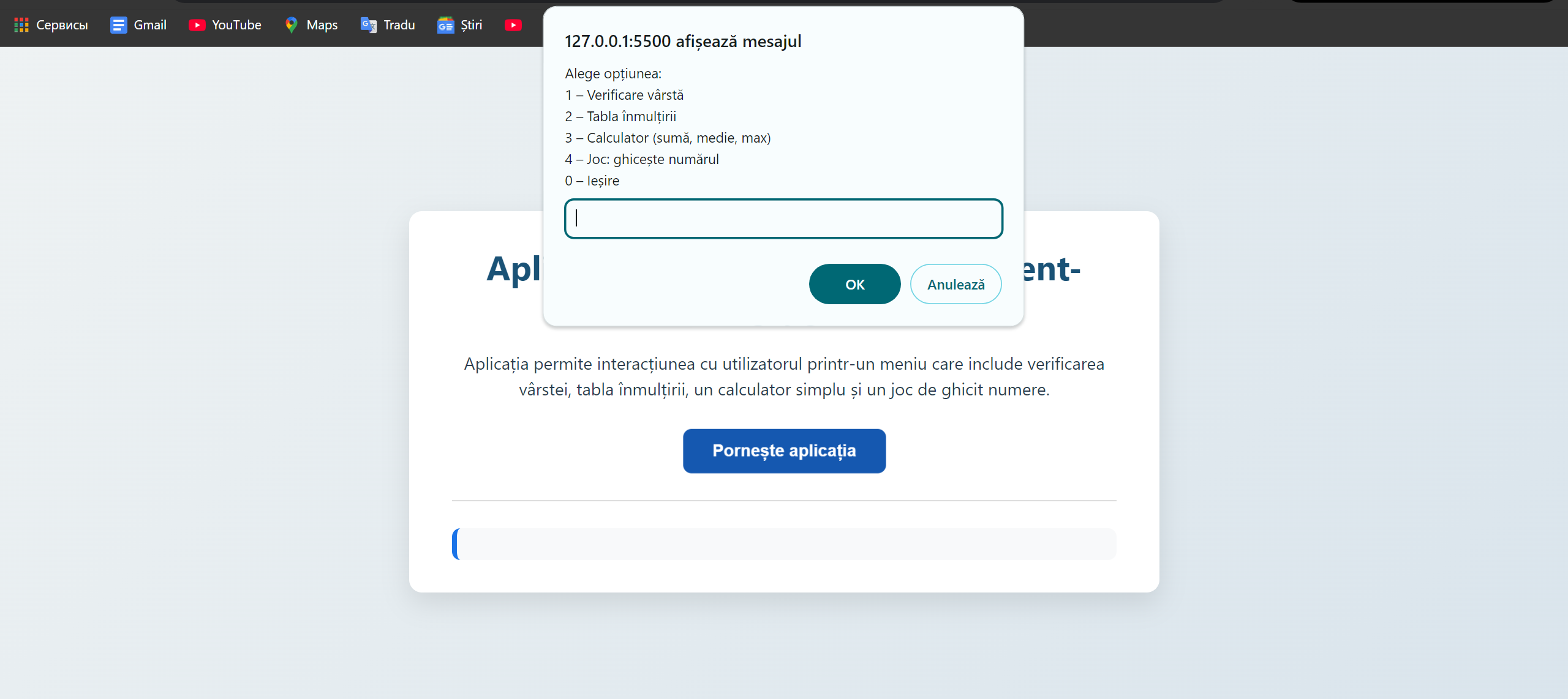
    afiseazaInPagina(`Încercări folosite: ${incercari}`);

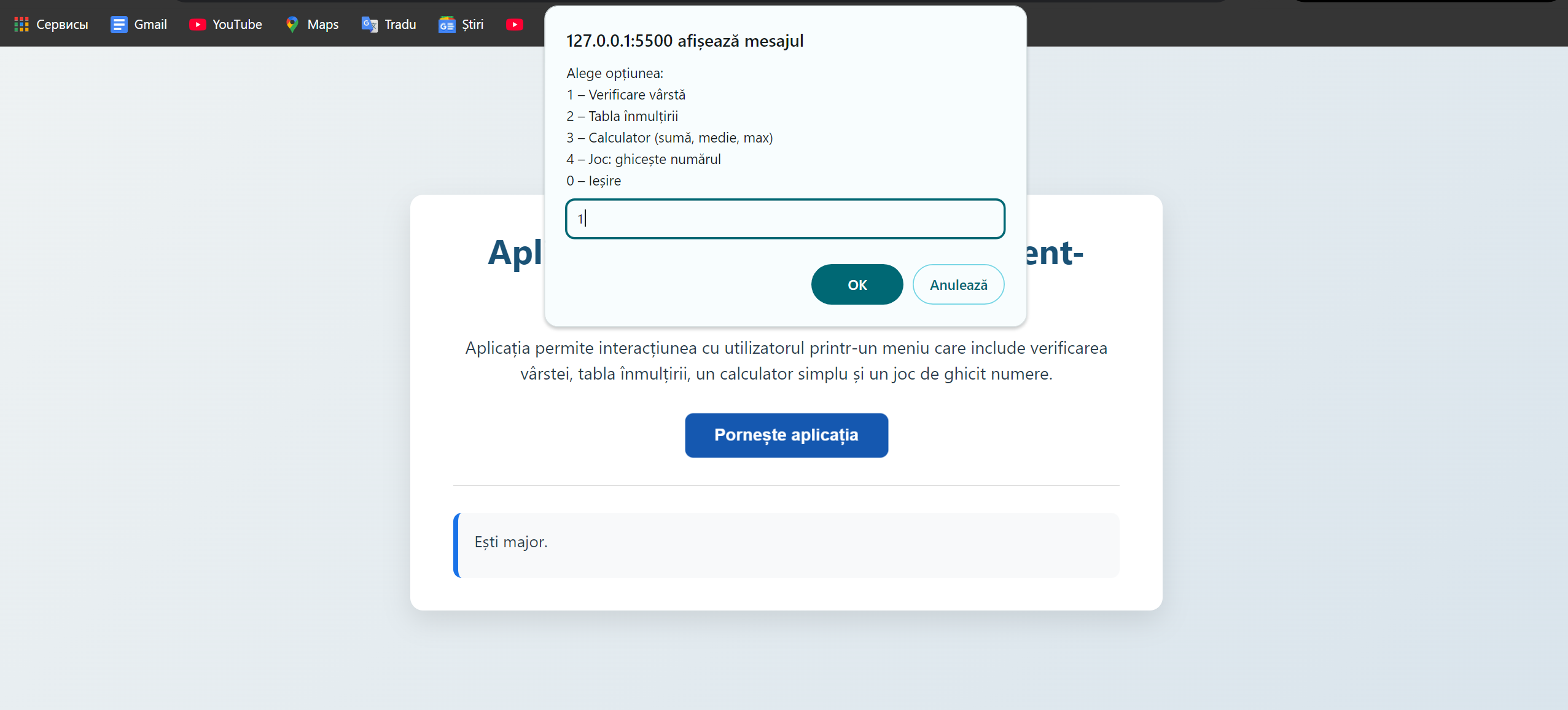
    afiseazaInPagina(castigat ? "Ai câștigat 🎉" : "Ai pierdut ❌");

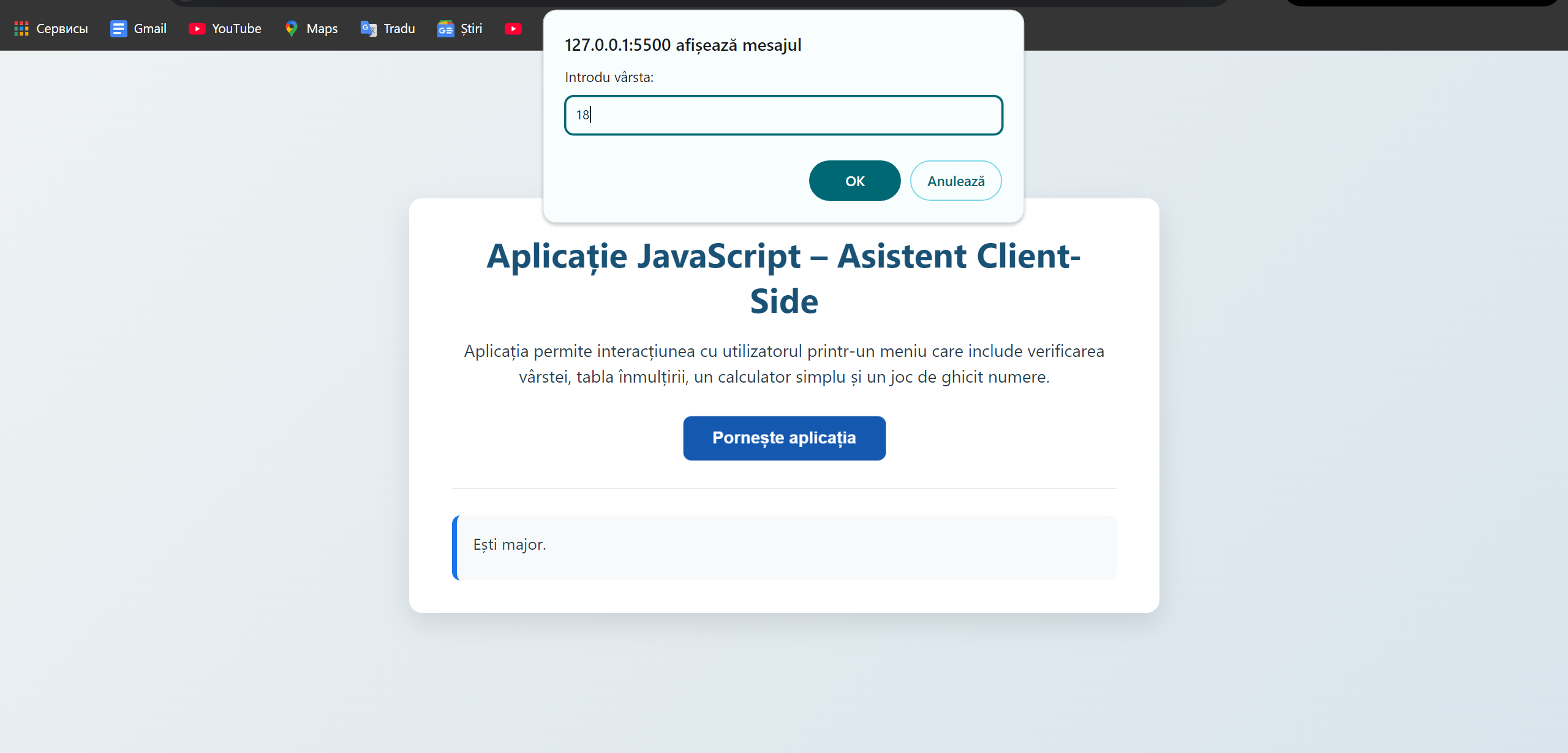
}

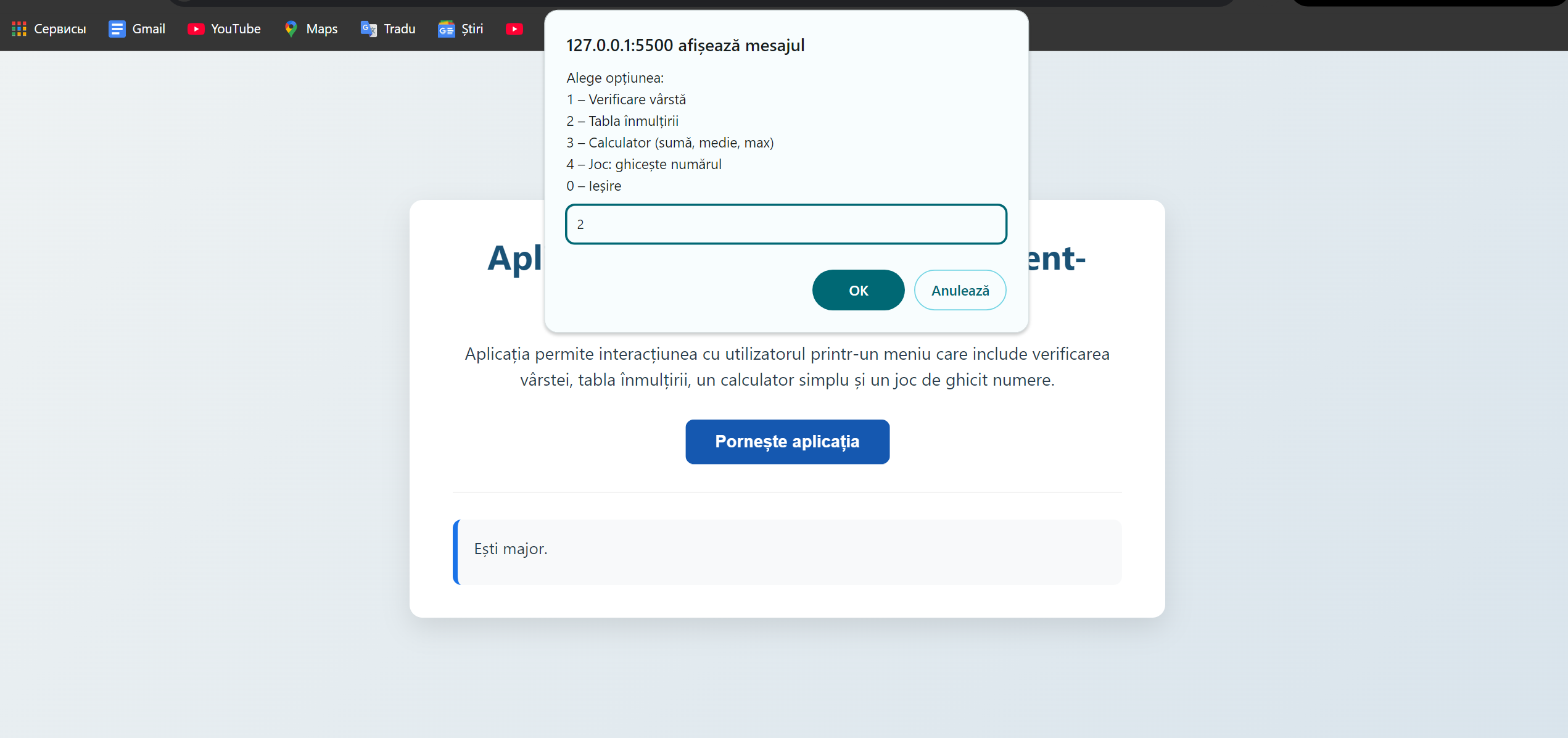
1. A fost demonstrată utilizarea corectă a domeniului de vizibilitate al variabilelor prin folosirea const și let.
2. Rezultatele obținute în urma executării aplicației au fost afișate utilizând mesaje alert, în consola browserului și direct în pagina web.

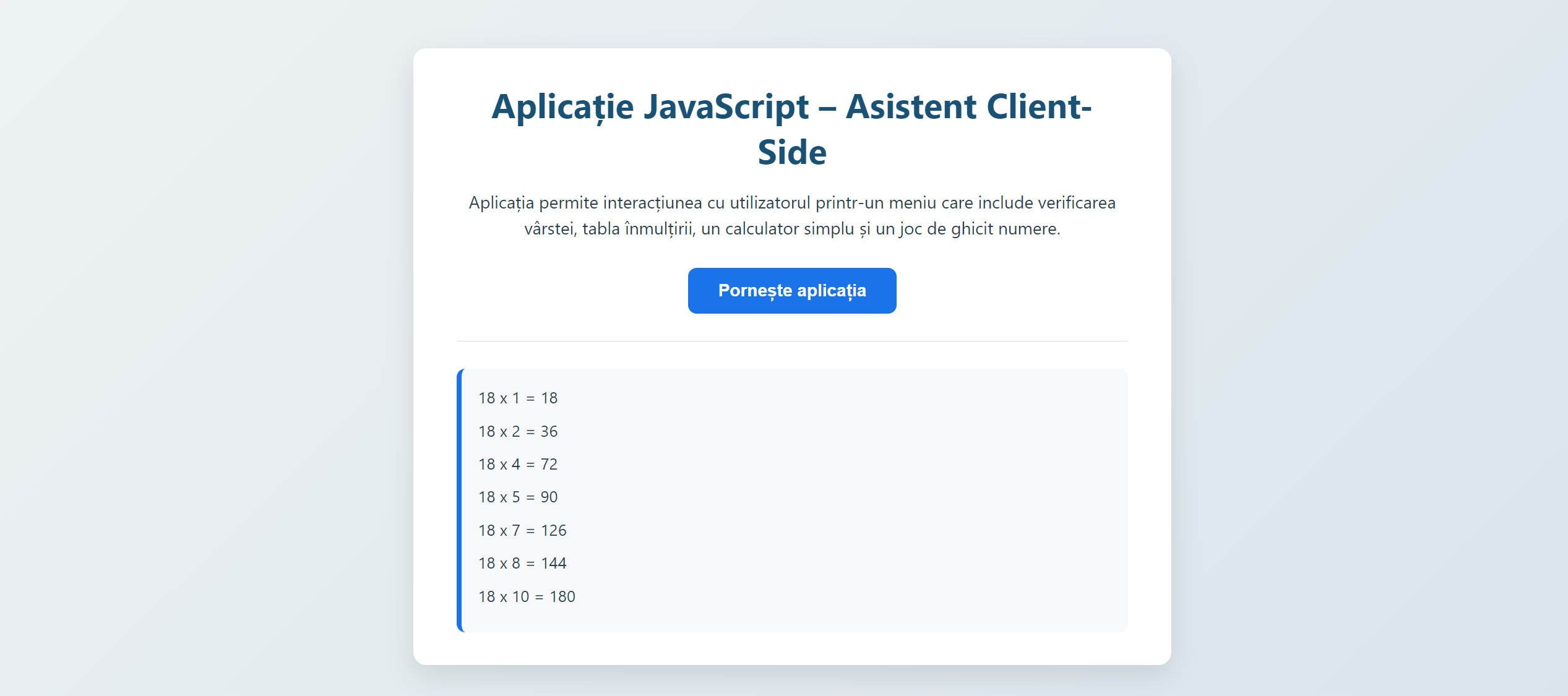
****

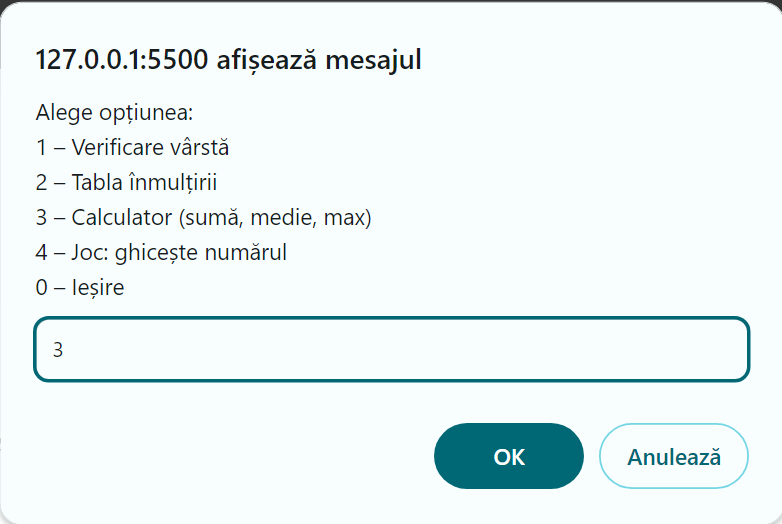
****

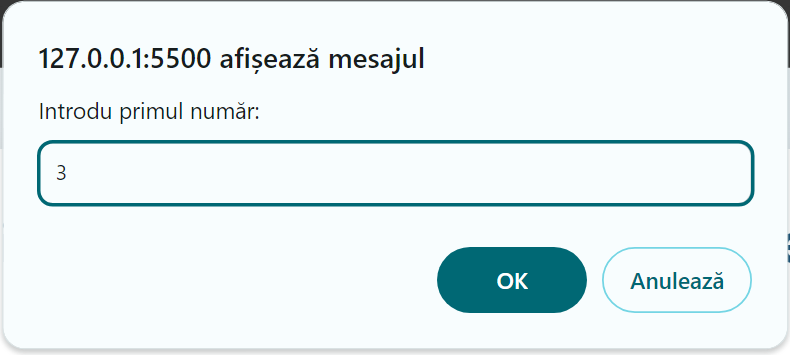
****

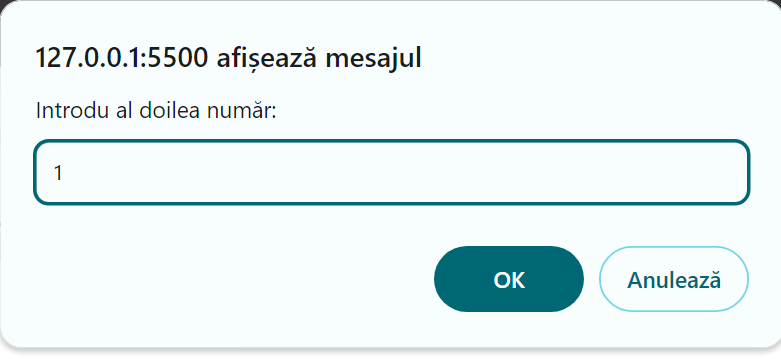
****

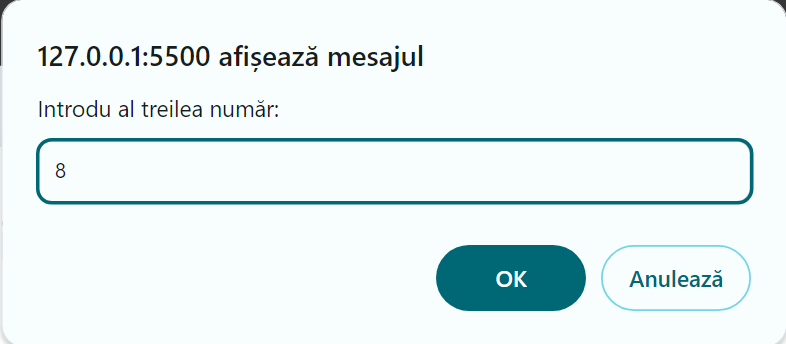
****

****

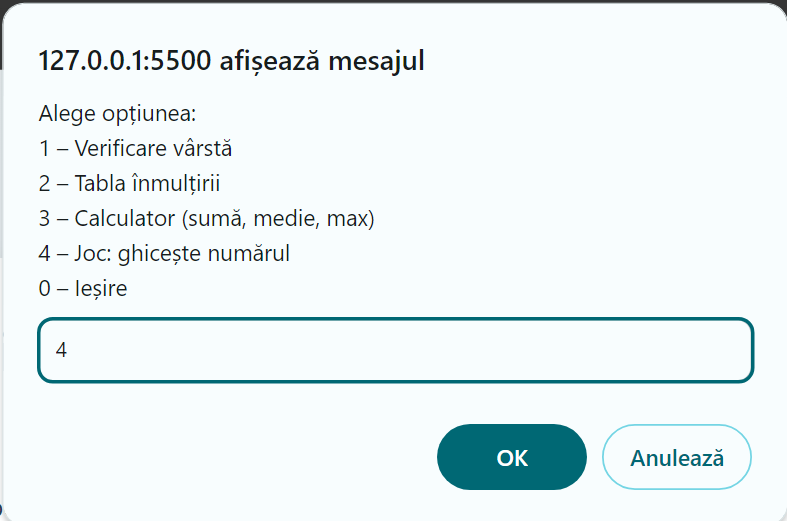
****

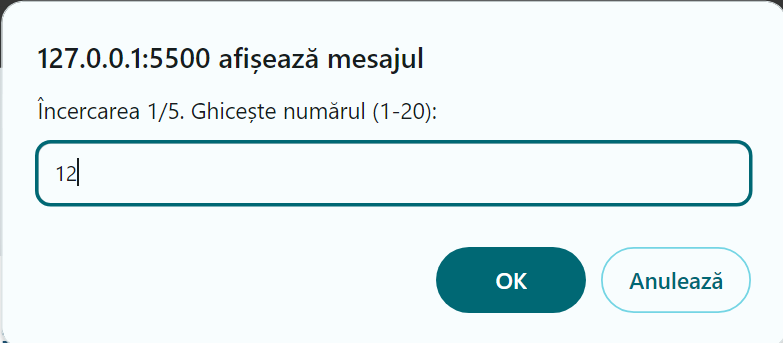
****

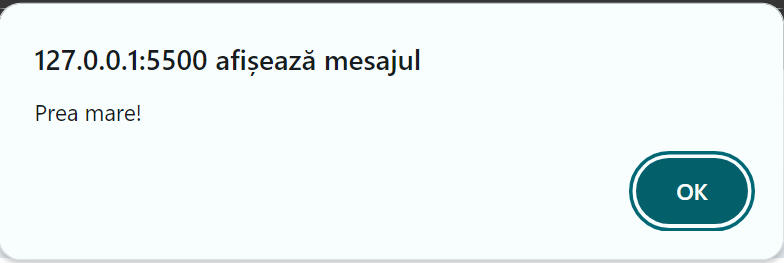
****

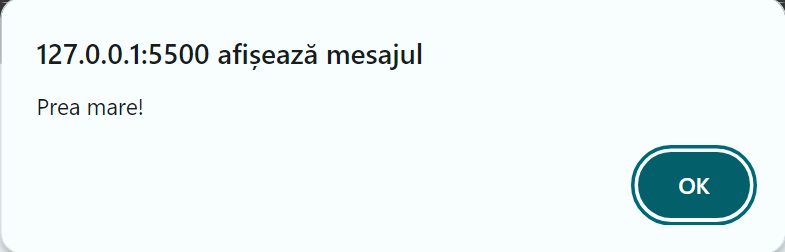
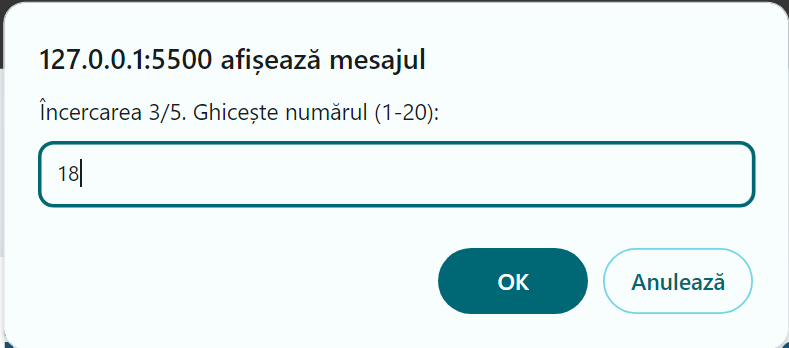
****

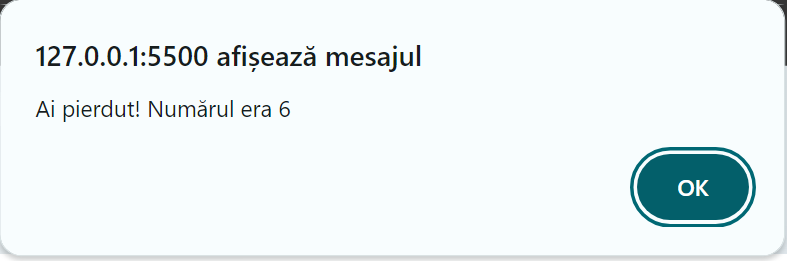
****

****

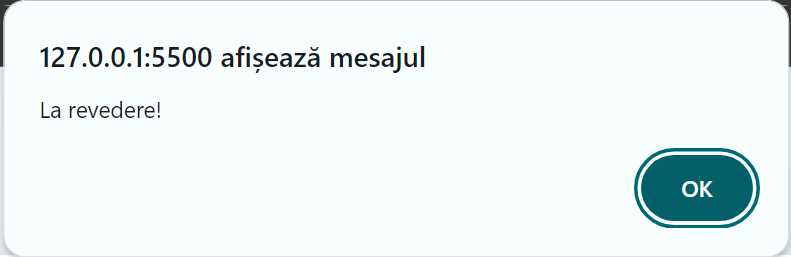
****

****

****

****

****

****

**CONCLUZIE**

În urma realizării lucrării de laborator, au fost aplicate cunoștințele teoretice privind limbajul JavaScript în dezvoltarea unei aplicații web client-side interactive. Lucrarea a contribuit la înțelegerea modului de organizare a codului, utilizarea funcțiilor, controlul fluxului de execuție și interacțiunea cu utilizatorul, respectând cerințele și bunele practici de programare.