# Planificarea proiectului

## Regulament de notare a proiectului

În acest document vom avea împărțirea în etape a proiectului cu detalii despre taskuri și penalizări.

### Bonusuri și penalizări

**În saptămâna cu punctaj normal poate fi în continuare cu bonus în următoarele situații:**

1. studentul a prezentat în săptămâna cu bonus dar avea greșeli sau taskuri incomplete și a decis să îndrepte situația
2. am reprogramat pe cineva din motive justificate: probleme tehnice (unuia dintre noi nu ne-a mers calculatorul sau netul etc), probleme administrative (nu a putut prezenta avand probleme cu căminul, cu secretariatul etc), probleme de sănătate, e om serios și ar fi terminat complet tema dar s-a aglomerat cu alte materii etc.
3. o amânare oficială, de comun acord cu toți studenții, fiind necesare explicații suplimentare la curs/laborator.

**Atentie! Perioada de punctaj normal nu va avea bonus aplicat dacă studentul se prezintă pentru prima oară cu etapa proiectului rezolvată și fără o amânare justificată.**
**Bonusul** e în valoare de ~~**10%.**~~ **15%**
**Penalizările sunt de 10%.**

### Alte precizări

- **Conteaza ora la care ati trimis etapa si nu momentul prezentarii.** Puteti trimite etapa si daca e undeva pe la 90% gata sau cu mici buguri si tot consider bonusul cat timp imi spuneti situatia, si chiar puteti remedia problema pana la prezentare. (evident nu se considera bonusul pentru o etapa care e abia inceputa sau cu foarte multe lucruri gresite). De asemenea consider bonusul si daca ati intarziat cateva minute (nu cateva zile!). Deci incercarea mea e sa nu vedeti deadline-ul ca fiind ceva stresant ci dimpotriva sa aveti curaj sa trimiteti ce ati lucrat si sa puneti interbari cu privire la ce nu a mers.
- **O etapa poate fi prezentata oricand in cadrul semestrulu**i insa e bine sa prezentati cat mai curand cand aveti codul proaspat in minte. De asemenea, daca aveti greseli ori lucruri lipsa la o etapa, care ar duce la depunctari, puteti sa recuperati punctele in etapa urmatoare, daca ati modificat/completat acele lucruri in cadrul proiectului.
- **În cazul depunctărilor rezultate din greșeli sau lipsa unor taskuri, se poate recupera punctajul dacă remediați acele probleme până la prezentările următoare.** Punctele nu sunt pierdute pentru totdeauna, puteți aduce completări în săptămâna 14 pentru etapa 1 fără a afecta bonusul sau penalizarea etapei deja scrise în tabel.
- **Va rog sa trimiteti etapele pe teams** (fisierul html), nu pe mail sau pe site. Cei care au pus proiectul pe github doar imi trimit linkul sau daca l-au trimis deja, **anunta pe teams ca au facut modificarea (mesajul cu anuțul trebuie trimis înainte de deadline, ca să pot verifica)**.
- Etapele de la 1 incolo trebuie prezentate pentru a lua puncte pe ele.
- **Bonusurile din cadrul unei etape pot fi rezolvate și mai târziu fără penalizări** (adică puteți, de exemplu să faceți bonusul dupa 3 săptămâni de la deadline) însă bonusului i se va aplica penalizarea sau bonusul de timp al etapei. De exemplu, etapa a fost făcută cu bonus de timp, și după 3 săptămâni de la deadline trimiteți și un task bonus, i se va aplica și lui bonusul de timp al etapei (iar dacă există penalizare de timp i se va aplica penalizarea). Motivul: încurajarea studentului de a face taskurile obligatorii la timp..

Linkuri utile pentru proiect

**https://pixabay.com/ (imagini gratuite)**
**https://www.videvo.net/ (videoclipuri gratuite - cele marcate cu "free")**

# Deadline-uri ID de bonus

Etapele 1-4: 7 aprilie 2024

# Etapele proiectului

| **Etapa** | **Perioadă bonus** | **Perioadă punctaj normal** | **De când încep penalizările săptămânale** |
|---|---|---|---|
| **Etapa 0 (0.3)** | - | nu are bonus; se trimite pana pe 02.03.2024 (inclusiv) fiindca daca nu aveti tema de proiect nu aveti cum sa lucrati la etapele urmatoare | |
| **Taskuri etapa 0 (0.3)**<br>1. Alegerea temei<br>2. Descrierea succintă a temei (google docs, 1/2-1 pagina). Va contine următoarele informații:<br>a. (0.02) Impartirea informatiilor, serviciilor etc. pe categorii si subcategorii.<br>b. (0.03) Identificarea efectiva a paginilor (încercați să fie doar 4-5) și partial a legaturilor dintre ele (ce informatii vor avea pagini separate si care informații vor fi subsecțiuni în aceeași pagină<br>c. (0.05) Stabilirea cuvintelor/sintagmelor cheie (o lista cu cuvintele cheie ale site-ului cât și cate o lista pentru fiecare pagină în parte)<br>d. (0.2) Căutarea unor site-uri similare (4-5 site-uri) ca tema pentru a observa modul de organizare a informației. Veti observa pentru ficare site cum au impartit informatiile si cum au facut designul. Veti nota pentru fiecare site ideile demne de implementat, dar și neajunsurile acelor site-uri (pentru fiecare site minim 2 lucruri pro si 2 contra). Veți pune linkuri către ele în fișier. | | | |
| **Etapa 1** | 04.03.2024 - 11.03.2024 | 12.03.2024 - 18.03.2024 | 19.03.2024 - |
| **Taskuri etapa 1 (punctaj recomandat 0.5)**<br>Creați prima pagină a site-ului (**doar prima pagină**; **fără stilizare** încă, fiindcă veți primi taskuri legate de acest aspect). Puteți pune în această pagină text care va fi mutat în alte pagini, mai târziu, dar **nu faceți încă mai multe pagini fiindcă le vom genera prin Node!** La prezentare vă rog să aveți pentru fiecare task notată linia din program la care l-ați rezolvat ca să nu dureze prezentarea mai mult de 3-4 | | | |

minute.

1. Creați un folder al proiectului care va cuprinde toate fisierele necesare site-ului vostru. Creați în el un fisier numit index.html. Deschideți acest fișier cu un editor de text care marchează sintaxa. Adăugați în fișier doctype și setați limba documentului în tagul html
2. Adaugati un title corespunzător conținutului textului. Folosiți 4 taguri meta relevante pentru a specifica: charset-ul, autorul, cuvintele cheie, descrierea.
3. Creați un folder (de exemplu numit "resurse") care va conține toate fișierele folosite de site, dar care nu sunt pagini html (de exemplu imagini, fisiere de stilizare etc). In el creati un folder numit **ico**. Adaugati un favicon relevant pentru temă. Folosiți https://realfavicongenerator.net pentru a genera toate dimensiunile necesare de favicon și codul compatibil pentru diversele browsere și sisteme de operare. Pentru favicon transparent, trebuie sa setati si o culoare a tile-ului (de background), care trebuie specificata și în tagul meta: <meta name="msapplication-TileColor" content="...culoarea aleasa de voi...">
4. Împărțiți body-ul în header, main, footer.
5. Folosiți minim un tag dintre: section, article, aside. Trebuie să existe măcar un caz de taguri de secționare imbricate (secțiune în secțiune). Puneți headingul cu nivelul corespunzător nivelului imbricării. Atenție, nu folosim headinguri decât ca titluri pentru tagurile de secționare. **Observație:** nivelul headingului trebuie să corespundă nivelului de imbricare a secțiunii (de exemplu un tag de secționare aflat direct în body are titlul scris cu h2, dar un tag de sectionare aflat intr-un tag de secționare care la rândul lui se află în body, va avea titlul scris cu h3
6. În **header** faceți un sistem de navigare ca în curs (nav cu listă neordonată de linkuri), cu opțiuni principale (care vor reprezenta paginile site-ului) și secundare (pentru opțiunea "Acasă", adică pagina principală, subopțiunile vor cuprinde linkuri către secțiunile paginii, care vor avea id-uri relevante). Folosiți în header h1 pentru titlul site-ului.
7. În cadrul secțiunilor folosiți minim 2 taguri dintre următoare taguri de grupare: p, ol, ul, blockquote, dl
8. Adăugați în pagină o imagine cu descriere, folosind figure și figcaption. Pe ecran mic (mobil) trebuie să se încarce o variantă mai redusă în dimensiune (bytes) a imaginii, pe tabletă o variantă medie, iar pe ecran mare varianta cea mai mare a imaginii. Folosiți un editor grafic pentru cropping și redimensionare pentru a obține cele 3 variante de imagini.
9. Textul trebuie să conțină toate cuvintele cheie identificate pentru pagina curentă. Puteți găsi mai multe sintagme cheie pe care le puteți folosi, cu **https://www.wordtracker.com/** sau https://app.neilpatel.com/en/ubersuggest/keyword_ideas

   Acestea trebuie să apară de mai multe ori în pagină, în taguri relevante.

10. În cadrul textului îndepliniți **3** dintre cerințele de mai jos, **la alegere**:
    a. marcați cuvintele și sintagmele cheie cu ajutorul tagului b
    b. marcați textul idiomatic (termeni științifici, în altă limbă, termeni tehnici, de jargon, etc) cu tagul i
    c. marcați textul de atenționare cu strong
    d. marcați textul accentuat cu em
    e. marcați textul șters (corectat sau care nu mai e relevant) cu tagul s și textul inserat în loc cu tagul ins
    f. marcați o abreviere cu abbr și cu atributul title specificați sintagma abreviată
    g. marcați un termen definit cu dfn
    h. marcați un citat cu tagul q
11. Creați următoarele linkuri speciale:
    a. un link extern (va fi in continutul paginii, nu in meniu, va face referire la alt site si se va deschide in fereastră nouă)
    b. un link în footer către începutul paginii,
    c. minim două linkuri care se deschid într-un iframe (se poate face ca în exemplul de curs, linkuri care deschid videoclipuri relevante de pe youtube în iframe). Atentie nu e vorba de src-ul iframe-ului ce de taguri <a> care la click se deschid in iframe. Iframe-ul va contine in mod default una dintre resursele specificate in linkuri
    d. Un link de tip download
12. Creați în pagină mai multe zone de details și summary. Pot fi întrebări frecvente, pot fi niște oferte pentru care afișăm titlul și utilizatorul le deschide pe cele care îl interesează, pot fi secțiuni explicative etc.
13. În footer se vor adăuga cu ajutorul tagului address informații de contact:
    a. telefon fictiv, marcat cu tagul <a> si URI Scheme-ul corespunzător
    b. adresă fictivă care la click deschide o locatie pe Google Maps (locatia in mod normal ar corespunde cu adresa dar voi veti pune drept locatie in maps, Facultatea de Matematica si Informatica)
    c. e-mail fictiv, marcat cu tagul <a> si URI Scheme-ul corespunzător în href
    d. Link care deschide o aplicatie de comunicare precum skype sau whatsapp pentru chat
14. În footer se va adăuga informație de copyright, folosind tagul small, simbolul specific de copyright cu codul html necesar (forma &cod;) și data creării paginii scrisă în limba română și pusă în tagul time cu **atributul datetime** corespunzător.
15. Pagina trebuie sa fie valida din punct de vedere sintactic. Deci verificati cu validatorul html. Validatorul va fi pregătit într-un tab, la prezentare, și pagina se va valida pe loc.

Se poate da bonus pentru o temă bine făcută sau pentru folosirea mai multor taguri decât minimul specificat (dar studentul trebuie să anunțe

la prezentare că le-a folosit)

Evitați pentru moment să adăugați alte taguri fiindcă vor apărea în taskurile următoare.

**Bonus (0.05) Folosirea unei formule scrise în MathML - formula trebuie să aibă sens în contextul site-ului.**

| | | | |
|---|---|---|---|
| **Etapa 2 (punctaj recomandat:-0.4)** | 12.03.2024 - 21.03.2024 | 21.03.2024 - 25.03.2024 | 26.03.2024 |

**Taskuri etapa 2 (punctaj recomandat:-0.4)**
**Atentie - unele cerinte au enunt diferit pentru fiecare student (si sunt marcate printr-un link). Trebuie sa va inregistrati pe site pentru a le vedea.**
**Daca stilizarea dintr-o cerinta nu va place, puteti sa imi cereti o alta varianta (imi scrieti pe chat). Culorile din imaginile si videoclipurile date ca exemplu nu trebuie respectate (folositi culorile din schema cromatică aleasă de voi).**
(0.025) Task schema cromatica: (cerință individuală)
(0.15) Task layout: (cerință individuală)
(0.05) Task design rudimentar:

a. Folosind variabile css, să se adauge o spațiere în stânga și dreapta paginii, comprimând un pic conținutul, dar aerisind pagina. Spațierea trebuie să fie identică în cele două direcții. Spațierea va fi mai mică pe ecran mediu și de valoare minimă pe ecran mic.
b. Folosiți grid-gap pentru a realiza o spațiere între celulele gridului paginii. Spațierea trebuie să descrească pe ecran mediu și mic.
c. Izolați vizual zonele paginii: header, footer, și zonele gridului folosind minim 3 dintre următoarele efecte css: background de culoare (aleasă din schema cromatică) diferită pe zone diferite, border, colțuri rotunjite ale box-ului, box-shadow
d. Folosiți padding pentru a distanța textul din zone de granițele zonelor. Padding-ul ar trebui să fie egal pentru toate zonele de text. Folosiți variabile CSS în cazul în care nu puteți asigura acest lucru prin simplii selectori.
e. Elementele media (imagini, videoclipuri etc.) care vin cu o lățime presetată vor primi lățimea în procentaje, setând, totuși, o lățime maximă si minimă pentru ele, pentru a nu avea efecte vizuale nedorite. Aceste valori pot să difere în funcție de dimensiunea ecranului

(0.025) Task iconuri și font extern. Folosiți in prima pagină a site-ului un font extern prin Google API. Folosiți în pagină, într-un loc relevant un icon static și unul animat (**diferite de iconurile cerute în eventuale alte taskuri**) din colecția Font Awesome (fie folosind kit - recomandat - fie folosind linkul CDN ca in exemplu: https://replit.com/@IrinaCiocan/curs4-exemple#fontawesome.html)
(0.05) Task tabel: (cerință individuală)
**Indicații de rezolvare:**
1. Porniți de la slide-ul cu tabelul din cursul 1 de HTML și schimbați conținutul ca să fie conform temei voastre.
2. Pentru taskul cu valori alternate folosiți pseudoclasele :nth-child(odd), :nth-child(even) cu elementele cerute (td pentru celule(coloane) și tr pentru rânduri)
3. Proprietățile de care aveți nevoie sunt în special la CSS Styling Tables (w3schools.com)
4. Bara de scroll se poate realiza cu ajutorul proprietății overflow.

(0.05) Task taburi iframe: (cerință individuală)
**Indicatii de rezolvare:**
1. Creati un container în care puneti linkurile care se deschid in iframe (cel care are clasa "taburi-iframe" in exemplul de mai jos). De asemenea grupați containerul linkuri și iframe-ul într-un alt container (cel cu clasa container-iframe din exemplul de mai jos)

```
<div class="container-iframe">
  <div class="taburi-iframe">
  <a href="https://www.youtube.com/embed/qtlhdIfojmc" target="ifr-video">Reteta tort vanilie</a>
  <a href="https://www.youtube.com/embed/dsJtgmAhFF4" target="ifr-video" >Reteta tort ciocolata</a>
  <a href="https://www.youtube.com/embed/Wtxbt-CpA2s" target="ifr-video" >Reteta tort căpșuni</a><br/>
  </div>
  <iframe name="ifr-video" width="560" height="315" src="https://www.youtube.com/embed/qtlhdIfojmc"
  frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture"
allowfullscreen></iframe>
</div>
```

2. Stilizati tagurile <a> sub forma de butoane (width si height setat, background, border)

3. Aplicati display:flex pe containerul care trebuie sa contina coloane (fie container-iframe, fie taburi-iframe in functie de cerinta)

(0.05) Link top: (cerință individuală)
**Indicatie de rezolvare:** Porniti de la exemplul: exemplu_layout-1 - Replit - cautati in html elementul cu id-ul "link-top", si fiul lui care e un div cu id-ul "triunghi". Preluati codul css pentru ele si il modificati pentru a obtine rezolvarea la acest task.
-
-
**Bonusuri:**

- **(0.05) Resetarea css-ului cu redefinirea spațierilor, dimensiunilor, culorilor, stilurilor bol si italic, eventual a bulleturilor și indicilor de listă, a stilului tabelelor). In afară de body și html care vor primi dimensiuni în unități fixe, toate celelalte elemente vor folosi unități relative. Se vor folosi variabile pentru valori care se repetă și depind logic unele de altele. Puteti folosi de exemplu: https://meyerweb.com/eric/tools/css/reset/**
- **(0.05)Stilizarea unei formule scrise în MathML.(părți diferite din formulă trebuie să aibă stil diferit, de exemplu culori diferite sau font italic/bold vs text normal).**

| **Etapa 3** | 18.03.2024 - 25.03.2024 | 26.03.2024 - 01.04.2024 | 02.04.2024 |
|---|---|---|---|

**Taskuri etapa 3 (punctaj recomandat: 0.4)**
**Atentie - unele cerinte au enunt diferit pentru fiecare student (si sunt marcate printr-un link). Trebuie sa va inregistrati pe site pentru a le vedea.**

(0.25) Task meniu: (cerință individuală)
(0.15) Stil printare (cerință individuală)

**Bonusuri:**

- **(0.05) Icon-ul meniului "hamburger" să fie creat cu 3 divuri (eventual puse într-un div container în locul unei imagini), cele 3 divuri vor primi background, width și height pentru a simula dreptunghiuri și vor fi pozitionate absolut în interiorul containerului.**
- **(0.05) Cand se trece pe ecran mic și apare iconul pentru meniu, apariția să fie făcută printr-o animație asupra divurilor care să implice schimbarea tuturor următoarelor proprietăți: culoarea celor 3 bare, o transformare geometrică, opacitate. Puteți schimba și alte proprietăți dacă doriți. Animația trebuie să aibă minim 3 cadre cheie.**
- **(0.05) Pentru bonusul anterior, fiecare bară din hamburger-menu să aibă asociată o animație, însă animațiile să înceapă succesiv cu o diferența de t milisecunde (de exemplu t=300). Delayurile diferite în cadrul animației se vor genera cu o instrucțiune for scrisă în sass.**
- **Se vor mai adauga.**

| **Etapa 4** | | | |
|---|---|---|---|

**Taskuri etapa 4 (punctaj recomandat 0.6)**
(0.5)**Trecerea site-ului pe node** și crearea de fișiere EJS conform cerințelor:

1. În folderul proiectului dați comanda **"npm init"** și setatți numele, autorul, descrierea și cuvintele cheie pentru proiectul vostru. Instalați express și ejs.
2. Se va crea în rădăcina proiectului un fișier index.js. În el se va crea un **obiect server express** care va asculta pe portul 8080. (sau alt port daca aveti deja folosit 8080)
3. Să se afișeze calea folderului în care se găsește fișierul index.js (**__dirname**), calea fișierului (**__filename**) și folderul curent de lucru (**process.cwd()**). Sunt __dirname și process.cwd() același lucru întotdeauna?
4. Se va folosi EJS pentru generarea (randarea) paginilor. Se va face un **folder numit views** în rădăcina proiectului. În el veți face un folder numit **pagini** (care conține paginile întregi) și altul numit **fragmente** (care conține părți de pagini (bucățele de cod html) ce pot fi refolosite pe mai multe pagini). Instalați în Visual Studio Code extensia **EJS language support**.
5. Din index (care va fi redenumit **index.ejs**) se vor decupa headerul și footerul și se vor pune în ejs-uri separate. De asemenea se va decupa partea de head care conține codul care nu se schimbă în funcție de pagină (de exemplu, tagul meta cu encodingul sau autorul, includerea faviconului, fișierelor css generale (nu specifice paginii) a scripturilor generale etc). Se va folosi funcția **include()** în fișierele ejs, pentru a include toate aceste fragmente în pagini

6. Se va realiza (dacă nu l-ați făcut deja) un **folder special cu toate resursele site-ului** (în stilul exemplului de la curs în care am pus toate fișierele statice, precum imagini, fișiere de stil, videoclipuri etc în folderul "resurse"). Numele folderului îl decideți voi, însă va trebui să fie structurat, de asemenea, în subfoldere în funcție de tipul și modul de utilizarea al fișierelor. Se va defini în program acest folder ca fiind **static**
7. Se vor schimba **căile fișierelor-resursă** folosite în pagini, astfel încât **să nu mai fie relative** ci stil cerere către server (de exemplu, /resurse/stiluri/ceva.css în loc de, de exemplu, ../resurse/stiluri/ceva.css)
8. Prima pagină (index) trebuie să se poată accesa atât cu **localhost:8080** cât și cu **localhost:8080/index**, **localhost:8080/home**. Realizați acest lucru folosinf un vector în apelul app.get() care transmite pagina principală
9. Veți declara un app.get() general pentru calea "/*", care tratează orice cerere de forma /pagina randând fișierul pagina.ejs (unde "pagina" e un nume generic și trebuie să funcționeze pentru orice string). Atenție, acest app.get() trebuie să fie ultimul în lista de app.get()-uri. Dacă pagina cerută nu există, se va randa o pagină specială de eroare 404 (în modul descris mai jos).
10. Se va da ca argument în funcția render o funcție callback function(eroare, rezultatRandare) care, în cazul în care mesajul erorii începe cu "Failed to lookup view" va afișa pagina pentru eroarea 404 (vezi mai jos). în cazul în care e altă eroare va afișa pagina de eroare generică (vezi mai jos), iar dacă nu sunt erori va trimite către client rezultatul randării.
11. Pentru randarea erorilor, veți folosi un fișier json, numit erori.json. Acesta va avea următoarele proprietăți:
    a. cale_baza: calea la care se găsesc imaginile corespunzătoare erorilor
    b. eroare_default: va fi un obiect JSON cu proprietățile: titlu (titlul paginii de eroare), text, imagine
    c. info_erori: un vector de obiecte. fiecare obiect descrie o eraore și are proprietățile: identificator (un cod numeric; pentru erorile http, precum 403, 404 e chiar codul http), status (boolean prin care indicăm dacă trebuie alt cod status decât 200 pentru răspuns), titlu (titlul erorii, pus in heading), text (descrierea erorii), imagine(o imagine descriptiva pentru eroare). Se vor adăuga în proiect în claea specificată în cale_baza imagini pentru erorile definite.
12. Se va crea un template (eroare.ejs) cu ajutorul căruia să se afișeze erorile. Acesta va avea, preluate din locals, titlul, textul imaginea erorii.
13. Se va crea o funcție care citește JSON-ul cu erorile și creează un obiect corespunător lui cu toate datele erorilor (pentru a le avea încărcate în memorie).
14. Se va crea o funcție de afișare a erorilor care va primi un obiect de tip Response identificatorul, titulul, textul și imaginea erorii. În cazul în care există o eroare cu acel identificator, și titlul, textul și imaginea nu sunt precizate, se preiau datele încărcate din JSON pentru afișarea erorii. Dacă una dintre cele 3 proprietăți ale erorii e dată ca argument în funcție, are rpioritate asupra datelor din JSON și se va afișa în pagină (de exemplu dacă titlul e dat ca argument, se afișează argumentul nu titlul citit din JSON). În cazul în care identificatorul nu se specifică, se afișează o pagină de eroare cu datele din eroare_default, însă totcu posibilitatea de a afișa titlul, textul și imaginea din argumente, dacă sunt precizate.
15. Veți mai face încă o pagină (cu puțin text sau imagini, ca să aibă conținut), de exemplu o pagină cu descrierea site-ului sau istoricul său, al firmei virtuale pentru care este făcut etc. Această pagină trebuie să poată fi accesată prin meniu (linkul să fie corect și să transmită o cerere de tip get). Nu faceti încă pagina de produse, findcă pe acelea le preluăm din baza de date. Nici paginile de înregistrare sau login, fiindca le tratăm separat.
16. În zona din layout de date despre utilizator vom afișa ip-ul utilizatorului (prin program). Deocamdată, site-ul fiind local, veți vedea mereu ip-ul de localhost (adică ::1). Ip-ul real se va vedea când adăugați site-ul pe Internet.
17. La o cerere către o cale din /resurse(de exemplu,localhost:8080/resurse/css/ ) fără fișier specificat (către folderul care ar conține fișierul) se va returna eroarea 403 Forbidden. Pagina de 403 va avea format similar cu cea de 404, folosind același template (eroare.ejs), dar textul și imaginea schimbate corespunzător, preluate din JSON
18. Să se adauge un app.get() pentru "/favicon.ico". Uneori browserele cer favicon pentru diverse răspunsuri primite de la server (nu neapărat fisiere html, unde putem specifica faviconul prin taguri link). Pentru această cerere vom trimite un favicon cu metoda sendFile().
19. La cererea oricărui fișier cu extensia ejs se va transmite o eroare de tip 400 Bad Request. Pagina de 400 va folosi același template (eroare.ejs), dar textul și imaginea schimbate corespunzător, preluate din JSON
20. Proiectul nostru va folosi niște foldere în care generează fișiere. Scrieți un vector cu numele folderelor de creat (vectorul va conține doar stringul "temp", deși pentru testare puteți pune și "temp1" pe care apoi îl ștergeți). Se va itera prin vector și se va testa dacă folderele există. Dacă un folder nu există, este creat. Peste tot unde aveți concatenare de căi folosiți path.join().

(0.05) Task video (cerință individuală)

(0.05) Task linkuri (cerință individuală)

| | | | |
|---|---|---|---|
| | | | |

**Etapa 5 (1.1p)**
(0.35) **Galeria statica** (cerință individuală)
(0.05) **Se va face un repository local si se va pune proiectul pe Github. Se vor adauga în .gitignore node_modules și folderul temp.**
(0.25) **Compilare automata scss.** Se vor realiza următoarele subpuncte:

a. **Pregătire cadru de lucru.** Se vor defini în obiectul global două proprietăți *folderScss* și *folderCss* care conțin căile din folderul de resurse (depinzând de __dirname). Se va adăuga folderul *backup* la lista folderelor create automat de aplicație (așa cum e și folderul temp)
b. **Funcția de compilare a scss-urilor.** Se va face o funcție **compileazaScss**(caleScss, caleCss){} care compileaza un fișier scss în fișier css. Primii 2 parametrii reprezintă căile către fișierul scss (inputul funcției) și fisierul css (outputul funcției). Dacă avem căi absolute se iau fișierele de la cele două căi, iar dacă sunt relative se vor considera relative la *folderScss*, respectiv *folderCss*. compilarea se va face cu ajutorul pachetului sass. Dacă numele/calea fișierului css lipsește, se va salva în folderCss rezultatul compilarii folosind numele fișierului scss, dar cu extensia css
c. **Salvare în backup.** În cadrul funcției compileazaScss, înainte de compilarea automată a scss-ului în fișierul css asociat, fișierul css vechi cu același nume va fi copiat în subcalea **resurse/css** a folderului backup. Orice folder din această subcale va fi creat dacă nu există deja. Se va afișa un mesaj de eroare în cazul eșecului copierii.
d. **Compilare inițială.** La pornirea serverului, toate fisierele scss din *folderScss* trebuie să fie compilate în fișierele css cu același nume folosind funcția compileazaScss. Înainte de suprascrierea fișierului css, acesta va fi copiat în folderul backup (suprascriind un backup cu același nume - sau dacă vreți să păstrați backup-urile anterioare puteți integra în nume o informație cu privire la timpul creării.
e. **Compilare pe parcurs.** Se va scrie cod (folosind fs.watch()) astfel încât să se urmărească modificările din folderul de fișiere scss. La modificarea/crearea unui fișier acesta va fi compilat automat în css. Fișierul css va acea același nume cu fișierul scss, având doar extensia scss schimbată în css. Înainte de suprascrierea fișierului css, acesta va fi copiat în folderul backup (suprascriind un backup cu același nume - sau dacă vreți să păstrați backup-urile anterioare puteți integra în nume o informație cu privire la timpul creării.

(0.2) **Customizare Bootstrap** cu schema cromatică aleasă de voi si cu dimensiuni de ecran diferite pentru ecrane medii și mari. Veți face un fișier numit custom.scss în care folositi fisierul scss al bootstrap, schimband valori pentru:

- culorile de background pentru minim 2 teme la alegere pe care aplicați tema cromatică schimbată (customizată de voi)
- culori de font (adică ale literelor. **Precizare**: nu neaparat pentru toata pagina, orice culoare de litere - de exemplu dintr-un buton)
- dimensiuni de ecran diferite pentru ecrane medii și mari
- dimensiunea razelor de border
- dimensiunea literelor headingurilor (h1,h2 etc)
- familia de font implicită
- încă una sau mai mule variabile alese de voi

**Corectare Bootstrap.** Atenție, este posibil ca integrarea bootstrap să afecteze aspectul site-ului deoarece pentru anumite elemente din pagină v-ați bazat pe aspectul implicit al acestora, prin urmare va fi nevoie să definiți reguli css astfel încât site-ul să revină la forma inițială. CSS-ul pentru bootstrap trebuie pus primul pentru a asigura suprascrierea proprietăților pentru selectorii-tag.
Veți folosi unul sau mai multe elemente de bootstrap care să ilustreze schimbările, dintre cele de mai jos:
Customizarea se va face ca la laborator folosid sass, iar **compilarea se va face automat la repornirea serverului folosind funcția compileazaScss!.**

(0.25) (**Efecte CSS**)
**Observație:** având în vedere că depășirea punctajului pentru lista de efecte de mai jos se transformă în bonus adunat la proiect, aceasta lista poate fi expandată pe tot parcursul semestrului (deci inclusiv după deadline-ul etapei 4), adăugând noi idei penru bonusuri, deoarece bonusurile pot fi prezentate si mai tarziu. **Atenție! Anumite efecte au cerință individuală asociată (care trebuie respectată, pentru a fi punctate) - enunțurile acestora au identificatorul prefixul "efect_css".** Cele care nu au textul "cerinta individuală" au enunțul întreg în acest fișier.
**Efecte css propuse** (se vor **alege** efecte css de implementat ca sa **însumeze 0.25** - nu e obligatoriu să le faceți pe toate din listă)

- (0.05) **Duotone** (cerință individuală)
- (0.15) **Reflexie** (cerință individuală)
- (0.025) **Scrierea textului pe coloane**, folosind proprietatea column-count. Se va alege o secțiune cu mai mult text pe care se va aplica proprietatea column-count. Pe ecran mic și mediu se va afișa o singură coloană. Între coloane se va afișa și o linie despărțitoare (column-rule)
- (0.025) **Schimbarea afișării implicite a textului selectat**, folosind pseduo-clasa ::selection - schimbați minim 2 proprietăți ale textului selectat, folosind variabilele definite pentru schema cromatică.

- (0.05) text care se plimba orizontal sau verticalprintr-o animație (keyframes) recurentă (după ce se termină mesajul, reapare). Elementul cu textul trebuie să fie responsive (să nu apara scrollbar orizontal pe pagină din cauza lui
- (0.05) **background fix la scroll** într-una din pagini, folosind background-attachment. Imaginea de background se va schimba (printr-o animație) dupa *t* secunde (*t* e ales de voi).
- (0.05) **Afișarea unui tabel astfel încât să fie responsive**, conform exemplului dat. Tabelul ales trebuie să aibă minim 4 coloane și să nu conțină celule cu rowspan/colspan.
- (0.025) **Afișarea unui tabel transpus** pe o dimensiune de ecran (media query) conform exemplului. Pe restul dimensiunilor de ecran se va afișa la fel.
- (0.1) **Stilizare hr** (cerință individuală)
- (0.05) **videoclip care se comporta ca un background**, conform exemplului de la https://css-tricks.com/full-page-background-video-styles/

**Bonus 1:**
(0.5) **galeria animata** (cerință individuală)
**- se poate considera ca bonus sa faceti mai multe efecte css care ar depăși cele 0.25 puncte.**
**Bonus 2:**
**(0.05) Fișierele salvate în backup (în funcția compileazaScss, în urma compilării scss -> css) să aibă în nume o informație de timp (de exemplu, în loc de a.css să fie fișierul a_timestamp.css, de exemplu a_1681124489791.css) pentru a putea salva mai multe versiuni ale aceluiași fișier.**

**Taskuri etapa 6 (baza de date + JavaScript introductiv) - afișarea produselor (punctaj recomandat 1.5p)**

**(1p) afisare + sortare/filtrare/calculare** (cerință individuală)
**(0.3p - fiecare e 0.05) Stilizare inputuri** . Inputurile de pe pagina de produse se vor stiliza cu ajutorul Bootstrap, folosind customizari facute de voi in sass (vezi etapa 5). Puteti sa mai adaugati variabile in fisierul de customizare, care sa ajute la stilizarea butoanelor si inputurilor.

a. **Butoanele (de filtrare, sortare, resetare) trebuie stilizate cu ajutorul temei bootstrap pentru care ati schimbat culoarea** (de exemplu primary sau secondary). **Raza si grosimea borderului** trebuie date de voi in fisierul sass de customizare prin variabile (daca nu ati facut asta deja). Butoanele si inputurile **vor fi dimensionate cu ajutorul claselor bootstrap**. Butoanele trebuie sa aiba iconuri (glyphicons) relevante din bootstrap: https://icons.getbootstrap.com/. Pe ecran mic se vor afisa doar iconurile fara text (realizati acest task in mod cat mai eficient - scris cat mai putin)
b. Inputul de tip textarea va avea un **floating label** (bootstrap). In cazul validarii esuate a valorii din textarea (vezi cerinta cu validarea din etapa 5), floating label-ul va fi de tip *is-invalid* (se va seta prin javascript) si se va corecta automat daca valoarea din textarea devine valida.
c. Fie inputurile de tip checkbox fie cele radio vor fi stilizate ca **toogle buttons**(din bootstrap). Butoanele deslectate trebuie sa fie desenate in stil outline (adica sa apara doar granita, fara background si textul sa fie colorat), iar la selectare sa apara cu background colorat - vezi clasele *btn-outline*). Atentie trebuie folosit bootstrap nu css!
d. **Asezarea inputurilor** in mod ordonat si aliniat pe coloane in cadrul paginii de produse se va face printr-un **grid bootstrap** (vezi clasele row si col, impreuna cu clasele asociate query-urilor media).
e. **Butonul pentru schimbarea temei sa fie un switch** din bootstrap (imagineasau iconul cu luna/soarele, ramane in pagina, cu acelasi comportament) Observatie pt cei care au mai multe teme: puteti inlocui cu alt element de bootstrap.
f. **Pentru inputul de tip range**,schimbati cu ajutorul customizarii bootstrap (prin variabile scss) dimensiunea bulinei care gliseaza (sa fie cu 50% mai mare fata de dimensiunea font-size-ului html-ului), schimbati de asemenea culoarea bulinei si culoarea sliderului. Variabilele necesare se gasesc in reference: https://getbootstrap.com/docs/5.0/forms/range/

**(0.2p) light/dark theme** cu variabile CSS (optional in SASS). Pentru moment va exista un buton în pagină care va face schimbarea. Butonul va fi reprezentat printr-o imagine cu soare (pt light) vs lună (pt dark) - poate fi un icon din fontawesome. Tema aleasă se va memora in localStorage si se va pastra tema la urmatoarea intrare pe pagina si pe restul paginilor site-ului.

**Bonusuri: TO DO**

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |